Voorwoord

Sinds een aantal jaar coach ik softwareontwikkelteams van bedrijven en help ik de teams met het verbeteren van hun productiviteit en de kwaliteit van de software. De coaching bestaat vaak uit training van programmeertechieken, maar vaak ook uit het helpen bij het beheren van projecten.

Als een team niet goed presteert, blijkt dat in veruit de meeste gevallen niet aan de kundigheid van het team te liggen, maar aan de manier waarop projecten worden beheerd en hoe de taken binnen het project worden verdeeld. Als een team nog niet bekend is met moderne Agile-ontwikkelmethoden, leg ik de principes uit. Meestal zijn alle betrokkenen snel enthousiast als we Scrum introduceren binnen de organisatie. Scrum heeft eenvoudige regels waarvan iedereen snel begrijpt dat ze de productiviteit en de kwaliteit kunnen verbeteren.

Iedere keer dat ik een bedrijf help met het implementeren van Scrum, worden mij dezelfde vragen gesteld. Over het algemeen gaan deze vragen niet direct over Scrum zelf, maar vooral over het invullen van details die niet specifiek door Scrum worden behandeld. 'Waarom schatten we user story's in storypoints en niet in uren?', 'Moeten bugs nu wel of niet op de productbacklog worden opgenomen?', 'Start de nieuwe Sprint direct na het einde van de vorige Sprint of is er nog tijd om opruimwerkzaamheden te doen?', 'Hoe kunnen we de meeste informatie uit een evaluatie halen?', 'Wat is de beste contractvorm met mijn klanten als we Scrum gebruiken?' enzovoort. Het blijkt dat deze vragen in bijna geen enkel boek over Scrum duidelijk worden beantwoord.

Daarmee was het idee voor dit boek geboren. Het duurde nog een paar jaar voordat ik de tijd vond om het boek ook daadwerkelijk te schrijven, maar na de inmiddels zo bekende bloed, zweet en tranen ligt het er dan eindelijk. Ik hoop dat uw vragen in dit boek zo veel mogelijk worden beantwoord en dat u net zo enthousiast over Scrum wordt als ik dat in de afgelopen jaren ben geworden.

Inhoud

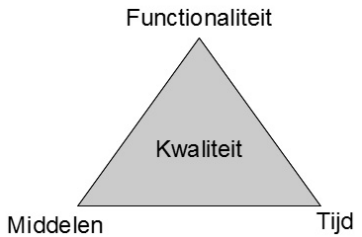
1

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 softwareontwikkeling door middel van de Scrum-methodiek. Het gebied waarvoor Scrum oorspronkelijk is ontwikkeld en waarin het 't meest wordt gebruikt.

De meeste mensen die iets met softwareontwikkeling te maken hebben gehad, hebben wel eens gehoord van de zogenoemde 'projectmanagementdriehoek' of 'IJzeren driehoek' (*Iron Triangle*). Deze driehoek geeft de relatie weer tussen de verschillende elementen van het projectbeheer: te ontwikkelen functionaliteit, 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. Deze randvoorwaarden zijn vaak tegenstrijdig. Als de beschikbare tijd voor het afronden van het project korter wordt gemaakt, heeft dat meestal tot gevolg dat de kosten (budget) zullen stijgen of dat de opgeleverde functionaliteit minder zal zijn. Als het beschikbare budget wordt verkleind, wordt daarmee automatisch de opgeleverde functionaliteit minder, maar kan het misschien wel eerder worden opgeleverd. De oppervlakte van de driehoek geeft de kwaliteit van het opgeleverde product weer.



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. Veelal bepaalt de klant (de opdrachtgever van het project) al twee van de drie punten. De klant wil dat het project tegen een vaste prijs wordt uitgevoerd, maar het moet ook over drie maanden klaar zijn. Het kost vaak veel moeite om uit te leggen dat daarmee de hoeveelheid functionaliteit die kan worden geïmplementeerd ook vast ligt. 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.

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

Scrum is een Agile-methodiek. Agile (lenig) is een groep van projectmethodieken die alle dezelfde uitgangspunten hanteren en waarvan de Scrummethodiek een van de oudste en bekendste is. Iedere Agile-methodiek, bijvoorbeeld eXtreme Programming (XP), Crystal Clear, Lean, Kanban enzovoort, focust op andere aspecten van het projectbeheer. 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. 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. U hoeft niet technisch onderlegd te zijn om Scrum te begrijpen en in te zien waarom het tot betere resultaten leidt dan ouderwetse ontwikkelmethoden.

Bij het introduceren van een nieuwe methodiek binnen een bedrijf zijn veel partijen betrokken. Voor Scrum geldt dat in hoge mate. Scrum heeft niet alleen invloed op de technische afdelingen, maar ook op andere afdelingen binnen de organisatie en zelfs daarbuiten. Scrum draait om communicatie en communicatie is niet iets wat alleen binnen de technische afdelingen plaatsvindt. Juist de communicatie tussen verschillende afdelingen onder-

ling, communicatie tussen de opdrachtgever en het ontwikkelteam, communicatie tussen de softwareontwikkelaars en de eindgebruikers enzovoort, is van belang voor het slagen van het project. 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 Agile. Dit gebeurt meestal door een van de vele artikelen op internet of een boek over Scrum te lezen. Omdat Scrum zeer 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 de meeste 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 die 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 vaak net zo belangrijk zijn. De vergadering is openbaar en managers (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 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 ant-

woord 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.

- Softwareontwikkelaars: mensen met een technische achtergrond die willen overstappen op de Scrum-methodiek of die de methodiek al gebruiken en met meerdere Scrum-teams willen gaan werken.
- Managers: personen in een leidinggevende positie die willen leren hoe ze hun team 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 softwareprojecten 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 (maatwerk)software afnemen 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 langzaam volwassen is geworden, 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 Scrum 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.

Hoe kunt u dit boek gebruiken?

Het gebruik van dit boek hangt af van uw huidige ervaring 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 meerder 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 mensen in de ICT-branch over het algemeen 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

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 te gebruiken, heb ik het meestal over hij. Er zijn helaas nog steeds niet veel vrouwen werkzaam in de ICT, maar de vrouwen die ik heb meegemaakt zijn minstens net zo goed in hun vak als de mannen en overal waar hij wordt gebruikt, kan ook zij worden gelezen.

2

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 of anagram, 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.

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 niet-techneuten moest de software gebruiksvriendelijker worden. Sinds Apple het op 'windows' georiënteerde systeem bedacht, dat later door 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 50 jaar niet rigoureus 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. Er zijn in de loop der jaren veel verschillende programmeertalen ontwikkeld. Elk met zijn eigen specifieke doelen en daarmee voor- en nadelen. Met de komst van de gestructureerde programmeertalen (de zogenoemde derdegeneratietalen) zoals C, C++, Java en later C# en voor het web PHP werd het voor de programmeurs eenvoudiger om complexe programma's te schrijven. 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.

Van oudsher werd software geschreven op de traditionele manier: we bedenken eerst wat we willen (functioneel ontwerp), daarna vertellen we tegen de techneuten wat we willen. De techneuten maken een abstract model (softwareontwerp) en als dat helemaal klaar is, gaan de programmeurs het ontwerp implementeren. Als alles is geïmplementeerd wordt het getest. Als alle tests op groen staan leveren we het programma op aan de klant die het juichend ontvangt, waarna we een feestje gaan 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.

Sinds die tijd is er veel veranderd. Behalve dat de software vele malen complexer is geworden, zijn de opdrachtgevers over het algemeen geen techneuten. Het vertalen van de wensen van de opdrachtgevers in de taal van de programmeurs is een vak apart geworden. Doordat de informatievoorzie-

ning tegenwoordig veel sneller gaat, veranderen de omstandigheden ook veel sneller. Een opdrachtgever kan op dag één heel goed weten wat de software zou moeten doen, maar omdat de concurrent met nieuwe functionaliteit komt, wordt het ineens veel belangrijker dat die functionaliteit ook wordt opgenomen in de software van de opdrachtgever. Traditioneel was dit altijd de nachtmerrie van iedere programmeur. 'Heb je net het hele softwareontwerp klaar, komt die vervelende klant weer met totaal andere eisen en wensen. Nu kan ik weer helemaal opnieuw beginnen.' Vanuit deze 'problemen' zijn creatieve mensen gaan zoeken naar oplossingen. Een van de oplossingen ligt in het gebruik van Agile-methodieken.

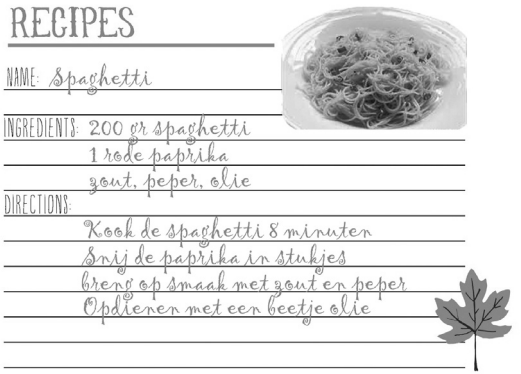
Agile-principes

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 manifacturing* of slanke productie. Lean manifacturing 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*).

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 softwareontwikkeling het best te vergelijken was 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).
- 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 programmeurs redelijk goed inschatten hoelang een softwareproject



Het recept voor eenvoudige spaghetti.

zal duren (en dus wat de kosten zullen zijn) als zij dezelfde soort functionaliteit al meerdere malen hebben gebouwd.

In de praktijk blijkt het schrijven van software 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 basissmaken 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.
- 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 een beter idee van het eindproduct.