Agile Scrum Handboek

Nader K. Rad & Frank Turley

Van Haren PUBLISHING

Agile Scrum Handboek

Andere uitgaven bij Van Haren Publishing

Van Haren Publishing (VHP) is gespecialiseerd in uitgaven over Best Practices, methodes en standaarden op het gebied van de volgende domeinen:

- IT en IT-management;
- Enterprise-architectuur;
- Projectmanagement, en
- Businessmanagement.

Deze uitgaven zijn beschikbaar in meerdere talen en maken deel uit van toonaangevende series, zoals *Best Practice, The Open Group series, Project management* en *PM series.*

Op de website van Van Haren Publishing is in de **Knowledge Base** een groot aanbod te vinden van whitepapers, templates, gratis e-books, docentenmateriaal etc. Ga naar www.vanharen.net.

Van Haren Publishing is tevens de uitgever voor toonaangevende instellingen en bedrijven, onder andere: Agile Consortium, ASL BiSL Foundation, CA, Centre Henri Tudor, Gaming Works, IACCM, IAOP, IPMA-NL, ITSqc, NAF, KNVI, PMI-NL, PON, The Open Group, The SOX Institute.

Onderwerpen per domein zijn:

IT and IT Management

ABC of ICT
ASL®
CATS CM®
CMMI®
COBIT®
e-CF
ISO/IEC 20000
ISO/IEC 27001/27002
ISPL
IT4IT®
IT-CMF™
IT Service CMM
ITIL®
MOF
MSF
SABSA
SAF
SIAM™
TRIM
VersiSM™

Enterprise Architecture

ArchiMate®
GEA®
Novius Architectuur Methode
TOGAF®

Business Management

BABOK® Guide
BiSL® and BiSL® Next
BRMBOK™
BTF
EFQM
eSCM
IACCM
ISA-95
ISO 9000/9001
OPBOK
SixSigma
SOX
SqEME®

Project Management

A4-Projectmanagement
DSDM/Atern
ICB / NCB
ISO 21500
MINCE®
M_o_R®
MSP®
P3O®
PMBOK® Guide
PRINCE2®

Voor een compleet overzicht van alle uitgaven, ga naar onze website: www.vanharen.net

Agile Scrum Handboek

Nader K. Rad

Frank Turley

Van Haren PUBLISHING

Colofon

Titel:	Agile Scrum Handboek
Auteurs:	Nader K. Rad & Frank Turley
Oorspronkelijke titel:	Agile Scrum Handbook
Oorspronkelijke uitgever:	Van Haren Publishing, 's-Hertogenbosch, 2018
Vertaling:	Theo Wanders, Maarssen
Tekstredactie:	Janneke Wolters
Uitgever:	Van Haren Publishing, 's-Hertogenbosch, www.vanharen.net
ISBN Hard copy:	978 94 018 0350 2
ISBN eBook:	978 94 018 0351 9
ISBN ePub:	978 94 018 0352 6
Druk:	Eerste druk, eerste oplage, november 2018
Lay-out en DTP:	Coco Premedia, Amersfoort – NL

Trademarks:
PRINCE2® is een registered trademark van AXELOS.
PRINCE2 Agile® is een registered trademark van AXELOS.
DSDM® is een registered trademark van Agile Business Consortium Limited.
EXIN Agile Scrum Master™ is een registered trademark van EXIN.

Inhoudsopgave

Over de auteurs

Nader K. Rad is auteur, spreker en adviseur over projectmanagement bij Management Plaza. Zijn carrière begon in 1997 en hij was betrokken bij veel projecten in verschillende industrieën. Hij heeft een groot aantal projectmanagementcursussen ontworpen, een aantal e-learningcursussen en hij heeft meer dan veertig boeken geschreven.

Nader is consultant en was een officiële recensent voor PRINCE2® 2017, PRINCE2 Agile®, P3.express en EXIN Agile Scrum Master™.

Meer over de auteur:	http://nader.pm
Website van de auteur:	https://mplaza.pm
LinkedIn profiel van de auteur:	be.linkedin.com/in/naderkrad

Frank Turley is al meer dan vijftien jaar projectmanager. Hij is een PRINCE2® Practitioner, een Scrum Master en een PRINCE2®- en projectmanagementtrainer en -coach. Hij heeft een aantal PRINCE2®- en projectmanagementgerelateerde boeken geschreven en is vooral bekend in de PRINCE2®-wereld vanwege zijn werk voor het creëren van het meest populairste PRINCE2®-zelfstudie trainingsmateriaal.

Meer over de auteur:	https://mplaza.pm/frank-turley/
Website van de auteur:	https://mplaza.pm
LinkedIn profiel van de auteur:	http://linkedin.com/in/frankturley

1. AGILITY (HET BEHENDIGHEIDS-CONCEPT)

Als je doel met het lezen van dit boek is om iets te leren dat je in je projecten ten goede kan komen, moet je eerst twee cruciale dingen weten die meestal verkeerd worden begrepen:

1. Wat je vaak hoort is: 'Agile is een mindset.' Het punt is dat Agile een mindset *nodig heeft*, net zoals al het andere. Maar het is onjuist om te zeggen dat het een mindset *is*. Zeggen dat 'Agile een mindset is' heeft slechts één praktisch gevolg: je bent erdoor in staat te werken zoals jij wilt. Noem het maar Agile zonder kritiek accepteren en zoeken naar verbeteringen.
2. Als je een beetje bekend bent met de manier waarop autoritaire systemen werken, weet je dat ze altijd een vijand nodig hebben. Dit vijandsbeeld vult de gaten in het systeem en helpt de menigte onder controle te houden. Veel Agile aanhangers gebruiken het woord 'Waterval' om naar de vijand te verwijzen, terwijl deze nooit duidelijk gedefinieerd is. Zij impliceren dat het gaat om de gevestigde projectmanagementsystemen. Als je doel is om succesvolle projecten te realiseren, heb je de illusie van een externe vijand niet nodig. Bovendien moet je onthouden dat elk succesvol systeem boven op bestaande systemen gebouwd is in plaats van dat het helemaal opnieuw begonnen is. Hoewel kritiek absoluut noodzakelijk is, moet die met respect en kennis gegeven worden.

Laten we beginnen over de echte aard van Agile.

1.1 PROJECTLEVERINGSMETHODE EN LEVENSCYCLUS

Wanneer je een stukje software ontwikkelt, worden de volgende stappen op een of andere manier uitgevoerd voor afzonderlijke functies, of voor de oplossing als geheel:

- Analyseren
- Ontwerpen
- Construeren/bouwen

- Integreren
- Testen

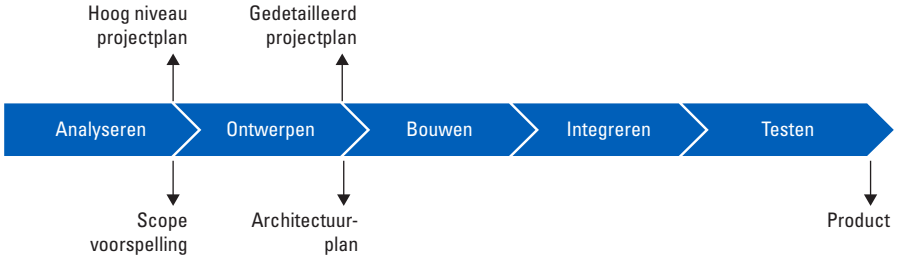
Je kunt natuurlijk andere namen voor die stappen gebruiken, ze samenvoegen, in minder stappen opsplitsen, of in meer; dat is prima. Deze stappen kunnen 'delivery (leverings)processen' genoemd worden.

Nu is de vraag: hoe ga je deze processen organiseren en uitvoeren? Denk aan een paar opties voordat je de rest van dit hoofdstuk leest.

En? Hoeveel opties heb je bedacht?

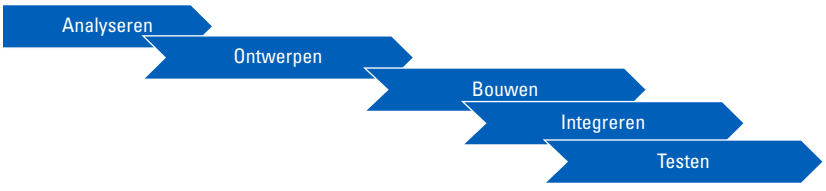
Je hebt misschien veel opties in gedachten, maar ze behoren allemaal tot een van de twee generieke vormen. Overigens zijn deze opties de 'ontwikkelingslevenscyclus' (development lifecycle).

Ons algemene levenscyclusmodel is iets als dit:



In dit levenscyclusmodel is elke processtap voltooid voordat we naar de volgende gaan; dat wil zeggen dat wij eerst alle eisen compleet analyseren en dan pas beslissen wat we als oplossing willen hebben; dan ontwerpen we de architectuur van de oplossing en zoeken we uit wat de beste manier is om alle eisen en kenmerken te realiseren en vorm te geven. Vervolgens gaan programmeurs aan de verschillende onderdelen werken en daarna worden de verschillende onderdelen geïntegreerd in één oplossing en wordt de oplossing getest.

Het is duidelijk dat de stappen elkaar kunnen overlappen. Je hoeft bijvoorbeeld niet te wachten tot alle oplossingsonderdelen compleet zijn voordat ze worden geïntegreerd en getest. De levenscyclus kan er als volgt uitzien:



Dit is geen fundamenteel verschil met het voorgaande model, we hebben nog steeds een volgorde van ontwikkelprocessen als de belangrijkste factor voor de levenscyclus.

Zoals je ziet, is dit type levenscyclus gebaseerd op de gedachte dat we willen begrijpen wat we moeten gaan produceren. We hebben een **upfront specificatie**, een **upfront ontwerp** en bijgevolg een passend plan. Daarom noemen sommigen het een **planningsgestuurde ontwikkeling**. We proberen te voorspellen wat we willen en hoe het geproduceerd kan worden, en daarom heeft het de naam 'voorspellend'. Voorspellende levenscycli zijn een normale en geschikte manier om veel projecten te ontwikkelen, zoals een bouwproject. Je plant en ontwerpt eerst en volgt vervolgens het plan. Dit is echter voor sommige soorten projecten geen prettige manier van werken.

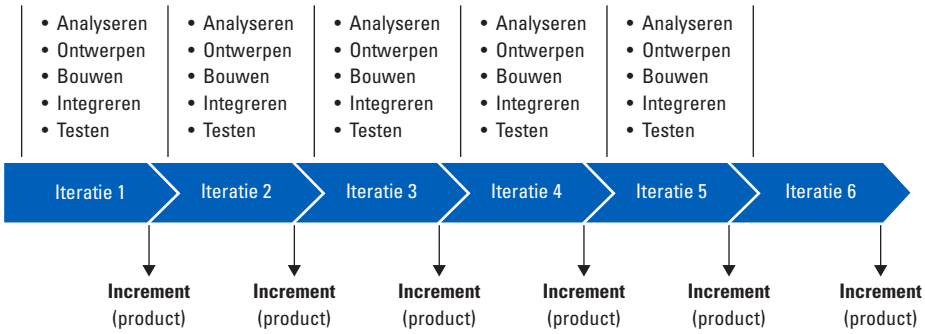
Denk aan een typisch IT-ontwikkelingsproject. Je kunt veel tijd besteden aan het opgeven van de eisen en het analyseren ervan, en vervolgens al het andere hierop baseren. Wat gebeurt er nu? De klant zal niet altijd blij zijn als hij het resultaat ziet! Hij zal vragen om wijzigingen. Wijzigingen zijn duur in deze levenscyclus omdat je misschien al het voorgaande werk moet herzien.

In deze branche is het gebruikelijk dat de klanten pas weten wat ze willen als ze het product zien. Wanneer zien ze het product? Tegen het einde van het project. Op dat moment zijn de kosten om het product te veranderen maximaal.

Om dit probleem op te lossen, kunnen we het comfort en de structuur van een voorspellende levenscyclus opofferen en er een gebruiken die het product incrementeel maakt en in delen oplevert, dat wil zeggen in meerdere versies, elke keer met meer functies. Dit is een luxe die we in IT-ontwikkelingsprojecten hebben en die niet elk ander project heeft: meerdere versies van werkende software, telkens met meer functies. Denk aan een bouwproject: daar zijn geen zinvolle incrementele opleveringen voor. Het product kan pas op het einde worden gebruikt.

Om eerlijk te zijn, dit nadeel van een bouwproject weegt op tegen het feit dat als je een project start om een ziekenhuis te bouwen, het niet veel uitmaakt hoeveel veranderingen er zijn. Het eindresultaat zal een ziekenhuis zijn, en niet bijvoorbeeld een pretpark! Echter, in IT-ontwikkelingsprojecten zou je een project kunnen starten om zoiets als een ziekenhuis te maken en eindigen met zoiets als een pretpark.

We kunnen dus een incrementele levering krijgen in IT-ontwikkelingsprojecten. Dit biedt de kans een levenscyclus te gebruiken als deze:



Er is geen echte voorspelling voor het eindresultaat in deze levenscyclus. In plaats van het eindproduct te voorspellen en daarop te vertrouwen hebben we korte perioden (**iteraties**) waarin we **incrementen** (delen) van het product maken. Wij laten de incrementen (de laatste versie van het product) aan de klant en de eindgebruikers zien, ontvangen hun feedback en beslissen wat te doen in de komende periode. Dus in plaats van te voorspellen, gaan we afhankelijk van de incrementen door met het project en gebruiken telkens de feedback. Hoe zou je deze levenscyclus willen noemen? 'Adaptief' is een geweldige naam: **adaptieve levenscyclus**.

Voor elke increment doorlopen we alle ontwikkelingsprocessen in de tijdsperiode die nodig is voor het maken van die increment. In de volgende periode herhalen we deze processen: we itereren. Dat is de reden waarom deze methode van ontwikkeling soms **iteratieve ontwikkeling** genoemd wordt. De tijdsperioden waarbinnen we itereren kunnen 'iteraties' worden genoemd. Dit is niet de enige naam die daarvoor wordt gebruikt; je weet mogelijk al minstens één andere naam voor iteraties. We komen verderop terug op dit onderwerp.

1.2 VOORSPELLENDE VERSUS ADAPTIEVE LEVENSCYCLUS

De voorspellende en adaptieve levenscycli hebben elk voor- en nadelen. De juiste keuze is afhankelijk van vele factoren, maar de belangrijkste is de aard van het product. Je kunt twee essentiële vragen stellen voordat je beslist welk type levenscyclus je nodig hebt voor je project:

1. Moet ik adaptief zijn? Want als dat niet nodig is, gaat het om een voorspellende levenscyclus! Die is eenvoudiger en gestructureerder. Een adaptief systeem is nodig als er een risico is dat je begint met het idee om zoiets als een ziekenhuis te bouwen en eindigt met zoiets als een pretpark.
2. Kan ik adaptief zijn? Deze vraag is nog belangrijker. Om adaptief te zijn, moet je de mogelijkheid hebben om iteratief te ontwikkelen en incrementeel op te leveren. Laten we nogmaals een bouwproject beschouwen: Kun je het gebouw iteratief ontwerpen? Kun je de constructie van een gebouw ontwerpen zonder de rest

van het gebouw te ontwerpen die bepaalt wat de belasting op de constructie is? Het antwoord is simpelweg *nee*! Het is niet mogelijk om bouwprojecten iteratief te ontwikkelen. Ook is incrementele oplevering niet mogelijk, zoals we eerder hebben besproken. We kunnen dus geen adaptieve levenscyclus gebruiken om een gebouw te bouwen (verwar dit niet met interieurontwerp en decoratie, of zelfs met renovatie, waarvoor we mogelijk wel een adaptief systeem kunnen gebruiken).

Mijn belangrijkste boodschap is dat voorspellend versus adaptief geen kwestie is van goed of kwaad. Als kleine oefening: denk aan een IT-project voor het upgraden van de besturingssystemen van driehonderd computers in een organisatie, of een IT-project voor de ontwikkeling van een netwerkinfrastructuur voor een zeer grote organisatie met kantoren op zes locaties. Welke ontwikkelingscyclus is naar jouw mening het geschiktst voor deze twee projecten?

1.3 AGILE VERSUS WATERVAL

'Agile' is de populaire naam voor systemen die de adaptieve levenscyclus gebruiken. Dat is hoe je Agile echt kunt definiëren, in plaats van te zeggen: 'Agile is een mindset'! Agile 'fans' gebruiken het woord Waterval om naar een voorspellende levenscyclus te verwijzen. Dan gebeurt dat vaak om te verwijzen naar een voorspellende levenscyclus in IT-projecten; je hoort mensen niet zeggen: 'Dit gebouw is gebouwd met behulp van een Watervalmethode.' Om er zeker van te zijn dat je alles weet als het gaat om terminologie, moet je je ervan bewust zijn dat het woord Waterval tegenwoordig praktisch een vloekwoord binnen de IT is, en je het recht hebt om je boos en beledigd te voelen als iemand je vertelt dat je Waterval gebruikt! Dat is waarom ik voorstel de meer formele naam in dit boek te gebruiken: 'voorspellende levenscyclus'.

1.4 IS AGILE NIEUW?

Agile wordt meestal geadverteerd als het nieuwe. Het gebruik van de term 'Agile' om te verwijzen naar de adaptieve levenscyclus is zeker nieuw, maar hoe zit het met de levenscyclus zelf?

Ik weet niets over jou, maar ik kan me moeilijk voorstellen dat veel projecten en programma's in de menselijke geschiedenis niet met enige vorm van een adaptieve levenscyclus zijn uitgevoerd. Kun je een voorbeeld bedenken?

Ik kan je er een geven. Denk aan een zeer populair initiatief (een project of programma) van vroeger (althans in het westen): oorlog voeren. Kun je een oorlog

met een voorspellende aanpak managen? Waarbij alles in het begin gepland en ontworpen wordt? Zeker niet. Je hebt misschien een plan op hoog niveau dat meer lijkt op een strategie dan op een plan, maar waarschijnlijk manage je de oorlog op één slagveld (een iteratie) tegelijkertijd (of op een paar tegelijk) en pas je op basis van het resultaat van elk gevecht de rest van het initiatief aan.

Geen prettig voorbeeld, maar een duidelijk voorbeeld waaruit blijkt dat adaptieve levenscycli niet nieuw zijn.

Wat is er dan nieuw?

Op een bepaald moment in de historie werden de zogenaamde 'wetenschappelijke managementbenadering' en het 'taylorisme' de norm, zozeer zelfs dat elke andere benadering als inferieur en zelfs verkeerd werd ervaren. Taylorisme was volledig en sterk gebaseerd op voorspellende systemen; daarom domineerden voorspellende systemen de hele wereld, om zo te zeggen.

Toen bereikten we de tijd dat steeds meer IT-ontwikkelingsprojecten werden geïnitieerd, en voorspellende levenscycli waren niet echt de beste manier om deze projecten te managen. Mensen probeerden het te tolereren, terwijl de druk toenam, tot er demonstraties en uiteindelijk revolutie van kwam! Zoals elke andere revolutie, verslindt deze zijn kinderen, maar dat is een onderwerp voor een andere keer.

1.5 HET AGILE MANIFESTO

Sommige mensen begonnen adaptieve systemen te gebruiken voor IT-ontwikkeling en geleidelijk aan structureerden ze deze in herhaalbare managementprocessen. Een groep van deze pioniers kwam in 2001 samen om het nieuwe systeem officieel te maken door het een naam te geven en daarvoor een manifesto op te stellen.

Laten we beginnen met de naam. Zoals de legende zegt, waren de twee laatste opties daarvoor 'Agile' en 'Adaptief'. Helaas was hun beslissing Agile. Adaptief zou veel beter zijn omdat dit de aard van de aanpak beschrijft en dat voorkomt veel misverstanden.

Dus het Agile Manifesto, dat beschikbaar is op de zeer geavanceerde en moderne website van AgileManifesto.org, is dit:

Wij laten zien dat er betere manieren zijn om software te ontwikkelen door het te doen en door anderen ermee te helpen. Daarmee komen we tot de volgende waardestatements:

mensen en hun onderlinge interactie	boven	processen en tools
werkende software	boven	allesomvattende documentatie
samenwerking met de klant	boven	contractonderhandelingen
inspelen op verandering	boven	het volgen van een plan

Dat wil zeggen dat hoewel de items aan de rechterkant waardevol zijn, wij toch aan de items aan de linkerkant meer waarde hechten.

Kent Beck	Ward Cunningham	Andrew Hunt	Robert C. Martin	Dave Thomas
Mike Beedle	Martin Fowler	Ron Jeffries	Steve Mellor	
Arie van Bennekum	James Grenning	Jon Kern	Ken Schwaber	
Alistair Cockburn	Jim Highsmith	Brian Marick	Jeff Sutherland	

Helaas is dit manifesto zelf nooit onderworpen geweest aan aanpassing tijdens zijn leven.

Een meestal over het hoofd gezien onderdeel van het manifesto is de laatste zin. Ik wil je graag uitnodigen om het manifesto opnieuw te lezen met de laatste zin in gedachten. Dus laten we deze vier **waardestatements (value statements)** nog eens bekijken.

Waardestatement 1: Mensen en hun onderlinge interactie boven processen en tools

Het over het hoofd zien van het belang van individuen en interacties is een snelle manier om te falen. Het zijn tenslotte de mensen die het project uitvoeren. Sommige managers denken dat ze problemen op dit gebied kunnen overwinnen door een meer geavanceerd 'systeem' te gebruiken, maar dat werkt zelden of nooit.

Velen van ons zijn teleurgesteld in het naïeve optimisme dat het implementeren van een geavanceerde tool problemen oplost die worden veroorzaakt door het over het hoofd zien van menselijke aspecten of zelfs methoden. Toch geven managers enorme sommen geld uit om tools te implementeren en te onderhouden, in de hoop dat ze toch hun magische werk doen. Het feit is dat tools alleen een systeem faciliteren; ze vervangen niet de behoefte aan een systeem zelf. Een

positief aspect is wel dat deze tools geavanceerde stukjes software zijn die jarenlange ontwikkeling en onderhoud nodig hadden, en veel projecten en banen creëerden, en die het ons mogelijk gemaakt hebben om te investeren in het denken over betere manieren om IT-ontwikkelingsprojecten te doen!

Het gedeelte over processen in dit statement is een beetje lastig. Het gaat eigenlijk over een bepaald type proces, niet om processen in het algemeen. Het gaat over processen die zijn ontworpen om de behoefte aan menselijke interacties en complexiteiten te vervangen. Ik ken persoonlijk managers die geloven dat als ze een beter proces hebben, zij geen hooggekwalificeerde professionals hoeven in te huren. Een belangrijk aspect van Agile systemen is dat ze *alle* menselijke aspecten ingebed hebben in hun processen, in plaats van ze gewoon in te schieten of alleen maar te praten over het belang van menselijke aspecten, wat helaas het geval is met gevestigde projectmanagementsystemen.

Kortom: processen die menselijke aspecten negeren of vervangen, zijn slecht, en processen die deze aspecten aanpakken en ze onderdeel maken van het systeem zijn goed.

Waardestatement 2: Werkende software boven allesomvattende documentatie

In tegenstelling tot het vorige waardestatement, dat correct is voor alle soorten projecten, is deze specifiek voor adaptieve systemen. Het verwijst naar het feit dat we niet vooraf documentatie moeten opstellen om te voorspellen wat er moet gebeuren in een project, maar dat we gewoon aan het werk moeten gaan, stukjes werkende software moeten creëren (incrementen) en deze geschikt maken voor gebruik.

Waardestatement 3: Samenwerking met de klant boven contractonderhandelingen

Elk project zou succesvoller zijn als er beter met de klant wordt samengewerkt. In adaptieve systemen is het zelfs meer dan belangrijk; het is noodzakelijk. De klant moet constant met je samenwerken voor het specificeren van nieuwe eisen en het controleren van en feedback geven op de incrementen die gemaakt zijn. Als de klant dat niet doet, kun je het product niet aanpassen.

Contractonderhandelingen voeren is iets waar we allemaal van houden. Een ideaal Agile project heeft alleen een tijd- en materiaalcontract en een klant die niet denkt dat leveranciers criminelen zijn. Dan zijn er niet veel contractonderhandelingen nodig omdat de twee partijen samenwerken aan het creëren van een waardevol product. Echter, een ideaal is maar een ideaal, en er zijn klanten

die nog steeds op zoek zijn naar fixed-scope en fixed-price contracten die in fundamentele tegenspraak staan met adaptieve methoden. Helaas is dit een bron van nooit eindigende contractonderhandelingen, vergelijkbaar met die in voorspellende projecten.

Waardestatement 4: Inspelen op veranderingen boven het volgen van een plan

Dit statement is vergelijkbaar met het tweede statement, en is specifiek voor adaptieve systemen. In plaats van een voorspelbaar, vooruitkijkend plan te hebben dat ons de weg wijst, zijn we afhankelijk van aanpassing. Dit laatste wordt in Agile meestal 'change' genoemd, waarschijnlijk omdat het klanten blij maakt te weten dat ze vrij zijn om te kunnen veranderen. In feite is het geen verandering, omdat we in adaptieve systemen geen initieel gebaselined plan hebben waar de 'verandering' van zou kunnen afwijken. Technisch gezien hebben we een continue stroom van nieuwe ideeën. Echter, laat het hun 'veranderingen' noemen voor de klanten.

- De kans is groot dat je tijdens het examen vragen zult krijgen over het Agile Manifesto. Het is geen slecht idee om het meerdere keren te bekijken en zelfs alle vier de statements te onthouden.

1.6 AGILE PRINCIPES

Het Agile Manifesto is aangenaam kort. De auteurs (van het Agile Manifesto) dachten echter dat het een goede zaak zou zijn om dit naar aanleiding van het nieuwe Agile idee verder uit te werken, dus creëerden ze de volgende twaalf principes:

Principe 1: Onze hoogste prioriteit is het tevreden stellen van de klant door het vroegtijdig en voortdurend opleveren van waardevolle software.

We doen tenslotte zaken en we hebben gelukkige klanten nodig. Dat is voor de hand liggend. Tegenwoordig geven we er de voorkeur aan om te zeggen dat de tevredenheid van de eindgebruikers de ultieme maat is, omdat dat profijt oplevert voor de klant, en vroeg of laat zal dat de klant op een duurzame manier tevredenstellen. Te idealistisch? Hoe kunnen we hen tevredenstellen? Dat is door software te maken die potentieel waarde genereert (bijvoorbeeld geld). Wanneer we vroeg en continu leveren zal er eerder waarde gegenereerd worden, en hebben we ook de mogelijkheid om aanpassingen te doen en iets te creëren wat de markt echt wil hebben en waarvoor men zal betalen, en niet iets wat we *verwachten* dat ze nodig hebben.