
INHOUD

Hoofdstuk 1: Inleiding	I
Voor wie is dit boek?	4
Software- of andersoortige projecten?	6
Hoe kunt u dit boek gebruiken?	6
Taalgebruik	6
Hij, of zij of het	7
Hoofdstuk 2: Wat is Scrum?	9
Definitie	9
Korte geschiedenis	10
Agile-principes	12
Populaire Agile-methodieken	19
Vergelijking met Watervalmethoden	23
Wat is Scrum niet?	28
Hoofdstuk 3: Scrum in theorie	31
Scrum-proces	31
Project starten	33
Productvisie (product vision statement of product goal)	34
Productbacklog	35
Opleverplan (release planning)	49
Mensen, rollen en verantwoordelijkheden	51
Sprint	57

Hoofdstuk 4: Starten met de Scrum-methodiek	69
Introduceren van Scrum binnen het bedrijf	70
Project kiezen	70
Scrum-team samenstellen	71
Huisvesting van het Scrum-team	87
Hoofdstuk 5: Scrum-project starten	89
Productvisie opstellen	89
Definition of Done opstellen	92
Productbacklog opstellen	96
Opleveringen plannen	117
Hoofdstuk 6: Sprints uitvoeren	125
Sprint	125
Sprint-planning	126
Sprint uitvoeren	135
Sprint-reviewvergadering	144
Sprint-evaluatie (retrospective)	146
Sprint afbreken	155
Sprint-nul uitvoeren (Sprint zero)	156
Oplever-Sprint uitvoeren (hardening Sprint)	157
Hoofdstuk 7: Projectdocumentatie bijhouden	163
Hoeveel documentatie?	163
Productvisie (product goal)	164
Productbacklog	164
Sprint-backlog	166
Technische documentatie	167
Functionele documentatie	168
Blokkades en procesverbeteringen	169
Beslissingsdocumenten	169
Andere documentatie	170

Hoofdstuk 8: Opschalen naar meerdere Scrum-teams 171

Communicatieproblemen	172
Teams op meerdere locaties	174
Organisatiestructuur	175
Scaled Agile Framework (SAFe)	177
Nexus	188
Large Scale Scrum (LeSS)	193
Welk raamwerk kiezen?	203
Opschaling algemeen	204

Hoofdstuk 9: Hoe ‘verkoop’ ik Scrum binnen mijn bedrijf? 213

Inleiding	213
Argumenten tegen Scrum	214
Verkoopargumenten	223
Het Scrum-introductieproject	228
Aandachtspunten voor de introductie	232

Hoofdstuk 10: Hoe ‘verkoop’ ik Scrum aan mijn klanten? 235

Agile-principes uitleggen	236
Verkoopargumenten	238
Contracten en Agile-methoden	240
Contractvormen	241
Conclusie	260

Hoofdstuk 11: Wat te doen als Scrum niet werkt? 263

Scrum lijkt eenvoudig, maar is lastig	264
Scrum is slechts gereedschap	265
Scrum is niet altijd het juiste gereedschap	266
Situaties die de implementatie van Scrum hinderen	268
Scrumbut	275
Alleen onderdelen van Scrum gebruiken	277
Als zelforganisatie zijn werk niet kan doen	277
Scrum-teams met tegenstrijdige belangen	278

Hoofdstuk 12: Hoe meten we succes met Scrum?	281
Wat is succes?	281
OKR-methode (Objectives and Key Results)	282
Kritieke Prestatie Indicatoren (Key Performance Indicators, KPI's)	284
OKR's en KPI's in Scrum	287
Succescategorieën	288
Hoofdstuk 13: Vraag en antwoord	297
Het Scrum-proces	298
Teamindeling en werkzaamheden	308
Hulpmiddelen	311
Literatuur	313
Index	321

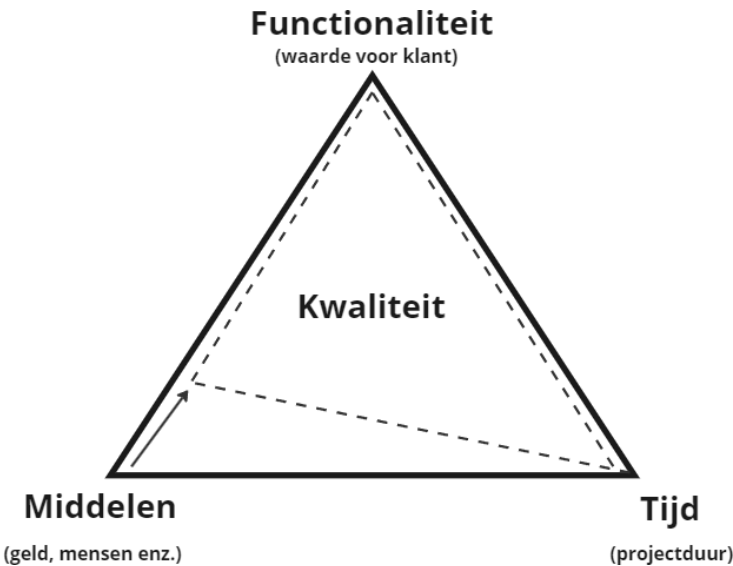
1 INLEIDING

Misschien bent u geïnteresseerd in Scrum omdat u in een bedrijf werkt waar Scrum wordt gebruikt en wilt u meer te weten komen over de methodiek? Misschien werkt u in een bedrijf dat Scrum nog niet gebruikt, maar het wil gaan gebruiken? Misschien wilt u software laten ontwikkelen door een softwarebedrijf dat zegt Scrum te gebruiken en wilt u weten wat dat inhoudt?

In ieder geval hebt u interesse in Scrum, anders zou u dit niet lezen. In dit boek wil ik u meenemen in de wereld van projectbeheer door middel van de Scrum-methodiek. Scrum is oorspronkelijk ontwikkeld in de wereld van de softwareontwikkeling en is daarin een van de meest succesvolle methodieken van de afgelopen decennia. Tegenwoordig wordt Scrum steeds vaker succesvol gebruikt buiten de IT-wereld. Het blijkt dat veel projecten in andere branches zich prima lenen voor Scrum.

In de wereld van projectbeheer wordt de term 'projectmanagement-driehoek' of 'IJzeren driehoek' (*Iron Triangle*) veel gebruikt. Deze driehoek geeft de relatie weer tussen de verschillende elementen van het projectbeheer: te ontwikkelen product, beschikbare middelen (zoals budget en mensen), de (doorloop)tijd van het project en de opgeleverde kwaliteit. De punten van de driehoek stellen de randvoorwaarden van het project voor. De oppervlakte van de driehoek bepaalt de kwaliteit van het eindproduct. Als we de kwaliteit van het eindproduct gelijk willen houden en daarmee dus het oppervlak van de driehoek, hebben wijzigingen in een van de punten direct een verschuiving van de andere twee punten tot gevolg. Als de beschikbare tijd voor het afronden van het project kor-

ter wordt gemaakt, heeft dat meestal tot gevolg dat de kosten (budget) zullen stijgen of dat de omvang (scope) van het product kleiner zal worden. Als het beschikbare budget wordt verkleind, wordt daarmee automatisch de omvang kleiner, maar kan het misschien wel eerder worden opgeleverd. Of de hoeveelheid beschikbare middelen wordt verkleind, waarmee de kwaliteit in gevaar komt (het oppervlak wordt kleiner). Deze driehoek gaat in principe op voor veel typen projecten en u zult vele varianten van de driehoek tegenkomen.



Afbeelding 1.1: De IJzeren driehoek.

Het is van belang te beseffen dat het vastleggen van twee van de drie punten in de driehoek automatisch ook het derde punt van de driehoek vastlegt als we de kwaliteit van het eindproduct (oppervlakte van de driehoek) niet willen verminderen. Veelal bepaalt de klant (de opdrachtgever van het project) al twee van de drie punten. De klant wil dat het project tegen een (lagere) vaste prijs wordt uitgevoerd, maar de opleverdatum en opgeleverde functionaliteit (waarde) mogen niet verminderen. Het kost vaak veel moeite om uit te leggen dat daarmee de kwaliteit van het eindproduct achteruit gaat. Als we praten over projectbeheer en projectbeheersing hebben we in feite altijd met deze vier factoren te maken. Een klant wil het liefst vaak vooraf bepalen wat hij krijgt, wanneer en tegen welke prijs. We zullen zien dat dit in de praktijk bijna niet mogelijk is. Als

de leverancier, de uitvoerder van het project, toch met zo'n project aan de slag gaat, zal meestal als eerste op de kwaliteit worden ingeleverd.

De Scrum-methodiek probeert met deze tegenstrijdigheden zo goed mogelijk om te gaan.

Scrum is een Agile-methodiek. Agile (vertaalt als lenig of flexibel) is een groep van projectmethodieken die alle dezelfde uitgangspunten hantieren en waarvan de Scrum-methodiek een van de oudste en bekendste is. Iedere Agile-methodiek, bijvoorbeeld Kanban, extreme programming (XP), Crystal Clear, Lean Development enzovoort, focust op andere aspecten van het projectbeheer. Veel van deze methodieken worden gebruikt in softwareontwikkeling, terwijl andere juist zijn ontwikkeld voor en door andere branches. Sommige (zoals extreme programming) richten zich voornamelijk op het technische aspect van softwareontwikkeling, terwijl andere (zoals Scrum) proberen het projectbeheer in goede banen te leiden door communicatiestructuren vast te leggen. Lean is ontstaan uit het optimaliseren van productieprocessen. Soms zijn verschillende methodieken goed te combineren omdat ze elkaar aanvullen. Doordat er zo veel verschillende methodieken zijn, is het vaak lastig voor iemand de voor- en nadelen van iedere techniek te overzien en te kiezen voor één bepaalde methodiek.

In mijn visie hebben alle methodieken bestaansrecht en het hangt vaak af van verschillende factoren, waaronder persoonlijke voorkeur, welke voor een bepaalde organisatie de beste methodiek is. In mijn ervaring is Scrum een methodiek die alle betrokkenen aanspreekt, omdat de basisregels zeer eenvoudig zijn. Ondanks dat Scrum zijn oorsprong in de softwareontwikkeling heeft, is de methodiek inmiddels gemeengoed bij vele andere soorten bedrijven en projecten. U hoeft niet technisch onderlegd te zijn om Scrum te begrijpen en in te zien waarom het tot betere resultaten leidt dan andere, niet-Agile methoden.

Bij het introduceren van een nieuwe methodiek binnen een bedrijf zijn veel partijen betrokken. Ook voor Scrum geldt dat in hoge mate. Scrum heeft niet alleen invloed op de projectafdelingen, maar (juist) ook op andere afdelingen binnen de organisatie en zelfs daarbuiten. Scrum draait om communicatie en communicatie is niet iets dat alleen binnen de uitvoerende afdelingen plaatsvindt. Juist de communicatie tussen verschillende afdelingen onderling, communicatie tussen de opdrachtgever en het productieteam, communicatie tussen de projectmedewerkers en de eindgebruikers enzovoort, is van belang voor het slagen van het pro-

ject. Om Scrum succesvol te laten zijn, is het van belang dat alle partijen begrijpen wat Scrum inhoudt.

Voor wie is dit boek?

Tijdens de coachingstrajecten die ik doe, merk ik vaak dat bedrijven al een beetje zijn voorbereid op het Agile-gedachtegoed. Dit gebeurt meestal door een van de vele artikelen op internet of een boek over Scrum te lezen. Omdat Scrum weinig regels kent die ook nog eens eenvoudig zijn, proberen de bedrijven direct de methodiek uit. De eerste stappen gaan dan vaak heel soepel, maar na een tijdje ontstaan de praktijkvragen.

Helaas gaan veel boeken over Scrum als methodiek alleen. Daarin wordt heel netjes verteld hoe je Scrum moet doen. De betere boeken behandelen veel theorie en praktijkvoorbeelden over wat er goed of fout ging tijdens het introduceren van Scrum. Bij alle Agile-methodieken zit het venijn in de details. De grote lijnen zijn erg eenvoudig, maar vaak worden de details onderschat.

Ik zie bijvoorbeeld bedrijven die de dagelijkse Scrum-vergadering (*daily standup*) met een korrel zout nemen en de vergadering doen wanneer het toevallig uitkomt. Dit lijkt op het eerste gezicht niet zo'n probleem. De dagelijkse Scrum-vergadering is immers bedoeld voor het team om aan elkaar te vertellen wat de voortgang is (zoals we uitgebreid zullen zien in hoofdstuk 3 en 6). Deze vergadering heeft ook nog een aantal subdoelen die minder belangrijk lijken, maar in de praktijk net zo belangrijk zijn. De vergadering is openbaar en managers of andere geïnteresseerden (van alle lagen van de organisatie) worden van harte uitgenodigd de vergadering bij te wonen. Het bijwonen van de vergadering door een manager geeft signalen af van betrokkenheid en interesse. Dit is belangrijk voor het zelfvertrouwen van het Scrum-team. Door de dagelijkse Scrum-vergadering niet iedere dag of steeds op een ander tijdstip te houden, kunnen managers en andere geïnteresseerden deze niet eenvoudig in hun drukke schema inpassen, waardoor ze de kans missen om direct betrokken te raken bij de projecten in het bedrijf.

Dit voorbeeld geeft aan dat juist de details erg belangrijk zijn voor het succesvol introduceren van Scrum in een bedrijf. In dit boek richt ik me vooral op de praktijkvragen die mij regelmatig worden gesteld en waarop het antwoord lastig is te vinden op internet. Natuurlijk behandel ik de theorie van Scrum, omdat zonder theorie geen praktijk mogelijk is. Scrum laat een groot aantal details van de invulling aan de gebruikers van Scrum

zelf over. Daarom maken we in de praktijk regelmatig gebruik van technieken die zijn ‘geleend’ uit andere methodieken, maar die prima passen in de structuur die Scrum ons biedt. De onderdelen die ik in dit boek behandel zijn soms ‘verplichte’ Scrum-onderdelen, maar vaak ook handvatten om met bepaalde situaties om te gaan die niet door Scrum tot in detail worden ingevuld.

Dit boek is bedoeld voor iedereen die wil starten met Scrum of al is gestart en tegen dagelijkse praktijkproblemen aanloopt.

- **Projectmedewerkers:** mensen die in teamverband projecten uitvoeren en die willen overstappen op de Scrum-methodiek of die de methodiek al gebruiken en met praktijkvragen zitten.
- **Managers:** personen in een leidinggevende positie die willen leren hoe ze hun team(s) het best van dienst kunnen zijn, maar ook hoe ze de Scrum-methodiek het best kunnen ‘verkopen’ binnen hun organisatie.
- **Verkopers en accountmanagers:** personen die binnen de organisatie verantwoordelijk zijn voor de verkoop van projecten en die willen weten hoe je potentiële klanten ervan kunt overtuigen dat het gebruik van de Scrum-methodiek uiteindelijk ook beter voor de klant is.
- **Directie:** beslissingsnemers binnen een organisatie die willen weten wat Scrum inhoudt en hoe ze hun organisatie kunnen omvormen naar een Agile-organisatie.
- **Opdrachtgevers:** personen die projecten laten uitvoeren bij een bedrijf dat de Scrum-methodiek gebruikt en die willen weten wat de voordelen daarvan zijn vanuit het standpunt van de opdrachtgever.

De eerste hoofdstukken behandelen de Scrum-methodiek zelf in theorie en praktijk. Nu Scrum volwassen is geworden, maar nooit ophoudt te veranderen, willen ook grotere organisaties (met meer dan duizend werknemers) Scrum gaan gebruiken. Het boek zou niet compleet zijn als het opschalen naar meerdere Scrum-teams niet zou worden behandeld. Maar net zo belangrijk is het ‘verkopen’ van de Scrum-methodiek aan uw klanten of binnen uw eigen bedrijf. Als niet iedereen het nut van Scrum inziet, is het overstappen naar Scrum bij voorbaat gedoemd te mislukken. In deze vierde editie ga ik ook in op een aantal zogenoemde hybride methodieken: methodieken waarbij onderdelen uit verschillende Agile-methodieken worden samengevoegd.

Software- of andersoortige projecten?

Zoals gezegd heeft Scrum zijn basis in de softwarewereld. Het bleek dat juist bij softwareprojecten de complexiteit dusdanig groot was dat er veel dingen fout gingen. Het gevolg: uitloop van projecten en ernstige budgetoverschrijding. Sinds een jaar of vijftien wordt Scrum ook veelvuldig gebruikt buiten de software-industrie. Vooral complexere projecten kunnen baat hebben bij een methodiek als Scrum. Ik heb geprobeerd dit boek voor een zo groot mogelijk publiek te schrijven, maar mijn achtergrond ligt in de ICT en veel van de voorbeelden die ik in de praktijk tegenkom, zijn daaraan gerelateerd. De voorbeelden die ik in dit boek gebruik zijn opzettelijk niet al te technisch van aard en ik heb als uitgangspunt genomen dat iedereen met enige ervaring in de uitvoering van projecten het zou moeten kunnen begrijpen. In ieder geval zou u in staat moeten zijn de voorbeelden te vertalen naar uw eigen projecten.

Hoe kunt u dit boek gebruiken?

Het gebruik van dit boek hangt af van uw huidige ervaring met en kennis van Scrum. Hebt u in de praktijk nog niet veel ervaring met de Scrum-methodiek, dan raad ik u aan het boek van voor naar achter te lezen. De hoofdstukken volgen elkaar logisch op en latere hoofdstukken bouwen verder op eerder opgedane kennis.

Hebt u al enige ervaring met Scrum, dan kunt u de volgorde van lezen zelf bepalen en kunt u het boek meer als naslagwerk gebruiken.

Vooral hoofdstuk 8, *Opschalen naar meerdere Scrum-teams*, hoofdstuk 9, *Hoe verkoop ik Scrum binnen mijn bedrijf?*, hoofdstuk 10, *Hoe verkoop ik Scrum aan mijn klanten?* en hoofdstuk 11, *Wat te doen als Scrum niet werkt?* zijn hoofdstukken die uitstekend afzonderlijk gelezen kunnen worden.

Taalgebruik

Ik heb ervoor gekozen dit boek te schrijven in het Nederlands. Hoewel veel mensen tegenwoordig de Engelse taal goed machtig zijn, lezen ze toch liever in hun eigen taal. Het probleem dat optreedt is de vertaling van veelgebruikte Engelstalige woorden. Een woord vertalen dat iedereen dagelijks in het Engels gebruikt levert meestal meer onduidelijkheid dan duidelijkheid op. Daarnaast is het overgrote deel van de informatie op internet te vinden in het Engels.

Ik heb geprobeerd die woorden, waar een eenduidige Nederlandse vertaling voor is, te vertalen en daarbij tussen haakjes de originele Engelse term op te nemen. Hierdoor weet u welke term er wordt bedoeld mocht u de Engelse term al kennen en kunt u eenvoudiger additionele informatie op internet opzoeken.

Hij, of zij of het

In dit boek gebruik ik regelmatig de derde persoon om iemand in een team aan te duiden. Omdat het nogal omslachtig is overal hij of zij of onzijdig te gebruiken, heb ik het meestal over hij. In de ICT-wereld, waarin ik veel bedrijven coach, zijn mensen van andere genders helaas nog steeds ondervertegenwoordigd. Die mensen die ik heb meegemaakt zijn minstens net zo goed in hun vak als de mannen en overal waar ‘hij’ wordt gebruikt, kan ook ‘zij’ of ‘het’ worden gelezen.

WAT IS SCRUM?

Hoewel Scrum als projectbeheersmethodiek al grote bekendheid heeft, is bij veel mensen nog onduidelijk wat het precies inhoudt en hoe Scrum is ontstaan. In dit hoofdstuk ga ik in op de geschiedenis en de basisprincipes van Scrum (en andere Agile-methodieken). Hoewel Scrum in steeds meer verschillende branches wordt toegepast, is de softwareontwikkelbranche op dit moment nog wel de grootste. Dit boek richt zich grotendeels op Scrum in een omgeving waarin software wordt ontwikkeld. Dit is niet omdat Scrum op andere gebieden niet succesvol is, maar simpelweg omdat mijn ervaring (met Scrum) voornamelijk ligt op het vlak van softwareontwikkeling.

De term Scrum is geen afkorting, anagram of acroniem, maar is direct geleend uit de rugbysport. De scrum is het moment waarop de bal opnieuw in het spel wordt gebracht en waar alle teamleden elkaar omarmen en samen proberen de bal te bemachtigen. De symboliek van de scrum refereert aan het samenwerken in een team om zo efficiënt mogelijk een bepaald doel bereiken.

Definitie

De basis voor alle Scrum-regels ligt in de officiële Scrum-guide. Deze is geschreven door twee van de grondleggers van Scrum, Ken Swaber en Jeff Sutherland (Swaber, K. & Sutherland, J., 2020). In de, op het moment van schrijven, laatste versie van de Scrum-guide wordt de definitie van Scrum als volgt beschreven: “Scrum is een lichtgewicht raamwerk dat mensen, teams en organisaties helpt om waarde te creëren door middel

van adaptieve oplossingen voor complexe problemen.” Wat mij opvalt aan deze definitie zijn de woorden “waarde”, “adaptief” en “complex”. Het doel van Scrum is een project op dusdanige manier te beheren dat toegevoegde waarde wordt gecreëerd voor de betrokken belanghebbenden (*stakeholders*) van het project. Hoe deze waarde vorm krijgt is aan de betrokkenen zelf. Dit kan waarde zijn in de vorm van geld, een product maar ook naamsbekendheid. Het “complex” geeft aan dat Scrum vooral is ontwikkeld voor projecten met vaak veel verschillende belanghebbenden, veel eisen, lastige vraagstukken of in een snel veranderende wereld. Allemaal eigenschappen waar veel softwareprojecten aan voldoen. Het is dus ook niet vreemd dat Scrum juist in die branche is ontstaan. Het woord “adaptief” verradt al iets over de methode. Zoals we verder in dit boek zullen zien, is het kunnen inspelen op veranderingen tijdens het project een belangrijk onderdeel van de Scrum-methodiek.

Korte geschiedenis

Al sinds 1960 wordt er software ontwikkeld voor computers. In den beginne waren de softwareontwikkelaars voornamelijk techneuten (lees nerds) die precies wisten hoe de computers intern werkten. De projecten waren kleinschalig en de mensen die met de computers werkten waren zelf ook techneuten. Naarmate computers toegankelijker werden voor het grotere publiek, moest de software gebruiksvriendelijker worden. Sinds Xerox het op ‘windows’ georiënteerde systeem bedacht, dat later door Apple en Microsoft groot is gemaakt, is het bedieningsgemak van computers sterk verbeterd. In feite kan tegenwoordig iedereen een computer bedienen. De computers zelf daarentegen zijn alleen maar complexer geworden. Hoewel de interne werking in de afgelopen vijftig jaar niet rigoureuus is veranderd, zijn computers dusdanig snel geworden dat er veel verschillende programma’s tegelijk draaien op zelfs de eenvoudigste mobiele telefoon. Door de noodzaak om gebruiksvriendelijke software te schrijven voor steeds complexere computers, is het schrijven van software zelf ook enorm complex geworden. Tegelijkertijd werd de vraag naar nog complexere software steeds groter. In plaats van een eigen stukje software voor ieder bedrijfsproces, moesten alle bedrijfsprocessen worden geïntegreerd in één programma. Het liefst moest deze software ook nog samenwerken met de software van andere bedrijven om zo snel mogelijk informatie uit te wisselen. Wie ooit met SAP of een ander ERP-product heeft gewerkt, weet hoe complex dit kan worden.

Van oudsher werd software geschreven op de traditionele manier: we bedachten eerst volledig wat het programma moest gaan doen (functioneel ontwerp), daarna vertelden we dit tegen de techneuten. De techneuten maakten een abstract model (softwareontwerp) en pas nadat dat helemaal klaar was, gingen de programmeurs het ontwerp implementeren. Nadat alles was geïmplementeerd, werd het getest. Als alle tests op groen stonden leverden we het programma op aan de klant die het juichend ontving, waarna we een feestje gingen vieren.

Inmiddels weet iedereen wel dat dit scenario helaas meestal niet zo verloopt. Dit heeft een aantal oorzaken. Deze traditionele manier van denken stamt uit de tijd dat software werd geschreven door techneuten voor techneuten. Deze mensen spraken min of meer dezelfde taal en konden vrij goed overbrengen wat de wensen waren. De omstandigheden tijdens het project waren over het algemeen stabiel. De software werd meestal aan dezelfde personen opgeleverd die ook bij het bedenken van de software betrokken waren. Het testen van software was tijdrovend, omdat de resultaten van de tests vaak pas uren later bekend waren. Soms moest men zelfs dagen wachten totdat de ponskaarten met het testresultaat werden teruggebracht van het computercentrum. Het was toen dus heel belangrijk om alle mogelijkheden vooraf te bedenken, om niet na drie dagen wachten te ontdekken dat er een fout in regel één van de code zat.

Volgens deze manier van denken loopt het project van de ene fase naar de volgende als een soort waterval (vandaar de naam *Waternvalmethode*). Mocht er tijdens een fase iets worden ontdekt dat in een eerdere fase niet helemaal (of helemaal niet) goed was, dan moest in feite alles opnieuw worden gedaan. Hierdoor kwam het regelmatig voor dat na een periode van bijvoorbeeld een jaar nog steeds niets bruikbaar was geproduceerd.

Het vreemde is dat deze manier van watervaldenken in veel grotere projecten nog steeds heerst, ondanks dat de wereld enorm is veranderd. Informatievoorziening is tegenwoordig vele malen sneller, maar ook de wereld om ons heen verandert continu. Projecten die over langere tijd lopen zijn al tijdens het project onderhevig aan grote veranderingen. De gebruikte projectbeheersmethodiek moet dus ook bestand zijn tegen deze veranderingen.

De moderne “Agile” projectbeheersmethodieken, waaronder Scrum valt, proberen hier een oplossing voor te bieden.

Agile-principes

De term Agile betekent letterlijk ‘lenig’ in de betekenis van flexibel en buigzaam. De term op zich is weer een verwijzing naar de term *Lean manufacturing* of slanke productie. Lean manufacturing is een managementfilosofie die al uit het begin van de twintigste eeuw stamt. De filosofie is verder uitgewerkt door de Japanse autofabrikant Toyota, die daarmee het productieproces van alle onnodige zaken ontdeden. Het basisprincipe achter Lean is dat de verspilling in een proces wordt geminimaliseerd door het proces continu te evalueren en te verbeteren (*inspect and adapt*). Door de stroom en opslag van onderdelen in het productieproces te balanceren, kan het proces in zijn geheel worden geoptimaliseerd en worden daarmee niet alleen de kosten verlaagd, maar wordt ook de flexibiliteit verhoogd.

Al in het begin van de jaren negentig van de vorige eeuw werden de bestaande softwareontwikkelmethoden nog eens goed tegen het licht gehouden. Oorspronkelijk dacht men dat grotere projecten het best te vergelijken waren met het koken van een gerecht door een kok:

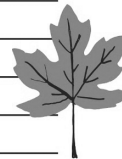
- Bedenk welk gerecht we willen hebben (functioneel ontwerp).
- Leg alle ingrediënten klaar (technisch ontwerp).
- Zet de pan op het vuur en doe alle ingrediënten in de juiste volgorde en op het juiste moment in de pan (ontwikkeling).

RECIPES

NAME: Spaghetti

INGREDIENTS: 200 gr spaghetti
1 rode paprika
zout, peper, olie

DIRECTIONS:
Kook de spaghetti 8 minuten
Snij de paprika in stukjes
breng op smaak met zout en peper
Opdieneren met een beetje olie



Afbeelding 2.1: Het recept voor eenvoudige spaghetti.

- Als het gerecht klaar is, moeten we proeven voordat we het opdienen voor onze klanten (testen).
- Als laatste dienen we het gerecht op (oplevering).

Als alle ingrediënten vooraf bekend zijn en het is een eenvoudig gerecht zoals de spaghetti in de afbeelding, is deze methode best mogelijk. Net als een kok, die iedere dag dezelfde spaghetti maakt, goed kan voorspellen hoe de spaghetti zal smaken en hoelang het duurt voordat het gerecht klaar is, kunnen projectmedewerkers redelijk goed inschatten hoelang een project zal duren (en dus wat de kosten zullen zijn) als zij eenzelfde soort project al meerdere malen hebben uitgevoerd.

In de praktijk blijkt het uitvoeren van complexere projecten (waaronder veel softwareprojecten) veel meer overeenkomsten te hebben met het bedenken van een nieuw soort gerecht door een chef-kok.

- De chef krijgt de opdracht van een Italiaans restaurant om een nieuw pastagerecht te bedenken voor hun menukaart.
- Hij bekijkt de huidige menukaart en bedenkt ongeveer wat voor soort gerecht hij wil maken.
- De chef gaat op zoek naar mogelijke ingrediënten voor het gerecht.
- Hij probeert eerst de basismaken samen te stellen.
- Hij proeft het gerecht.
- Na het proeven probeert hij te bedenken welke smaken nog ontbreken en zoekt hij de ingrediënten die die smaken kunnen toevoegen.
- Hij proeft opnieuw het gerecht.
- Hij voegt meer ingrediënten toe, maar bedenkt dat de vorige ingrediënten toch niet zo goed tot hun recht komen. De chef begint dus weer van voor af aan maar nu met andere ingrediënten (en met meer kennis over het eindproduct).
- Hij proeft opnieuw het gerecht.
- Tijdens het maken van het gerecht komt de chef op het idee voor een ander gerecht dat waarschijnlijk beter past tussen de andere gerechten. Hij begint dus weer opnieuw maar nu met opnieuw een beter idee van het eindproduct.
- Na een aantal keer proberen is de chef tevreden met het nieuwe gerecht en biedt hij het aan het restaurant aan.
- Het restaurant zet het op de menukaart en wacht op de reacties van de mensen die het bestellen.
- De chef verwerkt eventuele kritiek op het gerecht en past het aan zodat uiteindelijk de klant tevreden is.

RECIPES

NAME: *Nieuw gerecht*

INGREDIENTS: ~~200gr spaghetti~~ *200gr Farfalle*
1 rode paprika basilicum, champignons
zout, peper, ~~olie~~ 2 tomaten

DIRECTIONS:
Kook de farfalle beetgaar en de champignons
Snij de paprika in stukjes
bak de paprika en de champignons kort
Voeg de saus toe
leg de tomaten op het bord



Afbeelding 2.2: Recept voor een nieuw gerecht.

Dit proces is totaal anders dan het proces om een vooraf gedefinieerd gerecht te maken. De grootste verschillen zitten in de onzekerheid van het eindproduct. Hoewel de kok vooraf best een idee heeft van wat hij wil maken, kan hij niet precies zeggen wat het wordt en kan hij al helemaal niet zeggen hoe lang het zal duren voordat het gerecht is ontwikkeld. Het tweede verschil is de complexiteit. Als het recept vooraf bekend is, is het aantal mogelijkheden vrij beperkt. Alle ingrediënten zijn immers bekend. In het tweede voorbeeld zijn de ingrediënten vooraf niet bekend en moeten deze eerst worden gekozen. Het aantal combinaties van ingrediënten is zeer groot en wordt nog groter doordat twee ingrediënten die worden gebruikt gezamenlijk een andere smaak kunnen opleveren. Hoewel een ervaren kok weet welke ingrediënten het best samen kunnen worden gebruikt, kan het samenvoegen soms tot onverwachte resultaten leiden.

Het schrijven van (maatwerk)software heeft veel gemeen met het bedenken van een gerecht. Een ervaren programmeur kent veel basisoplossingen voor deelproblemen, bijvoorbeeld in de vorm van Design Patterns (Wikipedia, 2014), maar de integratie van een aantal basisoplossingen leidt soms tot onverwachte problemen. Het proeven van een recept is vergelijkbaar met het testen van software. Als we tijdens het proces vaak het product testen, weten we vroegtijdig of er iets mis gaat en kunnen

we bijsturen. Nog mooier is het als de opdrachtgever ook al bij de tests wordt betrokken, zodat die al vroegtijdig weet waar het eindresultaat naartoe gaat. Als de opdrachtgever tijdens het proces ontdekt dat zijn wensen toch niet helemaal juist worden geïnterpreteerd, kan hij nog voor het eind bijsturen. Van oudsher werd ervan uitgegaan dat projecten verliepen als in ons eerste receptvoorbeeld. Helaas was de praktijk anders. Uit onderzoek uit 2013 (Ambler, 2013) bleek dat maar vijftig procent van alle IT-projecten die werden uitgevoerd volgens dit concept, succesvol verloopt. 32 procent van deze projecten had problemen en achttien procent ging volledig fout. Hierbij moet de kanttekening worden gemaakt dat vooral de grotere projecten (met een budget boven 1 miljoen dollar) deze aantallen negatief beïnvloeden. Uit vervolgonderzoek in 2018 (Ambler, 2018) blijkt dat projecten over de gehele linie meer succesvol waren. Het gemiddelde aantal projecten dat als niet succesvol werd bestempeld was acht procent. In dit onderzoek is ook onderzocht wat de percentages waren van bedrijven die bij een of meerdere projecten een Agile-methodiek of Lean gebruikten. Dit bleek op respectievelijk 55 en 68 procent te liggen. Opmerkelijk is dat het percentage van mislukte projecten bij Agile met maar drie procent het minste was.

Agile Manifesto

In de 50 jaar dat we nu software ontwikkelen, zijn er allerlei processen en methoden ontwikkeld om een softwareproject in goede banen te leiden. Veel van de methodieken gingen uit van het concept dat, als je maar uitgebreid genoeg met de klant praat, je vooraf het hele project kunt beschrijven. Het was dus ook mogelijk vooraf redelijk te bepalen hoe de software moest worden gebouwd en hoelang het zou duren om die software te bouwen. Projecten konden dus best voor vaste prijzen (*fixed price projects*) worden verkocht, omdat de risico's niet zo groot waren. Pas rond 1990 werd men zich ervan bewust dat software ontwikkelen niet iets is dat volledig vooraf kan worden bepaald, maar dat het vaak een empirisch (proefondervindelijk) proces is. Er waren al een aantal pogingen gedaan om softwaremethodieken te ontwikkelen die beter omgingen met veranderingen tijdens het project (Scrum stamt uit 1993).

In februari 2001 kwamen zeventien ervaren mensen uit de IT-wereld bij elkaar in een skihut in de bergen van Utah (Amerika) om te praten over de problemen bij softwareprojecten. Zij wilden een alternatief bieden voor de bestaande zwaarlijvige softwareontwikkelmethodieken. Tijdens deze bijeenkomst werd het Agile “Software Development” Manifesto

(Beck, 2001) opgesteld. Dit manifest is een document waarin basisbeginselen worden genoemd die beschrijven wat de prioriteiten in een softwareproject zouden moeten zijn:

- personen en interacties boven processen en tools;
- werkende software boven lijvige documentatie;
- samenwerking met de klant boven onderhandeling over het contract;
- omgaan met verandering boven het volgen van een plan.

De personen die bij deze ontmoeting bij elkaar kwamen, konden niet vermoeden hoeveel invloed deze beginselen zouden hebben op de moderne manier van omgaan met softwareontwikkeling en sinds een aantal jaar ook op projectbeheer in het algemeen. Uit het manifest springen enkele dingen in het oog. De focus ligt op communicatie tussen alle betrokkenen (medewerkers maar ook opdrachtgevers), op het leveren van werkende software, maar ook op het menselijke aspect. Dit zijn nu precies de elementen waar veel projecten op stuklopen.

Hoewel het een open deur lijkt, is communicatie misschien wel het belangrijkste element van ieder softwareontwikkelingsproject en waarschijnlijk van ieder project in het algemeen. Nu zijn er veel communicatiemiddelen en methoden om de communicatie te verbeteren en ook de verschillende Agile-methodieken gebruiken ieder hun eigen manier van communiceren. Zoals we later in dit boek zullen zien, wordt bij Scrum veel aandacht besteed aan communicatie met de klant en communicatie in het team onderling.

Ook het leveren van werkende software lijkt nogal triviaal, maar het komt nogal eens voor dat een product wordt opgeleverd dat half werkt (of in ieder geval half is getest). Veruit de meeste programmeurs zullen de voorkeur geven aan het bouwen van software boven het schrijven van documentatie. Zij zullen het tweede punt van het manifest met open armen ontvangen. Het is echter wel zaak dat de tweede regel correct wordt geïnterpreteerd. Nogal eens wordt dit gelezen als: we leveren werkende software, geen documentatie. Documentatie is zeer belangrijk, mits deze documentatie functioneel is. Dikke documenten met specificaties die vaak na een week al zijn achterhaald, zijn geen functionele documenten. Ik heb deze documenten te vaak in een bureaula terecht zien komen zonder dat er ooit nog iemand naar keek. De tijd die is besteed aan het schrijven van het document had veel beter kunnen worden gebruikt voor het bouwen van werkende software. Natuurlijk moet er bij de software de nodige technische documentatie en functionele docu-

mentatie worden gemaakt. Het is tenslotte niet handig software op te leveren zonder gebruikshandleiding en ook het inwerken van een nieuw teamlid gaat een stuk eenvoudiger als er bijvoorbeeld technische schema's zijn.

Het derde punt uit het manifest wordt nogal eens uit het oog verloren. Vooral bij grote bedrijven hebben de juridische afdelingen hun handen vol aan het opstellen van volledig dichtgetimmerde contracten, die er bijna altijd van uitgaan dat het project mis zal gaan en verantwoordelijkheid wordt uitgesloten. In hoofdstuk 10 behandel ik een aantal contractvormen die meer uitgaan van samenwerking en het behalen van gezamenlijke doelen. Het vierde en laatste punt uit het manifest onderkent dat een planning een grof middel is om een project te beheersen. Bijna iedereen weet dat planningen al zijn achterhaald direct nadat ze zijn gemaakt. Waarom hechten we er dan toch zo veel waarde aan? Het is een schijnzekerheid waar we ons aan vast willen houden. Maar al te vaak wordt een planning gebruikt om medewerkers mee om de oren te slaan als het project niet verloopt volgens het plan. Het Agile Manifesto zegt dat we wel degelijk planningen maken, maar dat we veranderingen, die zeker gaan komen, moeten accepteren en zelfs omarmen. De mate waarin we binnen ons project om kunnen gaan met veranderingen geeft de mate van onze flexibiliteit aan. In de huidige snel veranderende wereld is deze flexibiliteit misschien de belangrijkste factor om als bedrijf te kunnen overleven.

Omdat bovenstaande vier basisbeginselen nogal breed kunnen worden geïnterpreteerd, is hieruit een aantal werkbare principes opgesteld. Deze principes vormen de handvatten voor alle Agile-projectmethodieken:

- 1 Onze hoogste prioriteit is het tevredenstellen van de klant door het vroegtijdig en voortdurend opleveren van waardevolle software.
- 2 Verwelkom veranderende behoeftes, zelfs laat in het ontwikkelproces.
- 3 Lever regelmatig werkende software op. Liefst iedere paar weken, hooguit iedere paar maanden.
- 4 Mensen uit de business en ontwikkelaars moeten dagelijks samenwerken gedurende het gehele project.
- 5 Bouw projecten rond gemotiveerde individuen. Geef hen de omgeving en ondersteuning die ze nodig hebben en vertrouw erop dat ze de klus klaren.
- 6 De meest efficiënte en effectieve manier om informatie te delen in en met een ontwikkelteam is door met elkaar te praten.

- 7 Werkende software is de belangrijkste maat voor voortgang.
- 8 Agile processen bevorderen constante ontwikkeling. De opdrachtgevers, ontwikkelaars en gebruikers moeten een constant tempo eeuwig kunnen volhouden.
- 9 Voortdurende aandacht voor een hoge technische kwaliteit en voor een goed ontwerp versterken agility.
- 10 Eenvoud, de kunst van het maximaliseren van het werk dat niet gedaan wordt, is essentieel.
- 11 De beste architecturen, eisen en ontwerpen komen voort uit zelfsturende teams.
- 12 Op vaste tijden, onderzoekt het team hoe het effectiever kan worden en past vervolgens zijn gedrag daarop aan.

Hoewel deze principes al dertig jaar geleden werden opgesteld, zijn ze na hooguit wat minimale tekstuele aanpassingen, nog steeds van toepassing. Sterker nog, het blijkt dat deze principes niet alleen voor softwareprojecten gelden, maar ook voor projecten in vele andere branches.

Klantbetrokkenheid

Een vaak gehoorde uitspraak is dat klanten lastig zijn en altijd over geld zeuren. In veel gevallen ontstaat dit beeld doordat de klant te weinig betrokken is bij het project. Bij de, in de volksmond genoemde ‘over-de-muurprojecten’, waarbij de klant na de functionele besprekingen lange tijd niets meer van de leverancier hoort, is het logisch dat de klant niet dagelijks bezig is met de voortgang van het project. Vaak heeft de klant het vertrouwen dat het bedrijf dat het product ontwikkelt het wel zal laten horen als er iets niet goed gaat. Nog los van het feit dat het terugkoppelen van ‘slecht nieuws’ naar de klant vaak als moeilijk wordt ervaren, is het bij veel bedrijven vaak intern niet eens bekend dat het project niet goed verloopt. Volgens het Agile-manifest is het dus erg belangrijk nauw met de klant samen te werken om samen tot een goed eindproduct te komen. Dit vergt ook van de klant een denkomslag. Aan klanten is namelijk lang voorgespiegeld dat de projectmedewerkers het heel goed af kunnen zonder al te veel klantcontact. Als we dan eenmaal een nauwe samenwerking met de klant kunnen opbouwen, merken we al snel dat zijn betrokkenheid vele malen groter wordt en als gevolg daarvan dat de klant veel minder gaat ‘zeuren’. Hij is tenslotte medeverantwoordelijk voor de voortgang van het project en hij ziet wekelijks (soms dagelijks) hoe hard het team werkt om zijn eindproduct te realiseren.