Het EXIN handboek voor Scrum Masters en Product Owners

Johann Botha

Titel: Het EXIN handboek voor Scrum Masters en Product Owners

Auteur: Johann Botha

Uitgever: EXIN Holding BV

ISBN: 9789076531151

Editie: September 2024

Inhoudsopgave

Lijst van figuren

Inleiding

Dit boek is de officiële literatuur voor de EXIN Agile Scrum Master en EXIN Agile Scrum Product Owner of Product Owner Bridge certificeringen.

Dit boek is geschreven met het expliciete doel om kandidaten te helpen zich voor te bereiden op de EXIN Agile Scrum Master, EXIN Agile Scrum Product Owner en/of EXIN Agile Scrum Product Owner Bridge certificeringen.

Aangezien deze certificeringen ***Advanced Level*** certificeringen zijn, gaan we ervan uit dat als u dit boek leest u bekend bent met de basisbegrippen van agile en Scrum als een Agile-methode.

Bent u dat niet, dan raden we aan ook het ***Agile Scrum Handboek*** van Nader Rad en Frank Turley, de ***Scrum Guide*** en de ***Nexus Guide*** te lezen.

Hoewel EXIN Agile Scrum Foundation geen voorvereiste is voor de certificeringen die binnen het bereik van deze publicatie vallen, raden we toch aan om die certificering te doen, tenzij u al goed vertrouwd bent met de basisbegrippen van Scrum.

Er zijn meer EXIN Agile-certificeringen. Kijk op de website van EXIN voor de meest actuele informatie: https://www.exin.com/

Voorwoord

Mijn betrokkenheid bij agile gaat ver terug – ik gebruikte zo'n twee decennia geleden een andere Agile-methode, Extreme Programming (XP), om software-ontwikkelingsprojecten te beheren. Toen ik aan het hoofd stond van een internet-startup voor een groot IT-conglomeraat, gebruikten we XP met een stel eigenzinnige programmeurs. Sindsdien heb ik in vele projecten en in consulting- en coaching opdrachten Lean, Scrum, ABC/DSDM, Kanban en DevOps methodes en concepten gebruikt.

Het natuurlijke gevolg van het gebruik van adaptieve en empirische methoden (zoals Agile Scrum) is dat je voorkeuren en vooroordelen ontwikkelt op basis van je eigen ervaringen. Ik zou willen stellen dat, aangezien agile vereist dat je verder kunt denken en handelen dan vanuit één methode, het prima is om te putten uit ervaring en de daaruit voortvloeiende voorkeuren. Ik heb echter geprobeerd om mijn persoonlijke vooroordelen en voorkeuren niet te laten doorklinken in dit boek en hopelijk ben ik daarin geslaagd. Dit boek gaat primair over Scrum en ik heb geprobeerd me zo goed mogelijk aan dit onderwerp te houden.

Dat gezegd hebbende, Scrum maakt vaak gebruik van methoden en technieken die niet van Scrum zijn. Twee simpele voorbeelden: Kanban wordt gebruikt als middel om iteraties visueel te beheren en flow te bereiken en MoSCoW van ABC/DSDM wordt gebruikt als een mogelijk middel om de product backlog te ordenen.

In deze geüpdatete derde editie zijn enkele anti-patronen verwijderd die onderdeel waren van de oude tekst. Allereerst is het probleem met schalen aangepakt en Nexus is nu de enige manier van schalen die wordt behandeld in dit boek.

Ik hoop dat u plezier heeft van dit boek en dat het praktisch genoeg is om het niet alleen te gebruiken voor de voorbereiding op het EXIN-examen, maar ook als een naslagwerk in uw taken als Scrum Master, Product Owner of Agile Coach.

Hoogachtend,

Johann Botha. Augustus 2024 – Johannesburg

1 Zet de toon voor Agile

1.1 Agile denken

Grote 'A' Agile (zelfstandig naamwoord): betrekking hebbend op of de aanduiding van een methode van projectmanagement, vaak gebruikt voor softwareontwikkeling, die wordt gekenmerkt door de verdeling van taken in korte werkfasen, frequente herbeoordeling en aanpassing van plannen.

Kleine 'a' agile (bijvoeglijk naamwoord): in staat om snel en gemakkelijk te bewegen; in staat om snel te denken, te begrijpen en te reageren.

De eenvoud van deze definities, die te vinden zijn op de website van PM-Partners[1] in Australië, zetten precies de toon.

1.1.1 Agile als een ‘project’-aanpak (Agile met een grote ‘A’)

Agile projectmanagementbenaderingen zijn ideaal voor empirische projecten zoals softwareontwerp en -ontwikkeling en dat geldt voor alle projecten in een complex systeem. Het aantal variabelen en onbekende factoren in deze projecten maakt een traditionele Waterval-aanpak onpraktisch.

Agile projectmanagementbenaderingen leggen de nadruk op het begrip *ontdekking* (discovery) dat voortkomt uit experimenteren en een aanvaardbaar middel lijkt om een project dat blijft evolueren op te leveren.

Het begrip ‘agile’ werd voor het eerst gedefinieerd op 11 februari 2001 in het Snowbird skiresort in de bergen van Utah. Deze baanbrekende bijeenkomst resulteerde in het Agile Software Development Manifesto, wat nu bekend staat als het Agile Manifest.

Op deze historische bijeenkomst waren vertegenwoordigers aanwezig van veel van de toen populaire Agile-methoden, die nu nog steeds in gebruik zijn. Alle deelnemers waren ervan overtuigd dat het nodig was om een alternatief te bedenken voor de documentgedreven, gecompliceerde softwareontwikkelingsprocessen die toen gebruikelijk waren.

De redenen voor het ontwikkelen van het Agile Manifest door deze visionairs waren onder andere de frustratie over lange leveringscycli, veel planning die waardeloos bleek en de voortdurend veranderende vereisten, omdat de wereld om hen heen veranderde tijdens een project. Deze frustraties leidden tot de creatie van het Manifest en de twaalf bijbehorende principes, die uitleggen hoe Agile-organisaties helpt om meer waarde, sneller en uitzonderlijke kwaliteit te leveren.

[1] https://www.pm-partners.com.au/gaining-true-business-agility-with-agile/

Men realiseerde zich al snel dat wat goed werkt voor softwareontwikkelingsprojecten, ook goed zou kunnen werken voor andere projecten waar de eisen onduidelijk zijn of zich voortdurend ontwikkelen.

Een niveau van onzekerheid in projecten is normaal. Vrijwel alle projecten hebben, in meerdere of mindere mate, te maken met ambiguïteit, veranderende eisen en leren van fouten naarmate het project vordert.

Agile en Agile-methoden zijn daarom de enige manier om om te gaan met een omgeving waarin de toekomst onduidelijk is, de deliverables (vereiste resultaten) niet duidelijk kunnen worden gedefinieerd aan het begin van een project en waar gaandeweg wordt geleerd.

Waarom? Omdat een empirische benadering de verstandigste aanpak is wanneer we te maken hebben met onzekerheid, ambiguïteit en de voortdurende verandering die zich voordoet in complexe adaptieve systemen (CAS), zoals organisaties en projecten.

Empirisch – gebaseerd op ervaring of proefnemingen (Van Dale).

Voorbeeld: Zij laten zich leiden door praktische ervaring, niet door theorie.

Wat het gebruik van conventioneel (zogenoemd Waterval) projectmanagement zo beperkt, is de veronderstelling dat er een gedetailleerde planning aan de uitvoering van een project kan en moet voorafgaan.

Er lijkt een algemene misvatting te bestaan dat Agile-methoden niet vooruitplannen, terwijl het tegenovergestelde waar is. Het is lastig om ook maar één Agile-project te vinden waar geen tijd is besteed aan plannen.

Het verschil is dat er bij een Agile-aanpak gepland wordt op basis van wat al bekend is. Er wordt vastgelegd wat nog onbekend is en de mensen die opleveren, hebben de taak om de vragen te beantwoorden terwijl ze bezig zijn met opleveren. Normaal gesproken, hebben deze mensen veel geleerd van eerdere fases van het project en begrijpen ze de context waarin de software moet werken. Meestal ontstaat een beter begrip van de context naarmate het project vordert en zich ontvouwt.

Iedereen die ervan uitgaat dat zij aan het begin van een project een perfect, gedetailleerd plan kunnen maken, zou gewezen moeten worden op het volgende: ‘Niemand is slim genoeg of heeft zoveel geluk dat ze alles kunnen plannen en voorspellen, al helemaal niet van tevoren’.

Het volgende dat in herinnering gebracht moet worden is dat *projecten veel meer te maken hebben met verwachtingen van klanten dan met specificaties*. Hoe goed er ook vooraf wordt geprobeerd om specificaties te definiëren, het is bijna nooit mogelijk de verwachtingen van klanten en gebruikers vast te leggen en te kwantificeren.

Soms moet de klant, de gebruiker of de consument iets tastbaars zien voordat hij of zij het kan accepteren of afwijzen, omdat het ofwel aan de verwachtingen voldoet of niet.

De woorden van Henry Ford illustreren dit punt: “Als ik mijn klanten had gevraagd wat ze wilden, zouden ze hebben gezegd: ‘een sneller paard’.”

Er wordt vaak gezegd dat de output (dat wat we leveren) er voor klanten niet toe doet; het is het resultaat of de uitkomst (dat wat klanten kunnen doen of bereiken met wat we leveren) dat ertoe doet. Dit betekent dat klanten tevreden zijn wanneer ‘wat’ het project oplevert (de gespecificeerde output), hen in staat stelt te bereiken wat zij willen of nodig hebben (het resultaat).

Waarde wordt dus altijd gecreëerd door de leverancier en de klant samen – iets dat het onthouden waard is.

Overweeg daarom de volgende vraag:

Hoe vaak was u betrokken bij een project waarvan de oplevering heel dicht bij de met de klant overeengekomen specificaties lag, maar de klant toch ontevreden was?

Het antwoord op deze vraag is waarschijnlijk vaak, zo niet meestal: “Voordat we dingen op een Agile-manier gingen doen.”

Er zijn twee redenen voor hogere klanttevredenheid wanneer Agile wordt gebruikt:

- Agile betrekt gebruikers en klanten door hen deel uit te laten maken van het project, wat hun de gelegenheid geeft om in een vroeg stadium frequente feedback te geven.
- Agile geeft de mogelijkheid om te leveren tegen een verwachting van twee weken, in plaats van een verwachting van twee jaar.

De wereld verandert en dat doen de behoeften van klanten ook.

1.1.2 agile als een manier van denken en handelen (agile met een kleine 'a')

Een groot deel van de aanvankelijke discussie in Snowbird ging over een andere manier van werken en hoe technieken uit TQM (de Plan Do Check Act (PDCA)-cyclus) en Lean (ofwel het Toyota-productiesysteem) van toepassing zijn op softwareontwikkeling.

Het is dan ook geen wonder dat als we kijken naar de twaalf principes van Agile, er een opvallende gelijkenis is met de principes die we vinden in Lean.

De bedoeling was niet om één manier van softwareontwikkeling te ontwikkelen. In plaats daarvan lag de nadruk op hoe projectteams bepaalde zaken zouden moeten waarderen (**dit** BOVEN *dat*) en dit te vertalen naar een set van principes om toe te passen op softwareontwikkeling.

Het is normaal dat Agile-methoden werden ontwikkeld, omdat de uitvoerders van principe naar praktijk moesten gaan en alle Agile-methoden van nu gebruiken de twaalf principes als leidraad of moreel kompas.

Nu is misschien een goed moment voor het aanhalen van de vier waarden van Agile-teams en de twaalf gedefinieerde principes.

Waarde 1.	**Individuen en interacties**	BOVEN	*processen en hulpmiddelen*
Waarde 2.	**Werkende software**	BOVEN	*uitgebreide documentatie*
Waarde 3.	**Samenwerking met de klant**	BOVEN	*contractonderhandelingen*
Waarde 4.	**Inspelen op verandering**	BOVEN	*het volgen van een plan*

Alleen gekeken naar de waarden, is het duidelijk dat het belangrijker is dat klanten waarde krijgen dan alle andere dingen die in projecten gedaan worden. Merk op dat deze uitspraken niet zeggen dat processen, tools, documentatie, contracten en plannen niet belangrijk zijn. De waarden stellen alleen dat het belangrijker is om te praten en werken met mensen, hun iets te geven dat voldoet aan wat zij nodig hebben, hen te betrekken bij het proces van waarde leveren en te weten wanneer hun omstandigheden en behoeften zijn veranderd.

De Agile-principes bouwen voort op de waarden en maken ze uitvoerbaar. Dit zijn:

Principe 1. **Onze hoogste prioriteit is het tevredenstellen van de klant door het vroegtijdig en voortdurend opleveren van waardevolle software**; omdat klanten gelukkiger zijn als ze iets ontvangen waarmee ze hun resultaten sneller en vaker kunnen bereiken, in plaats van lang te wachten op de volgende release.

Principe 2. **Verwelkom veranderende behoeftes, zelfs laat in het ontwikkelproces.** Agile-processen benutten verandering tot het concurrentievoordeel van de klant. Omdat de omgeving en het landschap van de klant snel veranderen, veranderen ook hun behoeften. Dit is geen wijziging van de scope, het is de realiteit. Wie er niet mee om gaat, faalt. Laten we niet vergeten dat voor elke nieuwe vereiste er hoogstwaarschijnlijk een oude vereiste is die niet langer geldt.

Principe 3. **Lever regelmatig werkende software op.** Liefst iedere paar weken, hooguit iedere paar maanden. Dit raakt aan principe 1, maar is niet precies hetzelfde. Hier wordt gezegd dat er niet alleen sneller opgeleverd zou moeten worden wat klanten kunnen gebruiken, maar ook in een voorspelbaarder tempo of met regelmaat (zie ook principe 8).

Principe 4. **Mensen uit de business en ontwikkelaars moeten dagelijks samenwerken gedurende het gehele project.** Betrek belanghebbenden vroeg en vaak. Als het mogelijk is, probeer ze dan zelfs deel te laten uitmaken van elke opleveringsiteratie en niet alleen van de sprint review. Gebruikers van software kunnen vaak deel uitmaken van Scrum teams. Wie kent de vereisten tenslotte beter dan de persoon die het werk doet?

Principe 5. **Bouw projecten rond gemotiveerde individuen.** Geef hun de omgeving en ondersteuning die ze nodig hebben en vertrouw erop dat ze de klus klaren. Dit betekent dat als Agile gebruikt wordt zonder een cultuurverandering in de organisatie, het hoogstwaarschijnlijk zal falen.

Principe 6. **De meest efficiënte en effectieve manier om informatie te delen in en met een ontwikkelteam is door met elkaar te praten.** Vergeet niet dat 80% van communicatie non-verbaal is! We leren om te gaan met een minder fysiek verbonden wereld, maar in een ontwikkelteam is face-to-face interactie cruciaal. Let wel: in een post-Covid-19 wereld is werken in virtuele teams de nieuwe realiteit. Het is echter nog steeds aan te raden om face-to-face communicatie te gebruiken wanneer dat mogelijk is en video te gebruiken bij virtuele teamvergaderingen.

Principe 7. **Werkende software is de belangrijkste maat voor voortgang.** Kan de software het werk doen? Kan het worden gebruikt? En levert het de beoogde resultaten op? Zo niet, dan is er gefaald.

Principe 8. **Agile-processen bevorderen constante ontwikkeling.** De opdrachtgevers, ontwikkelaars en gebruikers moeten een constant tempo eeuwig kunnen volhouden. Nogmaals, een principe dat nauw samenhangt met principes één en drie. In Agile zal er nooit een ‘wanneer’ zijn voor het hele project, maar er zal een ‘wanneer’ zijn voor datgene waar het team op dit moment aan werkt. Omdat de omgeving waarin de business zich bevindt voortdurend verandert, zullen de vereisten dat ook doen. Dit betekent dat Agile-projecten nooit af zijn!

Principe 9. **Voortdurende aandacht voor een hoge technische kwaliteit en voor een goed ontwerp versterken agility.** Dit betekent dat het van vitaal belang is dat teams over de juiste vaardigheden beschikken, dat zij de regels van goed ontwerp volgen en dat zij geen technische schuld (technical debt) laten opbouwen.

Principe 10. **Eenvoud, de kunst van het maximaliseren van het werk dat niet gedaan wordt, is essentieel.** Dit principe kan ook het Goudlokje-principe worden genoemd: alles moet precies goed zijn. Het is bedoeld als tegenwicht voor principe negen, dat ingaat op het concentreren op kwaliteit en detail. Toch moet er voor ogen gehouden worden dat meer niet altijd beter is. Er moet worden gedaan wat nodig is om het te laten werken; niet minder, maar ook niet meer.

Principe 11. **De beste architecturen, eisen en ontwerpen komen voort uit zelfsturende teams.** Dit principe is het moeilijkste voor organisaties die net aan hun Agile reis beginnen, omdat het een verandering impliceert van de managementstijl, van het meten van prestaties en soms van organisatorische structuren, inclusief een herdefinitie van rollen en verantwoordelijkheden. Het punt is hier om de details over te laten aan de specialisten die het werk doen, namelijk de teamleden – dus niet micromanagement. De meeste werknemers van tegenwoordig zijn kenniswerkers en hebben geen supervisie nodig; ze hebben motivatie en ondersteuning nodig!

Principe 12. **Op vaste tijden, onderzoekt het team hoe het effectiever kan worden en past vervolgens zijn gedrag daarop aan.** Dit impliceert een niveau van volwassenheid in teams en de bereidheid om het eigen lot in handen te nemen. Om dit te laten slagen, is organisatorische verandering nodig en moet het leiderschap van de organisatie een omgeving van groot vertrouwen bouwen.

Nu zou het duidelijk moeten zijn waarom er kan worden gezegd dat agile met een kleine 'a' een mindset is, geleid door principes, waarvan klantgerichtheid waarschijnlijk de belangrijkste is. Agile met een kleine 'a' gaat over het ontwikkelen

van weloverwogen culturele, organisatorische veranderingen en gedragsveranderingen, die beginnen bij het leiderschap in de organisatie.

Geen enkel gebruik van de technieken in dit boek, het produceren van artefacten, het hernoemen van rollen of het beoefenen van de hier beschreven rituelen zal Agile met een grote 'A' doen slagen in een organisatie, tenzij de organisatie agile met een kleine 'a' wordt en dat is geen eenvoudige zaak. Als het gaat om de vraag waarom zoveel Agile-initiatieven mislukken, dan is het definitieve antwoord dat er eerst agile moet zijn met een kleine 'a' voordat Agile met een grote 'A' kan lukken in een organisatie.

Het is echter de menselijke natuur om gefixeerd te raken op een model of methode die lijkt te werken en dan die methode te bepleiten. Methoden zijn allemaal Agile met een grote 'A'. Maar als de twaalf principes in een organisatie worden toegepast, met de bedoeling om business agility te bereiken, ongeacht de methode, dan is dat agile met een kleine 'a'. Dit heeft meer kans van slagen en komt het dichtst bij de geest van het Agile Manifest.

Je zou je kunnen afvragen waarom dit boek überhaupt nodig is.

De hier beschreven Agile en Scrum methoden, verantwoordelijkheden en technieken zijn heel waardevol en zullen geweldige resultaten opleveren voor de organisatie, zolang dit de organisatie niet knecht. Het zijn nuttige tools en ze zullen goed van pas komen; maar laat nooit een Agile-methode dicteren wat en hoe dingen gedaan moeten worden, want dan is de echte organisatorische wendbaarheid verloren. Agile met een grote 'A' staat altijd ten dienste van agile met een kleine 'a' en niet andersom.

Aangezien dit boek praktisch gericht is, op hoe het meeste te halen valt uit Scrum als een Agile-methode, zal Agile vanaf dit punt met een grote 'A' geschreven worden, behalve in specifieke omstandigheden waar verwezen wordt naar 'agile gedrag', een 'agile mindset', of 'Agile-principes'.

1.2 Een pleidooi voor Agile

Iemand overtuigen van de voordelen van Agile is in het begin lastig en kan het beste op persoonlijk niveau gebeuren. In dit korte hoofdstuk wordt een mogelijke aanpak gepresenteerd, ter overweging. Deze aanpak heeft bewezen goed te werken.

De gekozen benadering heeft de vorm van een interactief verhaal of rollenspel met een scepticus en een voorstander als de twee hoofdpersonen. Dit verhaal gaat over een IT-project, maar het kan vrij gemakkelijk aan een andere omgeving worden aangepast door het scenario en enkele van de eigenschappen te veranderen. Aangezien de meeste mensen online initiatieven zowel hebben zien slagen als mislukken, lijkt dit een geschikt voorbeeld om te gebruiken. Wat achtergrondinformatie over de context:

XYZ B.V. is een bloeiende detailhandelaar in de snelgroeiende markt van consumptiegoederen. Het bedrijf is al tien jaar succesvol en heeft de

detailhandelsactiviteiten uitgebreid van twee naar vijftien verkooppunten verspreid over de tien grootste stedelijke gebieden van het land.

Recentelijk staat XYZ onder druk door opkomende online verkoopkanalen en, hoewel het bedrijf een uitgebreide online aanwezigheid heeft, realiseert het management zich dat zij snel online capaciteit moeten ontwikkelen. Als zij dit succesvol doen, kunnen zij gebruik maken van hun bestaande marktpositie en hun relaties om de nieuwe concurrentie het hoofd te bieden.

Er zijn twee opties om het gewenste doel te bereiken die overwogen kunnen worden.

Optie 1 – Conventionele Waterval projectaanpak

- Onderzoek naar wat de concurrentie doet (2 maanden)
  - Uitgevoerd door IT
- Onderzoek beschikbare technologie (2 maanden)
  - Uitgevoerd door IT
- Gedetailleerde specificaties bepalen voor online verkoopcapaciteit en een business case opstellen (4 maanden)
  - Eisen worden verzameld bij de business door een business analist van IT
- Een online verkoopmogelijkheid ontwerpen (4 maanden)
  - Uitgevoerd door IT
- Ontwikkelen van die online capaciteit (12 maanden)
  - Uitgevoerd door IT-silo's, ontwikkeling, testen, releasemanagement, IT-operatie
- **Totale projecttijd** **24 maanden**

<u>Vragen aan de business</u>

1. Vindt u dat bovenstaande een redelijke weergave is van dit soort projecten in uw bedrijf?
   - Normaal gesproken is het antwoord **ja**.
2. Kunt u, in het hierboven beschreven concurrentiescenario, twee jaar wachten op een oplossing?
   - Normaal gesproken is het antwoord **nee**.
3. Wat zou er gebeuren als u twee jaar moet wachten?
   - Normaal gesproken beschrijft het antwoord onheil.
4. Wat is volgens u een redelijke termijn om met deze nieuwe marktdeelnemers te kunnen concurreren?
   - De antwoorden variëren gewoonlijk van 4 tot 6 maanden.
5. Denkt u dat de vereisten die in maand 5 van het bovenstaande scenario zijn vastgesteld en overeengekomen, 19 maanden later, wanneer het project hopelijk is voltooid, nog dezelfde zijn?
   - Normaal gesproken is het antwoord **nee**.
6. Denkt u dat de business case die in maand 8 van het project ter goedkeuring is voorgelegd, 16 maanden later nog geldig is?

- Normaal gesproken is het antwoord **nee**.

7. Is uw ervaring dat dit soort projecten binnen budget en op tijd worden afgerond?
    - Normaal gesproken is het antwoord **nee**.
8. Met welke compromissen zou u genoegen nemen?
    - De antwoorden variëren, maar in het algemeen komt het volgende naar voren, niet in volgorde van belangrijkheid:
        a. Een minder goed product in kortere tijd
        b. Uitbesteding van de winkel aan een andere partij
        c. Gebruik van een bestaand platform
        d. Meer geld uitgeven om het sneller gedaan te krijgen

Hoewel b. en c. geldige opties zijn en kunnen worden onderzocht, hebben zij beide duidelijke nadelen – namelijk verlies van controle over de klant en lagere marges. Meer geld uitgeven kan haalbaar zijn als de onderneming over veel liquide middelen beschikt, maar de ervaring leert dat deze optie vaak niet levensvatbaar is of in ieder geval niet populair.

Optie 2 – Een Agile-projectaanpak

- Identificeer algemene vereisten op hoog niveau op basis van de eigen behoeften van de organisatie en wat concurrenten doen (1 maand)
    - Uitgevoerd door de business, met hulp van IT
- Eén of twee productassortimenten identificeren die het beste online zouden verkopen (2 weken)
    - Uitgevoerd door de business, met hulp van IT
- Identificeer en onderzoek de minimale capaciteit die nodig is om deze producten online aan te bieden (ontwerpsprint van 2 weken)
    - Uitgevoerd door IT met hulp van de business
- Bouw basis winkelwagenmogelijkheid (sprint van 4 weken)
    - Uitgevoerd door IT
- Bouwen en testen gebruikersinterface (sprint van 4 weken)
    - Uitgevoerd door IT
- Oplossing verfijnen (sprint van 2 weken)
    - Uitgevoerd door IT
- Start basisfunctionaliteit
- **Totale tijd tot online-functionaliteit beschikbaar is (4 ½ maand)**
- Verbetering van de capaciteit van de online winkel om uiteindelijk alle producten op te nemen die levensvatbaar worden geacht om online te verkopen (sprints van 4 weken)
    - Uitgevoerd door IT op basis van feedback van de business over het gedrag van kopers/klanten
- Tijd die nodig is om over een verfijnde en veelzijdige online-functionaliteit te beschikken **ca. 18 maanden**

Vragen aan de business

1. Bent u blij dat op de datum van lancering de online-winkelfunctionaliteit niet aan al uw eisen voldeed?
   - Normaal gesproken is het antwoord **nee**.
2. Wilt u liever dat aan alle eisen is voldaan als dat betekent dat u nog eens 19 maanden moet wachten?
   - Normaal gesproken is het antwoord **nee**.
3. Denkt u dat de eisen, of uw begrip van de eisen, in de 18 maanden na de lancering zullen veranderen?
   - Normaal gesproken is het antwoord **ja**.
4. Zijn alle vooraf vastgestelde vereisten die in het verleden voor typische tweejarige projecten zijn vastgesteld, noodzakelijk of geldig gebleken?
   - Normaal gesproken is het antwoord **nee**.

Terugdenkend aan uw ervaringen met eerdere projecten, na de periode van twee jaar tussen projectdefinitie en projectoplevering:

5. Heeft u gevraagd om het systeem bij te werken en waarom?
   - Normaal gesproken is het antwoord **ja –** er zijn nieuwe eisen ontstaan, markten-, klanten-, de concurrentie zijn veranderd, enz.
6. Hoeveel nieuwe eisen zijn er gedurende de periode van twee jaar ontstaan?
   - Dit antwoord kan van bedrijfstak tot bedrijfstak verschillen, maar gemiddeld is het antwoord **nogal wat**.
7. Hoeveel van de huidige kenmerken zou u als essentieel beschouwen?
   - Dit antwoord kan van bedrijfstak tot bedrijfstak verschillen, maar gemiddeld is het antwoord: **de meeste, maar niet alle**.
8. Als u het management van XYZ B.V. was, welke van de twee gegeven scenario's zou u dan gekozen hebben?
   - Meestal is het antwoord **optie 2.**

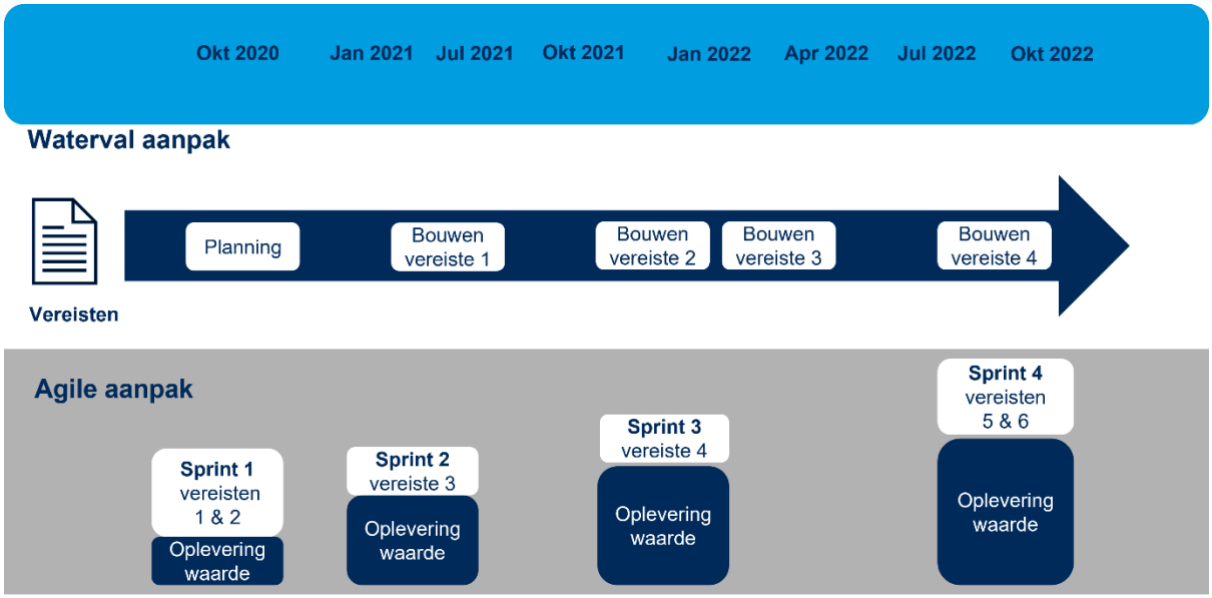
Dit brengt ons tot een illustratie, vereenvoudigd tot een praktisch overzicht.

Net als geld, heeft ook tijd waarde en als een waarde/tijd-vergelijking wordt gemaakt, is Agile een inkoppertje.

Agile is in potentie geschikt voor elke organisatie. Sommige zullen meer uitdagingen ondervinden bij het veranderen van denkwijzen en manieren van werken dan andere, maar alle organisaties kunnen het. Degene die dat niet doen, zullen hoogstwaarschijnlijk geen dag langer overleven in dit snel veranderende digitale tijdperk.

Cultureel gezien zal Agile echter niet in alle omgevingen werken en **tenzij er bij het topmanagement op zijn minst enige bereidheid bestaat om Agile-concepten te omarmen**, is het beter er niet naar te streven.

Figuur 1 Agile creëert sneller meer waarde dan traditioneel projectmanagement



Afbeelding gemaakt door EXIN op basis van: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC - Kdp Print Us.

De vraag die het topmanagement zichzelf moet stellen is: "Falen we als we het business agility-concept niet omarmen?" Het antwoord 'ja' impliceert de bereidheid om feiten onder ogen te zien. Agile worden is geen gemakkelijke opdracht, noch is het een gemakkelijke beslissing; het vraagt durf en moed om het door te zetten. Maar het geluk is met de dapperen.

1.3 Kritische succesfactoren om Agile te bereiken

Agile is een aanpak die waarde levert aan klanten door middel van korte iteratieve leveringscycli, door een grote betrokkenheid van de klant, door crossfunctionele teams die verantwoordelijkheid nemen voor de kwaliteit van wat ze opleveren. Meer beschrijven dan dit is al het begin van verdieping in een specifieke Agile-methode.

In het algemeen kunnen enkele kritische succesfactoren worden gedefinieerd, ongeacht welke Agile-methode gebruikt wordt.

1.4 Leiderschap en gedrag, cultuur, ethiek en vertrouwen

Hoewel elke organisatie, of een deel ervan, Agile-methodes kan gebruiken, werkt Agile het beste als door het leiderschap de juiste omgeving wordt gecreëerd.

Agile is, net zoals Lean, in de eerste plaats een cultureel en gedragsmatig fenomeen en in essentie werken de gedragsveranderingen die voor Agile nodig zijn, het beste als het leiderschap van de organisatie ze begrijpt en toewijding toont om ze mogelijk te maken. Deze veranderingen in cultuur worden aanzienlijk beïnvloed door de leiderschapsstijl in de organisatie.

In de afgelopen 20 jaar is veel gesproken over dienend leiderschap, maar er is weinig veranderd in de manier waarop management zich gedraagt, handelt en leidinggeeft. Wat de implementatie van Agile in organisaties meer dan wat ook succesvol maakt, is een verandering in de houding van managers op alle niveaus van de organisatie.

De reden waarom dit zo belangrijk is, is dat autonome, zelfregulerende, crossfunctionele teams Agile-projecten uitvoeren. Om dit succesvol te maken, moet er een fundamentele verschuiving plaatsvinden van de management- en leiderschapsstijl.

In de woorden van Michael Shota, het leiderschap moet begrijpen dat het niet gaat om Agile Doen, het gaat om Agile Zijn en Agile Zijn begint bij het leiderschap van de organisatie.

Leiders in organisaties waar Agile succesvol wordt toegepast, tonen een oprechte belangstelling voor de manier waarop werknemers hun taken uitvoeren, inspireren werknemers om zich te ontwikkelen en zij fungeren als coaches, trainers en degenen die de voorwaarden scheppen, in plaats van als toezichthouders.

Succesvolle Agile-implementaties vragen om **vertrouwen** en **openheid**, dat wil zeggen: een niet aflatende betrokkenheid van het management tegenover alle medewerkers en een oprecht verlangen om de mensen in de organisatie te helpen om het beste te worden dat ze kunnen zijn.

Dit vraagt ook om **open, eerlijke en rechtvaardige** communicatie en dat werknemers vol vertrouwen contact kunnen opnemen met het leiderschap en de structuren in de organisatie.

Op de vraag *Hoe creëer je een omgeving van vertrouwen en openheid?* is het antwoord dat het management moet leren er op te vertrouwen dat hun mensen het juiste doen.

Werknemers zullen nooit de nodige open en eerlijke communicatie vertonen die nodig is om Agile en Lean te laten lukken, tenzij ze weten dat ze het management kunnen vertrouwen en dat ze niet gestraft zullen worden als ze iets nieuws proberen, de waarheid spreken en eerlijk zijn over wat er is gebeurd.

Ernest Hemmingway definieerde het als volgt: “De beste manier om uit te vinden of je iemand kunt vertrouwen, is door die persoon te vertrouwen.”

Het volgende voorbeeld laat zien waarom omgevingen met een hoge mate van vertrouwen zo veel gunstiger zijn voor organisaties dan omgevingen met een lage mate van vertrouwen:

Weinig vertrouwen
Het management vertrouwt er niet op dat de werknemers hun werk goed doen en daarom schrijven zij de werknemers voor hoe zij hun werk moeten doen. Vaak zijn de door het management voorgeschreven methoden onpraktisch, verouderd, of omslachtig en om hun werk gedaan te krijgen, doen de werknemers het werk op een manier die het gevraagde resultaat mogelijk maakt.

Het gevaar is dat het vangnet en de controlemechanismen die deel uitmaakten van de voorgeschreven methode, het proces, of de procedure, nu ook aan het wankelen worden gebracht.

Als er iets fout gaat, zal niemand erkennen het proces niet te hebben gevolgd en fouten worden vaak weggemoffeld door werknemers die proberen negatieve consequenties als gevolg van hun afwijkende werkwijze te ontlopen. Het resultaat is dat de omgeving nog onstabieler wordt, tot de zaak uiteindelijk instort.

Als onderdeel van de oorzakenanalyse wordt een heksenjacht geopend om de schuldige te vinden en te ‘straffen’.

Het probleem is dat zelfs als de schuldige wordt ontdekt en gestraft, de organisatie en het management verliezen. De schade is al aangericht en wat verloren is, is verloren. Verrassend genoeg is dit schadelijke gedrag de modus operandi in veel organisaties.

Richard DiPilla zei: “Het heeft geen zin om schakers aan te nemen en ze vervolgens als schaakstukken te behandelen. Het heeft geen zin om slimme mensen aan te nemen en ze vervolgens domme regels te laten volgen.”

In tegenstelling tot bovenstaande, worden nu de gedragingen onderzocht die in een Agile-omgeving worden verwacht.

Groot vertrouwen
Het management heeft professionele mensen in dienst en verwacht van hen dat zij hun vaardigheden en talenten inzetten voor de verbetering van de organisatie. Medewerkers werken samen in kleine teams, van meestal tussen de drie en negen personen, hebben een doel en beslissen onderling hoe ze dat doel bereiken en hoe het werk moet worden gestructureerd om het doel binnen de gestelde termijn te bereiken.

Omdat de teamleden dichter bij het werk staan, begrijpen zij veel beter hoe de dingen in de praktijk werken en wat er gedaan moet worden.

Ze ontwerpen hun eigen werk en maken daarbij gebruik van door het management vastgestelde heuristische controles[2] die ervoor zorgen dat risico's tot een minimum worden beperkt en optimale prestaties worden geleverd.

Als er iets fout gaat, communiceert het team openlijk en zoekt een permanente oplossing, om herhaling te voorkomen.

Het management ondersteunt het team met advies, haalt blokkades weg, faciliteert en bemiddelt tussen teams.

[2] Heuristische controles definiëren gewoonlijk gedragsregels voor bepaalde omstandigheden en geen acties. Als je weet hoe je je moet gedragen en welk resultaat wordt verwacht, kun je beslissen wat de meest geschikte actie is. Als de controle echter een actie definieert, zal de actie altijd worden ondernomen, of die nu passend is of niet!

Omdat mislukking wordt aanvaard als deel van het leven, maar herhaling niet wordt getolereerd, worden problemen voortdurend opgelost en stapelen ze zich bijna nooit op tot catastrofes. De organisatie leert voortdurend en prestaties worden voortdurend verbeterd.

- Welke van de twee werkomgevingen is het meest wenselijk?
- Waar zou je liever werken?

Uit bovenstaand voorbeeld kan worden geconcludeerd dat één van de belangrijkste gedrags- en culturele veranderingen die in een Agile-organisatie plaatsvindt, het worden van een **lerende organisatie** is. Het is nodig om te transformeren naar een omgeving met een hoge mate van vertrouwen, waar mensen niet bang zijn om problemen aan te pakken zodra ze die zien en ze op te lossen voordat ze een probleem worden voor of in de organisatie.

Het idee van een lerende organisatie werd gepopulariseerd door Peter Senge in zijn boek *The Fifth Discipline* (1990) en het is zeker de moeite waard om een voorbeeld aan dit boek te nemen. Senge stelde de volgende vijf kenmerken van een lerende organisatie voor:

1. **Systeemdenken**, wat betekent dat iedereen de organisatie als een systeem begrijpt. Hoe beter iedereen het systeem begrijpt, hoe meer zij zullen leren en hoe beter de organisatie zal worden.
2. Systeemdenken impliceert ook engagement van het individu om het proces van **leren en persoonlijk meesterschap** de norm te maken in de organisatie. Leren wordt méér dan alleen het verwerven van kennis. Het wordt een focus op het vermogen om productiever te zijn. Individueel leren is de verantwoordelijkheid van het individu en niet uitsluitend die van de organisatie en ‘persoonlijk meesterschap’ wordt dagelijks in praktijk gebracht.
3. Aannames van mensen (**mentale modellen**) veranderen. Deze aannames kunnen alleen veranderen als een confronterende houding plaats maakt voor een open cultuur, die onderzoek en vertrouwen bevordert.
4. Het ontwikkelen van een **gedeelde visie** is een essentiële, zelflerende motivator voor werknemers, omdat deze een collectieve identiteit creëert die focus en energie geeft voor leren. Het toepassen van de praktijk van een gedeelde visie creëert een geschikte omgeving voor het ontwikkelen van vertrouwen door communicatie en samenwerking binnen de organisatie en het moedigt medewerkers aan hun ervaringen en meningen te delen, waardoor de effecten van organisatorisch leren worden versterkt.
5. **Teamleren** vindt plaats door het delen van opgebouwde individuele kennis. Het voordeel van team- of gedeeld leren is dat medewerkers sneller groeien en dat het probleemoplossend vermogen van de organisatie wordt verhoogd door verbeterde toegang tot (crossfunctionele) kennis en expertise. Individuen zijn betrokken bij de dialoog en discussie om hierdoor gedeelde betekenis en begrip te bereiken. Individuen hebben concrete manieren om kennis te delen nodig.

1.4.1 Agile en de impact ervan op de organisatiestructuur

Theoretisch kan Agile in elke organisatie worden toegepast. Als het goed gedaan wordt, zal Agile onvermijdelijk de samenhang in de organisatie gaan veranderen, met de grootste impact op de structuur en de manier waarop rollen en verantwoordelijkheden zijn gedefinieerd.

Omdat met Agile-projecten worden uitgevoerd door autonome crossfunctionele teams, leidt dit onvermijdelijk tot plattere organisaties en multi-getalenteerde werknemers met meerdere vaardigheden, die minder supervisie nodig hebben.

Het is erg aannemelijk dat de organisatie van de toekomst er precies zo uit zal zien. Naarmate meer mensen in het werkveld kenniswerkers zijn, wordt de behoefte aan complexe managementstructuren minder groot.

2 Agile invoeren

Tegenwoordig bestaan er veel Agile-methoden. Scrum is de meest gebruikte methode. Enkele van de andere populaire methoden worden beschreven in Bijlage A van dit boek.

Lean is de gemeenschappelijke voorouder van alle Agile-methoden en technieken. Lean en Agile delen gemeenschappelijke elementen en hoewel het in dit boek de bedoeling is om te focussen op Agile en Scrum, zullen er van tijd tot tijd methoden en technieken aan bod komen die niet uniek zijn voor Scrum.

Zodra Agile Scrum in een organisatie wordt toegepast, is het zinvol om ook andere methoden verkennen, omdat daar misschien parels te vinden zijn die ook bruikbaar zijn.

De praktijken, principes en het gebruik van artefacten die in dit boek worden gedefinieerd, zijn het absolute minimum om te kunnen zeggen dat er ‘Agile’ gebruikt wordt.

Toch is het, in elk geval in het begin, een goed idee om het bij één methode te houden. Scrum is vanwege de eenvoud zeer geschikt om mee te beginnen.

De reden om te beginnen met één methode en deze even aan te houden, is dat zo eerst gefocust kan worden op het bewegen van mensen en organisatorische elementen in de richting van het inbedden van Agile in de cultuur.

Te veel focus op artefacten, tools en technieken die nuttig kunnen zijn, leidt de aandacht af van wat echt belangrijk is. Agile is bovenal een mentaliteitsverandering, daarna een organisatorische verandering en een gedragsverandering.

Alles in Scrum is specifiek ontworpen om de reis te faciliteren. Laat je niet te snel afleiden; afleiding werkt destructief en leidt tot mislukking. Het is raadzaam om dicht bij de basis te blijven totdat de algehele mindset in de organisatie is veranderd en de weerstand tegen de nieuwe manier van werken is weggeëbd, alvorens methoden en technieken uit andere frameworks te introduceren en daarmee te experimenteren.

Agile laten lukken in een organisatie vraagt toewijding en focus.

In de bijlage van dit boek worden andere methoden bekeken, maar voor nu zal de focus liggen op Scrum als de gekozen Agile-methode.

2.1 De uitdaging bij het invoeren van Agile

Het is belangrijk om te begrijpen dat, net als bij Lean, de hele organisatie waarde creëert. Dat gebeurt niet door een enkel productieproces, dienstproces of alleen door een enkel team. Het is de organisatie die waarde creëert voor klanten en de hele organisatie combineert al haar processen, functieoverstijgend, om waarde te creëren voor een klant in de vorm van een product of een dienst.

Met Lean worden functieoverstijgende en procesoverstijgende waardestromen in kaart gebracht. Er moet wel opgepast worden dat er niet uitsluitend gericht wordt op producten of de product backlog.

Op dezelfde manier is het ook met Agile onwaarschijnlijk dat er de waarde gecreëerd wordt die de klanten zoeken, tenzij de teams die de oplossingen creëren multifunctioneel en multidisciplinair zijn.

Succesvolle levering door samenwerkende teams daagt echter de hiërarchie, de structuren en de gevestigde belangen in de organisatie uit.

De bedenkers van Scrum zeiden – Scrum is eenvoudig te begrijpen, maar moeilijk om echt onder de knie te krijgen!

Het moeilijke is dat er veel moet veranderen in de organisatie om Agile en Scrum met succes en effectief te gebruiken en die veranderingen zijn echt lastig. De invoering van Agile of Scrum is een omvangrijk verandermanagementinitiatief dat niet mag worden onderschat en zorgvuldig moet worden gepland en uitgevoerd.

Er zijn verschillende methoden en technieken voor verandermanagement en het is onmogelijk om voor te schrijven welke er gebruikt moet worden; als een methode werkt, blijf die dan gebruiken. Ervaring wijst uit dat evolutionaire verandermethoden een grotere kans van slagen hebben dan revolutionaire verandering.

Een agile mindset bestaat dus voor een deel uit geduldig zijn en de verandering niet te overhaasten – haastige spoed is zelden goed.

Business change management is niet de focus van dit boek, dus de gebruikelijke verandermanagementbenaderingen zullen hier niet worden beschreven. Het is echter nuttig om op hoog niveau twee mogelijke benaderingen te beschrijven om Agile in een omgeving te introduceren.

2.2 Verandering beheren

Weerstand tegen verandering is zinloos, omdat verandering onvermijdelijk is en dat geldt ook voor het afstemmen op een Agile-methode om incrementeel waarde te leveren.

Als organisaties niet om kunnen gaan met de *menselijke* kant van de verandering, zal de implementatie van Agile mislukken.

Jeffrey M. Hiatt en Tim J. Creasey hebben in hun boek *Change Management – The People Side of Change* (2012), verandermanagement gedefinieerd als een proces, dat een herhaalbare cyclus volgt en gebruik maakt van een holistische set tools; en als een competentie, omdat die verandering mogelijk maakt en het vermogen creëert om de effectiviteit van de organisatie te verhogen.

Zij definieerden vijf grondbeginselen van verandermanagement die de mensen helpen om om te gaan met verandering en deze te accepteren.

1. We veranderen met een reden
De verandering heeft als doel om een toekomstige toestand te bereiken met een specifiek en gewenst resultaat. De redenen voor verandering variëren sterk van organisatie tot organisatie en kunnen kostenvermindering, grotere klanttevredenheid en in het geval van Agile, meer waarde voor klanten omvatten.

De behoefte aan verandering wordt altijd ingegeven door een kans of door een probleem dat moet worden opgelost.

2. Organisatorische verandering vraagt individuele verandering
Nieuwe tools of processen zijn niet voldoende om organisatorische verandering te bewerkstelligen. Er zijn mensen in de organisatie nodig die nieuwe waarden aannemen en op nieuwe manieren gaan werken.

3. Organisatorische resultaten zijn het collectieve resultaat van individuele verandering
Verandering is een individuele gebeurtenis, dus er zijn menselijke factoren om rekening mee te houden. Dat betekent dat hoe hoger de adoptie door werknemers, hoe dichter de organisatie is bij het bereiken van de gewenste resultaten.

4. Verandermanagement is een kader voor het beheer van de menselijke kant van verandering
Weerstand tegen verandering is de norm in organisaties, vooral in omgevingen die voortdurend veranderen. Het managen van de menselijke kant van verandering zorgt voor een hogere adoptiesnelheid en -capaciteit.

5. We passen verandermanagement toe om de voordelen en gewenste uitkomsten van verandering te realiseren
Het hoofddoel van verandermanagement is het tot stand brengen van de gewenste toekomstige toestand en het stimuleren en het ondersteunen van de verwezenlijking van de verwachte resultaten.

2.3 ADKAR® en ADAPT

Hiatt en Creasey creëerden een model voor het faciliteren van verandering op individueel niveau. Zij noemden dit het ADKAR-model, een acroniem voor: Awareness (Bewustzijn), Desire (Verlangen), Knowledge (Besef), Ability (Vermogen) en Reinforcement (Bekrachtiging). Verandering is alleen geslaagd als iedere werknemer deze vijf mijlpalen heeft behaald.

Van de methoden die in dit boek worden beschreven, liggen de ADKAR- en ADAPT-methode waarschijnlijk het dichtst bij conventioneel verandermanagement.

De tweede methode is Lean-management, maar daarover later meer.

ADAPT[3] is gebaseerd op het ADKAR-model[4] en is een populaire methode voor verandermanagement die door PROSCI® is ontwikkeld, met slechts een subtiele wijziging ten opzichte van de oorspronkelijke aanpak.

Scrum-goeroe Mike Cohn paste het ADKAR-model aan naar ADAPT, omdat hij stelde, dat in een omgeving waar het meeste werk kenniswerk is, Besef en Vermogen onafscheidelijk zijn. Als zodanig combineerde hij ze onder de noemer Vermogen. Cohn vindt ook dat Bekrachtiging een te vage term is en hij verving deze uiteindelijk door twee stappen, namelijk Promotie en Overgang.

ADAPT staat dus voor:

- Awareness (Bewustzijn)
- Desire (Verlangen)
- Ability (Vermogen)
- Promotion (Promotie)
- Transfer (Overgang)

Om een algemene beschrijving van ADAPT te geven:

Awareness (Bewustzijn)
Pas als we ons realiseren dat de status quo niet langer wenselijk is, kan echte verandering beginnen, maar er bewust van worden dat wat in het verleden werkte, niet langer werkt, kan buitengewoon moeilijk zijn.

Over het algemeen worden mensen zich langzaam bewust van de noodzaak te veranderen. Dit is vaak te wijten aan een gebrek aan inzicht in het grote geheel.

De noodzaak om Scrum in te voeren, kan het gevolg zijn van een samenloop van factoren die niet voor iedereen zichtbaar is. En als de noodzaak tot verandering slechts voor enkele mensen duidelijk is (bijvoorbeeld voor diegenen die de daling van verkoop aan nieuwe klanten hebben gezien), zullen geruchten over verandering zich gaan verspreiden.

Het is van vitaal belang dat iedereen in de organisatie zich bewust is van de feiten die aan de verandering ten grondslag liggen, zodat veranderingen niet leiden tot negatief gedrag en negatieve gevolgen voor de organisatie.

Om ervoor te zorgen dat bewustzijn op feiten is gebaseerd en niet op speculatie, moet het probleem duidelijk gecommuniceerd worden en moeten er bewijzen met objectieve feiten en cijfers geleverd worden.

Er moet uitgelegd worden hoe de verandering het probleem zal oplossen en het kan een goed idee zijn om dat praktisch te doen door een proefproject uit te voeren, waarmee wordt bewezen dat de verandering het probleem inderdaad oplost.

---

[3] Cohn, M. (2010) *Succeeding with Agile: Software Development Using Scrum*. Addison-Wesley. Hoofdstuk 2.
[4] Meer informatie over het ADKAR-model is te vinden op https://www.prosci.com/methodology/adkar.

Er kunnen verschillende redenen zijn waarom de organisatie Agile wil introduceren, maar de boodschap kan duidelijk en eenvoudig gehouden worden – houdt het bij de belangrijkste redenen, twee of drie en niet meer dan dat.

Desire (Verlangen)
Het kan moeilijk zijn voor mensen om te aanvaarden dat het huidige proces geen succes oplevert voor de organisatie of niet de gewenste resultaten oplevert voor de klanten.

Wat projectmanagement betreft is dat ongetwijfeld waar, zoals de meesten van ons ofwel geleerd hebben, ofwel uit ervaring met de traditionele manier waarop projecten worden uitgevoerd weten.

Het heeft altijd goed gewerkt, waarom moet er nu veranderd worden?

Het is een feit dat als er wat tijd met mensen doorgebracht wordt om te praten over hun negatieve ervaringen met projecten, het al snel duidelijk wordt dat het niet altijd goed heeft gewerkt.

Als de negatieve uitkomsten tegenover elkaar gezet worden en er uitgelegd of getoond wordt hoe Agile kan helpen deze uitkomsten te voorkomen, ontstaat een verlangen naar meer positieve uitkomsten.

Als mensen bijvoorbeeld gevraagd wordt of wat ze nu nodig hebben en wat ze twee jaar geleden nodig hadden hetzelfde is, dan zullen ze vertellen dat dat niet zo is. Bij Watervalprojecten zullen ze ook erkennen dat wat wordt opgeleverd vaak verouderd is, niet langer nodig is of helemaal niet is afgestemd op de huidige behoeften.

Het wordt zo relatief eenvoudig om de oude manier, die leverde wat niet meer nodig is, te vergelijken met een Agile-aanpak. Het gebruik van Agile, waarbij het projectteam zich altijd richt op de huidige eisen en in korte cycli levert tegen de huidige behoeften, zal hen doen zeggen: “Ja, dit is wat we willen en dit is beter dan de oude manier van werken.”

Om verlangen naar verandering te creëren, moeten belanghebbenden zien dat er een betere manier is en dat de organisatie dringend betere resultaten nodig heeft. Belanghebbenden gaan meestal akkoord als er goede argumenten zijn voor de verandering naar Agile.

Maar wees gewaarschuwd: er hoeft niet net gedaan te worden alsof het verleden allemaal fout was; er moet wel vooral gefocust worden op nieuwe voordelen die echt een onmiddellijk of groot verschil zullen maken. Het is nog beter om een proefproject (pilot) te doen om te laten zien dat het lukt en goed werkt. Zorg ervoor dat het project succesvol is!

Als dingen beginnen te veranderen, help mensen dan door de moeizaamheid van de verandering heen naar de nieuwe manier. Wees geduldig en help mensen om de oude manier van werken los te laten. Laat de betrokkenen zien dat de verandering niet zo eng is en veilig kan worden doorgevoerd.

Betrek een zo breed mogelijke groep belanghebbenden bij de verandering en denk ook eens aan stimulansen om te veranderen. Prikkels als motor voor verandering

zijn echter afhankelijk van de context en de cultuur waarin de veranderingen plaatsvinden. Wees daar gevoelig voor.

Tom Peters heeft ooit gezegd dat het niet het moeilijkste is voor mensen om nieuwe ideeën te aanvaarden, maar om hen oude ideeën te doen vergeten.

De enige manier om mensen te laten loslaten, is ze iets beters laten zien ter vervanging!

Al het bewustzijn en de wil van de wereld zullen een team alsnog nergens brengen als het niet ook het vermogen verwerft om Agile te zijn.

Ability (Vaardigheid)
Net als in het voorbeeld van Tom Peters hierboven, vraagt Scrum niet alleen van teamleden dat ze nieuwe vaardigheden aanleren, maar ook dat ze oude afleren.

Enkele grotere uitdagingen waar Scrum-teams mee te maken krijgen, zijn de volgende:

- aanleren van nieuwe technische vaardigheden;
- leren denken en werken als een team;
- het concept van zelfsturing;
- leren hoe werkende producten en diensten kunnen worden gecreëerd binnen korte, vaste tijdscycli van één tot vier weken (time-boxes).

De juiste training is belangrijk. Training alleen is echter niet genoeg; in het begin zullen teams ook bij de hand genomen moeten worden en coaching nodig hebben. Het is raadzaam om iemand te zoeken van buiten het bedrijf, met veel ervaring en een geschiedenis van succesvolle Scrum-implementaties, die de Agile Scrum coach voor het team kan zijn.

Verwacht niet te snel te veel. Hoewel mensen verantwoordelijkheid gegeven moet worden voor de nieuwe en correcte manier van doen, hoeven er geen onredelijke doelen gesteld te worden.

Inspecteer, meet, geef feedback en betrek belanghebbenden bij het oplossen van problemen en vraagstukken.

Onthoud dat met alleen training het team er nooit helemaal klaar voor is om Agile te werken; er moet begonnen worden, want een deel is leren tijdens het werken en leren van het maken van fouten. Als het Scrum-team nog onbekend is met Scrum, is het beter om strikt de regels te volgen, totdat er meer ervaring is met Scrum en Agile werken.

Promotion (Promotie)
Een business zal niets verkopen als de waren niet op de markt gebracht worden. Waarom zou dat voor veranderingen anders zijn?

De voordelen van de nieuwe methode of van het loslaten van het verleden moeten voortdurend ‘verkocht’ worden om ervoor te zorgen dat de nieuwe aanpak, in dit geval Agile Scrum, op de juiste manier wordt ingebed en geïnstitutionaliseerd in de organisatie.

Er zijn drie doelstellingen in de Promotiefase:

1. Aangezien er hoogstwaarschijnlijk klein begonnen zal worden, moet het proefproject de basis leggen voor de volgende stap in het invoeren van Agile Scrum. Het promoten van huidige successen en het creëren van bewustzijn van de nieuwe aanpak zal een vliegende start geven aan de volgende ronde verbeteringen.
2. Het tweede doel is om agile gedrag in bestaande teams te versterken door de prestaties van die teams bekend te maken in de organisatie.
3. Het derde doel is het creëren van bewustzijn en belangstelling bij diegenen buiten de groepen die direct betrokken zijn bij het invoeren van Scrum.

Promoten betekent succesverhalen verspreiden en een sfeer uitstralen waar anderen deel van willen uitmaken. Soms is een presentatie een geweldige manier om de verandering te promoten.

Door belanghebbenden te laten deelnemen aan activiteiten zoals (serious) games en simulaties, krijgen zij ervaringen uit de eerste hand mee zonder de pijn en negatieve gevolgen te ervaren die gepaard gaan met veranderingen in het echte leven.

Transfer (Overgang)
Stel, het proefproject is gedaan en het was een daverend succes – maar de organisatie is niet in staat om het momentum vast te houden, omdat de rest van de organisatie niet volgt. Hoe komt dat? Omdat Agile niet alleen inhoudt dat werkwijzen veranderen, maar ook dat er op een andere manier gedacht wordt. Als de oude manier van denken in de rest van de organisatie blijft bestaan, zal dat uiteindelijk zelfs de beste start die een organisatie kan maken in de kiem smoren.

Als de implicaties van werken met Scrum niet worden overgedragen aan andere afdelingen, zal het immuunsysteem van de organisatie de verandering voortdurend aanvallen en uiteindelijk de strijd winnen.

Dit betekent niet dat de rest van de organisatie ook Scrum moet gaan gebruiken, maar wel dat de rest van de organisatie op zijn minst compatibel moet worden met Scrum.

Hier mogen veranderende meetwaarden, HR- en ander beleid en zelfs bestuurs- en controlestructuren niet over het hoofd worden gezien. Prestaties van een Agile-team kunnen beter niet op dezelfde manier gemeten worden als van een team in een traditionele hiërarchische organisatie. Oude manieren van management zullen vervangen worden door een nieuwe manier van managen, meten en structureren van teams. Controles zullen veranderen en de meest fundamentele zijn project- en programmamanagement, HR-beleid en rollen en verantwoordelijkheden, managementstructuren, prestatiemanagement, productmanagement en zelfs financiering, budgettering en financiële praktijken in de organisatie.

De invoering van Agile Scrum zal gevolgen hebben voor de hele organisatie; de implicatie is dat uiteindelijk alles in de organisatie zal veranderen en liever eerder dan later.

Het laatste woord over ADAPT
Net als Scrum zelf, is ADAPT een iteratieve aanpak. Het begint met het besef dat verandering nodig is en dat de huidige manier van werken niet langer acceptabele resultaten oplevert. Naarmate het besef zich verspreidt, krijgen sommigen, de pioniers (early adopters), de behoefte om Scrum uit te proberen in een poging de situatie te verbeteren.

Met vallen en opstaan, ontwikkelen deze pioniers het vermogen om succesvol met Scrum te zijn in de organisatie. Een nieuwe status quo kan ontstaan met een klein aantal teams dat Scrum met succes gebruikt, binnen een bredere organisatie die dat niet doet.

Naarmate deze eerste Scrum-teams hun toepassing van Scrum blijven verbeteren, beginnen ze hun successen te promoten – soms informeel zoals tijdens de lunch met vrienden in een ander team, of formeler zoals in een afdelingsbrede presentatie of een gepland veranderprogramma.

Dit helpt mensen in andere teams vooruit te gaan, van Besef naar Verlangen naar Vermogen en al snel beginnen ook de andere teams hun successen te Promoten.

2.4 Wat wordt er bedoeld met ‘producten’

In Agile Scrum is het woord *product* een inclusief woord. Sommige mensen gebruiken het woord product al op deze manier en zeggen dat producten zowel goederen als diensten kunnen zijn en in feite zijn producten in veel gevallen zowel goederen als diensten.

Anderen stellen diensten tegenover producten en hebben het over producten en diensten; in dat geval is product geen allesomvattende term.

Aangezien dit werk gebaseerd is op algemeen geaccepteerde Scrum-praktijken, wordt het standpunt ingenomen dat in Scrum, ‘product’ een inclusieve term is en dat diensten als zodanig ook producten zijn.

3 Lean-management

ADAPT kan worden gezien als een methode voor verandermanagement. Een andere vorm die verandering mogelijk maakt, is veel meer organisch en evolutionair.

Het invoeren van Lean als manier om een organisatie te leiden en te beheren betekent een radicale verandering in managementstijl, gedrag van de werknemers, organisatiestructuur en de manier waarop rollen en verantwoordelijkheden worden gedefinieerd.

Lean is echter eerder een evolutionaire managementaanpak dan een poging om revolutionaire organisatorische verandering te bewerkstelligen. Dat betekent dat het tijd vergt en dus een niet aflatende inzet van het leiderschap van de organisatie.

Alle Agile-methoden zijn gebaseerd op de funderingen die door Lean zijn gelegd. Het opbouwen van organisatorische wendbaarheid en een Lean-organisatie worden, is in de kern hetzelfde.

Het doel van dit boek is niet om Lean te onderwijzen, maar enkele Lean-principes en -praktijken die de basis vormen voor succesvolle Agile-implementaties, zullen worden besproken.

3.1 Lean als strategie en managementaanpak

Lean kan vooral worden begrepen als een operationele strategie en een manier om een bedrijf te ontwerpen, om te vormen en te leiden, waarbij de nadruk ligt op twee belangrijke onderwerpen:

- maximaliseren van waarde voor de klant;
- terugbrengen van verspilling.

Na de Tweede Wereldoorlog werd Japan geconfronteerd met aanzienlijke tekorten aan grondstoffen en de leiding van Toyota begreep dat een reeks innovaties de enige manier was om zowel in continuïteit van de processtroom als in een grote verscheidenheid van het productaanbod te voorzien.

Toyota richtte zich op het minimaliseren van grondstoffen die nodig zijn voor de productie van auto's en tegelijkertijd het verkorten van de doorlooptijd tussen de bestelling van de klant en de levering van het voertuig.

Deze ideeën en methoden werden bekend als het Toyota Productie Systeem (TPS). Dit systeem verlegde de focus van de productie-ingenieur: van individuele machines en hun gebruik, naar de flow (doorstroming) van het product door het gehele proces, gebaseerd op de vraag van de klant.

Toyota ontdekte al snel dat fabrieksarbeiders veel meer konden bijdragen dan alleen spierkracht; het ligt voor de hand dat de mensen die de machines bedienen met innovatieve ideeën kunnen komen en het productieproces kunnen verbeteren.

Twee fundamentele principes vormen de basis van Lean:

- **Just-in-time-productie**: alleen doen wat nodig is wanneer het nodig is.
- **Jidoka**: standaardisatie en automatisering.

Naarmate Toyota zich ontwikkelde, werd TPS onderdeel van het allesomvattende *The Toyota Way*, dat in 2001 werd gepubliceerd. Vandaag de dag zijn de Jidoka en just-in-time principes nog steeds relevant, maar de Toyota Way filosofie waarbinnen zij nu zijn ondergebracht heeft twee hogere principes opgenomen:

- respect voor mensen;
- continue verbetering.

Lean heeft fundamentele verbeteringen te bieden voor de organisatie van nu: van ontwerp en initiële opzet tot operatie en transformatie. Deze concepten draaien om langetermijnresultaten voor klanten en belanghebbenden, het opbouwen en afstemmen van procescapaciteit in de hele onderneming, culturele voorwaarden voor duurzaam succes en continue verbetering waar iedereen, elke dag en overal bij betrokken is.

Lean bedrijven zijn succesvoller dan niet-Lean bedrijven. Ze creëren meer waarde en zijn beter in staat talent te behouden. Ze zijn ook beter in staat tot langdurig succes.

Figuur 2 Lean-principes

Afbeelding gemaakt door EXIN op basis van: Botha, J. (2021). *Lean fundamentals for IT [Courseware].* GetITright.

Ten eerste helpt Lean om zich te richten op waarde voor de klant en door dit te doen, voegen organisaties meer waarde toe aan hun producten en diensten, terwijl ze bronnen van verspilling wegnemen en hun wendbaarheid en aanpassingsvermogen vergroten. Door meer verbonden te zijn met klanten en eindgebruikers en de interactie met hen te verbeteren, kan een organisatie de klanttevredenheid en klantloyaliteit aanzienlijk doen toenemen.

Ten tweede verbetert de Lean-organisatie voortdurend de procesprestaties. Haar diensten zijn van hogere kwaliteit, de levertijden zijn korter en de project- en operationele efficiëntie nemen voortdurend toe.

Lean-organisaties beseffen ook dat de meeste werknemers van nu kenniswerkers zijn en dat de sterkste troef van een organisatie haar mensen zijn. De mate van betrokkenheid van de werknemers is hoger in Lean-organisaties en zo ook hun motivatie en werktevredenheid.

Ten derde is er financiële winst te verwachten door het **terugbrengen van verspilling in het proces, het optimaliseren van waardetoevoegend werk, het tijd vrijmaken om meer waarde toe te voegen**. Organisaties zijn hier niet toe in staat als ze geen inzicht hebben in het 'proces' dat door de hele organisatie heen wordt gebruikt om waarde te creëren. De techniek van het maken van een **waardestroomdiagram (Value Stream Map, VSM)**, verschaft de nodige inzichten om verspilling niet alleen te verminderen maar zelfs te elimineren.

Eén van de manieren om verspilling tegen te gaan, is het verkorten van de tijd tussen orderontvangst en levering door het werk in het systeem te stroomlijnen en alle processen en systemen te veranderen in een **pullsysteem** in plaats van een **pushsysteem**.

Optimalisering van de werkstroom verbetert ook de cashflow, aangezien de noodzaak van grondstoffenvoorraden wordt verminderd. Er moet benadrukt worden dat meer winstgevendheid niet het primaire doel is van Lean. Hoewel meer winstgevendheid te verwachten is, is het een secundair gevolg van het verbeteren en verminderen van de inspanning en tijd die worden besteed aan niet-waardetoevoegende activiteiten.

Ook slechte kwaliteit kan een drijfveer zijn voor de invoering van Lean. Slechte kwaliteit heeft zowel binnen als buiten de organisatie gevolgen. Aspecten, zoals reputatieschade en verlies van vertrouwen van de klant, kunnen de organisatie opzadelen met aanzienlijke kosten of boetes. Verspilling van talent of een onverwacht aantal defecten veroorzaken ook stress in de organisatie. Als gevolg van inspecties, herbewerking of demotivatie zullen de kosten ook stijgen.

Lean verschuift paradigma's. Een traditioneel paradigma is bijvoorbeeld dat kennis macht is. Een organisatie kan afhankelijk worden van een paar kritieke medewerkers die weten hoe processen werken. Het is beter om ervoor te zorgen dat alle medewerkers ontwikkeld en bekwaam zijn, in plaats van afhankelijk te worden van enkele medewerkers.

Alle Agile- en Scrum-principes en werkwijzen zijn gebaseerd op Lean-principes en -praktijken, waaronder het concept van zelfsturende teams, decentralisatie van beslissingsbevoegdheid, het creëren van een veilige omgeving waarin iedereen kan bijdragen en zijn of haar mening kan geven, waarin iedereen gedreven is om te leren met alle mogelijke middelen; leren van fouten, zonder straf, hoort daarbij!

4 Scrum en continue verbetering

Agile en Scrum gaan van nature over continue verbetering; dat is waarom een Agile-aanpak iteratief is.

Centraal in continue verbetermethoden, zoals de PDCA-cyclus van Deming, staan het idee van empirisme en het concept van constant en voortdurend leren en de verbetering die het gevolg is van leren. Deze behoren ook tot de centrale thema's van Lean en Agile.

Agile kan worden gezien als een manier om continue verbetering te realiseren, in plaats van als een manier om projecten uit te voeren.

Nieuwe vereisten van bedrijven, klanten of gebruikers stimuleren een nooit eindigend programma van continue verbetering en het creëren van meer waarde voor de belanghebbenden.

Verbetering wordt gedreven door wat op dit moment belangrijk is en de flexibiliteit om onmiddellijke problemen aan te pakken – en zelfs van richting te veranderen als dat nodig is!

Dit moet echter worden gedaan op een geplande en voorspelbare manier. Het gaat om het creëren van een cadans van verandering en het leveren van waarde in de organisatie en het in stand houden van deze cadans.

Het in stand houden van een cadans van positieve verandering, door gebruik te maken van Scrum als een methode voor continue verbetering, betekent dat Scrum voorspelbaarheid en stabiliteit creëert in de omgeving waarin het wordt gebruikt.

5 Grondbeginselen van Scrum

5.1 Een samenvatting van Scrum

Scrum is een framework op hoog niveau dat werd gecreëerd om organisaties te helpen complexe projecten incrementeel op te leveren. Omdat de bedoeling van de auteurs was dat organisaties hun eigen methoden zouden ontwikkelen, in lijn met wat het beste werkt, om de blanco's in te vullen, is Scrum opzettelijk op hoog niveau.

Stel dat we Scrum vergelijken met andere Agile-methoden, zoals SAFe, Disciplined Agile Delivery (DAD) en ABC/DSDM. De lichtgewicht structuur is evident, maar dit maakt Scrum niet minder bruikbaar. Integendeel, juist de vrijheid om een eigen Agile-methode te ontwikkelen, maakt Scrum tot een krachtig hulpmiddel.

Omdat Scrum een framework op hoog niveau is, is het niet voorschrijvend en niet gedetailleerd. Echter, de fundamentele concepten van Scrum zijn in de loop der jaren zorgvuldig ontworpen en getest en vormen nog steeds de basis van een succesvolle Scrum-implementatie. De beste manier om Scrum te gebruiken is *adopteren* en *aanpassen*, terwijl de basis in stand wordt gehouden.

Het beste is om te beginnen met de manier van leidinggeven en die een paar maanden te volgen voordat andere aanpassingen worden gemaakt. Het is niet altijd vanzelfsprekend waarom Scrum een bepaalde manier van doen voorstelt en het kan enige tijd duren voordat duidelijk wordt waarom Scrum een slimme zet is.

Scrum wordt vaak gedefinieerd als:

> Een framework waarbinnen mensen complexe, adaptieve problemen kunnen aanpakken, terwijl ze productief en creatief producten van de hoogst mogelijke waarde leveren. Scrum is: lichtgewicht, eenvoudig te begrijpen en moeilijk te beheersen.

Het laatste deel van de verklaring is interessant. Als Scrum zo lichtgewicht en eenduidig is, waarom is het dan zo moeilijk onder de knie te krijgen ('beheersen')?

Er zijn verschillende redenen, maar de volgende springen in het oog:

1. Omdat het niet de manier is waarop organisaties traditioneel worden geleid.
2. Omdat het niet de manier is waarop organisaties traditioneel zijn gestructureerd.
3. Omdat het berust op gedragingen van de betrokkenen die niet gebruikelijk zijn in traditionele organisaties.
4. Omdat er niet één manier is om Scrum te gebruiken; bewustzijn van de situatie is daarom van vitaal belang.

Kortom, Scrum is moeilijk onder de knie te krijgen, omdat het om een aanzienlijke verandering in de organisatie vraagt en deze veranderingen zijn in het begin moeilijk. Maar zodra een organisatie en de mensen daarbinnen de verandering hebben

omarmd en de voordelen zien voor de organisatie, de klanten en de werknemers, zal het onmogelijk worden om terug te keren naar de oude manier van werken.

De eerste grote organisatorische verschuiving is dat in Scrum al het werk wordt uitgevoerd door zelfsturende teams, wat een zeer ongebruikelijk concept is voor organisaties met traditionele managementstructuren.

In Scrum zijn er slechts drie eindverantwoordelijken en niet één daarvan is een traditionele managementrol.

Het tweede punt dat in het begin veel twijfel oproept, is dat er geen perfect gedetailleerd plan van het hele project is vóór de start, zodat er geen einddatum en Gantt-grafiek zijn om aan de muur te hangen.

In eerste instantie kan het management hier sceptisch tegenover staan, omdat het een blanco blad lijkt waarop niemand zich wil vastleggen.

Het probleem met de traditionele methode is dat zelfs als er een verbintenis aangegaan wordt en er een perfect plan uitgewerkt wordt, de realiteit nooit in overeenstemming is met het plan en beloften vaak niet kunnen worden nagekomen. We weten allemaal dat dit de realiteit is – we hoeven het alleen maar te erkennen.

Onvolmaakte plannen zijn niet het resultaat van een gebrek aan ijver van de kant van de projectdeelnemers. Maar de veronderstelling dat we de toekomstige staat van veranderingen in een complex adaptief systeem (CAS), zoals een organisatie, perfect kunnen plannen, is onjuist.

Scrum erkent dat het onmogelijk is om vooraf een perfecte planning te maken. In plaats van het onmogelijke proberen, biedt het een betere oplossing: voortdurend werken aan wat belangrijk is voor de business en dat uitvoeren binnen een breder (lichtgewicht) plan. Dit maakt het mogelijk om te sturen op het realiseren van waarde in plaats van op deadlines.

5.2 Een andere manier van werken

De basis van het Scrum framework en de bijbehorende aanpak is het gebruik van zelfsturende teams met slechts drie eenvoudige eindverantwoordelijkheden (rollen), een paar geplande gebeurtenissen en tools (artefacten) en eenduidige regels.

Deze fundamentele bouwstenen zijn essentiële elementen van Scrum en nodig om Scrum te laten werken. Hoewel organisaties die Scrum gebruiken veel vrijheid hebben om de manier waarop ze dingen doen aan te passen en te veranderen, heeft de ervaring geleerd dat alles kapotgaat als er geknoeid wordt met deze fundamentele componenten.

Het is daarom raadzaam om vast te houden aan de basiselementen van het framework en de bijbehorende 'regels'. Elk onderdeel in Scrum dient een specifiek doel en is essentieel voor het succes van het framework.

Scrum werd ontwikkeld voor het managen van softwareprojecten, maar veel organisaties gebruiken het net zo effectief voor product- en serviceprojecten in een andere context.

Als iteratieve waardelevering en incrementele kennisoverdracht aan de orde van de dag moeten zijn, dan is Scrum de juiste koers.

Binnen het Scrum vocabulaire, wordt het team dat het werk doet de Developers genoemd. De term Developers is één van de drie Scrum eindverantwoordelijkheden, maar dit betekent niet dat het team enkel bestaat uit Developers. In deze bredere context moet dit eerder gezien worden als het team dat ontwikkelt, bouwt en oplevert wat opgeleverd wordt in die specifieke sprint of iteratie. Het Scrum-team omvat de Scrum Master, de Product Owner en alle Developers.

Het maakt niet uit of de organisatie een softwareontwikkelingsbedrijf is, industriële producten maakt, diensten verleent, of zelfs een school is. Leren is iteratief en incrementeel – daarom wordt Scrum in vooruitstrevende scholen als de primaire onderwijsmethode gebruikt. Tenslotte gaat het bij Scrum om kleine, autonome en crossfunctionele teams die voortdurend waarde creëren. In een bredere context verliezen deze teams hun autonomie niet om grote teams te worden, maar in plaats daarvan netwerken ze, werken ze samen en interopereren ze, terwijl ze elk een autonoom team blijven. Daarom heeft Scrum een significante impact op traditionele hiërarchische organisaties die dit framework aannemen.

5.3 Een beetje essentiële theorie achter Scrum

Let op: Dit boek is geschreven als een gids voor verdere Scrum-studie en veronderstelt kennis van de basis van Scrum. Is dat niet het geval, lees dan het *Agile Scrum Handboek* (Rad, 2021).

De bedoeling van dit boek is om te helpen een betere Scrum Master of Product Owner te worden. In dit boek wordt aandacht besteed aan de verantwoordelijkheden van en de praktijken gebruikt door Scrum Masters en Product Owners.

Hou in gedachten dat Scrum Masters op de hoogte moeten zijn van het Product Ownerschap en vice versa.

De ideeën achter Scrum zijn niet nieuw en kunnen worden teruggevoerd op empirische procesbeheersing en de beginselen en praktijken die in Lean worden toegepast.

Het idee is dat kennis en ervaring ontwikkeld wordt door dingen te doen en beslissingen te nemen op basis van kennis – niet op basis van veronderstellingen. Het betekent ook dat we soms iets niet weten en dat de enige manier om het antwoord te vinden een empirische benadering is.

In de wetenschap is de basis van een empirische benadering dat een hypothese wordt ontwikkeld en vervolgens een test wordt gemaakt om deze hypothese te testen. Na het testen van de hypothese, wordt de hypothese bewezen of weerlegd – hoe dan ook, er is meer duidelijk dat ervoor.

Dit is één van de grote voordelen van het gebruik van Scrum in tegenstelling tot Waterval-methoden, waar alle projectplanning vooraf wordt gedaan en waar het grootste deel van deze planning gebaseerd is op veronderstellingen, die vaker onjuist dan juist blijken te zijn.

In Scrum is het geen geheim dat iets niet bekend is; dit wordt erkend en er wordt geprobeerd zo snel mogelijk een antwoord te vinden met een empirische aanpak.

Dit is één van de drie pijlers van empirische procesbeheersing: **transparantie**. De experimenten en de lessen die geleerd worden wanneer er geprobeerd wordt te begrijpen wat er moet worden gedaan, zijn de overige twee: **inspectie** en **aanpassing**.

5.4 De drie pijlers van Scrum

Dus misschien is het tijd om nog eens naar de drie pijlers te kijken.

- transparantie
- inspectie
- aanpassing

Transparantie
Transparantie betekent dat de processen waarin het werk wordt gedaan en wat we weten over het werk dat moet worden gedaan, zichtbaar zijn voor de teamleden, aangezien zij eindverantwoordelijk zijn voor het verrichte werk en de resultaten ervan.

Het wordt voor iedereen veel begrijpelijker als gemeenschappelijke standaarden, taal, methoden, tools, maatstaven en een gemeenschappelijke woordenschat worden gebruikt. Werk wordt zichtbaar gemaakt, zodat iedereen het te verrichten werk kan zien en begrijpen.

Inspectie
Inspectie betekent dat voortdurend gecontroleerd wordt of wat er gedaan wordt werkt en of de veronderstellingen (de hypothese) juist en geldig zijn. Wanneer het werk is gedaan, worden de manier van werken en de bijbehorende Scrum artefacten regelmatig geïnspecteerd en gevalideerd, of veranderd, vastgelegd of verbeterd, wat de laatste van de drie pijlers is.

Aanpassing
Aanpassing betekent dat als een element van het proces, of het werk, afwijkt van de gestelde norm, het proces, of het werk dat wordt gedaan, wordt aangepast om ervoor te zorgen dat snel wordt teruggegaan naar de gestelde norm en dat kwaliteit wordt geleverd.

5.4.1 Scrum-gebeurtenissen

Scrum kent slechts vijf gebeurtenissen of rituelen en elk daarvan biedt een gelegenheid voor inspectie en aanpassing.

Dat zijn:

- sprint
- sprint planning
- daily scrum
- sprint review
- sprint retrospective

En hoewel verfijning van de product backlog niet is gedefinieerd als een gebeurtenis, is het wel een belangrijke activiteit.

Veel organisaties onderschatten de culturele verandering die nodig is om Scrum, of andere Agile-methoden of frameworks, in te voeren. Er zijn belangrijke culturele implicaties.

We weten allemaal dat een organisatiecultuur niet van de ene dag op de andere kan worden veranderd, maar om Scrum en Agile succesvol te laten zijn, moeten organisaties een reeks waarden onderschrijven die het toepassen van Scrum onderbouwen.

Hoogstwaarschijnlijk falen de meeste organisaties die hun doel opgeven om meer Agile te worden en Agile-methoden te gebruiken, in dit opzicht.

Vergeet niet dat de kern van Scrum empirisme en Lean thinking is. Het is gebaseerd op de overtuiging dat kennis voortkomt uit ervaring en dat beslissingen worden genomen op basis van wat er wordt waargenomen en wat er wordt geleerd. Lean thinking helpt om simpeler over de wereld na te denken, om te richten op wat werkt en om dingen eenvoudig te houden. De aanpak van Lean om verspilling terug te brengen is niet alleen gemaakt voor middelen die worden gebruikt, maar ook voor het proces van het bouwen van nieuwe en waardevolle producten en diensten.

Omdat Scrum, Lean, en andere Agile-methoden om effectief te werken afhankelijk zijn van een set van waarden, is het verstandig om onszelf af te vragen: “Zijn deze verenigbaar met de huidige waarden en cultuur van de organisatie?” Hoe verder weg deze waarden zijn, hoe moeizamer de overgang naar en de adoptie van Scrum of Agile zal zijn voor de organisatie.

Maar wat zijn deze waarden eigenlijk?

5.5 Agile Scrum-waarden

Scrum-teams opereren als zelfsturende, autonome eenheden die verantwoordelijk en aansprakelijk zijn voor wat het team doet. Wanneer het team faalt of fouten maakt, moeten ze, om te leren en te groeien, hiervan leren en in staat zijn om de fout te herstellen zonder inmenging van buitenaf, zolang ze maar waarde leveren aan klanten. Het leveren van waarde betekent anderzijds dat het team de eisen van de klant zal moeten interpreteren en valideren en ervoor moet zorgen dat deze en de bijbehorende resultaten worden bereikt door de eisen op te splitsen in de taken die moeten worden uitgevoerd om de waarde te leveren.

Eén van de kernpunten hier is dat het team zich veilig voelt en weet dat, zolang ze van hun fouten leren, er geen negatieve consequenties zullen volgen. Negatieve consequenties zijn waarschijnlijk de belangrijkste oorzaak dat Scrum en Agile falen in organisaties.

De Scrum waarden zijn:

- betrokkenheid (commitment)
- moed (courage)
- focus (focus)
- openheid (openness)
- respect (respect)

Als deze waarden niet nageleefd en gerealiseerd worden, zullen de drie pijlers van Scrum in elkaar storten

Bovendien moet er geloofd worden en erop vertrouwd worden dat alle betrokkenen, zowel binnen de organisatie als onder de klanten, in deze waarden geloven en ze ondersteunen.

In wezen leven de teamleden niet alleen de waarden na en geloven ze erin; de teamleden zorgen er ook voor dat de waarden binnen de organisatie op elkaar zijn afgestemd.

Laten we eens kijken wat de waarden in de praktijk betekenen.

Betrokkenheid
Betrokkenheid betekent dat elk teamlid zich inzet voor het team en ook voor het werk dat zal worden verricht.

Moed
Teamleden durven hun stem te laten horen en de andere teamleden luisteren. Het team pakt problemen zo snel mogelijk aan om hun beloften te kunnen nakomen.

Focus
De teamleden richten zich individueel en collectief op het klaren van de klus, zodat zij kunnen leveren wat er is beloofd, binnen de randvoorwaarden.

Openheid
Iedereen maakt fouten en niet iedereen kan alles doen. Teamleden zijn open en als ze een fout hebben gemaakt geven ze dat toe; als ze een belemmering tegenkomen, of niet weten hoe ze een opgedragen taak moeten uitvoeren, zijn ze daar ook open over. Het verbergen van fouten heeft een domino-effect dat het team niet kan toestaan. Om dit mogelijk te maken, vertrouwen zij erop dat ze niet zullen worden gestraft, omdat zij hun mond hebben opengedaan. Helaas zijn negatieve consequenties nog steeds de norm in veel organisaties.

Respect
Teamleden behandelen elkaar met respect en helpen elkaar waar en wanneer zij kunnen.

5.6 Een samenvatting van Scrum-verantwoordelijkheden

5.6.1 Het Scrum-team

De fundamentele eenheid van Scrum is een klein team dat het Scrum-team wordt genoemd. Het Scrum-team bestaat uit een Scrum Master, een Product Owner en Developers. Binnen een Scrum-team zijn er geen sub-teams of hiërarchieën. Het is een samenhangende eenheid van professionals gericht op één doel: het product goal (doel van het product).

Scrum-teams zijn zelfsturend en crossfunctioneel; ze kunnen dus zonder inmenging van buiten het team besluiten en kiezen hoe ze hun werk het beste kunnen doen. Omdat de Product Owner de klant en de business vertegenwoordigt, zijn de prioriteiten van de business via de Product Owner bekend bij het Scrum-team.

Scrum-teams bestaan idealiter uit crossfunctionele Developers. De leden van het team zouden alle benodigde vaardigheden moeten hebben om het gekozen deel van het werk te voltooien binnen een gebeurtenis in een timebox die een sprint wordt genoemd. Het team zou redelijk onafhankelijk moeten zijn van vaardigheden buiten het team. Dit betekent dat het inzetten van externe Developers zo veel mogelijk wordt vermeden; maar op sommige momenten, als een specifieke vaardigheid mist, kan een externe Developer voor die gelegenheid worden gebruikt. Daarnaast maakt het werken in een klein team het gemakkelijker om te leven volgens de Scrum waarden. Grotere teams hebben de neiging meer defecten te creëren dan kleine teams.

Het Scrum-team moet klein genoeg zijn om agile te blijven en groot genoeg om een significante hoeveelheid werk binnen een sprint te voltooien. Een Scrum-team van 10 mensen of minder wordt aanbevolen.

Waarom zo klein? Het is bewezen dat kleinere teams beter communiceren, dat de nadelen van overspecialisatie minder snel optreden en dat kleine teams productiever zijn.

Als teams te groot worden, kan overwogen worden om het grote team te reorganiseren in meerdere samenhangende Scrum-teams, allemaal gericht op hetzelfde product. Omdat de teams zich richten op hetzelfde product, kunnen ze hetzelfde product goal, dezelfde product backlog en dezelfde Product Owner hebben.

Scrum-teams zijn verantwoordelijk voor alle productgerelateerde activiteiten. Dit omvat met name de samenwerking met belanghebbenden, de verificatie van de vereisten, de voortgang van teamactiviteiten en teampraktijken, terwijl voortdurend wordt geëxperimenteerd met hoe de werkwijze kan worden verbeterd. Om dit te bereiken investeert het team tijd in onderzoek en ontwikkeling en al het andere dat nodig kan zijn om het team en de manier van werken te verbeteren. Scrum-teams worden door de organisatie gemachtigd om hun werk te beheren. Het team heeft echter breed overleg zodat ze begrijpen wat er moet gebeuren en wat de waarde is

voor klanten en gebruikers. Deze overeenstemming is de basis voor het definiëren van wat de definition of done (DoD) wordt genoemd, een kritieke maat voor de prestaties van het Scrum-team.

Het hele Scrum-team is verantwoordelijk voor het creëren van een waardevol, bruikbaar increment in elke sprint.

Een sprint is een gebeurtenis die één tot vier weken in beslag neemt en waarin het Scrum-team één of meer incrementen van waarde ontwikkelt.

Een bruikbaar increment betekent iets dat kan worden ingezet en gebruikt door belanghebbenden en dat voor hen van waarde is.

Let wel dat een sprint meerdere incrementen van bruikbare waarde kan omvatten.

Scrum definieert drie specifieke verantwoordelijkheden binnen het Scrum-team:

- de Developers
- de Product Owner
- de Scrum Master

Later in het boek zal in meer detail op effectieve teams worden ingegaan, het ontwerp van een team en hoe een team precies moet worden samengesteld. De primaire methode om ervoor te zorgen dat een Scrum-team consistent werkt, in een duurzaam tempo, is het gebruiken van sprints. Sprints zijn daarom een fundamentele bouwsteen van Scrum.

Hier zijn enkele beginselen voor een goed Scrum-team:

- **Scrum-teams zijn zelfsturend**. Niemand vertelt het Scrum-team hoe ze product backlog items (vereisten) omzetten in incrementen die waarde vertegenwoordigen (producten/diensten).
- **Het Scrum-team is crossfunctioneel**, met alle vaardigheden die een team nodig kan hebben om een productincrement te maken. Dit is het ideaal, maar is niet altijd mogelijk.
- **Scrum kent geen titels of posities**, ongeacht het werk dat door de persoon wordt uitgevoerd. De Scrum Master, de Product Owner en de Developers zijn niet zo strikt als traditionele 'rollen', maar eerder eindverantwoordelijkheden. Het hebben van een bepaalde 'rol' of ‘eindverantwoordelijkheid’ op een bepaald moment, betekent dat iemand ervoor zorgt dat bepaalde dingen worden gedaan.
- **Scrum kent geen sub-teams**, ongeacht de domeinen in het werk, zoals testen, architectuur, de operatie, of bedrijfsanalyse.
- Individuele Scrum-teamleden kunnen gespecialiseerde vaardigheden en aandachtsgebieden hebben, maar **de eindverantwoordelijkheid ligt bij het Scrum-team** als geheel.
- De verantwoordelijkheden van de **Scrum Master en de Product Owner kunnen niet gecombineerd worden in één enkele persoon**. Is dit wel het geval, dan is scheiding van taken belangrijk om conflicterende verantwoordelijkheden te vermijden.

5.6.2 Developers

Developers zetten zich in om in elke sprint elk aspect van een bruikbaar increment te maken en beschikken over de specifieke vaardigheden die nodig zijn om de incrementen te ontwikkelen.

De vaardigheden, capaciteiten en zelfs de vaardigheidsniveaus van Developers lopen vaak sterk uiteen en zullen in de eerste plaats worden bepaald door het werkdomein dat in de sprint aan de orde is.

Developers zijn altijd verantwoordelijk voor:

- het maken van het sprint plan en de sprint backlog;
- het bevorderen van kwaliteit door vast te houden aan een definition of done (DoD);
- het aanpassen van het plan en de manier van werken waar en wanneer nodig, om ervoor te zorgen dat er vooruitgang wordt geboekt in de richting van het sprintdoel;
- het elkaar verantwoordelijk houden, als professionals.

5.6.3 De Product Owner

De Product Owner is een enkele persoon, geen commissie, die verantwoordelijk is voor het maximaliseren van de waarde van het product door samen te werken met Developers en de Scrum Master.

Product Owner zijn is bij voorkeur een fulltimebaan; er moet aandacht aan worden besteed om multitasking of potentiële knelpunten in de projecten te voorkomen. Product Owners geven de behoeften van alle belanghebbenden in de product backlog weer en ze worden getraind in hoe ze de eindverantwoordelijkheden van een Product Owner vervullen. Als iemand de product backlog wil veranderen kan dit gedaan worden door de Product Owner te overtuigen van de voordelen van de wijzigingen.

- De Product Owner is eindverantwoordelijk voor het creëren, beheren en onderhouden van de product backlog, het beheren van het product budget en het coördineren van de productlancering.
- **De Product Owner ontwikkelt en communiceert het doel van het product (product goal) en stelt een duidelijke product backlog vast**, die bestaat uit vereisten die ervoor zorgen dat het product goal wordt bereikt. Vereisten worden gedefinieerd als product backlog items.
- Product Owners ordenen de product backlog items in overeenstemming met de doelen van de organisatie.
    - Let op: het ordenen van de backlog wordt grotendeels bepaald door de business prioriteit, maar dat is niet het enige criterium.
- **Product Owners wonen Scrum-bijenkomsten bij waar ze optreden als de stem van de klant (voice of the customer, VoC),** de vereisten uitleggen en de Developers helpen om de product backlog items en hun volgorde in de product backlog beter te begrijpen.

- **Product Owners coördineren verschillende Scrum-teams** om ervoor te zorgen dat werk niet dubbel wordt gedaan, dat afhankelijkheden zichtbaar en duidelijk worden gemaakt voor iedereen en dat werk en afhankelijkheden op elkaar worden afgestemd en goed georkestreerd.
- **Product Owners zorgen ervoor dat de product backlog wordt verfijnd (product backlog verfijning) en dat dit regelmatig gebeurt**. Het verfijnen van de backlog omvat het wijzigen van de volgorde, het uitsplitsen van de vereisten naar het juiste niveau van detail en het krijgen van feedback en hulp van de Developers om dit te doen.

Gezien het belang van deze eindverantwoordelijkheid, mag Product Ownerschap niet worden toegewezen aan iemand die geen of weinig gezag heeft en die niet wordt gerespecteerd door klanten.

De Product Owner is eindverantwoordelijk, maar kan verantwoordelijkheid delegeren aan anderen in het team.

Hoewel dit niet expliciet wordt vermeld, is het raadzaam Product Ownerschap in de onderneming (strategisch, tactisch en budgettair) te combineren tot één eindverantwoordelijkheid.

De Product Owner is niet alleen de vertegenwoordiger van de business, maar ook een businessvertegenwoordiger. Dit betekent dat ze uit de business/klant kant van de organisatie geselecteerd worden, niet uit de project/technische kant. De Scrum Guide (Scrum.org, 2020) stelt specifiek dat Product Owners alleen kunnen slagen als de hele organisatie hun beslissingen respecteert.

De beslissingen van de Product Owner worden zichtbaar gemaakt in de inhoud en ordening van de product backlog en door middel van een increment dat kan worden geïnspecteerd en herzien tijdens een sprint review.

Als de Product Owner een beslissing neemt, is dat hetzelfde als wanneer de business die beslissing heeft genomen. Product Owners met dit niveau van autoriteit en zeggenschap blijken het meest effectief te zijn.

5.6.3.1 Product Owner als de stem van de klant

Stem van de klant (voice of the customer, VoC) is een term die in Lean wordt gebruikt om de verwachtingen, voorkeuren en behoeften van de klant te beschrijven.

Als vertegenwoordiger van de klant heeft de Product Owner een diepgaand begrip van de wensen en behoeften van klanten en gebruikers.

We kunnen dus stellen dat de Product Owner de stem is van de klant en van de gebruiker.

Dit gezegd hebbende, Developers gaan nog steeds in gesprek met gebruikers en klanten tijdens een iteratie, om de vereisten en behoeften beter te begrijpen en te verfijnen. De Product Owner mag nooit een poortwachter worden die end-to-end communicatie belemmert.

Als vertegenwoordiger van de klant leeft het Scrum-team de volgende **agile-principes** na:

1. Onze hoogste prioriteit is de klant tevreden te stellen door vroegtijdige en continue levering van waardevolle software.
    a. Let op: in een andere context kan het om een ander product of andere dienst gaan.
2. Verwelkom veranderende eisen, zelfs laat in de ontwikkeling. Agile-processen maken gebruik van verandering om de klant concurrentievoordeel te geven.
3. Mensen van de business en Developers werken gedurende het hele project dagelijks samen.
4. Eenvoud (de vaardigheid om de hoeveelheid werk te minimaliseren en tegelijkertijd de waarde te maximaliseren) is een essentiële Agile-vaardigheid.

5.6.3.2 Eigenschappen van goede Product Owners

Goede Product Owners hebben allemaal dezelfde soort eigenschappen. Hier zijn een aantal kenmerken van een effectieve Product Owner.

1. Product Owners zijn **sterk in communiceren**, ze kunnen goed communiceren met elk publiek. Ze stemmen hun taalgebruik en hun communicatiemethoden af op het publiek. Maar bovenal zijn goede Product Owners goede luisteraars; ze luisteren, observeren, stellen onderzoekende vragen en reflecteren op wat ze hebben geleerd.
2. Product Owners hebben een **grondige kennis van de business**, zodat ze de behoeften van de business kunnen vertalen naar acties. Ze komen uit de business en niet uit het project. Daarmee hebben ze het juiste niveau van inzicht en zijn ze in staat complexe beslissingen te nemen en compromissen te sluiten wanneer de behoeften van de belanghebbenden niet duidelijk zijn, of erger nog, met elkaar in conflict zijn.
3. Product Owners zijn senior genoeg om het **respect van de business** te krijgen wanneer zij beslissingen nemen namens de business.
4. Ze kunnen ook **onderscheid maken tussen wat nodig is en wat gewenst wordt.** Belanghebbenden zullen veel eisen bij de Product Owner neerleggen als ze de kans krijgen. Product Owners zijn in staat onderscheid te maken tussen wat er nodig is (must-haves) en wat aardig is om te hebben (nice-to-haves). Hier kan de MoSCoW techniek[5] bij helpen.
5. **Oplossingsgericht**: Veel eisen of behoeften van de business worden uitgedrukt in negatieve termen (problemen) en Product Owners hebben het vermogen een discussie te faciliteren die zich niet richt op wat er mis is, maar op hoe het probleem kan worden opgelost.
6. **Resultaatgericht**: het eindproduct is niet essentieel; essentieel zijn de *uitkomsten* die worden bereikt (wat de gebruiker, organisatie of klant zal kunnen doen nadat een increment is opgeleverd).

[5] Voor een volledige uitleg, zie het stuk over MoSCoW.

Enige andere vaardigheden en eigenschappen dragen bij aan wat een effectieve Product Owner is, maar de bovenstaande zes zijn absoluut essentieel.

5.6.4 De Scrum Master

De eindverantwoordelijkheid van een Scrum Master lijkt niet op die van een traditionele projectmanager.

Een Scrum Master lijkt veel meer op een Scrum-half in rugby, waar de term Scrum ook vandaan komt. Het is hun taak om ervoor te zorgen dat de Scrum de bal krijgt, die vervolgens de taak heeft om de bal naar de backline te brengen (waarde creëren voor het bedrijf), die daarna de bal gebruikt om een try (doelpunt) te scoren. Als je niet vertrouwd bent met het rugbyspel, is het niet nodig om het te leren om vertrouwd te raken met Scrum, dus maak je geen zorgen over deze analogie.

Scrum Masters helpen met het coördineren van het werk, organiseren bijeenkomsten, helpen teamgenoten moeilijkheden te overwinnen en laten de vooruitgang zien met visuele tools, zodat iedereen precies weet hoeveel vooruitgang er is geboekt, welke problemen er nog zijn en hoe teamgenoten elkaar kunnen helpen wanneer dat nodig is.

Scrum Masters zijn verantwoordelijk voor het promoten en ondersteunen van Scrum als een methode en proces. Zij bereiken dit door iedereen te helpen de Scrum theorie en praktijk te begrijpen. Er kan gesteld worden dat een Scrum Master eindverantwoordelijk is als:

- trainer
- facilitator
- coach

Als onderdeel van de verantwoordelijkheid als coach en facilitator wordt niet alleen aandacht besteed aan het technische aspect van Scrum; het menselijke aspect, vooral in het begin, is ook belangrijk. De Scrum Master treedt dan op als de drijvende kracht van de verandering, creëert een samenhangend team en pakt destructieve teamdynamieken aan.

Scrum Masters zijn leiders die het Scrum-team dienen (servant leaders). Zij doen dit door:

- Ervoor te zorgen dat doelen, scope en producten/diensten worden begrepen.
- Door manieren te vinden om het beheer van de product backlog te verbeteren.
    - Ze beheren de product backlog niet daadwerkelijk, maar komen met suggesties om het beter te doen.
- Scrum Masters helpen de Developers om **product backlog items**, hun volgorde, prioriteiten en waarde **te begrijpen**.
    - Hoewel dit de primaire verantwoordelijkheid is van de Product Owner, assisteert de Scrum Master hierin.

- Scrum Masters helpen de teamleden **Scrum en alle aspecten ervan te begrijpen**. Ze helpen Scrum zo goed mogelijk te gebruiken en aan te passen aan de context van het team en de organisatie.
- Scrum Masters zijn facilitators van Scrum gebeurtenissen.
- Wat vaak over het hoofd wordt gezien, is dat de Scrum Master **het team beschermt tegen onderbrekingen en afleidingen van buitenaf. Ze nemen belemmeringen weg** om de teamprestaties te verhogen en te ondersteunen en maken communicatie mogelijk.
    - Scrum Masters creëren ook een omgeving die optimaal is om als team te werken binnen de context van wat het team te doen heeft.

De Scrum Master is bij voorkeur niet tegelijkertijd een Developer. Deze praktijk leidt tot een gebrek aan aandacht voor de eindverantwoordelijkheid van de Scrum Master en kan leiden tot belemmeringen.

In andere Agile-methoden kan een Scrum Master een projectmanager heten (wat geen goed idee is), een iteratiemanager, een Agile coach of een teamcoach.

Als coach helpen Scrum Masters, op grond van hun ervaring met Scrum, het team uit te zoeken wat de beste manier is om Agile-principes en Scrum-methoden en -technieken toe te passen binnen hun specifieke context en situatie.

Wees voorzichtig met het aanstellen van bestaande projectmanagers zonder de juiste Scrum Master training. Als de typische commando- en controlestructuur, die gewoonlijk in Waterval-projecten wordt gebruikt, toegepast wordt, dan zal Scrum falen, spectaculair falen.

Merk ook op dat het gebruik maken van voormalige tech-leiders als Scrum Masters nadelige effecten kan hebben, omdat zij, op basis van hun ervaring in hun rol als teamleider, gewend waren gestructureerde aanwijzingen te geven over wat er gedaan moet worden; de Scrum Master neemt echter geen beslissingen voor de Developers. Als afgewogen wordt wie er als Scrum Master aangesteld kan worden, is het raadzaam de voorkeur te geven aan mensen die hiervoor getraind zijn en die zich vrijwillig melden om Scrum Master te zijn. Vooral als er net begonnen wordt met Agile Scrum, is het goed een competente coach te betrekken: iemand met veel ervaring om de zaken op gang te krijgen. Agile Scrum is niet alleen een nieuwe werkwijze, maar ook een radicale breuk met de traditionele manier van werken in de meeste organisaties.

Een team krijgt met veel belemmeringen te maken en een ervaren coach kan uiteenzetten wat succesvol gaat zijn, wat niet werkt, welke lessen er te leren zijn en wat er ten koste van alles vermeden moet worden.

Ervaren coaches zijn in de eerste plaats meesters in organisatieverander-management en in de tweede plaats Agile Scrum experts. Maar ze moeten dus wel ook Agile Scrum experts zijn –dat maakt hen superexperts in het helpen van de organisatie tijdens de moeilijke overgangsperiode.

5.6.4.1 Eigenschappen van goede Scrum Masters

We hebben het al gehad over de verantwoordelijkheid van de Scrum Master en de impliciete vaardigheden die nodig zijn voor een goede Scrum Master.

Hier zijn enkele van de kenmerken van en vereisten voor een effectieve Scrum Master. Sommige van deze kenmerken zijn ‘zachter’ dan andere; laten we echter beginnen met enkele van de inhoudelijke vaardigheden die Scrum Masters nodig hebben.

Scrum Masters zijn coaches en trainers, wat impliceert dat zij een gedegen kennis van Scrum en Agile theorie hebben.

- **Scrum Masters moeten voortdurend bijleren** en hun technische en theoretische repertoire voortdurend uitbreiden. Dit omvat niet alleen de theorie en de praktijk van Scrum en Agile, maar ook de nieuwste tools en technieken die door andere teams succesvol worden gebruikt en aantonen wanneer en hoe de organisatie of hun team daar voordeel mee kan doen.
- **Scrum Masters zijn organisatoren maar geen supervisors**; zij bouwen en beheren het systeem dat door het team wordt gebruikt om te itereren en doen dit op een inclusieve en consultatieve manier. Dit betekent dat de Scrum Master een georganiseerd persoon is, die het hoofd koel houdt wanneer alles rondom hem of haar verkeerd loopt.
- **Scrum Masters kennen de gereedschappen die het team gebruikt goed**, omdat zij meestal verantwoordelijk zijn voor het onderhouden en bijhouden van tools en gegevens, zoals het Scrum-bord en burn-down charts.
- **Scrum Masters zijn goede trainers en coaches**. Een goede Scrum Master weet niet alleen wat te doen, maar is ook in staat uit te leggen hoe en waarom dingen op een specifieke manier gedaan worden in Scrum, aan iedereen die erbij betrokken is.
- **De Scrum Master zorgt er in eerste instantie voor dat het team Scrum beheerst**; deze vaardigheid is een absolute must. Hoewel de Scrum Master deel uitmaakt van het team, moet deze ook teamleden coachen en ze aanmoedigen om beter te worden en samen te werken. Niemand zal de sterktes en zwaktes van elk teamlid beter kennen dan de Scrum Master en zij of hij is in staat individuen te helpen maximaal bij te dragen aan het team.
- Hoewel de Scrum Master niet technisch hoeft te zijn, **helpt het als hij of zij technische kernvaardigheden heeft** of op zijn minst een goed begrip van de technische context waarin het team functioneert. Dit betekent ook: een goed begrip hebben van de omgeving, de sleutelrolspelers, wat te doen, of met wie er gepraat moet worden als dingen fout gaan.

Het is relatief eenvoudig om van tevoren na te gaan welke Scrum Masters de nodige inhoudelijke vaardigheden hebben om zowel de rol van Scrum Master te vervullen als de Scrum Master voor een specifiek team te zijn.

Andere vaardigheden zijn niet zo gemakkelijk vooraf te controleren.

- **Goede Scrum Masters zijn snelle leerlingen** en pakken in hoog tempo de nodige soft skills op om in een specifieke omgeving te functioneren.

- Wanneer mensen als een team werken, zullen er conflicten zijn en **Scrum Masters zijn in staat conflicten op te lossen** en het team te laten werken als een eenheid, in plaats van als individuen.
- Zoals eerder vermeld, zijn **Scrum Masters leiders die dienen** (servant leaders). Leiders geven het goede voorbeeld en zijn bereid om bij te dragen en hun deel van het werk te doen. Dienend leiderschap (servant leadership) stelt de behoeften van het hele team boven die van het individu en helpt anderen naar hun beste kunnen te presteren.

Samengevat, een goede Scrum Master is:

- verantwoordelijk
- bescheiden
- collaboratief
- toegewijd
- invloedrijk
- deskundig

5.7 Overzicht van Scrum-gebeurtenissen

Praten over Scrum is altijd een kip-en-ei-vraag. Als we het hebben over hoe dingen werken, zijn er vragen over rollen of verantwoordelijkheden. Als we het eerst hebben over verantwoordelijkheden, zullen er zeker vragen komen over hoe dingen werken.

Het is raadzaam om eerst over verantwoordelijkheden te praten voordat er ingegaan wordt op hoe de dingen werken, want dit is misschien het minste van twee kwaden.

Meestal worden de geplande activiteiten van Scrum **gebeurtenissen** genoemd en de gebruikte of gegenereerde middelen worden soms **artefacten** genoemd.

De vereisten van belanghebbenden worden verzameld en vastgelegd als user story’s, die één of meer product backlogs vullen. Deze zullen later in meer detail worden beschreven.

Elk product heeft een eigen product backlog en een Product Owner die eindverantwoordelijk is voor het maximaliseren van de waarde van het product als resultaat van het werk van het Scrum-team. De Product Owner is geen projectmanager, maar bij voorkeur de business owner van het betreffende product of de betreffende product portfolio.

Story’s in de product backlog worden geordend door de Product Owner, die met andere leden van het Scrum-team zal samenwerken om de story’s in de product backlog binnen de kortst mogelijke tijd op te leveren, op basis van hun prioriteit. Dit zal continu worden gedaan totdat alle verzamelde eisen zijn verwerkt.

De oplevering tegen de vereisten gebeurt in korte, kleine, projectachtige, iteratief geplande gebeurtenissen, die **sprints** genoemd worden.

Een Scrum-team zal overleggen met hun Product Owner en vereisten selecteren die ze kunnen voltooien binnen een bepaalde tijd (sprint). De vastgestelde tijdslimiet is

meestal hetzelfde voor alle sprints. Sprints zijn **time-boxed**, oftewel gebeurtenissen met een vast tijdsbestek, meestal tussen de één en vier weken lang.

Sommige organisaties stellen een vaste tijd vast voor alle sprints, bijvoorbeeld vier weken, terwijl andere organisaties het goed vinden dat ze variëren, afhankelijk van wat er gedaan moet worden. Over het algemeen wordt de lengte van sprints niet voorgeschreven, hoewel het, vooral in het begin, waardevol is om voor sprints een vaste lengte te bepalen.

Als startpunt is het aan te raden om met vaste sprints van vier weken te werken. Het gebruik van time-boxes, bij het plannen en uitvoeren van werk, zorgt voor meer voorspelbaarheid, wat het gemakkelijker maakt om beter te begrijpen wat de sprint gaat opleveren aan product of functionaliteit en wanneer deze worden vrijgegeven.

Als terugkerende problemen verhinderen dat een sprint met een time-box korter dan vier weken afgemaakt kan worden, kan de time-box verlengd worden in overleg met de Product Owner, op voorwaarde dat deze niet langer wordt dan vier weken.

Het Scrum-team functioneert autonoom. Zij kiezen en plannen hun eigen werk en zijn gezamenlijk eindverantwoordelijk voor wat ze opleveren. Als de sprint mislukt, faalt iedereen samen.

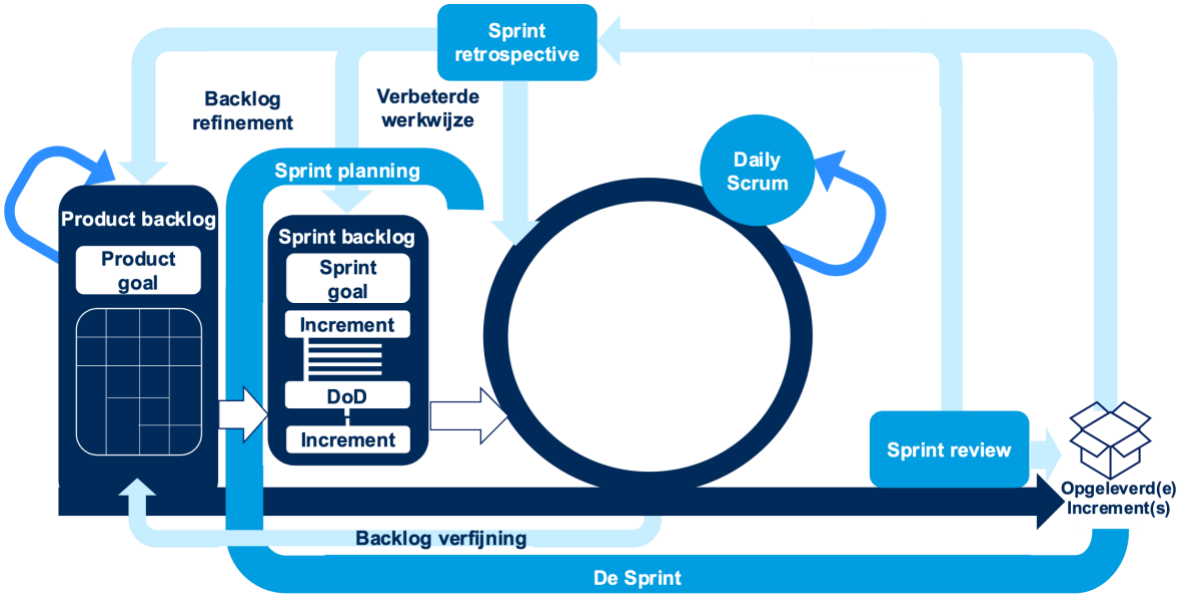
Er is eerder aangegeven dat vereisten in de product backlog worden gedefinieerd als user story’s. Hiermee zijn ook de uitkomsten gedefinieerd die door het team worden bereikt. Meer details over user story’s volgen later.

Het Scrum-team deelt de vereisten op in kleinere vereisten die tijdens een sprint kunnen worden opgeleverd. Het Scrum-team deelt dan, als onderdeel van de sprint planning, de kleinere story’s op in taken. Daaruit kunnen de teamleden de meest geschikte taken voor actie selecteren.

Als onderdeel van de sprint planning, wordt een definition of done (DoD) vastgesteld voor elk increment dat wordt opgeleverd in de sprint. Definitions of done dienen als borging, om te controleren of wat er gepland en toegezegd was, ook gedaan en opgeleverd is. Let wel, een sprint levert ten minste één increment op.

Sprint planning is ook time-boxed. Voor een sprint van vier weken bijvoorbeeld, mag de sprint planning niet meer dan acht uur in beslag nemen.

Figuur 3 Een algemeen overzicht van alles in Scrum



Afbeelding gemaakt door EXIN op basis van: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC - Kdp Print Us.

Het team volgt de vooruitgang tot de oplevering. De primaire gebeurtenis die hiervoor wordt gebruikt, heet de **daily scrum**. Tijdens deze time-boxed gebeurtenis die niet langer duurt dan 15 minuten, rapporteren teamleden over de voortgang, bespreken ze moeilijkheden en informeren ze elkaar over het deel van het werk waar ze mee bezig zijn. Soms maken afhankelijkheden het nodig dat het werk tussen teamleden zorgvuldig wordt gecoördineerd in de daily scrum of in superstructuren. Een dergelijke superstructuur, Nexus genaamd, zal later in detail worden behandeld.

Aan het einde van de sprint, demonstreert het team het product of de functionaliteit die potentieel bruikbaar en inzetbaar is voor klanten of gebruikers. Functionele of implementeerbare producten of functies worden ook wel verzendklaar (shippable product) genoemd als het product de volgende tests heeft doorstaan: issues, integratie, prestaties en bruikbaarheid. Deze gebeurtenis heet de **sprint review** en voor een sprint van vier weken zal deze niet langer duren dan 4 uur.

Let wel: Een product dat klaar is om op de markt te worden gebracht, kan door de gebruikers worden gebruikt en creëert waarde voor de gebruiker en voor de business. Het heeft misschien niet alle gevraagde functionaliteit maar er wordt wel gevalideerd of het werkt en de gewenste resultaten oplevert. Een verzendklaar product kan bestaan uit één of meerdere incrementen, zolang het maar in één sprint wordt opgeleverd.

Na de sprint review, evalueert het team de eigen prestaties tijdens de sprint om te zien hoe ze dingen in de toekomst beter kunnen doen. De gebeurtenis die voor deze zelfevaluatie wordt gebruikt, heet de **sprint retrospective**. Sprint retrospectives zijn time-boxed gebeurtenissen die ongeveer drie uur duren voor een vier weken durende sprint.

Hoewel dit niet officieel erkend is als Scrum gebeurtenis, besteden Scrum-teams vaak een bepaalde hoeveelheid tijd, meestal ook time-boxed, aan het helpen van de Product Owner bij het verfijnen van de product backlog (product backlog refinement).

Let wel: Product backlog verfijning werd vroeger ‘grooming’ (verzorging) genoemd, maar vanwege de negatieve connotatie die het woord grooming in sommige culturen heeft, werd de term veranderd in ‘verfijning’.

Product backlog verfijning helpt ervoor te zorgen dat product backlog items in de juiste volgorde staan voordat de volgende sprint begint.

Merk op dat, hoewel teams vaak specifieke tijd nemen om product backlog verfijning te doen, dit een doorlopende activiteit is, aangezien het team voortdurend leert en nieuwe afhankelijkheden kan ontdekken. Zodra dit gebeurt, wordt de product backlog onmiddellijk bijgewerkt met de nieuwste informatie of inzichten.

Het is hier niet de bedoeling om elke sprintgebeurtenis volledig te beschrijven, maar eerder om een overzicht te geven van de flow van gebeurtenis naar gebeurtenis in Scrum. Het volgende hoofdstuk zal specifiek ingaan op Scrum-gebeurtenissen.

5.8 Scrum-gebeurtenissen

In het vorige overzicht werden vijf Scrum-gebeurtenissen genoemd. Dat zijn:

- de sprint;
- de sprint planning;
- de daily scrum;
- de sprint review;
- de sprint retrospective.

We zullen nu kort elk van deze gebeurtenissen beschrijven in dit hoofdstuk. Merk op dat Scrum gebeurtenissen erg eenvoudig en eenduidig zijn.

5.8.1 De sprint

Sprints zijn de hartslag van Scrum, waar ideeën worden omgezet in waarde.

Het zijn gebeurtenissen met een vaste lengte van één maand of minder, vanwege de consistentie. Een nieuwe sprint begint onmiddellijk na afsluiting van de vorige.

Al het noodzakelijke werk om het product goal (doel van het product) te bereiken, inclusief sprint planning, daily scrums, sprint review en sprint retrospective, wordt gedaan binnen een aantal sprints.

Tijdens de sprint:

- worden geen verandering gemaakt die de sprint goal in gevaar brengen;
- nemen kwaliteit en kwaliteitseisen niet af;
- wordt de product backlog verfijnd, als dat nodig is;
- kan de scope van het werk worden verduidelijkt en heronderhandeld met de Product Owner naarmate het team meer te weten komt over de product backlog items (PBI's) die in de sprint zijn opgenomen.

Voorspelbaarheid wordt vergroot door inspectie en aanpassing van activiteiten terwijl deze vorderen in de richting van de product en sprint goals. Als sprints te lang zijn, kan de sprint goal ongeldig worden (en daardoor niet meer zijn afgestemd op veranderingen in de product backlog), kan de complexiteit toenemen en kan risico toenemen. Kortere sprints kunnen snellere leercycli genereren en risico's beperken. Iedere sprint kan gezien worden als een kort project.

Er kunnen veel tools worden gebruikt om voortgang te voorspellen, zoals burn-down charts, burn-up charts, of cumulatieve flows. Hoewel deze nuttig kunnen zijn, vervangen zij niet het belang van empirisme. In complexe omgevingen is niet bekend wat er zal gebeuren. Alleen wat al gebeurd is, kan worden gebruikt voor toekomstgerichte besluitvorming.

Een sprint kan worden geannuleerd *als de sprint goal achterhaald is*. Alleen de Product Owner heeft de bevoegdheid om de sprint te annuleren.

Echter, wanneer het team zich realiseert dat een sprint niet kan worden voltooid om welke andere reden dan ook, meestal wanneer Developers merken dat werk niet kan worden afgerond in de time-box vanwege onverwachte complexiteit, is het beter om de volgorde van de product backlog te herzien en na te gaan welk werk moet worden geprioriteerd.

5.8.2 Sprint planning

Sprint planning is de start van de sprint, met het kiezen van het werk dat wordt uitgevoerd in de sprint. Dit plan wordt gezamenlijk gemaakt, door het hele Scrum-team.

De Product Owner zorgt ervoor dat de aanwezigen klaar zijn om de belangrijkste product backlog items te bespreken en hoe hun relatie is met de product goal. Het Scrum-team kan ook andere mensen uitnodigen de sprint planning bij te wonen en advies te geven.

Sprint planning heeft betrekking op de volgende onderwerpen:

Waarom heeft deze sprint waarde?
De Product Owner stelt voor hoe het product aan waarde en nut kan winnen in de huidige sprint. Het hele Scrum-team werkt dan samen om een sprint goal te definiëren die uitdrukt waarom de sprint waarde heeft voor de belanghebbenden. De sprint goal is klaar aan het einde van de sprint planning.

Wat kan in deze sprint worden gedaan?
In gesprek met de Product Owner, selecteren de Developers items uit de product backlog die worden opgenomen in de huidige sprint. Het Scrum-team kan deze items tijdens de discussie verfijnen, wat het begrijpen en de zekerheid vergroot.

Het bepalen hoeveel werk binnen een sprint kan worden voltooid, kan een uitdaging zijn, vooral als het Scrum-team onbekend is met Agile en Scrum. In deze situatie is het raadzaam voorzichtig te zijn met het communiceren met belanghebbenden over verwachtingen met betrekking tot de snelheid (velocity); hoewel een goed begrip van hun verwachtingen nuttig is om deze te kunnen managen.

Hoe meer de Developers weten over de prestaties van het team in het verleden, de capaciteit die ze hebben en de definition of done (DoD), hoe zekerder ze zullen zijn in hun voorspellingen voor de sprint.

Hoe zal het gekozen werk worden gedaan?
De Developers plannen voor elk geselecteerd product backlog item van een increment, het noodzakelijke werk dat voldoet aan de definition of done (DoD). Dit wordt vaak gedaan door de product backlog items te ontleden in kleinere werkitems van een dag of minder. Dit wordt uitsluitend gedaan naar goeddunken van de Developers. Niemand anders vertelt hun hoe ze product backlog items omzetten in waardevolle incrementen.

De sprint goal, samen met de product backlog items die geselecteerd zijn voor de sprint, plus het plan van opleveren, worden aangeduid als de sprint backlog.

Sprint planning is time-boxed tot een maximum van acht uur voor een sprint van een maand. Voor kortere sprints is deze gebeurtenis meestal korter.

5.8.3 Daily scrum

Het doel van de daily scrum is om de voortgang richting sprint goal te inspecteren en, waar nodig, de sprint backlog aan te passen, door het aankomende geplande werk af te stemmen.

De daily scrum is een 15 minuten durende gebeurtenis voor de Developers van het Scrum-team. Om complexiteit te vermijden, wordt deze iedere werkdag van de sprint gehouden op dezelfde tijd en plaats. Als de Product Owner of de Scrum Master actief werkt aan items in de sprint backlog, nemen zij deel als Developers.

De Developers kunnen kiezen welke structuur en techniek ze willen gebruiken, zolang hun daily scrum zich maar richt op de voortgang richting de sprint goal en een uitvoerbaar plan oplevert voor de komende werkdag. Dit creëert focus en bevordert zelfmanagement.

Daily scrums verbeteren de communicatie, identificeren belemmeringen, bevorderen snelle besluitvorming en elimineren als gevolg daarvan de behoefte aan andere bijeenkomsten.

De daily scrum is niet het enige moment waarop Developers hun plan mogen aanpassen. Ze komen vaak gedurende de dag bij elkaar voor meer gedetailleerde discussies over het aanpassen of herplannen van de rest van het werk van de sprint.

In vorige versies van de Scrum Guide, werden drie specifieke vragen genoteerd die elk teamlid zou moeten beantwoorden. Deze waren:

- Waar werk ik vandaag aan?
- Wat heb ik gisteren voltooid?
- Wat kan ik niet doen vanwege een belemmering?

Velen interpreteerden deze lijst als de enige vragen die mogen worden gesteld en beantwoord en dat is waarom ze werden verwijderd uit de 2020 Scrum Guide. Scrum-teams kunnen alles vragen, beantwoorden en bespreken wat relevant is om

ervoor te zorgen dat taken worden afgerond en de flow behouden blijft. Het is geheel aan het team hoe ze de 15 minuten gebruiken om het werk voor de dag te plannen en uit te voeren en om te gaan met belemmeringen en problemen.

5.8.4 Sprint review

Het doel van de sprint review is het resultaat van de sprint te inspecteren en eventuele aanpassingen te bepalen. Het Scrum-team presenteert de resultaten van hun werk aan de belangrijkste belanghebbenden en de vordering richting product goal wordt besproken.

Tijdens deze gebeurtenis bekijken het Scrum-team en de belanghebbenden wat er is bereikt in de sprint en wat er is veranderd in hun omgeving. Gebaseerd op deze informatie, overleggen de aanwezigen over wat er nu moet gebeuren. De product backlog kan ook worden aangepast om tegemoet te komen aan nieuwe kansen. De sprint review is een werksessie en het Scrum-team moet vermijden dat het beperkt blijft tot een presentatie.

De sprint review is de één na laatste gebeurtenis van de sprint en is time-boxed tot een maximum van vier uur voor een sprint van een maand. Voor kortere sprints is de gebeurtenis meestal korter.

5.8.5 Sprint retrospective

Het doel van de sprint retrospective is manieren te vinden die de kwaliteit en de effectiviteit verhogen.

Het Scrum-team inspecteert hoe de laatste sprint is verlopen met betrekking tot individuen, interacties, processen, tools en hun definition of done (DoD). De elementen die geïnspecteerd worden, variëren vaak met het domein van het werk. Veronderstellingen die hen op een dwaalspoor brachten, worden geïdentificeerd en de oorsprong wordt onderzocht. Het Scrum-team bespreekt wat goed ging tijdens de sprint, welke problemen werden ondervonden en hoe die problemen al dan niet werden opgelost.

Traditioneel spraken teams specifiek over drie items, maar de discussie is, net als bij daily scrum meetings, niet beperkt tot deze onderwerpen. Het kan echter nuttig zijn om deze vragen toch te stellen tijdens een sprint retrospective:

- Wat moeten we niet langer doen?
- Wat moeten we blijven doen?
- Wat moeten we verbeteren in toekomstige sprints?

Merk op dat de vragen enkel helpen om een gesprek op gang te brengen en het Scrum-team kan nuttige veranderingen identificeren die de doeltreffendheid vergroten. De meest impactvolle verbeteringen worden zo snel mogelijk opgepakt. Ze kunnen zelfs worden toegevoegd aan de sprint backlog, voor de volgende sprint.

De sprint retrospective sluit de sprint af. Het is een time-boxed gebeurtenis van maximaal drie uur voor een typische sprint van een maand en meestal korter voor kortere sprints.

5.9 Scrum artefacten

Alle hier genoemde artefacten zullen in detail worden behandeld, te weten:

- de product backlog;
- de sprint backlog;
- een increment.

Hoewel ze niet zijn gedefinieerd als artefacten, zijn de volgende concepten nauw verbonden met de Scrum artefacten:

- de product goal;
- de sprint goal;
- de definition of done (DoD);
- user story's (en de variaties daarop, waaronder epics en functionaliteit);
- uitwerking naar taken.

Het verband tussen de drie artefacten en de tweede lijst kan worden omschreven als de commitment om een artefact te creëren of te onderhouden:

- De commitment voor de product backlog is de product goal.
- Voor de sprint backlog, is het de sprint goal.
- Voor het increment, is het de definition of done (DoD*).*

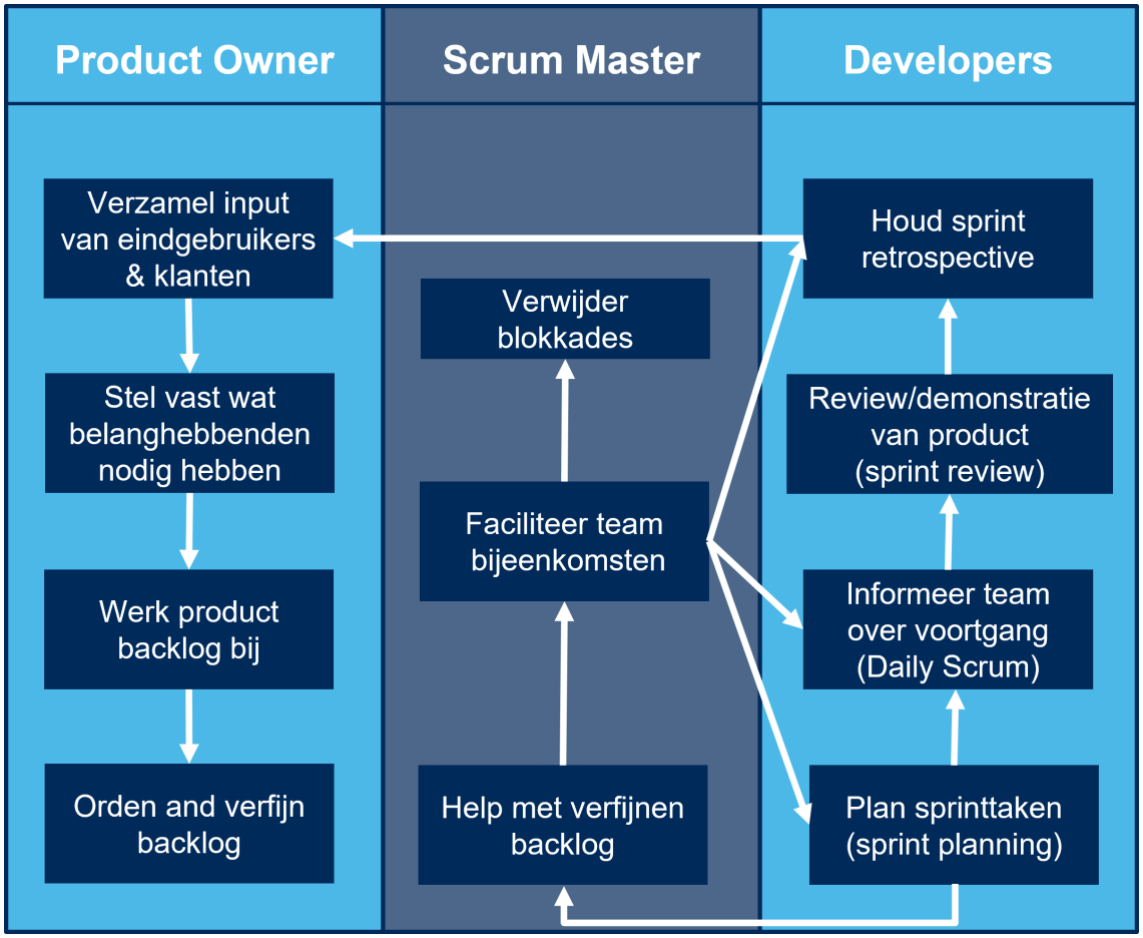
Hoewel het geen commitments zijn, worden user story's gebruikt om product backlog items te beschrijven en tijdens de sprint planning worden ze opgesplitst in bijbehorende taken.

Misschien is het opgevallen dat de verantwoordelijkheden zoals ze in Agile Scrum zijn gedefinieerd bijzonder eenvoudig zijn. Het zou zinloos zijn om nog een hoofdstuk te wijden aan wat een Scrum Master of Product Owner doet en dan alleen hun taken te beschrijven in een gebeurtenis of in het gebruik of creëren van een artefact, of in feite een andere activiteit.

In plaats daarvan is voor de rest van dit boek ervoor gekozen een concept of artefact te bespreken en daarbij de verantwoordelijkheid te beschrijven van de Developers, de Scrum Master of de Product Owner, in een specifieke gebeurtenis of als onderdeel van het concept of artefact dat wordt besproken.

De onderstaande grafiek dient als een samenvatting van de verantwoordelijkheden in Scrum en de bijbehorende activiteiten. Dit is geen proces flow, maar eerder een overzicht van de relaties tussen activiteiten en Agile Scrum verantwoordelijkheden.

Figuur 4 Relaties tussen Scrum activiteiten (dit is geen procesdiagram)



Afbeelding gemaakt door EXIN op basis van: Botha, J. (2021). *[Courseware]*. GetITright.

6 Andere activiteiten van Scrum-teams

6.1 Portfolio, producten en roadmaps

Strikt genomen geeft Scrum geen richtlijnen voor portfoliomanagement; de richtlijnen van Scrum beginnen op productniveau. De realiteit is dat producten niet uit de lucht komen vallen.

In iedere organisatie is er een opeenstapeling van gebeurtenissen waarbij belangrijke beslissingen moeten worden genomen, inclusief de planning, om ervoor te zorgen dat beslissingen in overeenstemming zijn met de context van de organisatie en dat ze weloverwogen zijn.

Er is een cascade van planningsactiviteiten en gerelateerde beslissingen binnen organisaties. Planning begint op strategisch niveau, vervolgens op productmanagementniveau en dan uiteindelijk tot waar het werk wordt gedaan in de operatie (Developers). Let wel: de operationele omgeving kan worden onderverdeeld in *operations run* (lopende activiteiten) en *operations change* (projecten). Scrum richt zich vooral op het laatste.

Voor Scrum om te werken en om Scrum te kunnen schalen, is het nodig de grenzen van gebeurtenissen te begrijpen, ofwel op welk niveau van de cascade vindt de gebeurtenis plaats, van strategie tot *operations run*. In een Scrum context kan dit er ongeveer zo uitzien:

- organisatorische doelstellingen;
- portfoliobeslissingen;
- backlog samenstelling;
- backlog verfijning;
- geschaalde planning van levering;
    - Nexus, Scrum@Scale of zelfs releaseplanning;
- sprint planning.

Let wel dat, hoewel het nuttig is om over beslissingen te praten, in een Agile-omgeving deze beslissingen niet heilig zijn en het veranderen van beslissingen of plannen nooit een probleem mag zijn (inspelen op verandering boven het volgen van een plan). Zorg ervoor dat er geen tijd wordt verspild aan onnodig plannen. Doe de minimale hoeveelheid planning op elk niveau van de cascade. Door dit te doen, zal het veel eenvoudiger zijn om plannen te wijzigen als er meer geleerd (empirisme) wordt over omstandigheden, behoeften van klanten, of als het begrip van de werkelijkheid verandert in de loop van de tijd.

Voormalig president van de Verenigde Staten Dwight D. Eisenhower zei ooit: ‘In het voorbereiden van de veldslag heb ik altijd vastgesteld dat plannen nutteloos zijn, maar planning onmisbaar.’

Denk aan de woorden van Eisenhower; planning is onontbeerlijk, omdat het deel uitmaakt van onze reis van leren en begrijpen. Focus dus op planning, niet op het

plan, moedig verandering en leren aan en maak het veranderen van een plan zo eenvoudig mogelijk.

6.2 Portfolioplanning

In veel grote organisaties wordt het werk beheerd in een breder portfolio. Dit portfolio kan bestaan uit producten, systemen, waardestromen, toeleveringsketens, investeringen of zelfs programma's.

Het doel van portfoliobeheer en -planning is te bepalen welke producten de doelen en doelstellingen van de organisatie ondersteunen.

Op basis van de prioriteiten van de organisatie, kan dan begrepen worden in welke volgorde de organisatie aan producten moet werken en onder welke voorwaarden ze zijn voldaan.

Bij portfolioplanning wordt een brede groep van belanghebbenden betrokken en ook de Product Owners. De planningshorizon voor een portfolio is de lange termijn en kan 6 tot 12 maanden zijn.

De vraag is: Hoe zullen de producten in het portfolio de organisatie dichter bij haar doelen en doelstellingen brengen? En, als Agile-methoden gebruikt blijven worden, kan de visie van de organisatie op het portfolio worden uitgedrukt in een portfolio backlog, bestaande uit product roadmaps voor elk product in het portfolio?

6.3 Productvisie

Het startpunt is het voor ogen hebben van de essentie van een potentieel product en een ruwe schets van hoe een product kan worden gecreëerd. De Product Owners en belanghebbenden (stakeholders) komen bij elkaar om de visie voor een nieuw product vast te stellen, met een planningshorizon die zich kan uitstrekken over meer dan een jaar. Het advies is om het bij een kortere planningshorizon te houden als dat kan, omdat de wereld snel en vaak verandert.

De deelnemers denken na over het doel van het product en zorgen ervoor dat de discussie altijd wordt teruggekoppeld naar de organisatiedoelen of -doelstellingen waar het product aan zal bijdragen.

De *belangrijkste resultaten van productvisie* zijn: het definiëren van een product goal, het ontwikkelen van een product backlog op hoog niveau en het creëren van een visie op hoe de backlog zal worden omgezet in waarde door het definiëren van een product roadmap op hoog niveau (een levend en evoluerend document). Deze output wordt input voor de algemene portfolioplanning.

De initiële product backlog op hoog niveau, geeft de Product Owner een breed startpunt om de algehele en grote/brede functionaliteit, of voordelen die belanghebbenden verwachten van het product, te identificeren, zodat het verzamelen van gedetailleerde vereisten kan beginnen.

Hoewel een product roadmap een doeltreffende tool is om het doel van het product te communiceren, evenals de incrementele manier waarop het product zal worden

gebouwd en geleverd, blijkt het vaak heel anders uit te pakken dan aanvankelijk gedacht.

6.4 Producten en product goals

Scrum-teams gebruiken de product goal als referentie voor de planning, omdat de product goal een toekomstige staat van het product beschrijft en aangeeft wat de bovenliggende organisatorische doelen en doelstellingen zijn die het product ondersteunt. De product goal maakt deel uit van de product backlog. De product goal wordt normaal gesproken opgeschreven voordat er begonnen wordt met het definiëren van product backlog items en is de primaire ‘check’ dat items die aan de product backlog worden toegevoegd, de organisatie dichter bij het bereiken van de product goal zullen brengen.

Om de product goal en een product backlog vast te stellen, moet eerst de Scrum-opvatting van wat een product is gedefinieerd worden.

Scrum definieert **een product** als een middel of tool om waarde te leveren aan iemand**.** Dit betekent dat we niet alleen willen begrijpen welke 'waarde' het product levert, maar we willen ook weten wie die 'iemand' is.

Duidelijke grenzen zijn nodig. Weten wie de belanghebbenden zijn, wat hun rol is, waarvoor ze eindverantwoordelijk zijn, en wat voor hen van waarde is, is een goed begin.

In Scrum is er geen onderscheid tussen producten en diensten; producten kunnen dus fysiek zijn of heel abstract.

De product goal is de lange termijn doelstelling van het Scrum-team. Elke sprint goal is een stap naar het bereiken van de product goal en het team moet één doel bereiken of loslaten alvorens een volgend doel op te pakken.

6.5 Product goal en waarde voor de business

De product goal is het resultaat van de strategie van het product en de waarde die ermee gecreëerd wordt en een verklaring op hoog niveau van wat een product is.

Maar hoe verhoudt de waarde zich tot een product goal?

De business zal doelen en doelstellingen definiëren die de organisatie moet realiseren om haar strategie te behalen. Producten in het portfolio stellen de organisatie soms in staat om deze doelen te bereiken. Het gebruik van het woord ‘soms’ is bewust; binnen organisaties bestaan ook andere middelen om doelen te bereiken.

Het verschil tussen organisatorische doelstellingen en product goals is dus dat de eerste verband houden met de bedrijfsstrategie en de laatste met productmanagement.

Wat de *product goal* in Scrum anders maakt dan het algemene gebruik van de term product goals in product management, is dat het woord in Scrum op een meer

specifieke manier wordt gebruikt. In oudere versies van Scrum was er een verwijzing naar de productvisie, en het gebruik van de term ‘product goal’ ligt dichter bij deze term dan het algemene gebruik van het woord in productmanagementkringen.

De product goal is een korte beschrijving van de lange termijn doelstelling of toekomstige toestand van het product. Er kan gezegd worden dat de product goal het WAT van het product is, soms aangevuld met een WAAROM.

Er kan ook gezegd worden dat de product goal de algemene toewijding weerspiegelt van Scrum-teams die aan een product backlog werken; de product goal zorgt voor focus.

De organisatie in staat stellen haar strategische intentie te bereiken is op zich waardevol en dat is het belangrijkste doel van het product.

Echter, het is belangrijk te bedenken dat over het algemeen producten de organisatie en/of haar klanten faciliteren en dat zij daarom op verschillende manieren waardevol zijn. Producten creëren dus vaak zowel interne waarde, door organisatorische processen en werkmethoden te verbeteren, als externe waarde, door eindgebruikers in staat te stellen iets te doen dat zij voorheen niet konden, of iets dat zij voorheen wel deden, beter te doen. Producten doen dit op verschillende manieren en hetzelfde product of zelfs dezelfde functionaliteit kan andere waarde creëren voor andere gebruikers van het product. Het leveren van waarde, als nuttige/waardevolle incrementen van een product, gaat verder dan de scope die is gedefinieerd in de product goal.

Hoewel het leveren van waarde in incrementen verband houdt met de algehele product goal, beschrijft de product goal niet alle waarde van het product.

Er moet worden opgepast om de product goals ingewikkeld te maken door alle vormen van waarde die het product creëert, vast te leggen. Een ingewikkelde product goal is minder bruikbaar, omdat dit de focus belemmert.

Beschouw de product goal als een beschrijving van het belangrijkste strategische doel of doelstellingen die het product mogelijk maakt.

De volgende vraag die in dit verband gesteld kan worden, is: Wat is waarde of waardevol?

Waarde wordt bepaald door degene voor wiens probleem het product of de functionaliteit een oplossing biedt. Hoewel waarde een zeer subjectief begrip is en moeilijk te definiëren, proberen we toch dit te doen.

De voornaamste reden is dat het creëren en leveren van waarde middelen gebruikt en dus tastbare kosten met zich meebrengt. Daarom kan de vraag over waarde niet worden beantwoord tenzij de volgende vraag beantwoord is: “Is de gecreëerde waarde in monetaire termen meer waard dan de kosten om de waarde te creëren?”

En dit is waar het interessant wordt in Agile-omgevingen.

6.6 Meten van waarde in reële termen

Verwacht niet dat traditionele financiële kengetallen de winst kunnen meten die wordt behaald door Agile te zijn of de waarde meten van producten in reële termen.

In plaats daarvan kan er beter gericht worden op verbetering en op metingen van kernindicatoren. Financiële rapporten veranderen de prestaties in de organisatie bijna nooit positief – maar focus op de resultaten van dag tot dag doet dat wel!

Een traditioneel geïndustrialiseerd wereldbeeld drijft in grote mate de opmaak en het gebruik van conventionele financiële meetwaarden aan, een slechte match voor Agile-omgevingen. Deze uitdaging is niet nieuw voor Agile-omgevingen, maar werd snel duidelijk toen organisaties Lean begonnen te omarmen in de jaren 1990.

Waarom zijn traditionele financiële kengetallen zo problematisch en vaak misleidend in Agile-omgevingen?

Hoewel bewijzen van het gebruik van financiële kengetallen dateren van 300 v. Chr., heeft verspreiding van het bedrijfsmatig investeren in ondernemingen het nut van financiële kengetallen duidelijk gemaakt. Het probleem dat zij oplosten was eenvoudig. Het gebruik van kengetallen stelt beleggers in staat investeringen in bedrijven die qua samenstelling, bedrijfstak, risico en markt sterk van elkaar verschillen, met elkaar te vergelijken. Met behulp van kengetallen kunnen beleggers overwegen of zij moeten investeren in een bedrijf dat houten kistjes maakt in India, een autofabrikant in de Verenigde Staten, of een financiële instelling in de Europese Unie. Door naar de belangrijkste kengetallen te kijken, wordt het nemen van een beslissing eenvoudiger.

De kengetallen waar het hier om gaat, omvatten alle traditionele boekhoudkundige kengetallen (ratio’s):

- **Liquiditeitsratio’s** meten het vermogen van een onderneming om zowel korte- als lange termijn verplichtingen te voldoen.
- **Hefboomratio's** meten de hoeveelheid kapitaal die afkomstig is van schulden.
- **Efficiëntieratio's** (ook bekend als financiële activiteitsratio's) meten hoe goed een onderneming gebruik maakt van haar activa en middelen.
- **Rentabiliteitsratio's** meten het vermogen van een onderneming om inkomsten te genereren in verhouding tot de inkomsten, de activa op de balans, de bedrijfskosten en het eigen vermogen.
- **Marktwaarderatio's** evalueren de aandelenprijs van een bedrijf.

Het is niet de bedoeling hier te stellen dat geen van de traditionele ratio's nuttig zijn in een Agile-omgeving. Echter, veel van deze ratio's zullen een heel ander verhaal vertellen als ze gemeten worden over een maand, zes maanden, een jaar, vijf jaar, of zelfs tien jaar, afhankelijk van hoe ver er vooruit gekeken wordt.

Het probleem met traditionele boekhoudpraktijken en de daaruit voortvloeiende financiële ratio's is dat accountants financiële verslagperioden zien als iets met een begin en een eind. Het is ook eenvoudig om de traditionele metingen en

meetwaarden te gebruiken voor Watervalprojecten – per slot van rekening is de definitie van een Watervalproject dat het een begin en een einde heeft. Dan is het eenvoudig om het rendement op investeringen (return on investment, RoI) voor een project te berekenen door de kosten te vergelijken met de gerealiseerde baten.

De manier waarop wordt gekeken naar de prestaties van de business in een Agile-omgeving is heel anders.

Hier ligt de focus op het voortdurend verbeteren van alle maatstaven – er is geen begin en einde. Daarom hebben de meeste traditionele financiële ratio's en boekhoudkundige praktijken geen zin bij Agile of Lean.

In feite kan het gebruik van traditionele ratio's leiden tot korte-termijnverbeteringen met negatieve langetermijnresultaten. In een Agile-omgeving is het juist aan te bevelen om lange-termijnverbeteringen aan te brengen, zelfs als dat negatieve korte-termijnresultaten brengt.

De lessen van Lean kunnen heel goed toegepast worden als gekeken wordt naar de boekhoudpraktijken in Agile-organisaties. Bedrijven kwamen er al snel achter dat er radicaal andere financiële praktijken nodig waren, praktijken die een brede systeemvisie van de organisatie omvatten en wat 'goede financiële resultaten' zijn. Toen dit besef doordrong, was Lean-accounting geboren.

Het gebruik van rendement op investeringen (Return on Investment, RoI), intern rendement (Internal Rate of Return, IRR) en netto contante waarde (Net Present Value, NPV) als maatstaven voor het nemen van Agile-investeringsbeslissingen wordt lastig.

Hier zijn enkele redenen waarom oude veronderstellingen misschien niet meer gelden:

- In Agile is er **geen start- of einddatum van een project**. Er zou zelfs beargumenteerd kunnen worden dat er in werkelijkheid geen project is, maar alleen korte incrementen die waarde leveren.
- Omdat er ruimte is voor of zelfs aangemoedigd wordt dat vereisten in de loop van de tijd evolueren, is er **geen vooraf bepaald of eindig beeld van wat er in de loop van de tijd zal worden gedaan**.

RoI, IRR, of NPV worden problematische meetwaarden in een Agile-omgeving als er investeringsbeslissingen genomen moeten worden. Een andere set van metingen is daarom nodig.

Wat het nog complexer maakt, is dat wordt verwacht dat continue verbetering, wat van oudsher uitsluitend een operationele activiteit was, deel uitmaakt van elke sprint, waardoor de grenzen tussen ‘operationele projecten’ en ‘operationele run-omgevingen’ nog meer vervagen.

Net als in Lean-omgevingen (van Lean-accounting), worden de meest geschikte maatstaven voor succes nu:

- verbeterde flow (doorstroming);
- toename of handhaving van kwaliteit;
- verbeterde prestaties (levering, doorlooptijden, productiviteit).

en als gevolg daarvan:

- meer winstgevendheid.

De focus van Lean (Agile) accounting is een brede, systeemvisie. Er kan gerapporteerd worden over verbetering in een specifiek venster om koerscorrecties door te voeren, maar het is belangrijk om te begrijpen dat deze rapporten een tussentijdse meting zijn en niet noodzakelijk de gezondheid van het bedrijf op lange termijn weerspiegelen.

Deze veranderde visie betekent dat de praktijk om een afdeling of een project te financieren niet langer zinvol is. Aangezien de inkomsten van een organisatie rechtstreeks verband houden met de prestaties van haar producten, is het verstandiger om producten te financieren, waarbij de financiering zowel project-/veranderings- als operationele runs betreft. Er moet nagedacht worden over de hele waardestroom die het product creëert, verkoopt, onderhoudt en onderhoudt – ja, end-to-end. Deze systeemvisie wijkt sterk af van wat accountants op de universiteit leren!

De nieuwe Lean/Agile-manier om naar RoI te kijken is meer holistisch. Om de resultaten van investeringen effectief te meten, is het noodzakelijk om het totaal van alles wat middelen verbruikt in de waardestroom, af te zetten tegen de winst die wordt behaald door een product aan klanten te verkopen. Rapportage moet een in de tijd gesplitste instantie worden die prestaties vergelijkt, in de loop van de tijd, op een continuüm, in plaats van een rapportage die de bedrijfsprestaties weergeeft als een reeks niet met elkaar in verband staande gebeurtenissen.

Er kan worden aangevoerd dat de totale kosten van de waardestroom die een product oplevert, afgezet tegen de inkomsten die eruit voortvloeien, nog steeds in wezen RoI is en ja, dat is zo. Maar hoe helpt dat in de toekomst? RoI uit het verleden is een zeer onbetrouwbare maatstaf voor de toekomstige RoI van een product.

De tekortkoming van ratio's zoals RoI is dat het achterlopende indicatoren zijn – ze kunnen laten zien dat er niet het gewenste rendement op de investering hebben behaald, maar ze zijn nutteloos als maatstaf om koerscorrecties aan te brengen. Het is precies om deze reden dat Lean-accounting zich richt op verbetering als een leidende indicator van prestatie.

Interessante kanttekening: Eén van Toyota's doelstellingen is om alles in het bedrijf als geheel, elk jaar met 2% te verbeteren. Het klinkt als weinig, maar ze hebben dit doel bereikt sinds 1950 en het opgebouwde effect is enorm!

Als er deze maand, de vorige maand en de maand daarvoor verbeteringen zijn aangebracht, betekent dit dat het bedrijf goed op weg is naar een betere RoI. Als er deze maand niets werd verbeterd, moet er onmiddellijk iets gedaan worden om de koers bij te stellen.

Leden van de Agile Alliance hebben tussen 2012 en 2016 projecten uitgevoerd om standaardisatie van Agile-accounting te bereiken, maar te oordelen naar de output van het initiatief, hebben ze grotendeels gefaald om agility te zien als een geïntegreerde, waarde-stroom-brede, continue aanpak om financiële resultaten in Agile-omgevingen te beoordelen.

Lean-accounting is voorlopig de meest geschikte manier om naar investeringsbeslissingen in een Agile-omgeving te kijken, maar het vergt een aanzienlijke verschuiving in het begrip van wat boekhouden is en hoe dit in een organisatie moet worden gedaan.

Dus, wat is de conclusie van dit alles?

Focus op verbetering en op leidende indicatoren. Financiële verslagen veranderen de prestaties van de organisatie nauwelijks positief – maar focus op de prestaties van dag tot dag doet dat wel.

6.7 Meer over het beheren van de product backlog

Zonder een fatsoenlijke product backlog werkt Scrum gewoonweg niet. De product backlog is het hart van alle Scrum planningsactiviteiten – het is de enige bron van waarheid voor het Scrum-team. Het beschrijft al het werk dat de teams te doen hebben en de relatieve prioriteit van het werk. Als ze niet in de product backlog staan, bestaan de vereiste en bijbehorende activiteiten niet.

Product backlogs beschrijven niet alleen wat de klant nodig heeft, maar ook alle technische taken die op de achtergrond moeten gebeuren om ervoor te zorgen dat aan de behoefte van de klant wordt voldaan. We kunnen daarom de product backlog beschrijven als een geprioriteerde lijst (of beter gezegd, een geordende lijst, zoals later zal blijken) van zowel functionele als niet-functionele vereisten die moeten worden geleverd tijdens het project. Zodra een vereiste is vervuld, wordt het gerelateerde item verwijderd uit de product backlog.

Sommige bedrijven vermelden ook taken die worden uitgevoerd om aan de vereisten te kunnen werken, zoals planning, middelen en andere voorbereidende stappen die nodig zijn om de bal aan het rollen te brengen.

Elke backlog is eigendom van een Product Owner, die eindverantwoordelijk is voor het beheren van die specifieke product backlog. Het is hun taak om ervoor te zorgen dat de backlog actueel en geordend is. Zij doen dit echter met hulp van de rest van het Scrum-team.

Een gangbare praktijk is ervoor te zorgen dat de product backlog aan vier specifieke criteria voldoet. Dit concept, dat aanvankelijk werd gedefinieerd door Mike Cohn, wordt nu algemeen gebruikt in de Scrum gemeenschap. De criteria worden voorgesteld door het acroniem DEEP:

- **D**etailed appropriately (voldoende gedetailleerd);
- **E**stimated (geschat);
- **E**mergent (zich ontvouwend);
- **P**rioritized (geprioriteerd).

Er is al aangegeven dat Scrum de term Prioritized schuwt, omdat deze in verschillende contexten en organisaties verschillende betekenissen heeft. Scrum geeft de voorkeur aan het gebruik van de term **Ordered** (geordend).

DEEP in een Scrum context wordt dus DEEO:

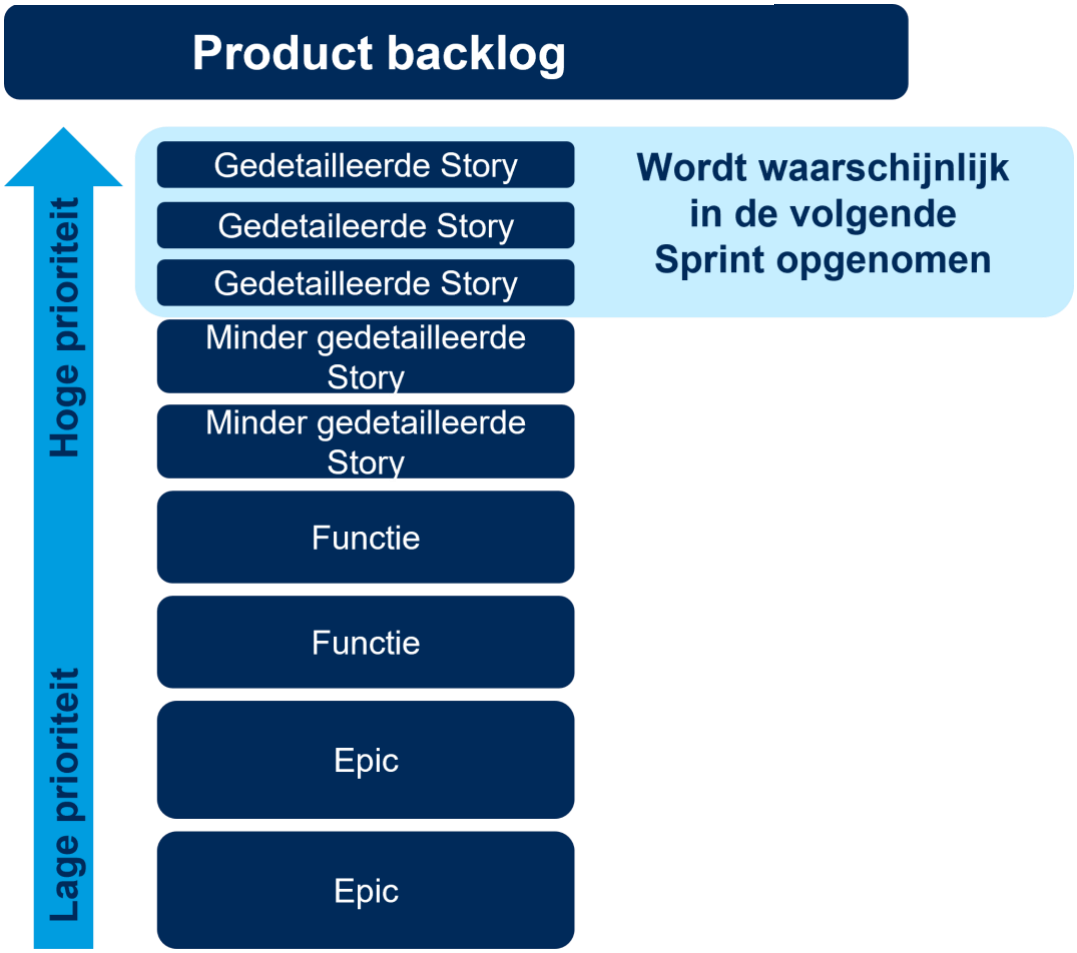
- **D**etailed appropriately (voldoende gedetailleerd);
- **E**stimated (geschat);
- **E**mergent (zich ontvouwend);
- **O**rdered (geordend).

6.7.1 Detailed appropriately (Voldoende gedetailleerd)

Hoewel de product backlog niet de plaats is waar de vereisten opsplitst zijn in gedetailleerde activiteiten, moet deze toch gedetailleerd genoeg zijn om zicht te geven op de tijd die nodig is om het werk voor de vereiste te doen.

Vereisten worden normaliter geschreven als story's, maar story's kunnen ook verschillende niveaus van detail hebben.

Figuur 5 Product backlog detaillering



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2018). *Scrum Masters and Product Owners [Courseware]*. GetITright

Detaillering neemt toe naarmate de story richting bovenkant van de product backlog gaat, grotere story's worden uitgewerkt in kleinere (verwerkbare) story's.

Items met een hoge prioriteit moeten zo gedetailleerd zijn dat de inzet en de vaardigheden die nodig zijn om de vereisten te realiseren, kunnen worden bepaald.

Belangrijke items, waaraan waarschijnlijk gewerkt zal worden tijdens de volgende twee sprints, worden daarom ontleed tot het juiste niveau tijdens product backlog verfijning en tijdens sprint planning.

Dit betekent dat naarmate projecten vorderen, het ontleden van story's of vereisten naar meer detaillering, voortdurend zal plaatsvinden. Het wordt echter alleen gedaan als het nodig is en niet vooraf voor alle story's of vereisten, want als de story voor het eerst wordt toegevoegd aan de product backlog is het niet eens zeker of dit item en alle gerelateerde vereisten belangrijk zijn, of zelfs in de backlog zullen blijven. Vaak worden vereisten met een lagere prioriteit uit de backlog verwijderd of kunnen er andere vereisten opduiken die deze zelfs gaan vervangen.

In Agile wordt inzet niet verspild aan zaken die niet worden gebruikt – dit is wat wordt bedoeld met ‘net genoeg’ werk doen om succesvol te zijn.

OPMERKING: epics en functionaliteiten zijn aanvullende praktijken die helpen de omvang en complexiteit van story’s te beschrijven; de Scrum Guide maakt geen onderscheid tussen verschillende soorten story’s.

6.7.2 Estimated (Geschat)

Alle items in de product backlog moeten een schatting hebben van de inzet die nodig is om de story te voltooien. Het is duidelijk dat hoe groter de story is, hoe moeilijker dit in te schatten is. Schatting van epics zal daarom een lage mate van nauwkeurigheid hebben, maar dat maakt niet uit; story’s die epics worden gelaten, meestal van minder direct belang.

Schatten is een essentiële activiteit om sprint planning te kunnen doen. De definition of done (DoD) wordt overwogen bij het schatten van de inzet om de functionaliteit als een increment te realiseren. Dit houdt in dat elk increment een DoD heeft, omdat er meer dan één increment in een sprint kan zijn.

Het gebruik van crossfunctionele teams kan een groot voordeel geven bij het schatten van taken, omdat zij ontbrekende informatie kunnen verschaffen tijdens de uitvoering. Crossfunctionele teams hebben alle competenties die nodig zijn om het werk uit te voeren zonder afhankelijk te zijn van externe inbreng. Omdat er altijd minstens één teamlid is met de kennis en vaardigheden om de taak uit te voeren, is er dus altijd iemand die de taak kan schatten.

6.7.3 Emergent (Zich ontvouwend)

Wat bedoeld wordt met emergent is dat de product backlog niet een definitieve vaststelling is van wat er gebeurt gedurende de levenscyclus van een product. Vaak veranderen de eisen van de klant tijdens de levenscyclus van het product en worden nieuwe vereisten toegevoegd aan de backlog. Dit is echter niet de enige keer dat er items aan de backlog worden toegevoegd.

Regelmatig worden tijdens de incrementele ontwikkeling van een product nieuwe vereisten ontdekt die voorheen niet bekend waren. Dit zijn meestal afhankelijkheden waarvan andere vereisten afhankelijk zijn om te werken en vaak zijn dit geen functionele maar niet-functionele vereisten.

Wanneer dus wordt gezegd dat de product backlog emergent is, wil dat zeggen dat deze zal evolueren.

6.7.4 Ordered (Geordend)

Alle items in de product backlog worden gepresenteerd door ze te rangschikken, waarbij de items die het eerst worden gedaan boven in de product backlog komen te staan. De volgorde is een indicatie van het belang of de implementatievolgorde of de afhankelijkheid van andere belangrijke story’s voor de voltooiing of implementatie van de story in kwestie.

6.8 Product backlog verfijning

De product backlog is een steeds veranderend artefact en omdat er voortdurend items de product backlog binnenkomen, uitgaan of veranderen, moet deze zorgvuldig worden bijgehouden.

De DEEO-activiteiten waarover eerder is gesproken, zijn taken die horen bij product backlog onderhoud. Deze activiteit staat bekend als product backlog verfijning.

De Product Owner is eindverantwoordelijk voor verfijning, maar dit is geen activiteit die door één persoon wordt uitgevoerd. De Product Owner wordt hierin bijgestaan door Developers. Zij dragen bij met hun specifieke technische inzicht, om bestaande items in de product backlog of nieuwe items die worden toegevoegd aan de product backlog te herschikken en opnieuw in te schatten.

Developers dragen specifiek bij, omdat zij de technische knowhow hebben om in te schatten hoelang het zal duren om een story af te ronden en om te herkennen of een belangrijke story afhankelijk is van andere story's en dus hetzelfde belang heeft vanwege de afhankelijkheid. Scrum-teamleden zullen ook weten hoe grote story's kunnen worden opgesplitst, zodat ze gemakkelijk in een sprint kunnen worden uitgevoerd.

Ook al is de Product Owner eindverantwoordelijk voor het verfijnen van de product backlog, het is een zeer collaboratieve activiteit.

Hoewel backlog verfijning geen gedefinieerde Scrum gebeurtenis is, moeten Scrum-teamleden tijd inplannen voor deze activiteit.

Omdat het geen geplande (time-boxed) gebeurtenis is, vindt product backlog verfijning plaats wanneer het nodig is, vaak tijdens andere Scrum-gebeurtenissen. Als er problemen optreden, zoals veranderende vereisten, afhankelijkheden, of miscommunicatie over onontdekte vereisten, worden deze daar en dan aangepakt.

Enige mate van verfijning zal onvermijdelijk worden gedaan tijdens geschaalde of sprint planningen. En het is niet meer dan natuurlijk dat hetzelfde zal gebeuren tijdens de sprint reviews en sprint retrospectives. Afgeronde story's worden verwijderd en story's opnieuw geordend in het licht van wat er nu bekend is. Story's kunnen zelfs worden uitgewerkt en nieuwe story's toegevoegd als afhankelijkheden of nieuwe vereisten werden ontdekt tijdens de zojuist voltooide sprint.

Het kan een goed idee zijn om dagelijks tijd te besteden aan het bespreken van de product backlog en de sprint backlog en eventuele problemen die worden gevonden, bijvoorbeeld in de daily scrum.

Aanbevolen wordt dat Scrum-teams tot 10% van de tijd besteden aan verfijning. De meeste teams besteden echter veel minder tijd aan product backlog verfijning.

Product backlog verfijning omvat normaalgesproken de volgende activiteiten:

- **Nieuw ontdekte items worden toegevoegd aan de backlog**. Dit kunnen zowel nieuwe eisen van klanten zijn, als ontbrekende functionele en niet-functionele vereisten en afhankelijkheden, die tijdens de vorige sprint werden ontdekt.
- **Backlog items worden gerangschikt** met de belangrijkste boven aan de lijst.
- **Backlog items moeten de juiste omvang hebben** om sprint planning en schatting eenvoudiger te maken.
- **Grote of vage belangrijke items worden uitgewerkt** in kleine user story's die kunnen worden gebruikt in komende sprints.
- **Belangrijke items worden verfijnd** (beter beschreven) om sprint planning eenvoudiger te maken.
- **Afhankelijkheden worden vastgesteld** en de volgorde waarin ze worden uitgevoerd, wordt vastgelegd.
- **De backlog wordt opnieuw bekeken** om ervoor te zorgen dat alle nieuwe items zijn toegevoegd, de volgorde van uitvoering is bepaald, de omvang en de schattingen voor voltooiing juist zijn en items die niet langer nodig zijn, uit de backlog zijn verwijderd.

6.9 Creëren van product backlog items

Zoals eerder opgemerkt is het verzamelen van vereisten geen eenmalige gebeurtenis in Scrum. Hoewel het verzamelen vooral gedaan zal worden aan het begin van een project zou deze activiteit zich niet moeten richten op het krijgen van gedetailleerde vereisten en het uitwerken van epics in fijnmazige user story's.

Het uitwerken van grote story's is daarom een doorlopende activiteit die het beste vlak voor het werk gedaan kan worden. Op deze manier kunnen teams er zeker van zijn dat de vereisten waaraan gewerkt wordt, de laatste en actuele vereisten zijn en niet wensen van twee jaar geleden. Kortom, door dit zo te doen, wordt ervoor gezorgd dat waarop gericht wordt actueel is en maximale waarde creëert.

In de vorige sectie werd benadrukt dat vereisten niet statisch zijn en dat er voortdurend nieuwe eisen ontstaan, die vaak oudere eisen vervangen – dus te veel tijd besteden aan het uitwerken van story's waaraan niet zal worden gewerkt tijdens de volgende sprints, is eigenlijk verspilling; er wordt zo mogelijk werk gedaan dat nooit zal worden gebruikt. Dit is in lijn met het Agile-principe van 'net genoeg'.

In het algemeen kan gesteld worden dat wanneer er vereisten verzameld worden, men de vereiste en het resultaat wil begrijpen, en niet de details van hoe men daar zal komen.

Een goede manier om over vereisten na te denken is een overzicht te maken van wat er in het project zal worden gedaan door een stappenplan (roadmap) te maken en dan vast te stellen welke vereisten in welke fase van de roadmap worden opgepakt.

Het maken van een roadmap dwingt ook tot nadenken over de kernvereisten die snel kunnen worden opgeleverd en onmiddellijk waarde toevoegen. Vervolgens kan er gericht worden op de periferie en het verbeteren van het product.

Het verzamelen van deze vereisten op hoog niveau kan in een brainstormsessie worden gedaan met de belangrijke belanghebbenden. Begin de sessie met het definiëren van wat er geprobeerd wordt te bereiken en met het creëren van een productvisie. Welk probleem wordt er opgelost of welke waarde gaan wordt ontsloten?

Het product kan dan op hoog niveau worden opgesplitst in componenten en de ontwikkeling daarvan kan de basis vormen van de product roadmap.

Er wordt nog niet gepraat over details – dit is praten in grote lijnen. Het is eerst bepalen wat het minimum viable product (MVP) is. De focus ligt op de kernfunctionaliteit zonder welke het product niet zal werken.

Vergeet niet dat de product backlog zal worden verfijnd in de tijd en meer gedetailleerde en minder belangrijke eisen kunnen altijd later worden toegevoegd.

DSDM gebruikt een concept dat MoSCoW heet – en dat hier misschien nuttig is.

MoSCoW
MoSCoW is een prioriteringsmethode ontwikkeld door Dai Clegg, één van de personen die heeft bijgedragen aan de ontwikkeling van de Dynamic Software Development Method[6] en is bedoeld voor het rangschikken van items in een sprint. Het gebruik van MoSCoW in het proces van identificeren van vereisten kan heel nuttig zijn en het Scrum-team waardevolle inzichten verschaffen.

Bij het definiëren van een vereiste kan gevraagd worden, is het een:

- Must Have;
- Should Have;
- Could Have;
- Won’t Have.

In het begin kan er het beste gericht worden op de Must Haves en misschien wat Should Haves, omdat het initieel vaak moeilijk is om onderscheid te maken tussen die twee.

[6] DSDM, wordt nu ook wel ABC wordt genoemd, een afkorting van Agile Business Consortium.

Het verschil tussen Must Have en Should Have kan als volgt worden gedefinieerd:

- **Must-Have** vereisten zijn kritisch en moeten snel gerealiseerd worden, om de nodige waarde te leveren. Als deze niet gerealiseerd kunnen worden, is het project een mislukking.
- **Should Have** vereisten zijn belangrijk maar minder dringend. In het algemeen worden ‘Should Have’ vereisten geleverd tegen het einde van het project om er zeker van te zijn dat het product volledig functioneel of gemakkelijk bruikbaar is.
- **Could Have** vereisten zijn nice to have (aardig om te hebben) functionaliteit en Won't Have vereisten zijn uitgesloten, omdat er geen rechtvaardiging is om ze op te nemen in het project.

6.9.1 Uitwerken van niet-functionele eisen

Er is één uitzondering op de regel om vereisten niet onmiddellijk uit te werken, namelijk de niet-functionele vereisten.

Niet-functionele vereisten zijn de onderdelen van een initiatief waar de rest van het initiatief van afhangt, maar waar de klant niet om vraagt. Het is belangrijk om goed te weten wat er komt kijken bij het leveren van deze vereisten want niet-functionele eisen zijn de basis voor de rest van het project.

Niet-functionele eisen worden daarom zodra ze bekend zijn, zo goed mogelijk uitgewerkt door het team.

6.10 Verzamelen van eisen – producten en uitkomsten

Het begrijpen welke uitkomsten gebruikers en klanten wensen wordt in Scrum als zeer belangrijk gezien. Inzicht hierin maakt fundamenteel deel uit van de manier waarop Scrum vereisten verzamelt vanaf het begin.

Een minimumvereiste voor opname in de product backlog is dus een goed begrip van WELKE belanghebbenden de vereiste nodig hebben (dit zorgt voor context en helpt om afhankelijkheden te identificeren), vervolgens WAT vereist is en WAAROM het vereist is.

Het is om deze specifieke reden dat user story’s zo gestructureerd zijn:

Als < belanghebbende ROL>,

wil ik < het WAT van de vereiste>,

zodat < het WAAROM van de vereiste>.

Als op basis van eerder verzamelde vereisten wordt gewerkt, wees dan heel voorzichtig. Het is af te raden om eerder verzamelde vereisten te vertalen naar user story’s.

Traditionele technieken voor het verzamelen van vereisten beogen precies het tegenovergestelde van wat hierboven is beschreven. Deze technieken proberen de vereisten fijnmazig en gedetailleerd te definiëren.

Deze gewoon vertalen of kopiëren naar de product backlog zal resulteren in een zeer complexe backlog met weinig inzicht in prioritering en te creëren waarde.

Als er gewerkt wordt vanuit traditioneel gedetailleerde vereisten is het aan te bevelen om hieruit vereisten op hoog niveau te destilleren door vereisten te groeperen (het tegenovergestelde van uitwerken tot items) om uit te komen op minder items, die eenvoudiger kunnen worden gerangschikt en geschat.

Een techniek die hier gebruikt kan worden is alle gedetailleerde vereisten op post-its schrijven en een affinity map creëren met groepen van gerelateerde vereisten. De groepsbeschrijvingen kunnen dan als grofmazige vereisten worden toegevoegd aan de product backlog.

Het gebruik van deze techniek zal er ook toe leiden dat vereisten die in de categorieën Could Have en Won't Have zouden passen, worden geschrapt of een lagere prioriteit krijgen.

Er zijn veel verschillende manieren waarop user story’s en andere story’s kunnen worden weergegeven in de product backlog. Sommige mensen geven de voorkeur aan een voorgedefinieerd formaat, anderen niet. Een nadeel van het gebruik van een voorgedefinieerd formaat is dat het dwingt om op een bepaalde manier over vereisten na te denken.

Soms bestaan user story’s letterlijk uit alleen de regel hierboven en soms bevatten ze een heleboel details, zoals belanghebbenden, prioriteiten, afhankelijkheden, gerelateerde activiteiten, of zelfs wie er in een sprint aan de story moet werken. In de meeste gevallen is het wat overdreven om dit allemaal vast te leggen.

Het user story format van hierboven zorgt ervoor dat de absoluut noodzakelijke informatie wordt vastgelegd in het backlog item. Met de story-indeling wordt antwoord gegeven op de vraag WAT er bereikt moet worden, voor WIE en WAAROM. Let wel, de WAT en WAAROM zijn beide ook prestatie- of acceptatiecriteria.

Merk op dat Scrum niet voorschrijft om vereisten in een user story formaat te definiëren, maar het wordt sterk aanbevolen om deze methode te gebruiken.

6.11 Meer over user story’s

User story’s worden story’s genoemd, omdat verondersteld wordt dat de belanghebbende over de vereiste vertelt. Niet in klinische termen, maar eerder in de vorm van een verhaal. Het verhaal van wat ze doen of wat ze moeten doen en waarom het belangrijk voor hen is om dat te doen.

Dit laatste is eigenlijk heel belangrijk! Eerder werd benadrukt dat iteratieve oplevering niet alleen over output gaat, maar over het helpen van de belanghebbenden om hun uitkomsten te realiseren, vaak en zo snel mogelijk. Het WAAROM van het verhaal van de persoon is de uitkomst.

Het vertellen van een verhaal over het hoe en waarom helpt de persoon die het verhaal optekent om te evalueren of het logisch is. Het gewoon noteren van eisen op de traditionele manier doet dat niet.

Als het niet logisch is, kan degene die het verhaal vastlegt om opheldering vragen – zij of hij kan ook vragen waarom en hoe.

Hoe wordt het precies gedaan? Waarom wordt het op die manier gedaan? Waarom wordt het überhaupt gedaan? Is het goed om het op een andere manier te doen als hetzelfde resultaat wordt bereikt?

Gesprek leidt tot begrip en begrip leidt tot meer kennis over hoe waarde te creëren voor belanghebbenden, meestal gebruikers en klanten.

Figuur 6 Voorbeelden van een story- en een taakticket

Afbeelding gemaakt door EXIN op basis van: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us.

Het gebruik van de verhaalvorm is een buitengewoon krachtige manier om vereisten vast te leggen en te begrijpen.

Als < belanghebbende ROL>,

wil ik < het WAT van de vereiste>,

zodat < het WAAROM van de vereiste>.

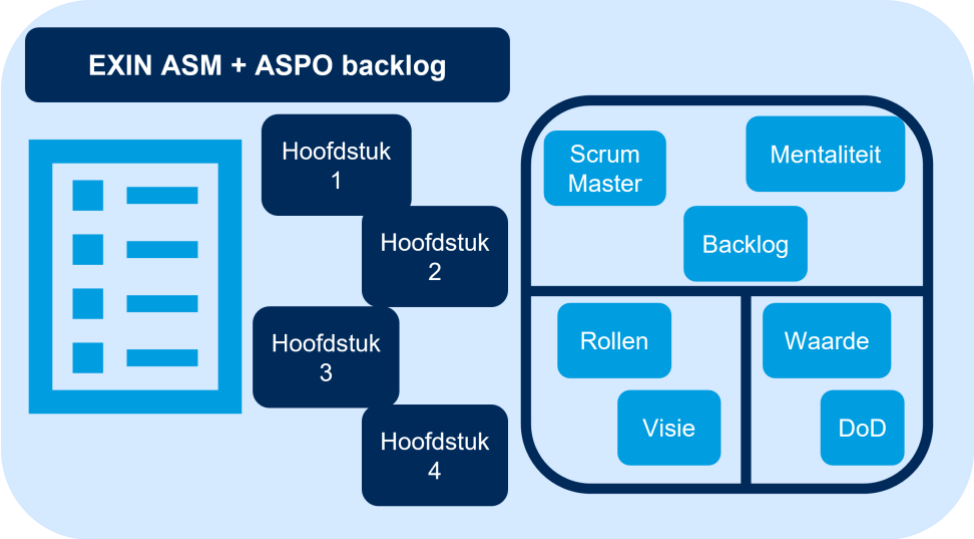
Er kan worden gekozen om de eisen op te slaan in een tool, maar het is aan te bevelen ze vast te leggen op fysieke kaarten of post-its en de voortgang van de oplevering uit te zetten op een Kanban-bord.

De afbeelding hieronder is een voorbeeld. Het is eigenlijk de Kanban voor het schrijven van dit boek – met de product backlog aan de linkerkant. In dit voorbeeld is te zien dat de story's heel eenvoudig zijn en alleen de uitkomst vastleggen. Dat is voldoende informatie om het product (in dit geval het boek) te kunnen schrijven – het is met andere woorden *net genoeg*.

Zelfs in dit persoonlijke voorbeeld, is er waarde in het creëren van een backlog, het definiëren van vereisten en het verfijnen van vereisten tijdens het proces.

In dit geval zijn de lichtblauwe post-its uitgewerkte story's die behoren tot één van de epics van een hoger niveau, hier weergegeven door een grotere, donkerder post-it. En elk van de verschillende kleuren stickers heeft een eigen unieke betekenis, met vaak aanvullende informatie erop genoteerd. Dit is niet de norm; het is een sterk aangepaste manier van werken. Elk Scrum-team zal in de loop van de tijd de eigen manier van werken aanpassen.

Figuur 7 De product backlog



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2021). *[Courseware].* GetITright.

Vergeet niet dat het vastleggen van de vereisten slechts het begin is van een reis; vereisten zullen worden verfijnd en verfijnd tot ze klaar zijn en worden opgeleverd.

6.12 User story's en het uitwerken van taken

Het volgende hoofdstuk gaat in op schattingstechnieken en het uitwerken van taken. Het is belangrijk om hier op te merken dat er pas aan story's gewerkt kan worden als bekend is hoeveel werk er nodig is om de story op te leveren.

Het uitwerken en inschatten van taken is de laatste stap in de sprint planning. De Scrum Master en de Product Owner mogen de Developers niet beïnvloeden bij het maken van de schatting. De Product Owner moet aanwezig zijn tijdens de activiteit om onduidelijkheden toe te lichten, mochten die zich voordoen tijdens het schatten.

Taken uitwerken is geen eenmalige gebeurtenis en hoewel het wordt gedaan op een hoog niveau bij het starten van de sprint planning of een geschaalde implementatieplanning, het moet uitgebreid worden gedaan voordat het werk in een sprint begint. Vergeet niet dat het zelfs tijdens een sprint opnieuw kan gebeuren als dat nodig is, dus net als product backlog-verfijning, herhaalt het zich totdat het werkt en resultaat oplevert.

Theoretisch kunnen taken even lang duren als een sprint, maar dat is een slecht idee, niet praktisch. Dus dit roept twee vragen op:

- Hoelang is een taak?
- Als de taak langer is, wat doen we er aan?

Het is altijd het beste dat taken overeenkomen met de cadans van de daily scrum, omdat tijdens deze bijeenkomst het team juist praat over het werk voor die dag.

Het is onvermijdelijk dat er taken zijn die langer dan een dag in beslag nemen, maar als er goed gekeken wordt, kunnen ze opgesplitst worden in deeltaken en kan er een schatting gemaakt worden voor elk van deze deeltaken.

Taken opsplitsen op een te laag niveau is ook niet productief – de regel is dat een taak wordt opgesplitst op het niveau waarop deze kan worden geaccepteerd door een teamlid op basis van zijn of haar vaardigheden en niet lager.

Een aanbevolen maximum tijdsduur om een taak te voltooien, is 8 uur. Soms is het verleidelijk om meer story's toe te voegen aan een user story die geselecteerd is om aan te werken. Het is niet aan te raden om ze te combineren voor de volgende sprint, omdat dit de story te groot kan maken om nog hanteerbaar te zijn.

Als er geen taakopsplitsingen op het niveau van 8 uur mogelijk is, kan er overwogen worden om een moment van extra feedback te introduceren terwijl teamleden over voortgang rapporteren tijdens de daily scrum. Dit kan een nuttige praktijk zijn. Als er na een dag nog werk in een taak zit, rapporteert het teamlid niet hoeveel van de toegewezen tijd zij of hij aan de taak heeft besteed, maar eerder hoeveel van de geschatte tijd er nog over is om de taak te voltooien.

Deze werkwijze heeft twee grote voordelen:

1. **Het dwingt mensen na te denken over werk** en leert hun betere taakinschattingen te maken, en
2. **Het helpt de Scrum Master om de voortgangsmeting** bij te werken zoals in een burn-down chart, die de tijd weergeeft die nog rest om de iteratie te voltooien. Hier zullen puristen zeggen dat een burn-down chart geen half-voltooide taken weergeeft. In theorie is dit correct, maar de voordelen van het reflecteren op onvoltooide taken helpt het team om beter te plannen en motiveert het teamlid dat bezig is met de taak, want niemand wil zeggen: 'Ik heb vandaag niets bereikt'.

Bij het maken van planningen en het opsplitsen van taken kunnen de volgende ‘emmers’ gebruikt worden om de duur van taken in te schatten.

Figuur 8 Geschatte werktijd toekennen aan activiteiten in een emmersysteem

Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2019). *Agile: A Manager’s Guide to Unlocking Business Value.* Amazon Digital Services LLC - Kdp Print Us.

6.13 Creëren en onderhouden van de product backlog en de roadmap

In de vorige sectie hebben ging het over het verzamelen van vereisten en de noodzaak om te beginnen met grofmazige vereisten (hoog niveau).

De Scrum Guide beschrijft de product backlog als een geordende of gerangschikte lijst van alle bekende productvereisten. Het is de enige bron van alle wijzigingen aan het product om uiteindelijk aan deze eisen te voldoen, die allemaal de product goal en de doelstellingen van de organisatie ondersteunen.

Let op het woord ‘alle en allemaal’. Het is belangrijk om ervoor te zorgen dat ook niet-functionele vereisten in de backlog worden opgenomen. Het kan zelfs een goed idee zijn om belangrijke kwesties op te nemen die nog moeten worden opgelost die tijdens sprint reviews of retrospectives als niet-optimaal zijn gevonden.

Een product backlog kan beginnen met slechts vier of vijf grofmazige user story’s (epics) om deze vervolgens uit te werken in volgorde van prioriteit na elke iteratie. Een goede vuistregel is om niet meer dan 100 items in de backlog te hebben – met een maximum van ongeveer 300.

Alles wat meer is, zal het verfijnen van de product backlog vrijwel onmogelijk maken en het beheer van de backlog extreem moeilijk.

Het advies is om de 100-item regel aan te houden en om, zodra dit aantal hoger oploopt, te bekijken welke items uit de backlog kunnen worden geschrapt. Doorloop de oefening opnieuw om gedetailleerde items met lage prioriteit te combineren tot epics, zoals hierboven beschreven toen we het hadden over het werken met reeds bestaande vereisten die op traditionele wijze zijn verzameld.

Als er begonnen wordt met vier of vijf vereisten op hoog niveau is het maken van een product roadmap eenvoudig – wijs een geschatte volgorde van uitvoering toe en het is klaar.

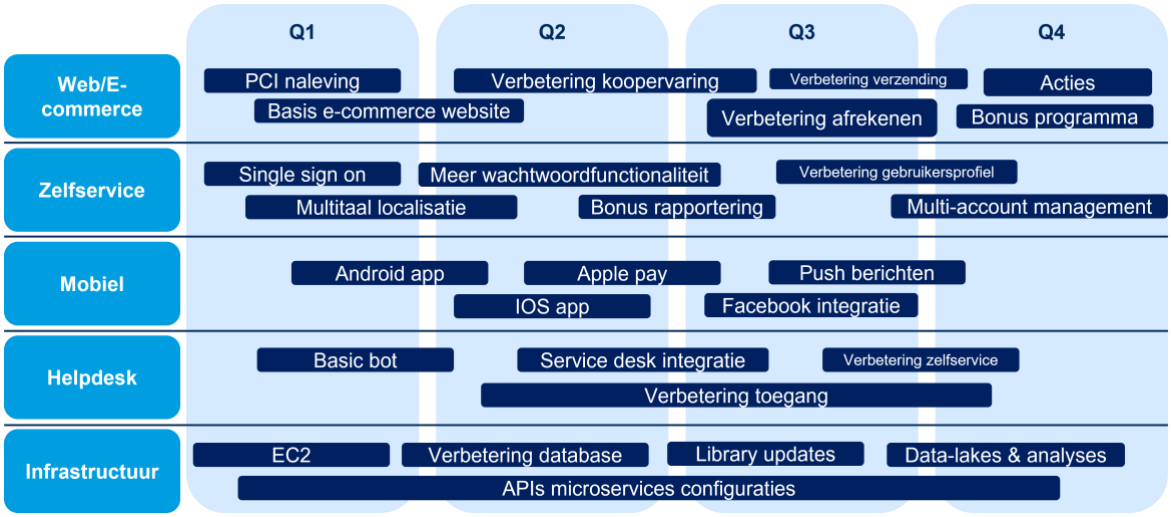
Meestal is het echter niet zo eenvoudig.

Om een roadmap te maken, zouden de vereisten op hoog niveau gegroepeerd moeten worden in thema's en deze gebruikt worden als leidraad voor het maken van een roadmap. Over het algemeen kunnen thema's worden omschreven als een groep vereisten die betrekking hebben op producten of productcategorieën, op gebruik door de business (inclusief ondersteunde processen), of op functionaliteit.

Thema's zijn echter in de praktijk niet de beste manier om roadmaps te maken in complexe projecten, vooral wanneer producten of productcategorieën, worden gebruikt als leidraad. Thema's zijn eenvoudiger en effectiever te maken op basis van clusters van gerelateerde functionaliteit.

Dit is echter ook niet altijd ideaal, aangezien het doel van een roadmap is om een brede groep van belanghebbenden een indruk te geven van hoe waarde zal worden gecreëerd. De meeste belanghebbenden zullen een roadmap op basis van thema's niet begrijpen als deze te technisch zijn; dit heeft alleen zin als de thema's betrekking hebben op wat zij doen of willen doen – dus vereisten.

Figuur 9 Voorbeeld van een product roadmap



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value*. Amazon Digital Services LLC - Kdp Print Us.

Als er thema's gedefinieerd worden, wees dan heel duidelijk over waarom dit gedaan wordt en ook over hoe het creëren van thema's kan helpen of hinderen. Vraag jezelf altijd af: Zullen de gebruikers en klanten begrijpen wat we bedoelen met het thema? Zullen ze begrijpen welke functies bij welk thema horen?

Er is geen eenvoudig antwoord op de vraag hoe een product roadmap gemaakt moet worden. Laten we eens kijken naar een aantal redenen daarvoor.

Een product roadmap wordt gebruikt om de visie, richting, volgorde van oplevering van een product te delen. Het is geen vastgelegde tijdslijn zoals een traditioneel projectplan of de bijbehorende Gantt chart.

De bedoeling is belanghebbenden te informeren en op de hoogte te houden van wat er gebeurt en dat ze zich daar prettig bij voelen.

Product roadmaps zijn daarom vloeibaar. Ze veranderen naarmate wat als bedrijfskritisch wordt gezien, evolueert en naarmate Scrum-teams aan items beginnen te werken en afhankelijkheden identificeren.

Product roadmaps worden ook vaak gebruikt als een hulpmiddel binnen het project of als visueel communicatiemiddel. Hier wil het team, naast de genoemde punten, ook de voortgang van een product in de tijd zien. Als de product roadmap op deze manier wordt gebruikt, wordt het een actieplan en een manier om de voortgang te laten zien.

Gebruik niet dezelfde roadmap voor beide doelgroepen. Als de teamview van de roadmap gebruikt wordt voor andere belanghebbenden, met name het management, dan wordt de roadmap een verplichting naar klanten toe. Dit is een slecht idee, want over drie maanden kan de roadmap er heel anders uitzien.

Let wel: Dimensies die door Scrum-teams worden gebruikt als prestatie-indicatoren, mogen nooit worden gebruikt als een voortgangs- of prestatieweergave voor algemene belanghebbenden.

6.14 Welke criteria worden gebruikt voor het ordenen van items in de backlog?

Steven Covey zei in ‘The Seven Habits of Highly Effective People’ dat prioriteit het product is van ‘dringend’ en ‘belangrijk’, maar in tegenstelling tot de manier waarop de meeste mensen met deze twee termen omgaan, wordt prioriteit als volgt toegekend:

1. belangrijk en dringend;
2. belangrijk;
3. dringend;
4. onbelangrijk en niet dringend.

Maar wat betekenen deze termen?

- **Belangrijk** betekent dat de business aanzienlijk waardeverlies zal lijden of dat er aanzienlijke gevolgen zullen zijn als het niet gedaan en niet snel gedaan wordt.
- **Dringend** betekent dat er iemand is die er belang bij heeft dat het gedaan wordt, bijvoorbeeld een verkoper heeft beloofd dat een functie in de volgende release is opgenomen.

Er kan worden gesteld dat de twee criteria waarde en risico zijn. Waarde en risico zijn daarom de twee bepalende factoren voor het belang en tot op zekere hoogte voor de urgentie.

Het is duidelijk dat het heel belangrijk is om beloften aan klanten na te komen, maar als dit betekent dat de organisatie verlies lijdt door eerst iets te doen, ligt het voor de hand wat er gedaan moet worden. Waarde en risico zijn belangrijker dan korte

termijn beloften. Het is beter om samen met belanghebbenden de waarde te bepalen van de dingen die gedaan moeten worden dan om 'goedkope' korte termijn beloften te doen die misschien moeilijk kunnen worden nagekomen.

Als te veel items als belangrijk worden gezien, vormt dat een aanzienlijk bedrijfsrisico – omdat het betekent dat niets belangrijk is. Help belanghebbenden met deze analogie te begrijpen hoe het bepalen van belang en de volgorde van items in een product backlog wordt gedaan.

Een andere manier is gebruik te maken van de **Pareto regel** – slechts 20% van de items in de product backlog zullen belangrijk zijn. Vraag belanghebbenden om te beslissen welke 20% van de items in de product backlog dat zijn.

Hoewel Product Owners uiteindelijk de volgorde van items in de backlog bepalen, is het beter om een breed scala van belanghebbenden te betrekken bij het begrijpen van en beslissen over de volgorde van backlog items.

Gebruik het toegewezen relatieve belang van items, gebaseerd op overeengekomen criteria, om items in de backlog te rangschikken. Wat als eerstvolgende wordt gedaan, staat boven in de product backlog.

Zorg er ook voor dat items boven in de backlog voldoende zijn uitgewerkt tot fijnmazige vereisten, inclusief een nauwkeurige inschatting van het werk dat met de items gepaard gaat.

Dit zijn immers de items die hoogstwaarschijnlijk in de sprint backlog van de volgende sprint zullen staan.

Het kan zijn dat tijdens het uitwerken duidelijk wordt dat sommige van de fijnmazige story’s belangrijk zijn en andere niet. Herorden de product backlog op zo'n manier dat alleen items die gedaan worden in de komende paar sprints boven in de product backlog blijven staan. De andere items kunnen lager in de backlog worden geplaatst, ook al zijn ze fijnmazig of belanghebbenden kunnen besluiten om ze helemaal te verwijderen uit de backlog.

Het kan ook nuttig zijn om na te gaan of er andere, gelijksoortige vereisten in de backlog staan die grotendeels of volledig dezelfde uitkomsten mogelijk maken. Is dit het geval dan is verwijderen van dubbele items eenvoudig uit te leggen. Als items op basis hiervan verwijderd worden, zorg er dan voor dat de afhankelijkheden en vereisten in kaart worden gebracht en begrepen.

Besteed niet te veel tijd aan het bepalen van de volgorde van minder belangrijke items; het doet er echt niet toe totdat duidelijk wordt dat ze nu belangrijk zijn geworden.

Essentiële items moeten goed worden begrepen en gedocumenteerd. Gebruik indien nodig, aparte documenten voor het onderbouwen van backlog items. Vaak wordt dit onderbouwen gedaan door terug te gaan naar belanghebbenden en meer vragen te stellen of door belanghebbenden te betrekken bij het uitwerken van epics.

Als sommige van de zeer fijnmazige items moeilijk te ordenen zijn, kan er besloten worden om ze te verpakken in één story of item dat gemakkelijker te begrijpen is en deze een positie geven in de backlog. Deze praktijk helpt afhankelijkheden te begrijpen. Het Scrum-team kan deze weer opsplitsen tijdens de sprint planning.

Herprioritering wordt gedaan elke keer dat de product backlog wordt verfijnd. Het is de eenvoudigste manier om de backlog zuiver en relevant te houden.

Het zal ook blijken dat al snel nadat de resultaten van de eerste iteratie beschikbaar zijn gemaakt voor gebruik, er ‘nieuwe’ vereisten binnenkomen. Ga er niet van uit dat deze nieuw zijn; veel ervan bestaan al als onderdeel van epics of story’s op hoger niveau.

Het kan nodig zijn deze items opnieuw te ordenen, als de vraag groot is en deze belangrijker worden.

Niet-functionele vereisten zijn de uitzondering. Ze vormen vaak een hogere afhankelijkheid en vragen daarom een hogere plaats in de product backlog. Zijn ze geïdentificeerd, dan moeten ze onmiddellijk worden uitgewerkt en opgenomen als fijnmazige vereisten. Risico is ook een goede indicator voor het bepalen van de volgorde.

Het is beter om risico's en onzekerheid zo snel mogelijk aan te pakken zodat extra werk op een later tijdstip wordt vermeden. Classificeer items met een hoog niveau van risico en onzekerheid als hoog en positioneer ze zo ver mogelijk naar boven.

Dit is vroeg experimenteren om aannames te kunnen valideren, wat van nature onzeker en riskant is. Door dit te doen, wordt voldaan aan één van de principes van Scrum – het gebruik van een empirische aanpak.

Ongeadresseerd risico is een grote molensteen rond de nek van het team in Scrum en het zal ze later naar beneden sleuren. Als iets dat al gedaan is, afhangt van een item dat uiteindelijk onhoudbaar, onwaar of onbruikbaar blijkt te zijn, betekent dit dat alle bijbehorende inspanningen verspild zijn.

Bij het opsommen van items in de backlog kan ervoor gekozen worden items met afhankelijkheden al te groeperen of items die nodig zijn om een ‘done’ op te leveren voor gebruikers. Een Scrum Scaling methode (Nexus) die specifiek deze aanpak gebruikt, zal later worden behandeld.

Wees voorzichtig met aan te nemen dat alle gerelateerde story’s noodzakelijk zijn voor de uitkomst van een sprint. Vaak kan een sprint enkele kernelementen opleveren, één of twee functionaliteiten uitsluiten en toch perfect bruikbaar zijn en waarde opleveren.

Wat de sprint planning betreft, is er de aanbeveling om Must, Should en Could functionaliteiten te kiezen. Should en Could functionaliteit maakt niet noodzakelijkerwijze deel uit van de definition of done (DoD) en wordt daarom gebruikt om een ‘buffer’ te creëren in een sprint.

Deze aanpak wordt vaak gebruikt als Scrum wordt geïntroduceerd, omdat het natuurlijk ongewenst is dat sprints mislukken. Dit kan leiden tot een situatie waarin het traditionele antwoord van de organisatie op falen, openlijk moet worden toegepast (strafmaatregelen).

Onthoud dat het doel is in een vroeg stadium waarde te ontsluiten voor belanghebbenden. Probeer altijd zo snel mogelijk echte waarde te leveren.

6.15 Communicatie met belanghebbenden

Eén van de grootste uitdagingen voor elke projectomgeving is weten wat als waardevol wordt beschouwd en weten wat waarde creëert.

Helaas is waarde net als schoonheid in de ogen van de toeschouwer. Wat voor de één waardevol is, kan door de ander als waardeloos worden gezien.

Het is ook belangrijk op te merken dat belanghebbenden vaak iets beschrijven als waardevol voor hen, maar dat bij nader onderzoek kan blijken dat niet alles even waardevol is. Hier helpt de MoSCoW-techniek om het gesprek te sturen en belanghebbenden te helpen onderscheid te maken tussen wat absoluut niet-onderhandelbaar is (een Must Have), wat belangrijk is en de efficiëntie of productiviteit sterk bevordert (een Should Have), wat leuk is om te hebben, omdat het dingen beter of gemakkelijker zou maken (een Could Have) en wat een toekomstige overweging zou moeten zijn maar op dit moment niet kritisch is (Wouldn't Have Now).

Tijd is een belangrijk onderwerp in de discussie over het ontwerpen en bouwen van nieuwe producten en diensten, omdat sommige functies de kern van een product of dienst vormen en andere zijn ondersteunende functies of verbeteringen. Het spreekt voor zich dat de kern eerst wordt gebouwd en dat Scrum-teams zich daarna richten op het volgende meest waardevolle.

Het bouwen van nieuwe producten en diensten met behulp van Agile Scrum lijkt erg op de Lean Startup aanpak beschreven door Eric Ries. Het is een iteratief proces en er moet snel geleerd worden wat belanghebbenden vinden van de kern van het product of de dienst die wordt opgeleverd.

Scrum-teams kunnen een minimaal levensvatbaar product (**minimum viable product, MVP**) definiëren en met belanghebbenden valideren of het MVP iets is dat ze kunnen gebruiken en waarde voor hen zal ontsluiten. Als dat zo is, moeten alle story’s die samen het MVP vormen, fijnmazig zijn en bovenaan de product backlog staan. Als dit gedaan wordt, zal het opvallen dat alles boven in de backlog Must Have items zijn. Maar als dit waar is, hoe kan MoSCoW dan nuttig zijn?

Gebruik het just-as-well (net zo goed) principe wanneer het Scrum-team de product backlog verfijnt en zoek naar werk dat gemakkelijk en effectief gedaan kan worden met het Must Have werk, ook al is het misschien een Should of een Could.

Waarschuwing: Gebruik dit principe niet bij het bespreken van functionaliteit met andere belanghebbenden, omdat dan onrealistische verwachtingen worden gewekt.

Gebruik het alleen in het sprint -team (of Nexus-team). En doe het alleen als de MoSCoW-techniek gebruikt gaat worden.

Het zal opvallen dat als user story's op hoog niveau (epics genoemd) opgesplitst worden, er natuurlijke clusters van kenmerken zijn die automatisch in de categorieën Must, Should en Could vallen.

Vaak kunnen MVP's worden getest en gebruikt, maar het zal moeilijk zijn om ze in de markt aan te bieden, omdat ze nog niet compleet genoeg zijn om klanten te verleiden. Voor de ontwikkeling van commerciële producten en diensten is meer nodig.

Het alternatief is de ontwikkeling van minimaal verkoopbare producten (**minimum marketable product, MMP**). Het idee van een MMP is gebaseerd op het Lean-principe dat minder meer is. Het MMP omvat de kleinst mogelijke functionaliteit die tegemoetkomt aan de behoeften van initiële gebruikers – zij zouden bereid zijn er geld voor te betalen. Het MMP is een manier om de doorlooptijd naar de markt (time-to-market) te verkorten, door een uitgeklede versie van het product of de dienst aan te bieden en na verloop van tijd extra functies toe te voegen om er een completer en rijker product van te maken.

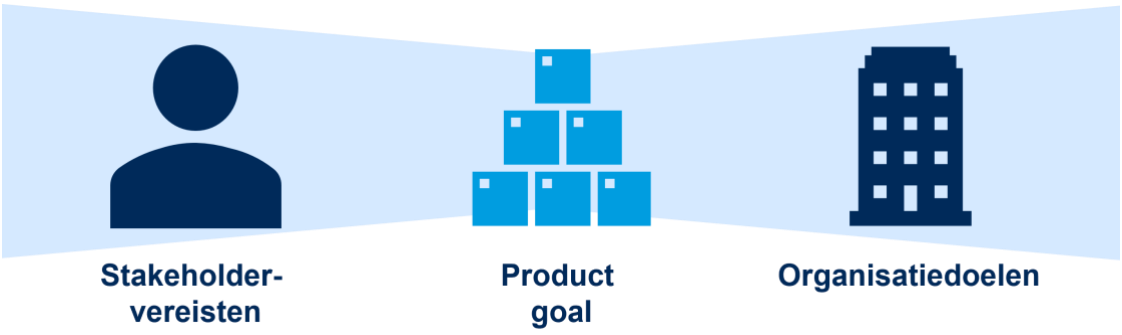
6.16 Een product goal definiëren

De product goal is een samenvatting van de bedrijfsdoelstellingen die door het product worden ondersteund en alle vereisten van de belanghebbenden, geëvalueerd en geordend in een product backlog. Alleen met een volledig beeld van het product, de vereisten van klanten en andere belangrijke belanghebbenden en hoe het product de organisatiedoelen ondersteunt, kan er een effectief product goal gedefinieerd worden.

Let wel: Voor alle Scrum-teams die werken met een product backlog, is de product goal een weergave van het Ware Noorden (True North). Elke sprint goal is in lijn met en een stap vooruit naar het realiseren van de product goal.

Product goals ontstaan niet vanzelf, aangezien zij boven de vereisten staan. Ze worden specifiek ontwikkeld.

Figuur 10 Wat is een product goal?



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2021). *[Courseware].* GetITright.

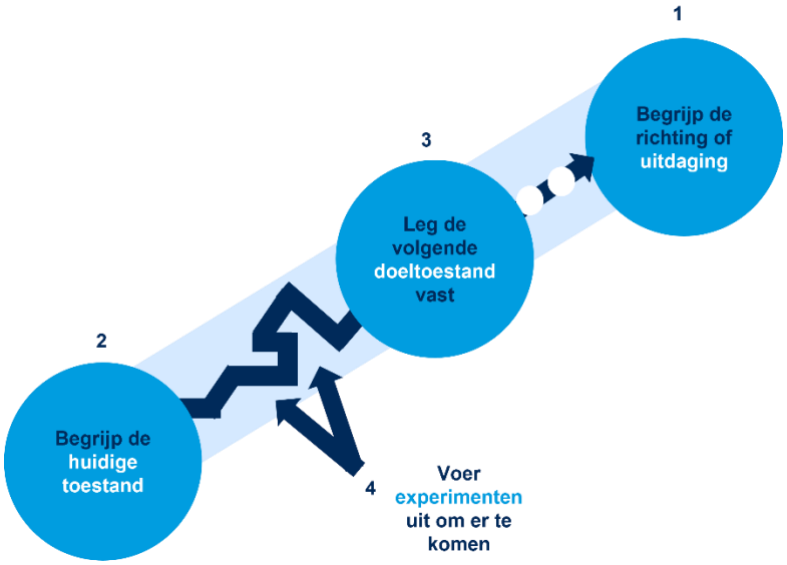
Een nuttige techniek is de **Toyota Improvement Kata** aanpak om niet alleen de product goal maar ook de product backlog te verduidelijken en te verfijnen, vooral in het begin.

Toyota Improvement Kata is, net als Agile, een wetenschappelijke en empirische benadering om problemen, doelen en doelstellingen te begrijpen. Traditioneel wordt ze niet gebruikt om doelen en doelstellingen te verduidelijken; maar er kan wel gesteld worden dat slecht gedefinieerde doelen of doelstellingen een probleem zijn.

Toyota Improvement Kata volgt in het algemeen deze vier stappen:

1) **Begrijp de richting of uitdaging**: Begrijp de richting, de grotere, waarschijnlijk ver verwijderde visie en wees duidelijk over wat die is.
2) **Begrijp de huidige toestand**: Onderzoek de huidige situatie en omschrijf deze feitelijk en duidelijk.
3) **Stel de volgende doeltoestand vast**: Bepaal een passend doel voor de volgende stap, dat de grenzen van de huidige kennis en mogelijkheden oprekt, de uitdaging dichterbij brengt en op een bepaald tijdstip zal zijn bereikt.
4) **Voer experimenten uit**: Experimenteer methodisch en wetenschappelijk om naar de volgende doeltoestand te komen.

Figuur 11 De Toyota Kata stappen



Afbeelding gecreëerd door EXIN gebaseerd op: Rother, M. (2018). *The Toyota Kata Practice Guide: Practicing Scientific Thinking Skills for Superior Results in 20 Minutes a Day.* Mcgraw-Hill Education

We kunnen vertrouwde technieken, zoals story's, gebruiken om de product goal te definiëren. Het is echter beter om voldoende input te hebben van belanghebbenden. Eerst moeten de organisatiedoelen begrepen worden, evenals de principes en het

doel van de organisatie, die de primaire, normaal gesproken langetermijn-organisatiedoelen aansturen.

Er kan begonnen worden met:

a. **Definieer bovenliggende organisatiedoelstellingen**: In < x jaar tijd> zal < onze organisatie> < datgene zijn/bereiken/hebben waarnaar we streven>.
b. **Om het gestelde doel te bereiken**: We moeten < de dingen die we moeten doen of gedaan krijgen, meetbaar definiëren – tegen wanneer>.
c. **Evalueer vervolgens waar we nu staan**, zodat we kunnen vaststellen welke kloof moet worden gedicht. We kunnen ervoor kiezen de kloof stap voor stap te dichten, bijvoorbeeld:
    - ‘Wij zullen < wat we zullen doen> tegen < wanneer> ‘.
    - Als het niet te ambitieus is, kunnen we dezelfde aanpak toepassen voor het algemene doel van de organisatie.
d. Het **actief beheren van de product backlog** vertegenwoordigt de experimenten die in de Kata worden uitgevoerd.

Vaak worden de manier en de volgorde waarop dingen gedaan worden, beïnvloed door de vereisten en prioriteiten van de belanghebbenden. Hun behoeften worden in aanmerking genomen als de stappen op hoog niveau, waarmee de doeltoestand bereikt kan worden, bepaald worden.

Puristen zouden zeggen dat Toyota's Improvement Kata niet op deze manier mag worden gebruikt. De effectiviteit is echter bewezen. Er zijn veel middelen en richtlijnen beschikbaar voor de Kata-aanpak en dit maakt het een geschikte methode om goede product goals te definiëren.

Net als Agile, steunt Kata verder op communicatie, teamwerk en samenwerking.

6.17 Verzamelen van vereisten

De primaire manier in Agile om vereisten te verzamelen is door user story’s vast te leggen.

Het user story format kan grofmazige en fijnmazige vereisten verzamelen door de verwachte uitkomsten van een vereiste in een enkele verklaring vast te leggen.

Story’s worden daarom vaak verzameld wanneer een product en product backlog voor het eerst worden gedefinieerd en in dit stadium zijn deze story’s grofmazig en groot, ook wel bekend als epics.

Naarmate het werk vordert in de richting van de oplevering van vereisten worden story’s opgesplitst in fijnmaziger story’s. Deze kunnen op hun beurt worden opgesplitst in taken, waaraan in een increment gewerkt kan worden.

Gebruik de INVEST[7] methode bij het definiëren van user story’s:

[7] https://searchsoftwarequality.techtarget.com/tip/7-techniques-for-better-Agile-requirements-gathering

- **Onafhankelijk (independent)**: Elke user story staat op zichzelf, onafhankelijk van andere story's.
- **Onderhandelbaar (negotiable)**: Story's zijn geen contracten, eerder mogelijkheden tot overleg en verandering.
- **Waardevol (valuable)**: Elke story voegt waarde toe voor gebruikers en belanghebbenden.
- **Inschatbaar (estimable)**: De nodige tijd en het budget van elke story zijn te berekenen, gebaseerd op domein- en technische kennis.
- **Klein of eenvoudig (simple)**: User story's zijn klein genoeg om eenvoudig in te schatten en te implementeren.
- **Testbaar (testable)**: De user story kan worden getest aan de hand van criteria die de story zelf uitlegt.

Mensen vinden deze manier van vereisten verzamelen en het voortdurend bijschaven vaak wat chaotisch. In die gevallen kunnen user story's worden aangevuld met andere informatie en documenten.

Aanvullende informatie is meestal ofwel technisch of heeft betrekking op niet-functionele eisen en kan nalevings- of procesnormen bevatten. Alle technische en niet-functionele eisen kunnen ook worden gedefinieerd in een story format waar dat mogelijk is.

Omdat eisen voortdurend worden bijgesteld, is het belangrijk belanghebbenden regelmatig te betrekken bij gesprekken of zelfs bij verfijningsactiviteiten.

Betrek ook andere belanghebbenden hierbij, met name gebruikers, wanneer vereisten niet duidelijk genoeg zijn om naar te handelen. Belanghebbenden zijn de experts als het gaat om vereisten. Zij kunnen ideeën delen, suggesties doen en zelfs helpen om vereisten te documenteren in elke iteratie.

Verfijn de product backlog voortdurend en werk de vereisten bij naarmate de tijd verstrijkt. Vereisten zijn niet statisch en veranderen daarom in de loop van de tijd. Het van tijd tot tijd betrekken van klanten en gebruikers bij het verfijnen van de product backlog helpt bij het identificeren van nieuwe vereisten en veranderende behoeften en prioriteiten van de business.

Als Developers beginnen te werken aan incrementen in een sprint ontdekken ze vaak informatie of afhankelijkheden die voorheen onbekend waren. Deze informatie wordt gebruikt om zowel de product- als de sprint backlogs te verfijnen.

Het verfijnen van de sprint backlog betekent het opnieuw plannen van werk in de huidige sprint, maar vaak hebben de nieuw ontdekte feiten gevolgen die niet in één sprint kunnen worden afgehandeld, in welk geval product backlog-verfijning nodig is. Let wel: product backlog-verfijning is een doorlopende activiteit en niet een gebeurtenis!

7 Agile-planning en -schatting

In dit hoofdstuk zullen verschillende benaderingen van schatting en planning worden besproken, maar in het algemeen worden in Scrum twee soorten schatting- en planningstechnieken gebruikt: velocity-gedreven en commitment-gedreven. Voorstanders van een bepaalde techniek, zijn zeer gepassioneerd over hun eigen techniek.

Velocity-gedreven schatting en sprint planning probeert werk aan een sprint toe te wijzen dat gelijk is aan de hoeveelheid werk die ze in de vorige sprint hebben voltooid (story points en T-shirt maten, bijvoorbeeld).

De typische stappen bij het gebruik van velocity-gebaseerde sprint planning zijn:

- **Bereken de snelheid (velocity) van het team**. Deze kan veranderd zijn sinds de laatste keer dat dit gedaan werd – vooral aan het begin van het Scrum traject.
- Gebaseerd op de product goal – **bepaal een sprint goal**.
- **Selecteer story's** in de top van de product backlog, gelijk aan de snelheid van het team.
- **Splits verhalen op in taken** en stel vast hoelang het zal duren om elk van die taken te voltooien.

Bij commitment-gedreven sprint planning committeert het team zich aan een item voordat het wordt toegevoegd aan de sprint backlog. Zodra het totaal van de items in de sprint backlog gelijk is aan de tijd van de time-box, stopt het team en worden er geen story's meer toegevoegd aan de backlog.

Omdat het team hier van nature werk vertaalt naar hoelang het zal duren, zal opvallen dat teams de voorkeur geven aan de ideale dagen- of uren-schattingstechniek.

Een sprint planning sessie zal meestal als volgt verlopen.

1. **Bepaal de sprint goal** op basis van de product goal
2. **Selecteer een product backlog item** van boven in de product backlog
3. **Splits de story op in taken** en bepaal hoelang het zal duren om deze te voltooien
4. Zorg ervoor dat het **team zich committeert** aan de haalbaarheid van de story en voeg de story toe aan de sprint backlog
5. Bekijk hoeveel tijd er over is in de sprint time-box
6. **Herhaal** het proces totdat er geen items meer kunnen worden toegevoegd aan de sprint backlog, omdat de sprint time-box vol is

Zoals eerder gezegd, zijn de meningen zeer verdeeld over welke aanpak en welke tools het beste werken om verschillende soorten schattingen te doen. De belangrijkste technieken zullen hier worden beschreven en het is aan het team om te beslissen welke het beste werkt.

Waarom zouden er plannen gemaakt moeten worden als Agile niet over gedetailleerd plannen gaat?

Planning aan het begin van de Agile-cyclus dient slechts één doel – er is voldoende kennis nodig om het product aan het portfolio toe te kunnen voegen, om zo effectief portfolio- en productmanagement mogelijk te maken.

Het niveau van planning is zodanig dat de juiste beslissingen genomen kunnen worden voor de komende zes maanden tot een jaar en dat er minstens een indruk bestaat van wat er na dat tijdsbestek kan gebeuren en welke investeringen daarvoor nodig zijn. Het doel is om fondsen en middelen toe kunnen wijzen om de doelstellingen van het ondernemingsportfolio te bereiken.

Het antwoord op planning vooraf is dus dat het ‘net genoeg’ moet zijn om beslissingen te kunnen nemen over prioriteiten voor de business en om financiële en andere middelen te kunnen toewijzen aan initiatieven die een optimale uitwerking op de business zullen hebben.

Initiële planning is altijd gebrekkig; dat weten we van Waterval-projecten – maar ze geeft wel inzichten en wordt een katalysator voor discussie, ontdekking en tactische plannen van de onderneming als geheel.

Eén van de belangrijkste bezwaren tegen een Agile-aanpak en specifiek het gebruik van Scrum is dat de organisatie dit overzicht op hoog niveau niet heeft om de juiste strategische en tactische beslissingen te kunnen nemen. De bezwaren zijn deels juist als we als puristen kijken naar Agile en Scrum. Echter, tegenwoordig heeft iedere Agile-methode een manier om deze initiële bezwaren te overwinnen en te corrigeren.

Wanneer later in dit boek schaalvergroting aan de orde komt, zullen enkele methoden worden genoemd die als aanvulling op de Scrum-aanpak worden beschouwd.

De aanpak van alle Agile-methoden is echter om op elk niveau net genoeg te plannen om de vragen te kunnen beantwoorden die op dat niveau van planning en/of uitvoering beantwoord moeten worden.

Wees dus niet bezorgd als de initiële plannen en schattingen verkeerd blijken te zijn – dit geldt zowel voor Waterval projecten als voor Agile-projecten. Initiële plannen zijn een paal in de grond, een basislijn; een plaats van waaruit meer geleerd kan worden, meer kennis verzameld kan worden en er verbeterd kan worden naarmate de levering van de echte waarde aan klanten dichterbij komt.

Bruce Martin tekende een grafiek in zijn artikel uit 1980 *The goal: to improve credibility in the reporting of engineering progress*, in de PMI's Project Management Quarterly. Die grafiek is vandaag nog steeds geldig. Dit artikel ging niet over Agile of Scrum, maar over Waterval-projecten – wat eens te meer bewijst dat de meeste bezwaren tegen Scrum gebrekkig zijn en gebaseerd op een foute veronderstelling.

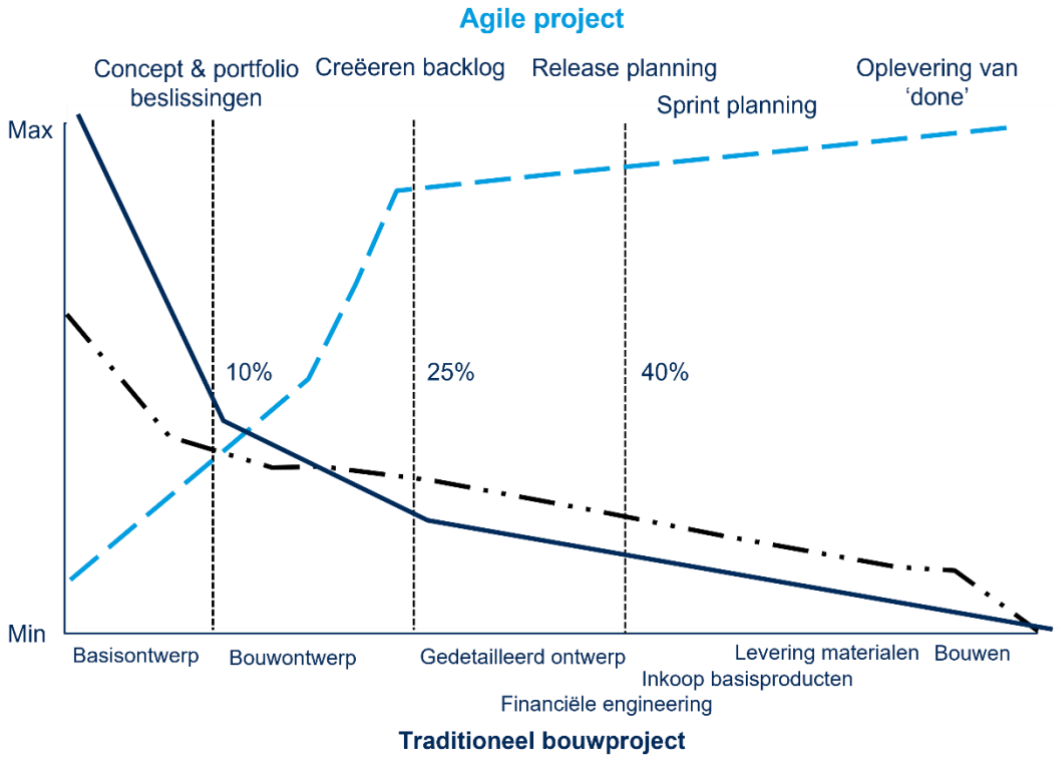
De aanname is dat aan het begin van een project gedetailleerde, Waterval-planning beter is dan hoog niveau Agile-planning (zie de aangepaste afbeelding hieronder).

Waarom die moeite doen, is de vraag. Wel, hoe vaker er een schatting op hoog niveau gedaan wordt en het portfolio verfijnd wordt, hoe beter men er in de loop van de tijd in worden. Dit is één van de voordelen van Agile schattingstechnieken – omdat een nieuw project en deliverables meestal vergeleken worden met een bekend, eerder project. Hoe vaker dat gedaan wordt, hoe meer er geleerd wordt en hoe beter dit zal lukken.

De betrouwbaarheid van een gedetailleerde planning is in een vroeg stadium van het proces twijfelachtig; hoe dichterbij de uitvoering van het werk komt, hoe betrouwbaarder de planning en raming en hoe minder behoefte er is aan metingen van kosten en schema's.

In het begin van het proces is gedetailleerde planning twijfelachtig; hoe dichter men bij de uitvoering komt, hoe betrouwbaarder de planning en schatting en hoe minder kostenbeheersing nodig is.

Figuur 12 De betrouwbaarheid van gedetailleerde planning



Afbeelding gecreëerd door EXIN gebaseerd op: Martin, B.A. (1980). *The goal: to improve credibility in the reporting of engineering progress.* Project Management Quarterly, 11(2), 14-22

Het proces van leren en verbeteren betekent dat het creëren en onderhouden van de product backlog en de roadmap ook nauwkeuriger zullen worden. Als gevolg daarvan zal het vertrouwen dat de organisatie heeft in het toekomstbeeld dat zij presenteren, toenemen.

Het creëren van dit algemene overzicht van alle projecten, inclusief schattingen, levert de input voor portfoliomanagement, dat op zijn beurt helpt de inspanningen te richten op wat van wezenlijk belang is voor de organisatie en strategische initiatieven te koppelen aan projecten die aan deze initiatieven zullen bijdragen.

Het helpt de vragen te beantwoorden: “Wat moeten we bouwen, wanneer en doet het ertoe?”

Planning helpt ook bij het aansturen van deelprojecten, zoals marketing, productbranding, verkoop en reclame en vaak ook bij het opstarten van opleidings- en bijscholingsinitiatieven, ter voorbereiding van de oplevering van het project.

Een goed plan heeft voldoende detail om planning te doen op het niveau waar het plan gebruikt gaat worden. Dit betekent dat Agile-planning iteratief is, met typische, indicatieve planningshorizonten waarin de uitvoering van de planning dichtbij is.

7.1 Wat maakt Agile-planning anders?

Professionals moeten oppassen dat ze planning niet op een traditionele manier benaderen. Conventionele projectplanning is lineair en gebaseerd op activiteiten.

In Agile wordt waarde niet op een lineaire manier geleverd, maar eerder door het voortdurend herprioriteren van de functionaliteit die de meeste waarde zal ontsluiten voor de klant.

Naast het nadeel van lineaire planning en uitvoering, door stop-start momenten telkens wanneer een vertraging invloed heeft op het kritieke pad, gaat de Waterval aanpak ervan uit dat volledige kennis van de toekomst beschikbaar is – wat niet waar is.

In waterval-projecten betekent elke vertraging in de oplevering van iets op het kritieke pad dus een vertraging in het project, dus in het leveren van waarde aan klanten.

Een ander probleem is dat volgens de wet van Parkinson projectteams de door vertragingen veroorzaakte ‘vrije tijd’ of ‘wachttijd’ zullen opvullen. Dit verschijnsel doet zich hoogstwaarschijnlijk voor, omdat de belangrijkste maatstaf in dit soort projecten is dat er wat gedaan wordt! De wachttijd wordt bijgevolg niet gebruikt voor niet-afhankelijke taken, omdat die niet de volgende op de lijst zijn. Als gevolg daarvan schuift het effect van de vertraging bijna altijd stroomafwaarts op.

Denk aan de vier waarden van Agile-teams:

Waarde 1.	**Individuen en interacties**	BOVEN	*processen en hulpmiddelen*
Waarde 2.	**Werkende software**	BOVEN	*uitgebreide documentatie*
Waarde 3.	**Samenwerking met de klant**	BOVEN	*contractonderhandelingen*
Waarde 4.	**Inspelen op verandering**	BOVEN	*het volgen van een plan*

Deze waarden maken deel uit van alles wat er gedaan wordt, dus ook van de manier waarop er gepland wordt.

Let wel: planning vindt in Agile plaats vlak voordat het werk uitgevoerd wordt. Dat is het beste moment, omdat er veel geleerd zal zijn sinds het initiële plan en het maken van de product backlog. Het iteratieve karakter van Agile en Scrum betekent dat de geleerde lessen ook zullen helpen om de juiste vragen te stellen en met het best mogelijke plan te komen.

Werk in teams, met verschillende teams op verschillende niveaus, zoals blijkt uit de bovenstaande planningshorizonten; en betrek een breed segment van belanghebbenden erbij, aangezien dat op elk niveau van plannen nuttig is. Converseer, vraag, informeer en kom met iets dat de wijsheid en het inzicht van de betrokken partijen het beste weergeeft.

Probeer niet te definiëren wat nog niet bekend is; laat dat liggen voor later. Elke iteratie van het plan zal zich ontwikkelen en meer gedetailleerd worden naarmate het werk dichterbij komt.

Plannen moeten licht zijn – volgens het Lean-principe ‘net genoeg’ om een goede beslissing te kunnen nemen. Niet meer, niet perfect, niet allesomvattend – net genoeg.

Bij het plannen is het raadzaam gebruik te maken van participerende en visuele technieken. Als de muur niet bedekt is met post-its en de borden vol gekrabbel, kan dat erop wijzen dat het team zijn best niet heeft gedaan.

Zorg er bovendien voor dat de planning goed overdraagbaar is naar het volgende niveau van deelnemers. Gooi niet iets over de muur en hoop het beste ervan. Als de partij die de volgende activiteit gaat doen niet in de kamer is, haal ze in de kamer, leg uit, discussieer, debatteer en bereik een consensus of overeenkomst.

Agile-planning is gebaseerd op de waarde die wordt gecreëerd door functionaliteit en het toekennen van prioriteit op basis van het creëren van waarde. Gedetailleerde planning wordt niet vooraf gedaan in Agile; activiteiten bestaan niet eens in Agile plannen op hoog niveau.

Deze aanpak heeft vier belangrijke voordelen;

1. Er is **geen planning op basis van activiteiten** – zo worden de valkuilen van de aanpak vermeden.
2. De aard van het te verrichten werk wordt overgelaten aan de specialist die de levering van de waarde tot stand zal brengen; hij of zij is veruit het best toegerust om te beslissen wat de beste manier is om het werk uit te voeren.
3. Omdat er geen zorgen hoeven te zijn over activiteiten, kan de **blik op de planning een niveau omhoog gebracht worden** en kan er nagedacht worden over de beste volgorde om waarde te ontsluiten voor klanten en de organisatie.
4. Omdat er geen gedetailleerd beeld is van alle activiteiten, zal **geaccepteerd moeten worden dat er een onvolmaakt beeld is van de toekomst** en voortdurend, bij elke stap in het traject, geprobeerd worden om een beter inzicht te krijgen in wat er gedaan moet worden en of de onzekerheid verminderd kan worden.

Dit laatste betekent ook dat er met de belanghebbenden gecommuniceerd wordt dat het portfolio- of roadmapperspectief op de projecten geen toezegging is maar een indicatie van de geplande voortgang en dat alles zomaar kan veranderen als de omstandigheden of prioriteiten van de organisatie of de klant veranderen.

We kunnen ons afvragen of deze aanpak veel onrust zal veroorzaken bij de klanten; klanten willen zekerheid.

Het eerlijke antwoord is *ja*, maar het is: *ja, in het begin wel*.

Waarom?

Omdat klanten betrokken blijven bij de planning gedurende de hele cyclus van waarde creëren en het is de taak van de Product Owner om hen aan boord te houden.

Eerder werd vermeld dat Scrum-teams betrokken zijn bij backlog-verfijning en het herprioriteren van items in de product backlog. De Product Owner, als vertegenwoordiger van de business, voert bij de verfijning van de backlog ook gesprekken met zijn of haar tegenhanger in de business over prioriteiten.

Bij het bespreken van de taken van het Scrum-team, werd gesteld dat zij het werk plannen voor de iteratie of sprint. Ook werd opgemerkt dat de Product Owner het werk plant en ordent in de product backlog (product niveau), om releases en iteraties zo goed mogelijk te plannen en uit te voeren. Klanten moeten de Product Owner op de hoogte houden van veranderende prioriteiten en de Product Owner moet zich ervan verzekeren dat de prioriteiten van de klant niet veranderd zijn.

Opname van een project in het portfolio van de organisatie vraagt een inschatting van de inspanning, het budget en de middelen die nodig zijn om het project te voltooien. Deze informatie wordt ook gebruikt om een prioriteit aan het initiatief toe te kennen, in relatie met andere initiatieven in de organisatie.

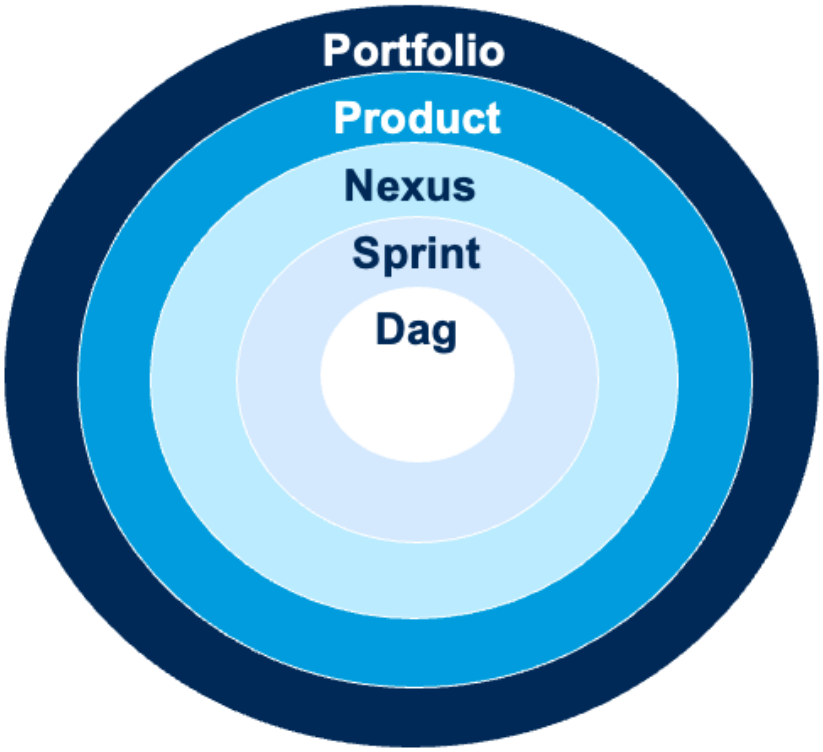
Dit komt neer op ten minste vier niveaus van schatting en planning die gedurende de levenscyclus van een project nodig zijn. In veel organisaties en bij complexe projecten kunnen dit er zelfs meer zijn – vier, vijf, of zelfs acht niveaus van planning en schatting. Het belangrijkste is dat alleen het niveau van planning en schatting wordt gedaan dat nodig is om specifieke vragen te beantwoorden.

De niveaus van verfijning van planning en schatting zijn meestal: Welke kennis is er nodig om te bepalen of een project aan het portfolio wordt toegevoegd, om een backlog te maken, om een product backlog te verfijnen, om geschaalde implementaties of releases te plannen, of om een sprint te plannen en uit te voeren?

Planning en schatting vinden ook plaats binnen een sprint: Wat gaan we in deze sprint doen en wat gaan we vandaag doen?

Planningen worden meer gedetailleerd naarmate men dichter bij de uitvoering komt.

Figuur 13 Agile gebruikt het principe 'net genoeg' planning in iedere laag



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us.

Planning en schatting op product-, release- en sprintniveau zijn al eerder aan de orde geweest, maar hoe zit het met planning en schatting vanuit een portfolio-perspectief?

Om deze vraag te beantwoorden, worden vragen worden gesteld bij het opstellen en bijwerken van het organisatie-portfolio.

Het doel van een organisatie-portfolio is het beheren van het verwezenlijken van de doelstellingen van de organisatie. De vraag is: Wat (welke initiatieven) moet worden gedaan om onze doelen en doelstellingen te bereiken? De initiatieven die zijn vastgesteld, worden vervolgens geëvalueerd en geordend.

- Wat moeten we eerst doen?
- Wat zal de grootste impact hebben?

Helaas is deze vraag niet eenvoudig te beantwoorden, omdat we altijd te maken hebben met beperkingen, meestal van middelen en budget.

De vraag wordt dan:

- Welke combinatie van dingen die we te doen hebben, zal de optimale uitwerking hebben?

Om deze vraag te beantwoorden en op dit niveau ramingen en planningen te doen, is er een aardig idee nodig van de waarde die door het initiatief wordt gecreëerd, hoelang het zal duren om die te realiseren en van de kosten en benodigde middelen. In de tweede plaats wordt er gekeken naar de onderlinge afhankelijkheden.

Het antwoord op deze vraag is in het beste geval een ‘geïnformeerde gok’, maar door vergelijkingen te maken met eerdere initiatieven en bekend werk, kan een schatting op dit niveau al heel snel oplopen tot meer dan 80% nauwkeurigheid. Voor portfolio-planning is een schatting van 80% goed genoeg!

Is het opgevallen dat het beantwoorden van deze vragen niet alleen betrekking heeft op de aard van het uit te voeren werk? Ook de snelheid van het betrokken team is van invloed!

Zolang het niet duidelijk is wie het werk zal doen, is het raadzaam te rekenen met de gemiddelde snelheid over de teams.

7.2 Wie worden bij de planning betrokken?

Dat is een goede vraag en het antwoord hangt af van het niveau van planning.

Voor portfolio-planning kan verwacht worden dat Product Owners (managers) samen met hun business sponsors (de klant) en senior en executive managers in het gesprek worden betrokken.

Bij het creëren van de product backlog zijn Product Owners, business sponsors (indien mogelijk) en in eerste instantie een brede vertegenwoordiging van de gebruikers betrokken. De toehoorders moeten voldoende inzicht hebben in de story’s die in de backlog komen en hun relatieve volgorde.

De keuze van het woord ‘volgorde’ is bewust; product backlogs maken niet alleen gebruik van belang en prioriteit om de volgorde te bepalen waarin het werk wordt gedaan. Andere factoren zoals risico, afhankelijkheden en vele andere kunnen een rol spelen.

Product backlog verfijning kan dezelfde betrokkenen hebben, maar in werkelijkheid is een brede vertegenwoordiging van gebruikers hoogstwaarschijnlijk niet nodig. Betrek ook de Developers die momenteel aan de backlog werken en de Scrum Master(s). Bijdragen vanuit de lessen geleerd door Scrum-teams, zijn van vitaal belang.

Een vergelijkbaar publiek is van toepassing voor geschaalde implementaties en sprint planning. De betrokkenheid van de klant en de gebruikers wordt minder naarmate de planning vordert in de richting van een specifieke sprint. Dat wil niet zeggen dat ze helemaal niet betrokken zijn. Scrum-teams met een hoge mate van betrokkenheid van gebruikers en klanten gedurende hun sprints, doen het veel beter dan teams die dat niet hebben. Er hoeft niet gewacht te worden tot een sprint review om feedback van gebruikers en klanten te krijgen.

Product Owners betrekken als het kan, meerdere teams van Developers en Scrum Masters bij het samenstellen, verfijnen en plannen van product backlogs.

Planning wordt steeds technischer naarmate de planning dichter bij een daadwerkelijke sprint komt. Het moet duidelijk zijn hoe verschillende teams kunnen werken aan dezelfde backlog, de afhankelijkheden en de gevolgen daarvan, hoe werk en activiteiten in de juiste volgorde geplaatst kunnen worden, welke communicatie nodig is en uiteindelijk wie wat gaat doen in een sprint. Dit is waarom een samenhangende geschaalde aanpak zo belangrijk is in een grotere en complexere omgeving.

Hoewel het geen goed idee is om alle soorten planning te zien als time-boxed gebeurtenissen, worden ze vaak wel gedaan als een time-boxed gebeurtenis.

Zowel geschaalde implementaties als sprints moeten doelen, doelstellingen en succes- of acceptatiecriteria hebben voor wat moet worden bereikt; op een geschaald implementatieniveau moet het echter wat vloeibaarder zijn. Een sprint heeft echter een duidelijk gedefinieerde sprint goal en een definition of done (DoD) voor elk increment, die meer gedetailleerde acceptatiecriteria kan omvatten.

7.3 Schattingstechnieken

Er is duidelijk behoefte aan raming bij het maken van plannen, maar de aard van het woord zegt al iets belangrijks.

Schatten is geen 100%; de kunst is te weten hoeveel goed genoeg is.

De twee factoren die in aanmerking worden genomen zijn **nauwkeurigheid** versus **inspanning**. Zoals met de meeste dingen in het leven, zal het plotten van verschillende pogingen hoogstwaarschijnlijk een typische normale verdeling (bell curve) vertonen. Eén manier om beter te kunnen schatten, is het verzamelen van meer gegevens, zodat teams er vanzelf beter in worden naarmate ze het vaker doen.

Hoe meer teamleden actief deelnemen en ervaring opdoen met schatten, hoe beter het team collectief zal worden in schatten. Alle Agile-schattingstechnieken steunen op de gezamenlijke bijdragen van het team.

Stel dat een story moeilijk in te schatten is. Het team kan dan overwegen de story op te splitsen in kleinere story’s die gemakkelijker in te schatten zijn en vervolgens de som van de kleine story’s te nemen als schatting voor de grote story. Deze techniek wordt disaggregatie (uitsplitsen) en aggregatie (samenvoegen) genoemd.

Maar wat als het team geen idee heeft?

We kunnen ervoor kiezen een expert of meerdere experts te vragen; dit zal echter niet resulteren in betrouwbare schattingen, omdat experts van buiten het team geen inzicht hebben in de vaardigheden en capaciteiten van het team. Het is echter beter dan gissen.

Teams kunnen ook proberen het huidige werk te vergelijken met werk dat eerder is gedaan. Rekenen met ideal days of story points is eigenlijk precies dat. Maar als er geen project is om mee te vergelijken, is het vragen van een externe mening de enige haalbare optie.

7.3.1 Herschatten

Bij het gebruik van story points en ideal days wordt geïmpliceerd dat zodra het werk wordt uitgesplitst, opnieuw moet worden geschat. Let wel, deze technieken worden gebruikt om het schatten van een functie of een story en alle bijbehorende complexiteit te vereenvoudigen, zonder het werk op te hoeven splitsen tot het niveau van de taken die het omvat.

Bijgevolg is het duidelijk dat wanneer taken worden onderverdeeld in de sprint planning, er een nieuwe schatting nodig is, omdat nu de som van de geassocieerde taken weer geaggregeerd kunnen worden naar de functie of story.

Er zou aangevoerd kunnen worden dat dit overbodig is, omdat het op dit niveau niet meer gaat om de story of functie. Echter, aggregatie is zinvol, omdat het team kan leren van de delta tussen de oorspronkelijke schatting en de nu meer gedetailleerde schatting en dat zal het vermogen van het team om dit soort werk in de toekomst te schatten aanzienlijk doen toenemen.

Eén van de gespreksonderwerpen in een retrospective kunnen de gevallen zijn waarin een aanzienlijke afwijking werd ontdekt tussen schattingen in het verleden, gedetailleerde schattingen en de werkelijke tijd van uitvoering.

Dit wordt specifiek aanbevolen als agendapunt, omdat het team hoogstwaarschijnlijk de Product Owner zal helpen met het verfijnen van de product backlog, direct na het afronden van de sprint.

Het opnemen van dit agendapunt helpt dus onmiddellijk bij het verbeteren van toekomstige schattingen.

Vergeet ook niet dat gedeeltelijk voltooide story’s onmiddellijk na een sprint weer aan de product backlog worden toegevoegd. Deze story’s zelf veranderen niet, maar de schatting kan veranderen omdat, omdat een deel van het werk misschien wel is gedaan. Story’s worden alleen uit de product backlog verwijderd wanneer ze helemaal klaar zijn.

In het deel over sprint planning worden enkele technieken geïntroduceerd die ook kunnen worden gebruikt bij het schatten van product backlog items.

7.4 Wat wordt of kan nog meer worden gedaan in sprint planning?

Even ter herinnering, de volgende vragen worden beantwoord tijdens de sprint planning:

- Waarom is deze sprint waardevol?
- Wat kan in deze sprint worden gedaan?
- Hoe zal het geselecteerde werk worden gedaan?

Aan het eind van de planningsactiviteit is het volgende duidelijk omschreven:

- de sprint goal;
- de product backlog items die geselecteerd zijn voor de sprint, en
- het plan voor het opleveren van items in de sprint backlog.

Hoewel eerder al een overzicht van sprint planning werd gegeven, wordt in deze paragraaf dieper ingegaan op technieken die tijdens sprint planning kunnen worden gebruikt.

7.4.1 Sprint planning

Sprint planning is een time-boxed gebeurtenis die ongeveer 1 uur duurt voor elke week van een sprint.

In de sprint planning, zal de Scrum Master de Developers coachen in het schatten van functionaliteit maar nooit beslissen over de schatting. Het Scrum-team komt met elkaar overeen om een aantal product backlog items te voltooien, die de organisatie dichter bij het bereiken van de product goal zal brengen. Deze overeenkomst wordt weerspiegeld in een sprint goal en in de items die worden toegevoegd aan de sprint backlog.

Hoe weten we hoeveel werk we kunnen aannemen?

In het begin zal er geëxperimenteerd worden met de geschatte tijd van backlog items totdat er gegevens beschikbaar zijn over de snelheid waarmee de Developers kunnen werken. Dit staat bekend als de snelheid (velocity) van het team en zal later in wat meer detail worden behandeld. Laten we ons voor nu richten op de time-boxed gebeurtenis die sprint planning heet.

De voorbereiding van een sprint begint vóór de sprint planning. De Product Owner zorgt ervoor dat de product backlog op de juiste manier is verfijnd, uitgewerkt en dat de essentiële items zo zijn geordend en gedimensioneerd dat een productieve discussie in de sprint planning sessie mogelijk is.

Het hele Scrum-team is betrokken bij de sprint planning. Het doel is om een overeenkomst te bereiken tussen de Product Owner en de rest van het team over welk werk in de sprint wordt opgenomen.

Scrum beoefenaars behandelen de termen sprint en increment vaak als hetzelfde, maar volgens de Scrum Guide, kan een sprint meerdere incrementen bevatten. Een increment wordt gedefinieerd als een concrete bouwsteen van de product goal.

Elk **increment** is een toevoeging aan de voorgaande incrementen en wordt grondig geverifieerd, zodat alle incrementen samenwerken. Om waarde te bieden, moet het increment bruikbaar zijn.

Binnen een sprint kunnen meerdere incrementen worden gecreëerd.

De som van incrementen wordt gepresenteerd tijdens de sprint review, waarmee empirisme wordt versterkt. Het is echter mogelijk dat een increment al voor het einde

van de sprint aan belanghebbenden wordt opgeleverd. De sprint review mag nooit worden beschouwd als uitgangspoort voor het opleveren van waarde.

Werkzaamheden kunnen alleen als onderdeel van een increment worden beschouwd als zij voldoen aan de definition of done van dat increment (DoD).

Deze definitie impliceert dat voor elk increment wordt gedefinieerd wat ‘done’ is, terwijl de sprint goal eerder de overkoepelende oplevering is.

Sprint planning begint met een gesprek over de ideale en breed gedefinieerde, uitkomsten van de sprint. De Product Owner geeft het team een overzicht van de items bovenaan de product backlog en herinnert hen, om de sfeer te bepalen, aan de product goal.

Dit vormt de achtergrond voor het definiëren van een eerste en breed gedefinieerde doelstelling voor de sprint. De term ‘breed gedefinieerd’ is opzettelijk, want als deze te smal wordt gedefinieerd, zal het team geen keuze hebben in wat ze gaan doen in de sprint – wat in strijd is met het Agile-principe van zelfsturende teams met de autonomie om te beslissen welk werk ze doen en hoe ze het doen.

Dit gesprek resulteert in de gedefinieerde sprint goal voor deze sprint en de sprint goals dragen bij aan de algemene product goal.

De sprint goal is een doelstelling voor een specifieke sprint en het team kan het doel bereiken door items te implementeren die in de product backlog staan. Sprint doelstellingen zijn het resultaat van een onderhandeling tussen de Product Owner en de Developers en zijn specifiek en meetbaar. Developers zorgen ervoor dat het doel realistisch is, want het is wat ze beloven tot stand te brengen.

Om de sprint goal te bepalen, biedt Roman Pichler[8] drie vragen ter overweging:

- **Waarom voeren we de sprint uit?** Waarom is het de moeite waard om een sprint te doen? Wat willen we bereiken?
- **Hoe bereiken we het doel?** Welk artefact, welke validatietechniek en welke testgroep worden gebruikt?
- **Hoe weten we dat het doel is bereikt?** Bijvoorbeeld, tenminste drie van de vijf gebruikers voeren de bruikbaarheidstest met succes uit in minder dan een minuut.

Het is ook belangrijk de volgende vragen te beantwoorden:

- **Waarom is deze sprint waardevol?** De Product Owner stelt voor hoe waarde en nut van het product verhoogd kunnen worden in de huidige sprint. Het hele Scrum-team werkt dan samen om een sprint goal te definiëren die laat zien waarom de sprint waardevol is voor de belanghebbenden. De sprint goal moet klaar zijn voor het einde van de sprint planning.
- **Wat kan gedaan worden in deze sprint?** In gesprek met de Product Owner, selecteren de Developers items uit de product backlog en nemen ze op in de

[8] https://www.Scrum.org/resources/blog/11-advantages-using-sprint-goal

huidige sprint. Het Scrum-team kan deze items in dit proces verfijnen, wat begrip en vertrouwen vergroot. Bepalen hoeveel binnen een sprint kan worden voltooid, kan een uitdaging zijn. Echter, hoe meer de Developers weten over hun prestaties in het verleden, hun capaciteit nu en hun definition of done (DoD), hoe zekerder ze zullen zijn in hun sprintvoorspellingen.

- **Hoe zal het gekozen werk worden gedaan?** Voor elk gekozen product backlog item plannen de Developers het werk dat nodig is om een increment te creëren dat voldoet aan de definition of done (DoD). Dit wordt vaak gedaan door product backlog items uit te splitsen in kleinere werk items van een dag of minder. Hoe dit wordt gedaan is uitsluitend ter beoordeling van de Developers. Niemand anders vertelt hun hoe ze product backlog items omzetten in incrementen van waarde.

Tussen deze twee sets van vragen, kan een goede sprint goal worden geformuleerd. Alles wat het Scrum-team vanaf dat moment doet, zal worden afgemeten aan hoe het bijdraagt aan de sprint goal.

De volgende stap is evalueren of het team in werkelijkheid het werk kan doen dat nodig is om de sprint goal te bereiken in de timebox van de sprint. Vaak zal het gesprek over deze sprint laten zien wat de stappen zijn in toekomstige sprints, direct na de huidige sprint.

Dit betekent dat Product Owners mogelijk items in de product backlog gaan herschikken, omdat de items in de sprint backlog de top van de lijst in de product backlog weerspiegelen en items voor de volgende sprint idealiter de volgende zijn. Dit helpt ook om de tijd die besteed wordt aan het kiezen van items voor de volgende sprint te minimaliseren, omdat de input van de Developers dan al terug te vinden is in de product backlog.

Het aantal product backlog items dat in de sprint wordt opgenomen hangt af van de capaciteit en de snelheid van het team. Als er voor het eerst met Scrum gewerkt wordt, zal dit een gok zijn, omdat het team geen gegevens heeft waarmee de snelheid van het team kan worden berekend.

Alle items die worden gekozen om op te nemen in de sprint, zijn opgesplitst in kleine, schatbare en testbare items. Ze zijn uitgewerkt voordat het team deze items toevoegt aan de sprint backlog.

Items zullen opnieuw worden geschat voordat ze worden toegevoegd aan de sprint backlog, omdat verschillende teams van Developers verschillende vaardigheden, capaciteit en snelheid kunnen hebben.

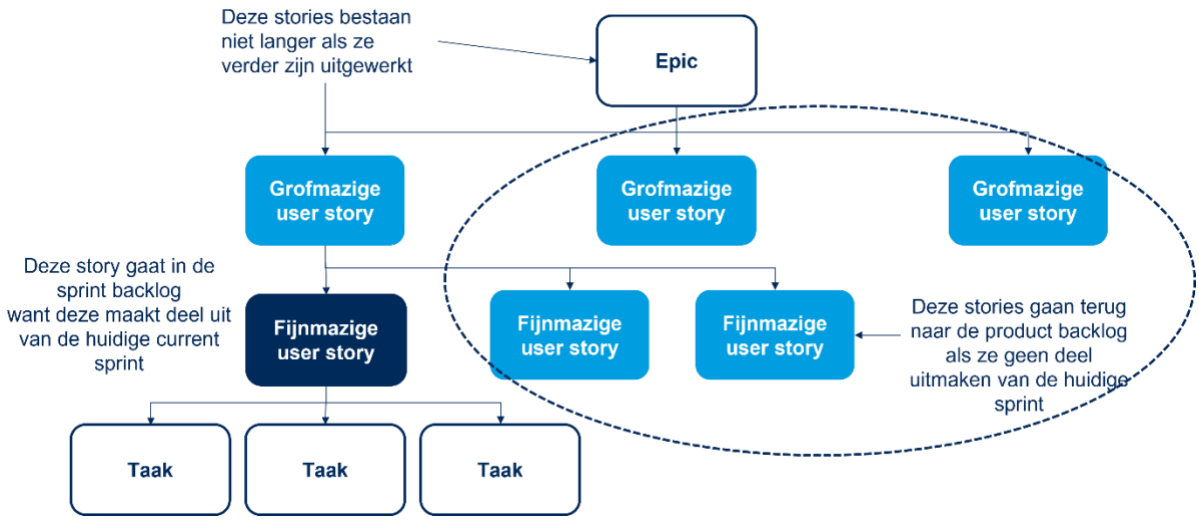
Probeer de items zo klein mogelijk te houden, zeker niet meer dan een paar dagen. Dit zal helpen om het werk op te splitsen in eenvoudige en gemakkelijk hanteerbare taken. Dit beperkt ook de work-in-progress (WIP) en maakt het eenvoudiger de voortgang te volgen met behulp van een burn-down of burn-up chart.

Let wel, user story’s mogen meerdere keren en over meerdere sprints worden uitgewerkt. Het is niet nodig om alle vertakkingen van een meer algemene story in deze sprint uit elkaar te halen, tenzij ze deel uitmaken van het werk dat in de sprint

wordt gedaan – haal alleen de vertakkingen uit elkaar waaraan in deze sprint wordt gewerkt.

Als het team in deze sprint niet werkt aan een vertakking van een uitgesplitste story, kunnen ze deze terugplaatsen in de product backlog met de juiste prioriteit en schatting.

Figuur 14 Van Epic tot taak – het uitwerken van vereisten



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2018). *Scrum Masters and Product Owners [Courseware[.* GetITrights.

Overweeg ook of er kwesties spelen in de context van het project, die van invloed kunnen zijn op de prestaties tijdens de sprint. Markeer afhankelijkheden en overweeg de implicaties.

Heel vaak is het nodig om te vertrouwen op een partij die niet tot de Developers behoort. Hoewel het beter is om deze situatie ten koste van alles te vermijden, komt het voor. Bedenk hoe deze afhankelijkheden kunnen worden opgeheven en als dat niet mogelijk is, hoe ze kunnen worden beheerd.

Een voorbeeld hiervan is een situatie waarin Developers een Cloud omgeving nodig hebben om in te werken. Aanvragen kan eenvoudig zijn, maar het krijgen is niet altijd te voorspellen. Kan er iets worden gedaan om ervoor te zorgen dat dit geen invloed heeft op de resultaten van de sprint? Zo ja, wat moet er dan gebeuren en is het mogelijk om de juiste stappen nemen?

Zodra het team een akkoord heeft bereikt over de items die in de sprint worden opgenomen, stelt het team een sprint backlog samen met de op te leveren items. De sprint backlog wordt regelmatig bijgewerkt; updates worden bij voorkeur één keer per dag gedaan om de sprint backlog zo actueel mogelijk te houden.

Het team moet het dan eens worden over een definition of done (DoD) voor elk increment dat deel uitmaakt van de sprint en het team verbindt zich aan de items die worden voltooid voor het increment en de sprint.

De DoD is de lijst van vereisten voor een succesvol increment. Sommige teams van Developers nemen het voltooien van alle backlog items op in hun definition of done (DoD), andere niet.

De definition of done (DoD) voorziet in een gedeelde opvatting over welk werk nodig is voor het increment. Als een voltooid product backlog item niet voldoet aan de definition of done (DoD), kan het niet beschikbaar worden gemaakt voor gebruik of worden gepresenteerd tijdens de sprint review. In plaats daarvan wordt het teruggeplaatst in de product backlog, voor opname in toekomstige sprints.

Als de definition of done (DoD) voor een increment een deel van een standaard van de organisatie omvat, volgen alle Scrum-teams deze, bijvoorbeeld een niet-functionele beveiligings- of kwaliteitseis. Als een vereiste geen organisatorische standaard is, creëert het Scrum-team een definition of done (DoD) die passend is voor het product en de producteisen.

Deliverables van Developers voldoen aan de definitie van done (DoD) en als meerdere Scrum-teams aan een increment werken, definiëren zij een gezamenlijke definition of done (DoD) en voldoen daaraan.

Op het moment dat een product backlog item voldoet aan de definitie van done (DoD), is een increment geboren.

Kanttekening: Teams die de MoSCoW techniek gebruiken, kiezen vaak items uit Must, Should en Could Haves. Meestal zal 80% Must Have zijn (items van hoog belang) en Should Haves (nuttig en waardevol maar niet essentieel) en 20% zal Could Haves zijn. Door deze aanpak te volgen, wordt een buffer gecreëerd in de sprint voor als er iets gebeurt dat niet is voorzien.

In een scenario waarin het team bufferitems (Should en Could Haves) opneemt, worden gewoonlijk alleen de Must Have story's opgenomen in de definition of done (DoD).

De standaardpraktijk in Scrum is om alleen items van de top van de product backlog op te nemen in de sprint backlog. Veel Scrum-teams hebben echter de DSDM-praktijk geadopteerd en als die aanpak werkt voor de organisatie, is dat ook een optie.

Bij het selecteren van items om op te nemen in de sprint backlog, is het van vitaal belang om rekening te houden met afhankelijkheden en relaties**.** Afhankelijkheden zijn vaak technisch of infrastructureel van aard. Als er een afhankelijkheid bestaat, vergeet dan niet om ook deze story's in de sprint op te nemen.

Wanneer is een story af?

Een story is af als deze voldoet aan de definition of done (DoD) voor het increment!

Story's en bijbehorende taken hebben acceptatiecriteria. Deze worden vastgesteld aan het begin van een sprint wanneer de story's worden opgedeeld in taken. Activiteiten en gerelateerde criteria kunnen veranderen naarmate ontwikkeling vordert en wanneer er meer ontdekt wordt over een functionaliteit of vereiste, maar

al het werk wordt getest aan de hand van acceptatiecriteria voordat het beschikbaar wordt gesteld voor gebruik.

Bovenstaande beschrijving klinkt nogal als de XP ‘test-driven development’ (test gedreven ontwikkeling) die gebruikt wordt bij softwareontwikkeling. In dit geval zou de laatste test van ‘done’ zijn om te vragen of de software inzetbaar, bruikbaar en veilig is.

Done is niet alleen voldoen aan de gestelde individuele criteria, maar betekent ook dat wat het team in het increment heeft voltooid, klaar is voor gebruik en bruikbaar is.

Kanttekening: Als het werk dat wordt gedaan de ontwikkeling van software is, betekent ‘done’ dat wat wordt opgeleverd de continual integration/continual delivery (continue integratie/continue levering, CI/DC) pijplijn in kan en het increment vrijwel onmiddellijk beschikbaar kan worden gemaakt.

CI/CD zijn concepten die worden gebruikt bij voorgedefinieerde tools en geautomatiseerde workflows om softwarecode zo snel mogelijk naar de live omgeving te brengen. De overkoepelende discipline die wordt gebruikt om software geautomatiseerd te testen en correct in te zetten, wordt vaak DevOps genoemd. Het integreren van DevOps en Scrum is een prachtige manier om waarde te creëren voor gebruikers en klanten, liever vroeger dan later.

Zodra het team het eens is over wat er in de sprint backlog komt, wordt de rest van de sprint planning besteed aan het uitsplitsen van het werk, het inschatten van taken en acceptatie van werk dat bij de user story hoort.

De taak van de Product Owner in het laatste deel van de sprint planning is om vragen te beantwoorden en acceptatiecriteria te verduidelijken. De Product Owner heeft geen rol in het evalueren hoe het team het werk doet; de inbreng van de Product Owner is beperkt tot het relatieve belang van backlog items.

Scrum Masters zorgen ervoor dat nieuwe kwesties en zorgen die tijdens de bijeenkomst naar voren kwamen, of veronderstellingen of afhankelijkheden die tijdens de planning werden ontdekt, worden vastgelegd en beter begrepen, liever vroeger dan later.

7.4.2 Dimensioneren van items

Schattingstechnieken werden eerder genoemd in de paragraaf over het creëren en verfijnen van de product backlog. Vergelijkbare technieken worden gebruikt bij het uitwerken van story’s in zowel de sprint als de product backlog.

Hoewel backlog items worden geschat en gedimensioneerd tijdens product backlog verfijning, worden de items in de sprint backlog opnieuw geëvalueerd in het licht van wat nu bekend is. Houd rekening met wijzigingen in de vereisten, de omgeving en het vermogen en de capaciteiten van de Developers bij het opnieuw evalueren van backlog items.

De juiste, realistische omvang bepalen van items is heel belangrijk; anders is het team misschien niet in staat om aan het einde van de sprint een 'done' op te leveren.

Vaak worden items opgenomen in de sprint backlog, maar blijkt tijdens het uitwerken van de taken dat de geschatte tijd of inspanning onrealistisch waren. Het team kan de samenstelling van de sprint backlog opnieuw bekijken en onderhandelen met de Product Owner dat items terug worden geplaatst in de product backlog als ze waarschijnlijk niet tijdens de sprint kunnen worden voltooid. De ruimte die overblijft, kan worden gevuld door andere, kleinere product backlog items in de sprint backlog op te nemen.

Bedenk wel dat als dit gedaan wordt, de sprint goal en de DoD's kunnen worden beïnvloed en mogelijk moeten worden herzien. Er worden bij voorkeur alleen wijzigingen aangebracht die geen invloed hebben op de sprint goal. Maar omdat het team nog steeds bezig is met plannen, kan het herzien van een sprint goal onvermijdelijk zijn.

Let wel: Als de sprint planning eenmaal klaar is, breng dan geen veranderingen aan die de sprint goal zullen beïnvloeden. Er zijn verschillende manieren om items te dimensioneren en deze methoden zullen hier kort worden beschreven.

7.4.3 Story points

Hoewel er geen officiële definitie is van wat een story point inhoudt, is een story point in algemene termen een relatieve meeteenheid, waarbij werk dat moet worden uitgevoerd wordt vergeleken met soortgelijk werk dat in het verleden werd uitgevoerd.

Story points zijn dus relatief en worden samen met de snelheid van een Scrum-team bekeken. Deze schattingsmethode beschermt beter tegen veranderingen in de werkomgeving.

Stel dat het werk aan het nieuwe item, in vergelijking met een deel van het eerder verrichte werk, half zo lang of drie keer zo lang zal duren. De eenheid die voor de vergelijking wordt gebruikt, moet echter dezelfde blijven. Het is beter om het middelste getal of het getal net eronder als referentie-eenheid kiezen en niet het kleinste getal. Dus als er bijvoorbeeld een schaal van 1 tot 10 gebruikt wordt, is de referentie-eenheid idealiter 3 of 4. Ook al zou de normale schaal 1 tot 10 zijn, toch kan het voorkomen dat er een story is van 20 story points; dit geeft enkel aan dat het een epic is die verder opgesplitst moet worden.

Op product backlog niveau is het voor Product Owners veel gemakkelijker om deze techniek te gebruiken, omdat zij vaak niet voldoende technisch onderlegd zijn om andere eenheden, zoals uren, te gebruiken.

Varianten van story points zijn onder meer het gebruik van T-Shirt-maten. Onderstaande tabel kan gebruikt worden als referentie voor een typische 1 tot 10 story points schaal:

Tabel 1 – T-shirt maten in vergelijking met story points

T-shirt maat	Aantal story points
XS	1 story point
S	2 story points
M	3 story points
L	5 story points
XL	8 story points
XXL	13 story points
XXXL	21 story points

Let wel: de Fibonaccireeks wordt hier als een relatieve referentie gebruikt.

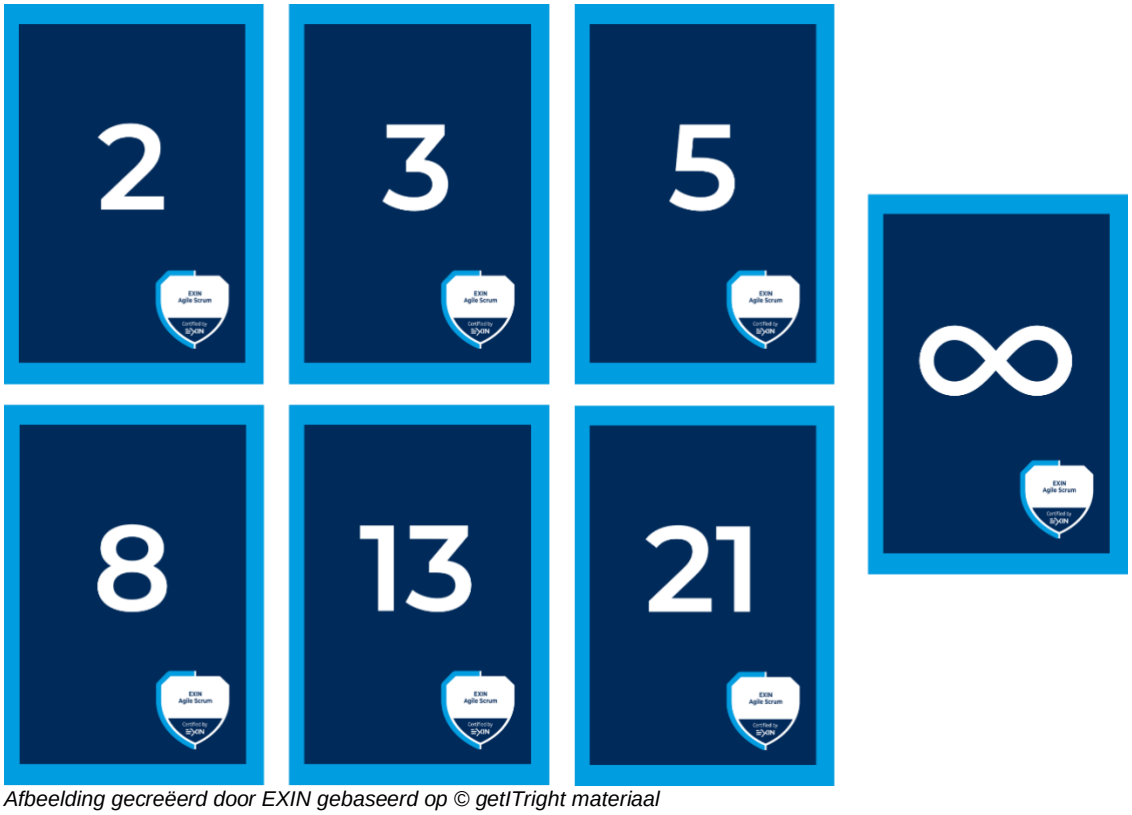
7.4.4 Planning of scrum poker

Planning poker lijkt eigenlijk meer op het spelen van Uno of Snap dan op het spelen van poker.

Elke Developer krijgt een stel kaarten met de Fibonaccireeks. Wanneer hen wordt gevraagd naar hun schatting van de omvang van het werk, legt elk teamlid de kaart op tafel met de waarde van hun schatting of het eerstvolgende hogere getal.

Opmerking: De Scrum Master en Product Owner zijn bij voorkeur niet betrokken bij het schatten; het gaat om de input van de Developers. Product Owners doen schattingen bij het toevoegen van items aan de product backlog; dit is een gelegenheid voor hen om beter inzicht en inbreng te krijgen van de mensen die het werk doen.

Figuur 15 Scrum Poker kaarten



Afbeelding gecreëerd door EXIN gebaseerd op © getITright materiaal

Het doel hier is te peilen in hoeverre teamleden het oneens zijn over de inspanning die nodig is om de activiteit of de story uit te voeren.

Als de spreiding klein is, is het redelijk eenvoudig om tot een consensus te komen – vaak wordt, uit voorzichtigheid, het hoogste getal gekozen.

De echte waarde ligt echter in de uitschieters. Bijvoorbeeld, als de meeste teamleden 3 of 5 story points zeiden, maar iemand in de groep zei 20, dan zijn er twee mogelijkheden. Ofwel begrijpt de persoon met de uitschieter niet de omvang en context van het werk dat gedaan moet worden, ofwel weet zij of hij iets dat de rest van het team niet weet.

Uitschieters worden altijd besproken door te vragen: *Waarom zeg je dat?*

Het eerste probleem is eenvoudig; door de context uit te leggen, zal de persoon waarschijnlijk besluiten de schatting in de volgende spelronde te herzien en als dat zo is, speel nog een ronde.

De tweede situatie leidt vaak tot een interessante discussie en nieuwe inzichten die noch de Product Owner noch de andere teamleden wisten of begrepen. Dit gesprek leidt meestal tot het ontdekken van nieuwe afhankelijkheden en risico's en vaak tot het creëren van extra story’s die aan de backlog worden toegevoegd.

Blijf rondes spelen tot een nauwe groepering van waarden op tafel ligt.

Zoals gezegd, als er tien mensen rond de tafel zitten en acht van hen schatten de taak op 3 en drie van hen schatten deze op 5 – ga dan voor 3. Als de helft 3 zegt en de andere helft 5, ga dan voor 5. Verspil geen tijd aan pogingen om iedereen op één lijn te krijgen; zolang de schattingen overeenkomen, is de uitkomst aanvaardbaar.

7.4.5 Ideal days of ideal hours

Schatten in ideal days of hours maakt gebruik van werkelijke tijd als indicator voor de inspanning. Een ideal day voldoet aan de volgende voorwaarden:

- Alles wat nodig is, is beschikbaar wanneer aan het werk wordt begonnen.
- De story die geschat wordt, is de enige waaraan wordt gewerkt.
- Er zullen geen onderbrekingen zijn.

Nogmaals, de schatting is niet alleen van het werk, maar van het werk **voor dat team**. Niet alle teams hebben dezelfde capaciteit of snelheid.

Overigens kunnen de getallen op de pokerkaarten net zo goed dagen of uren voorstellen, dus het spelen van Scrum poker kan ook worden gebruikt met ideal days en ideal hours.

7.4.5.1 Maar waarom ideal days of hours?

De schatting is gebaseerd op het aantal dagen/uren inspanning dat het team nodig zou hebben om een story af te krijgen als ze zonder onderbrekingen zouden werken. Dit vertegenwoordigt dus de ideale omstandigheden, zonder afleidingen, onderbrekingen en zonder afwisseling van taken. Dergelijke onderbrekingen vragen meestal ongeveer 2 uur per dag, wat betekent dat de ideal day een gemiddelde van 6 uur heeft.

De realiteit is dat zelfs als er gewerkt wordt met story points, dit op een bepaald moment moet worden omgewerkt naar tijd. Sommigen zullen aanvoeren dat als de snelheid van het team wordt gemeten met story points, dit niet hoeft; maar dat is semantiek. Zelfs als er gezegd wordt dat een team 60 story points kan afmaken in één sprint, dan nog is een sprint een time-boxed gebeurtenis, dus in feite hebben wordt er al teruggerekend naar tijd.

De echte waarde van story points, tenminste in het begin, is het veranderen van de cultuur in de organisatie. In de meeste organisaties wordt een gegeven tijdlijn standaard gezien als een toezegging. Dus door story points te gebruiken, wordt vermeden dat iemand zegt: “Je zei dat het maar 8 uur zou duren om dit te doen.”

Gebruik nooit ideal days of hours in gesprek met klanten. Zij zullen dat opvatten als een toezegging.

7.4.6 Snelle schatting met behulp van post-its

Een door Roman Pichler beschreven techniek kan worden gebruikt als de teams in tijdnood komen door een bepaalde schattingstechniek te gebruiken.

Schrijf de Fibonaccireeks op een bord met een behoorlijke ruimte tussen de getallen, geef elk teamlid een paar items om te schatten, vraag hen die op een post-it te schrijven en ze onder het getal te plakken dat zij geschikt vinden.

Als iedereen klaar is, laat alle teamleden het bord bekijken en de punten aanwijzen waarvan zij denken dat ze problematisch zijn en niet correct ingeschat. Ofwel de post-it moet verplaatst worden, ofwel de schatting moet herzien worden.

Een alternatief is dat het team er een kort gesprek over voert alvorens het item naar een betere plek op het bord te verplaatsen.

Raad de teamleden aan niet te kritisch te zijn over de schattingen; dat kan er namelijk toe leiden dat elke post-it wordt verplaatst of bediscussieerd. Dat is hier niet de bedoeling.

Deze techniek zal minder nauwkeurig zijn, maar stelt het team in staat snel veel gedaan te krijgen.

We zullen merken dat het gebruik van deze techniek in de loop van de tijd betere resultaten geeft, naarmate het team meer ervaring heeft.

7.4.7 Andere dingen die we moeten weten

Welke techniek er ook gebruikt wordt om story's of taken in te schatten, in het begin zal het team er hoogstwaarschijnlijk naast zitten. Maar hoe vaker dit gedaan wordt, hoe beter de schattingen zullen worden.

Sommigen zeggen dat alleen degenen met veel kennis van een taak of story mogen schatten, maar het hele team erbij betrekken, levert vaak nieuwe inzichten op. Ten tweede helpt dit het team meer inzicht te krijgen en draagt het uiteindelijk bij tot leren en groei.

Als een teamlid echt geen idee heeft, laat hem of haar dan afzien van het schatten. Niet elke beurt hoeft gespeeld.

7.4.7.1 Maar wat als we niet weten hoe het werk in te schatten?

Dan is er meer informatie nodig – zo eenvoudig is het. Situaties als deze ontstaan vaak wanneer een team te maken heeft met een nieuw type werk.

Het uitzoeken kan een nieuwe story worden die wordt afgerond voordat het werk aan het specifieke item kan worden voortgezet. Voeg een story toe aan de backlog om erachter te komen.

Dit wordt vaak gedaan door een model of prototype te maken, door onderzoek te doen en met mensen te praten die enige ervaring hebben.

7.4.7.2 Hoe zit het met niet-functionele eisen?

Wat is een niet-functionele eis? Mensen zien deze vaak als niet-gebruikerseisen, of technische of infrastructurele vereisten. Maar dat is niet hoe ze worden gedefinieerd in Scrum.

Een niet-functionele eis is een beperking op systeem- of productniveau die geen betrekking heeft op functionaliteit. De eis verwijst naar eigenschappen, zoals prestatie, nauwkeurigheid, portabiliteit, herbruikbaarheid, onderhoudbaarheid, interoperabiliteit, capaciteit en is voorwaarde voor product backlog items om volledig geaccepteerd te worden door de belanghebbenden**.** Niet-functionele eisen hebben vaak betrekking op kwaliteitscriteria, maar niet altijd.

Vaak hangt de acceptatie van functionele eisen af van een niet-functionele eis. Bij het schatten van functionele eisen, worden alle afhankelijke niet-functionele eisen ook overwogen.

Wanneer een niet-functionele vereiste op een Scrum of Kanban bord geplaatst wordt, zal het lijken alsof deze steeds terug op het bord geplaatst wordt. Dus, als er een vereiste is die steeds terugkomt, omdat andere items ervan afhankelijk zijn, is het hoogstwaarschijnlijk een niet-functionele vereiste.

Een aantal van deze niet-functionele eisen worden in elke sprint opgenomen, omdat het product of de dienst onbruikbaar of gebrekkig is hij 'done' is, maar niet aan deze eisen voldoet. Het is daarom belangrijk om ervoor te zorgen dat afhankelijke niet-functionele eisen worden opgenomen in de definition of done (DoD).

7.4.8 Andere story’s zijn user story’s geschreven voor andere belanghebbenden

Technische vereisten, zaken waar gebruikers of klanten niet om vragen maar die nodig zijn om andere items te laten werken, zijn gewoon een ander soort user story. Hier zijn de gebruikers intern, ze maken deel uit van het organisatorische systeem dat de uitkomst voor de gebruiker of klant mogelijk maakt.

In Lean zouden deze vereisten waarschijnlijk worden beschouwd als dat ze geen waarde toevoegen. Ze zijn noodzakelijk, niet omdat de klant er om heeft gevraagd of ervoor wil betalen – maar als ze worden weggelaten, zal er niet geleverd worden waar de klant om heeft gevraagd.

Sommige mensen verwijzen graag naar deze technische story’s als proces-story’s, infrastructuur-story’s en een heleboel andere namen. In Scrum zijn het allemaal user story’s – alleen de gebruiker is anders.

7.5 Verbeteringsactiviteiten als onderdeel van sprints

7.5.1 Hindernissen & Theory of Constraints (TOC)

Een deel van de verantwoordelijkheid van de Scrum Master is het helpen wegnemen van hindernissen die het Scrum-team tegenkomt. Een hindernis is alles wat een teamlid in de weg staat om op volle capaciteit te werken.

De Scrum Master hoeft niet te wachten tot iemand hem of haar informeert over een hindernis of beperking. Goede Scrum Masters maken gebruik van de POOGI-techniek die hieronder wordt uitgelegd, om proactief beperkingen te identificeren.

Mensen denken vaak aan hindernissen als iets dat werk onmogelijk maakt. Hoewel dat waar is, houden de meeste hindernissen het werk niet volledig tegen, maar kan het een obstructie of bottleneck zijn. Scrum-teams en met name Scrum Masters kunnen veel leren van de Theory of Constraint (TOC) zoals gedefinieerd door Dr. Eliyahu M. Goldratt in zijn boek uit 1984 getiteld 'The Goal'.

Het uitgangspunt van Dr. Goldrat is dat organisaties en teams kunnen worden gemeten aan de hand van variaties van drie maatstaven: doorvoer, operationele kosten en voorraad. Doorvoer kan worden gezien als de snelheid waarmee een systeem waarde genereert. Operationele kosten en voorraad samen kunnen worden gedefinieerd zoals Lean work-in-progress (WIP) zou definiëren. WIP zal later in wat meer detail worden behandeld.

Voordat een doel kan worden bereikt, moet ook worden voldaan aan noodzakelijke voorwaarden, zoals veiligheid, kwaliteit en wettelijke verplichtingen.

De meeste bedrijven willen winst maken door de verwerkingscapaciteit te verhogen, de voorraden te beperken en de bedrijfskosten te verlagen.

Wat heeft dit te maken met het managen van hindernissen en beperkingen in Scrum?

7.5.2 Focus op vijf stappen

De Theory of Constraints rust op de vooronderstelling dat de snelheid waarmee een doelgericht systeem het doel bereikt, door tenminste één beperking wordt gehinderd. Als er geen beperking zou zijn, zou de output van het systeem oneindig zijn, wat onmogelijk is.

Alleen door de flow door de beperking te verhogen kan de totale doorvoer worden verhoogd en de beste manier om dit te doen is door vijf eenvoudige stappen te volgen:

1. Identificeer de beperking(en).
2. Bepaal hoe de beperking(en) van het systeem kan/kunnen worden benut.
3. Maak al het andere ondergeschikt aan bovengenoemd(e) besluit(en).
4. Ondervang de beperking(en) van het systeem.
5. Als in de vorige stappen een beperking is opgeheven, ga direct terug naar stap 1.

De vijf focusstappen hebben als doel continue verbetering en zijn gericht op de beperking. Dit staat bekend als het proces van continue verbetering (Process of Ongoing Improvement, POOGI).

Maar wat wordt bedoeld met woorden als benutten, ondergeschikt maken en ondervangen?

In stap twee betekent het **benutten** van een beperking dat we, zonder een nieuwe investering te doen, begrijpen hoe het best mogelijke resultaat kan worden bereikt terwijl de beperking van kracht blijft.

Ondergeschikt betekent dat het werk zo ontworpen wordt dat het binnen deze grenzen kan worden gedaan. Het is van essentieel belang dat het werk blijft doorstromen – zelfs als dit betekent dat andere delen van het systeem worden vertraagd. Het evalueren van de doeltreffendheid van deze inspanningen is gemakkelijk waar te nemen door te kijken naar de flow en kan worden gedaan door WIP te beperken.

Dit zal tijd opleveren om de hoofdoorzaak, niet de symptomen, te identificeren en een manier te vinden om deze hoofdoorzaak(en) voorgoed weg te nemen. Zodra deze maatregelen zijn uitgevoerd, zal de beperking zijn opgeheven. Het is zeer waarschijnlijk dat een nieuwe beperking zal opduiken, waardoor de cyclus opnieuw begint.

Let er wel op dat de activiteiten die in stap twee zijn opgelegd om de beperking te benutten, kunnen worden opgeheven, aangezien de beperking nu niet meer bestaat.

De kans dat teams geruime tijd werken met beperkingen die alleen worden benut en niet worden ondervangen, is vrij groot. Het oplossen van sommige problemen vraagt investeringen en de problemen zijn vaak complex en groot genoeg om te worden uitgevoerd als een project of een initiatief tot verbetering.

7.5.3 De Continuous Improvement Backlog (CIB)

Hoewel Scrum niet specifiek een backlog van verbeteringen kent, hebben de beste Agile-organisaties deze wel.

Scrum stelt dat verbeteringen worden gedefinieerd als product backlog items in een product backlog, maar wat gebeurt er met verbeteringen die niet tot een specifieke product backlog behoren?

Veel Scrum experts stellen dat een backlog van algemene continue verbetering een plaats heeft in Agile-organisaties. Als een verbetering duidelijk bij een product hoort, is de product backlog de beste plaats om deze vast te leggen.

Maar als productgerelateerde verbeteringen worden opgenomen in een specifieke product backlog, welke items behoren dan tot de algemene Continual Improvement Backlog (CIB)?

Er zijn drie bronnen voor het verbeteren van product backlog items:

- **Resultaten van audit, continue verbetering van de dienstverlening (continual service improvement, CSI), of Kaizen** (empirisch leren en verbeteren) die niet productspecifiek zijn;
- **Beperkingen** die worden benut en die product- of systeem overschrijdend werken (bv. niet-functionele eisen);
- **Geïdentificeerde technische schuld** in de organisatie, haar systemen en processen en initiatieven om technische schuld af te lossen (zoals gedaan wordt in zowel Lean als DevOps en geschikt om ook te worden toegepast in een Scrum omgeving).

Het is niet de bedoeling van dit boek om deze initiatieven te onderzoeken; het is wel nuttig om te verduidelijken wat wordt bedoeld met het laatste punt op de lijst.

7.5.4 Technische schuld

Als de term opgezocht wordt op Internet, levert dat een definitie op die nauw verband houdt met softwareontwikkeling:

> Technische schuld [...] is een concept in softwareontwikkeling dat de impliciete kosten weergeeft van bijkomend werk, door nu een gemakkelijke (beperkte) oplossing te kiezen in plaats van een betere aanpak te gebruiken die meer tijd vergt.[9]

Maar, alle organisaties hebben technische schuld. Eenvoudig gezegd is technische schuld de totale som van al die dingen waarvan bekend is dat ze fout zijn in de organisatie en waarvan meestal gezegd wordt: Dit wordt opgelost zodra daar tijd voor is.

De realiteit is meestal dat de fout nooit zal worden verholpen.

Het probleem is dat al deze halfbakken, ondermaats presterende, met beperkingen beladen zaken zich langzaam opstapelen tot ze één grote beperking worden die een aanzienlijke negatieve impact kan hebben.

Er kan beter niet gezegd worden dat dit later opgelost zal worden, maar er moet tijd vrijgemaakt worden om het later op te lossen.

Dit is waar de Continual Improvement Backlog (CIB) van pas komt. Zodra het gezegd wordt, wordt het vastgelegd. Net als elk andere product backlog, heeft de CIB een Product Owner en in iedere sprint wordt iets gedaan om de items in de CIB aan te pakken.

7.5.5 Increment van verbetering

Zoals eerder vermeld, is een regel in Scrum dat in elke sprint minstens één increment wordt opgeleverd. Dit geldt ook voor de CIB.

Veel bedrijven kiezen ervoor om alle Scrum-teams incrementen van de CIB op te laten leveren. Sommige bedrijven hebben speciale Scrum-teams die uitsluitend werken aan CIB-items. Ongeacht de gekozen aanpak, is het opleveren van incrementen van verbetering één van de manieren waarop duurzame Agile-bedrijven worden gebouwd.

[9] https://en.wikipedia.org/wiki/Technical_debt

8 Wat gebeurt er nog meer tijdens een sprint?

Zoals eerder vermeld, beginnen sprints met sprint planning. Daarna wordt het werk gedaan om één of meer incrementen te voltooien. Vervolgens zijn er een sprint review en sprint retrospective. Gedurende de sprint heeft het Scrum-team daily scrums, waarin ze elkaar bijpraten en de planning aanpassen.

Er is al veel aandacht besteed aan sprint planning en meer inzicht daarover is gegeven in de paragrafen over planning en schatting.

De volgende sectie betreft de overige sprint-activiteiten en enkele gedachten over het gebruik van Scrum met DevOps.

8.1 Meer over de daily scrum

Elke dag op hetzelfde tijdstip vindt er een 15 minuten durende time-boxed bijeenkomst plaats waar Developers het werk van de komende dag plannen. De bedoeling is om de samenwerking en prestaties van het team te optimaliseren door het werk dat sinds het begin van de Sprint is gedaan te inspecteren, werk te plannen en voorspellingen te doen over het werk dat nog rest.

Zoals eerder vermeld, geeft de daily scrum iedere Developer gelegenheid drie vragen beantwoorden:

1. Gisteren heb ik...
2. Vandaag ga ik ...
3. Ik kon het niet, omdat ....

Hoewel deze vragen goed zijn en bruikbare inzichten kunnen opleveren, hebben ze voor velen geleid tot de misvatting dat dit alles is in de daily scrum. Het is niet verplicht om deze drie vragen te stellen en vragen die relevanter zijn voor het team, zijn ook prima. Dit is geheel aan het team; de vragen zijn slechts een middel om een gesprek op gang te brengen.

Inspectie en aanpassing zijn de kern van Scrum en de daily scrum is precies dat.

Developers gebruiken de daily scrum om de voortgang richting sprint goal te verifiëren en om te inspecteren hoe de trend is van het voltooien van het werk in de sprint backlog. Het gaat erom dat het Scrum-team de vinger aan de pols houdt en in staat is te reageren als realiteit en planning niet op één lijn liggen.

Vergeet niet dat het Scrum-team zelfsturend is en veel daarvan is het resultaat van het feit dat het team tijdens de sprint op de juiste manier reageert of acteert op veranderingen, uitdagingen of problemen en dat ze dat snel kunnen doen.

De Developers kunnen onmiddellijk reageren en de passende actie bepalen.

Let wel: De daily scrum is geen gedetailleerde planningsvergadering; de daily scrum is gericht op het identificeren van afhankelijkheden, uitdagingen en problemen. Vaak is het mogelijk om direct een passend antwoord te vinden, maar als dat niet mogelijk

is, wordt het vaststellen van de juiste actie onderdeel van het dagelijkse werk van één of meer teamleden. De leden van een Scrum-team komen vaak direct na de daily scrum bij elkaar voor het bespreken van details, of om de rest van het werk van de sprint aan te passen, of opnieuw te plannen.

De daily scrum verbetert de communicatie en is één van de belangrijkste middelen voor inspectie en aanpassing in Scrum.

Daily scrums minimaliseren ook de behoefte aan andere bijeenkomsten of maken ze overbodig. Ze identificeren belemmeringen voor het bereiken van de sprint goal. Ze zorgen ervoor dat iedereen op de hoogte is, dat beslissingen snel worden genomen en dat actie snel volgt.

De Developers zijn eigenaar van de daily scrum. Zij bepalen de agenda en het format. Hoewel de Scrum Master vaak de taak heeft om de bijeenkomst te faciliteren en ook om de voortgang bij te houden en bij te werken op een Kanban of burn-down chart, het is niet hun vergadering.

Het is de verantwoordelijkheid van de Scrum Master dat de daily scrum niet langer duurt dan 15 minuten. Als een probleem wordt geïdentificeerd en de oplossing niet duidelijk is, zal de Scrum Master vragen wie erbij betrokken moet worden. Als diegene beschikbaar is en tot een conclusie kan worden gekomen, zal de Scrum Master ervoor zorgen dat de daily scrum daarna verder gaat. Dit is de enige reden voor het onderbreken van de bijeenkomst door iemand anders dan een Developer.

8.2 Reviews en retrospectives

8.2.1 Meer over sprint reviews

De sprint review wordt gehouden aan het einde van de sprint om het increment te inspecteren en wijzigingen aan te brengen in de product backlog, als dat nodig is.

Houd er rekening mee dat de sprint review geen release- of implementatiepoort is, of een bijeenkomst om de status vast te stellen of iets goed te keuren. De bijeenkomst is bedoeld om feedback te genereren en samenwerking te bevorderen.

Als een sprint meer dan één increment heeft, wordt het increment geïmplementeerd of beschikbaar gesteld zodra het klaar is.

Nogmaals, de sprint review gaat over inspecteren en aanpassen.

Tijdens de sprint review bespreken het Scrum-team en de belanghebbenden wat er gedaan is tijdens de sprint en of er iets veranderd is dat niet volgens plan was.

Er wordt getoond wat gedaan is en hoe de incrementen van de sprint voldoen aan de definition of done (DoD).

Belanghebbenden werken dan samen om vast te stellen wat vervolgens kan worden gedaan om de waarde te optimaliseren.

De Scrum Master zorgt ervoor dat de bijeenkomst plaatsvindt en dat de aanwezigen het doel ervan begrijpen. Zij of hij zorgt er ook voor dat de vergadering binnen de time-box blijft.

De deelnemers van deze gebeurtenis zijn het Scrum-team en andere relevante belanghebbenden, meestal uitgenodigd door de Product Owner, zoals gebruikers, klanten, management.

Het is aan de organisatie om te beslissen welk format de sprint review aanneemt, gebaseerd op haar unieke behoefte, maar hier zijn enkele voorstellen van ervaren Agile Coaches over wat er gedaan kan worden in een sprint review:

- De Product Owner legt uit welke product backlog items zijn gedaan (Done) en welke niet zijn gedaan;
- Developers bespreken wat er goed ging tijdens de sprint, tegen welke problemen ze aanliepen en hoe die problemen werden opgelost;
- De Developers laten afgerond werk zien en beantwoorden vragen over het increment;
- De Product Owner bespreekt de product backlog zoals die er nu uitziet. Hij of zij geeft, als dat nodig is, verwachte doel- en opleveringsdata, gebaseerd op de voortgang tot nu;
- De hele groep bespreekt wat er nu gedaan gaat worden, zodat de sprint review waardevolle input oplevert voor de volgende sprint planning;
- Nagaan of de markt, of het potentiële gebruik van het product is veranderd en wat de meest waardevolle volgende stap is,
- Evalueren van het tijdschema, het budget, de potentiële mogelijkheden en de markt voor de volgende verwachte releases van functionaliteit en mogelijkheden van het product.

Het resultaat van de sprint review is een herziene product backlog waarin de product backlog items verwacht voor de volgende sprint, zijn vastgelegd. De product backlog kan ook in het geheel worden aangepast om tegemoet te komen aan nieuwe kansen of noodzakelijkheden.

8.2.2 Meer over sprint retrospectives

De sprint retrospective vindt plaats na de sprint review en voor de volgende sprint planning. Het is gewoonlijk een maandelijkse, drie uur durende bijeenkomst waar het Scrum-team bespreekt wat goed ging, wat verbeterd kan worden, wat vermeden moet worden in de toekomst en welke gebieden waarschijnlijk deel uitmaken van de volgende sprint.

De Scrum Master zorgt ervoor dat de gebeurtenis plaatsvindt en dat de aanwezigen het doel ervan begrijpen. Dit is opnieuw een gelegenheid voor het Scrum-team om te inspecteren en aan te passen door verbeteringen te identificeren.

Alle leden van het Scrum-team zijn aanwezig.

De Scrum Master moedigt het Scrum-team aan om hun proces en werkwijzen te verbeteren, om ze effectiever en prettiger te maken voor de volgende sprint. Tijdens

iedere sprint retrospective, plant het Scrum-team manieren om de kwaliteit van het product te verhogen door werkprocessen te verbeteren, of door de definition of done (DoD) aan te passen, als dat kan en niet in strijd is met product- of organisatiestandaarden.

Tegen het einde van de sprint retrospective, zal het Scrum-team verbeteringen geïdentificeerd hebben, die ze zullen implementeren in de volgende sprint. Het implementeren van deze verbeteringen in de volgende sprint is de aanpassing naar aanleiding van de inspectie van het Scrum-team zelf.

Hoewel verbeteringen op elk moment kunnen worden doorgevoerd, biedt de sprint retrospective een formele gelegenheid om te focussen op inspectie en aanpassing.

Veel Scrum-teams maken een backlog van verbeteringen die niet gemakkelijk kunnen worden doorgevoerd en nemen verbeteringsitems op in elke sprint, zodat er in elke sprint product increment(en) en verbeteringsincrement(en) zijn.

9 Complexe, grootschalige product backlogs

We weten nog: Elk product heeft maar één product backlog. Maar wat als het product groot en complex is en er veel Scrum-teams nodig zijn om de productvisie te realiseren?

De werkzaamheden van meerdere teams in één backlog bijhouden is ingewikkeld.

9.1 Methoden voor schaalvergroting

Er zijn verschillende manieren waarop Scrum kan worden opgeschaald en veel andere Agile-methoden richten zich specifiek op het schalen van Agile.

Er zijn veel pogingen gedaan om met complexiteit en schaalvergroting om te gaan, waaronder:

- SAFe;
- LeSS;
- Nexus;
- Scrum@Scale.

We zullen SAFe en LeSS hier niet proberen te beschrijven, omdat deze methodes vrij ingewikkeld zijn en veel overhead vragen. Nexus en Scrum@Scale zijn relatief eenvoudig en lichtgewicht en ze gebruiken alle Scrum praktijken vrijwel ongewijzigd.

Wat zijn die andere methodes dan?

Het **Scaled Agile Framework® (SAFe®)** is een set van organisatie- en werkstroompatronen voor het implementeren van 'Agile' praktijken op bedrijfsschaal. Het framework is een kennisbasis (body of knowledge) die gestructureerde richtlijnen bevat voor rollen en verantwoordelijkheden, hoe het werk te plannen en te beheren en bepaalde waarden.

Large-Scale Scrum (LeSS) kan worden gezien als een schaalbaar framework voor productontwikkeling, maar LeSS kan ook gezien worden als een schaalbaar framework voor organisaties. Het probeert de complexiteit van organisatie weg te nemen door problemen in productontwikkeling op andere en meer enkelvoudige manieren op te lossen. Let wel, enkelvoudig betekent niet gemakkelijk, zeker niet op korte termijn. Ook al is LeSS eenvoudig, het adopteren is moeilijk.

Nexus is een framework voor het ontwikkelen en ondersteunen van geschaalde product- en softwareontwikkeling. Het gebruikt Scrum als bouwsteen.

Evenzo steunt **Scrum@Scale** op de fundamentele bouwstenen van Scrum, maar sommige praktijken die hier worden gebruikt, weerspiegelen een oudere denkwijze en meer afhankelijkheid van langetermijnplanning, waardoor Nexus een eenvoudiger keuze is voor opschalen.

Softwareontwikkeling is complex en de integratie van het resultaat in werkende software heeft vele artefacten en activiteiten die worden gecoördineerd om tot een ‘done’ te komen. Het werk moet worden georganiseerd, in een bepaalde volgorde uitgevoerd, de afhankelijkheden opgelost en de uitkomsten geordend.

Veel softwareontwikkelaars hebben het Scrum framework gebruikt om als team een increment van werkende software te ontwikkelen. Echter, als meer dan één Scrum-team werkt in dezelfde product backlog en dezelfde broncode voor een product, hoe kunnen ze met elkaar communiceren, wanneer zij werk doen dat elkaar zal beïnvloeden? Als ze in verschillende teams werken, hoe zullen ze dan hun werk integreren en het geïntegreerde increment testen? Deze uitdagingen doen zich voor wanneer twee teams werk combineren en het wordt aanzienlijk ingewikkelder met drie of meer teams.

Dit geldt ook voor veel andere soorten projecten waarbij Scrum wordt gebruikt, dus wat hier wordt besproken, geldt voor complexe en onderling afhankelijke projectomgevingen.

In dit boek zal Nexus worden gepresenteerd als de primaire manier van schalen, omdat het kan worden beschouwd als de minst gecompliceerde aanpak. Natuurlijk staat het iedereen vrij de aanpak te kiezen die voor de organisatie het beste werkt.

Nexus is een exoskelet[10] om het Scrum framework. Een Nexus combineert het werk van meerdere Scrum-teams in een geïntegreerd increment. Nexus is consistent met Scrum en de onderdelen zullen bekend zijn bij diegenen die aan Scrum projecten hebben gewerkt. Het verschil is dat er meer aandacht wordt besteed aan afhankelijkheden en samenwerking tussen Scrum-teams en dat er in elke sprint minstens één ‘done’ geïntegreerd increment wordt opgeleverd.

9.2 Agile Scrum's visie op schaalvergroting

Aangezien het een fundamentele regel is om niet meerdere backlogs voor hetzelfde product te hebben, willen meerdere teams die vanuit dezelfde backlog werken, effectief communiceren en coördineren.

Eén mogelijkheid is om na te denken over de inhoud van de backlog op een hoger niveau dan functionaliteit. Zoals eerder beschreven zijn product backlog items uitgewerkt als epics, thema's en functies. Het is mogelijk om het werk op basis van thema's aan verschillende teams toe te wijzen. Deze aanpak zal de kans op conflicten en afhankelijkheden tussen de teams verkleinen.

Hier moet voorzichtig mee omgegaan worden, want op deze manier wordt een thema dan beschreven als een frame voor productfunctionaliteit, wat helpt om de backlog te structureren en te prioriteren. Dit is niet hetzelfde als de Strategische Thema’s zoals beschreven in SAFe. Dat zijn van elkaar te onderscheiden bedrijfsdoelstellingen die een portfolio verbinden met de strategie van de onderneming.

[10] https://en.wikipedia.org/wiki/Exoskeleton

In Agile Scrum zijn methoden gedefinieerd die helpen bij het schalen van grote en complexe opleveringen en backlogs. Later in het boek zal Nexus meer gedetailleerd worden geïntroduceerd. Het komt enigszins overeen met de traditionele Agile Scrum kijk op schalen, maar er zijn enkele subtiele verschillen in de aanpak.

Nexus introduceert een aantal nieuwe termen en terminologie. Het consequent gebruiken van deze terminologie kan waardevol blijken en de kans op misverstanden verkleinen. Later gaan we meer in op schaalvergroting en Nexus.

10 Visueel management, de scrum- en Kanban borden

Kanban is van oorsprong een Lean-concept. Omdat Agile en dus Scrum, is gebouwd op Lean-principes, is het toepassen van Kanban als methode heel relevant in Scrum. Dit laat Scrumban, ontwikkeld door David J. Anderson, ook zien: een combinatie van Kanban- en Scrum-technieken.

Visueel management is een sleutelbegrip in alle Lean en Lean gerelateerde managementtechnieken. Middelen voor visueel management brengen boodschappen sneller en boeiender over dan geschreven informatie of verbale communicatie.

Een voorbeeld. Als we elke dag met de auto naar het werk gaan, kunnen we precies zeggen bij hoeveel stopborden of verkeerslichten we stoppen? Maar we stoppen wel – anders hadden we al veel ongelukken veroorzaakt. Dat is de kracht van visueel management en information radiators (information radiators)!

Wanneer informatie visueel wordt weergegeven, is het voor teams veel gemakkelijker om een probleem te op te merken en actie te ondernemen. Het gebruik van information radiators betekent ook dat problemen, defecten en problemen worden blootgelegd op het moment dat ze zich voordoen, zodat het team snel en doeltreffend actie kan ondernemen.

Een veelgebruikte methode om de voortgang van een team naar herkenbare incrementen te visualiseren en te volgen en om problemen en defecten snel bloot te leggen, is het gebruik van een scrumbord.

Teams verwijzen vaak naar het scrumbord als een Kanban bord, wat strikt genomen juist is, omdat een scrumbord de eenvoudigste vorm is van een Kanban bord. Kanban als methode echter, houdt meer in dan het bijhouden van voortgang. Daarom wordt er onderscheid gemaakt tussen Scrum – en Kanban borden. Een Kanban bord is in het algemeen wat complexer dan een scrumbord.

Hier volgt een kort overzicht van de verschillen. De keuze van de te gebruiken methode wordt door het team gemaakt.

10.1 Het scrumbord

In eenvoudigste vorm is een scrumbord een visuele voorstelling van story’s en gerelateerde taken die aanwezig zijn in de sprint backlog. Dit is een effectieve manier voor het Scrum-team om een gevoel van isolatie of disconnectie met de sprint of het Nexus plan of de product goal weg te nemen.

Het bord heeft gewoonlijk vier (of meer) kolommen, gemarkeerd:

1) User story (of sprint backlog Item);
2) Doen (To do);
3) In uitvoering (In progress);
4) Gedaan (Done).

Een softwareontwikkelingsbord kan er ongeveer zo uitzien:

1) User story;
2) Doen (To do);
3) In uitvoering (In progress);
4) Test;
5) CD/CI (DevOps) pijplijn;
4) Gedaan (Done).

Dit is een voorbeeld van een softwareontwikkelingsbord en als er met behulp van Scrum waarde geleverd wordt binnen een ander gebied dan softwareontwikkeling, zal het bord er zeker anders uitzien! De Developers in de sprint kiezen de stappen die nodig zijn om de voortgang te volgen van 'To do' tot uiteindelijk 'Done'.

Zowel de story als de samenstellende taken worden gewoonlijk geschreven op wachtrij-kaartjes of post-its van verschillende groottes.

Taken worden gewoonlijk aan iemand toegewezen en kunnen, afhankelijk van de benodigde capaciteiten, opnieuw worden toegewezen naarmate ze van de linkerkant van het bord naar de rechterkant van het bord bewegen.

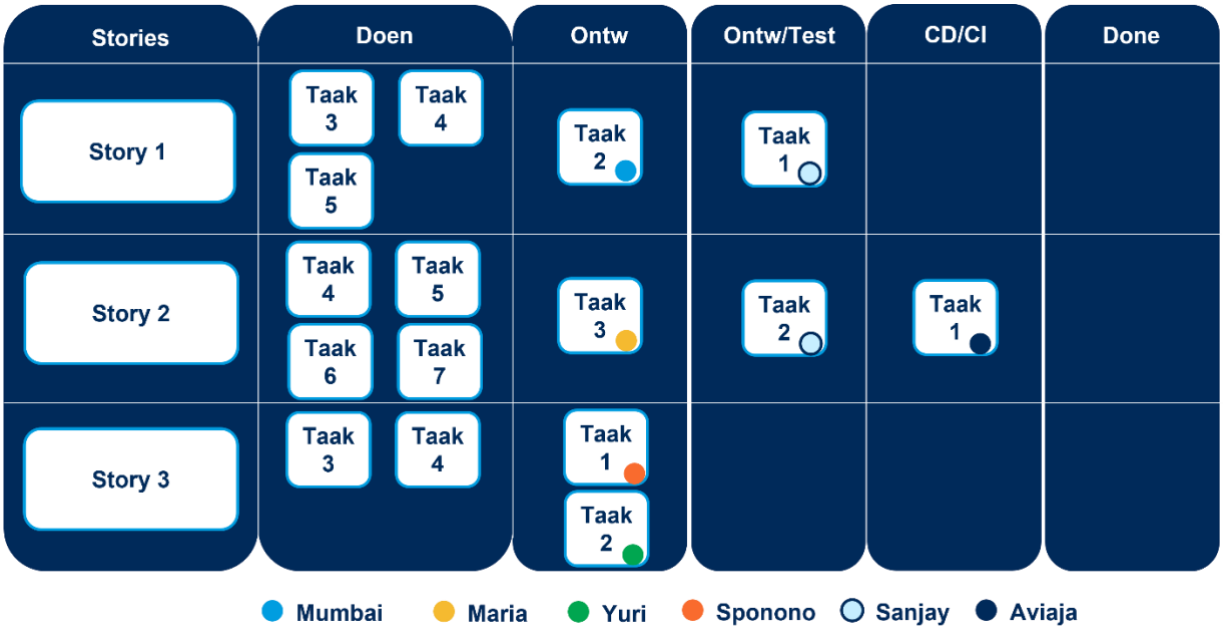
Let wel dat niet alle story’s vereisten van de klant zijn, sommige zijn afhankelijkheden en niet-functionele vereisten, of kunnen zelfs story’s van verbetering en de bijbehorende taken weergeven.

Het voorbeeld hieronder toont een zeer eenvoudig scrumbord met swimlanes voor verschillende story’s die deel uitmaken van de sprint. Op die manier is het gemakkelijk om te zien of een taak bij een story hoort en ervoor te zorgen dat het werk wordt voltooid en niet tussen teamleden heen en weer gaat. Dit voorbeeld is van het produceren van een stuurinrichting voor een auto. Swimlanes worden gebruikt om de taken bij de product backlog items (story’s) te houden.

Figuur 16 Een eenvoudig scrumbord met kolommen voor de soorten taken in de typische ontwikkelingscyclus in een specifieke sprint

Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us.

Figuur 17 Een typisch scrumbord voor een softwareontwikkelingsproject, gekleurde post-its laten zien aan wie de taak is toegewezen



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us.

Het is niet verplicht de stickers in het voorbeeld te gebruiken; vaak wordt alleen degene die de taak uitvoert op de kaart of *post-it* geschreven. Andere informatie die

op een kaart kan staan is de geschatte inspanning die nodig is om een story of een taak te voltooien, welke taken een hoge prioriteit hebben, of er afhankelijkheden zijn tussen *story's* en taken, of tussen taken en andere taken. Afhankelijkheden worden niet getoond in het plaatje, maar dit zal later specifiek worden behandeld bij het schalen met Nexus.

10.2 Waarom een scrumbord gebruiken?

Een scrumbord bevordert de zichtbaarheid en vereenvoudigt de communicatie. Een scrumbord stelt teamleden ook in staat om te zien waar zich problemen voordoen, zodat teamleden elkaar kunnen helpen (swarming of zwermen op een probleem) of plannen kunnen wijzigen als ze gebaseerd waren op de verkeerde veronderstellingen. Een scrumbord helpt dus bij inspectie, aanpassing en transparantie en bevordert daarmee teamwerk.

Een scrumbord helpt het team ook zich te committeren aan een sprint en het bijbehorende werk in de sprint, omdat iedereen alles kan zien wat gedaan moet worden.

Het bord wordt gemaakt als onderdeel van de sprint planning. In de sprint planning beslist het team welke items worden opgenomen in de sprint backlog. Vervolgens werken ze de story's uit in taken die nodig zijn om de story's op te kunnen leveren.

Het mooie van een scrumbord is dat er direct visueel gewerkt kan worden. Let wel: Scrum schrijft een scrumbord niet voor, maar het gebruik ervan is zeker in het voordeel van het team.

10.3 Een scrumbord gebruiken

Een scrumbord kan het beste gemaakt worden als de laatste activiteit van sprint planning. Hoewel vaak wordt gezegd dat de Scrum Master verantwoordelijk is voor het maken en onderhouden van het scrumbord, is dit NIET WAAR.

Het scrumbord behoort toe aan de Developers en hoewel de Scrum Master belast kan zijn met het onderhoud ervan, is dit noch een vereiste, noch een gegeven.

Het scrumbord behoort toe aan de Developers en helpt hen hun werk te optimaliseren.

Dus hoe wort een scrumbord gemaakt?

Developers maken een bord dat het werk dat ze gaan doen het beste laat zien. Er zijn twee voorbeelden gegeven van hoe een bord eruit kan zien, maar het bevat in ieder geval swimlanes voor Doen (To-Do), In uitvoering (In Progress, iets) en Gedaan (Done).

Als product backlog items (PBI's), meestal in de vorm van story's, worden toegevoegd aan de sprint backlog, worden ze ook toegevoegd aan het sprintbord. PBI's worden tijdens de sprint planning uitgewerkt in taken, meestal met een aantal attributen zoals duur, afhankelijkheden, toegewezen persoon, deadline, enz. Plaats alle gerelateerde taken in een swimlane naast de PBI waar ze bij horen, of door

gebruik te maken van dezelfde kleur post-its of kaartjes. In de voorbeelden hierboven zijn beide gedaan.

Door dit te doen is het initiële scrumbord gebouwd en klaar voor gebruik als de sprint planning voltooid is.

Tijdens de daily scrum, draait het gesprek om het werk dat op het scrumbord staat en hoe het verandert van dag tot dag. Als er gekozen wordt om nog steeds de drie vragen te gebruiken (dit is niet langer vastgelegd in de Scrum richtlijnen, maar ze kunnen nuttig zijn), zullen ze helpen bepalen welke kaart/post-it in welke kolom is en wie is toegewezen aan de taak. Het bord zal ook problemen of belemmeringen weergeven, meestal blokkers genoemd, die voltooiing van de taak verhinderen. Vergeet niet dat Developers werken aan de taak die zij kiezen, taken worden niet toegewezen tijdens de sprint planning.

In softwareontwikkelingsprojecten kan het scrumbord ook gebruikt worden om bugs (fouten) op te sporen en te rapporteren. Wanneer een bug wordt gevonden, kan deze worden toegevoegd aan het scrumbord om te worden opgelost in een tijdsbestek dat wordt overwogen door de Product Owner, op basis van prioriteiten.

Het verplaatsen van kaarten en het toewijzen van taken is een teamverantwoordelijkheid en NIET die van de Scrum Master. Teams vragen de Scrum Master echter vaak om de beheerder van het scrumbord te zijn en ervoor te zorgen dat het bord altijd de huidige status weergeeft van een sprint of gerelateerde iteraties in de sprint. Het komt dus vaak voor dat de Scrum Master scrumbordactiviteiten uitvoert.

10.4 Hoe verschilt Kanban van het gebruik van een scrumbord?

Het woord Kanban betekent letterlijk ‘kaart die je kunt zien’ en begon als een planningssysteem van voorraad bij Toyota (Lean). Later werd de techniek ook gebruikt als werkplanningsysteem en zo evolueerde Kanban tot de huidige vorm zoals die in Scrum wordt gebruikt.

De oorspronkelijke Kanban was een kaart die aangaf wanneer een item op een werkplek bijna op was, zodat het ‘just-in-time’ kon worden aangeleverd voor de volgende werkeenheid die zou worden gedaan.

Op dezelfde manier geeft een Kanban-taakplanningsbord een idee wanneer de volgende taak wordt uitgevoerd, zodat het werk 'just-in-time' kan worden gedaan.

En hier komt het belangrijkste verschil tussen een scrumbord en een Kanban bord.

Een Kanban bord houdt niet alleen de voortgang bij, maar ook de flow (doorstroming).

10.5 Waarom is flow belangrijk?

Het bereiken van de juiste flow (tempo) in het werk is net zo belangrijk als het werk gedaan krijgen. Er kan geen optimaal resultaat bereikt worden als er geen flow is in

een proces. Stel je voor dat alle verkeerslichten in de stad van de ene dag op de andere werden verwijderd – wat zou er dan met de verkeersstroom gebeuren?

De hoeveelheid verkeer verandert niet, maar de impact op de flow zou aanzienlijk zijn. Uiteindelijk zou alles tot stilstand komen.

Kanban als methode is gericht op optimalisatie van de werkstroom in een proces en verbetert daardoor de algehele voorspelbaarheid, efficiëntie en doeltreffendheid van het proces.

Let wel: Scrum is gebaseerd op empirische procesbeheersingstheorie. Centraal in empirische procesbeheersing staat een frequente cyclus van inspectie en aanpassing in een proces dat transparant is. Kanban praktijken richten zich specifiek op de verbeteringscyclus door regelmatige inspectie, niet alleen van de kwaliteit van het verrichte werk, maar ook van hoe het werk door het systeem of proces stroomt.

Het Kanban bord biedt de nodige transparantie om het werk en de werkstroom te inspecteren. Het team kan het werk of de werkmethoden regelmatig aanpassen om kwaliteit te bereiken en kan dat effectief en efficiënt doen.

Nu terug naar het scrumbord. In figuur 17 zien we dat Sanjay meerdere taken heeft. Als Sanjay de ene taak in de ochtend en de andere in de middag zou kunnen afronden, zou dat geen probleem zijn, maar wat als beide taken langer dan een dag duren?

Het betekent dat het testwerk voor Sanjay zich zou opstapelen en de volgende taak niet kan worden gedaan. Aviaja moet wachten tot Sanjay klaar is met testen voordat ze de software kan implementeren.

In dit scenario wordt Sanjay een bottleneck en maakt het niet uit hoeveel werk anderen voor of na Sanjay doen. De deliverables van het team worden bepaald door de blokker in het systeem, wat in dit geval Sanjay is. Zijn vermogen om synchroon met het werktempo van de rest van het team te testen, is essentieel om de werkstroom door de swimlanes op het scrumbord te maximaliseren.

Wat normaalgesproken in dit soort situaties gebeurt, is dat Sanjay probeert te switchen tussen beide taken, omdat hij geen knelpunt wil creëren. Van taak wisselen betekent dat hij zich niet op één ding tegelijk kan concentreren en dat goed doen. In plaats van twee dingen gedaan te krijgen, krijgt hij ofwel twee dingen slecht gedaan of helemaal niets.

Let wel: In zowel Lean als Agile wordt het wisselen van taken koste wat kost voorkomen!

Als Sanjay de enige tester in dit team is, moet het werk zo gepland worden dat er niet meer dan één taak in de kolom Dev/Test staat.

Zo kan ervoor gezorgd worden dat Sanjay zich op één taak kan concentreren en die goed kan afronden. Hij krijgt pas een andere taak als de test waar hij mee bezig is, is afgerond en het product aan Aviaja is overhandigd om uit te rollen en beschikbaar te maken voor gebruikers.

Dit concept wordt het beperken van werk in uitvoering genoemd, door voor work-in-progress (WIP) limieten vast te stellen. Door de hoeveelheid testen die op een bepaald moment worden gedaan te beperken, wordt het hele systeem van oplevering sneller.

Als Sanjay de enige Tester is, heeft de DevTest kolom een work-in-progress (WIP) limiet van één (1), om er zeker van te zijn dat het hele systeem goed werkt. Of er wordt een Tester aan het team toegevoegd, dan kan de WIP-limiet worden verhoogd naar twee (2).

Dit kan betekenen dat sommige Developers zullen stoppen met ontwikkelen als de WIP-limieten zijn bereikt.

Het klinkt misschien contra-intuïtief, dat het een goede zaak is om iemand te laten stoppen met werk. De realiteit is dat ze kunnen afmaken waar ze mee bezig zijn, maar ze zullen niet met nieuw werk beginnen want dat zal het knelpunt groter maken! Ja, het kan betekenen dat een teamlid ergens niet aan zal werken, maar hij of zij kan hulp aan anderen aanbieden om werk gedaan te krijgen, of, nog beter, worden ingezet om het werk weer te laten stromen. In dit geval kan Maria aanbieden om Sanjay te helpen met testen, op voorwaarde dat ze niet haar eigen werk test. Als een teamlid niet over de vaardigheden beschikt om knelpunten mee op te lossen, kan zij of hij gaan leren dat te doen en zo een nieuwe kernvaardigheid ontwikkelen.

Kortom: knelpunten worden vermeden in de planning, zodat het werk kan doorstromen. Eén van de belangrijkste doelstellingen van Lean, waar Scrum op is gebaseerd, is het elimineren van verspilling. Laten we het voorbeeld in het verkeer nog eens gebruiken. Verkeerslichten stoppen en starten het verkeer. In essentie maakt het verkeerslicht het verkeer trager; maar daardoor stroomt het beter door. Dus is het resultaat: sneller verkeer.

Hoe werkt dit?

10.6 Inzicht in de theorie van flow

Er zijn vijf Kanban (Lean) meetwaarden voor Scrum-teams om ervoor te zorgen dat werk beter en sneller gedaan wordt; het zijn[11]:

1. **Werk in uitvoering (Work in Progress, WIP):** WIP is al het werk dat begonnen is maar niet voltooid. De reden is niet belangrijk. Wel is van belang dat andere taken niet kunnen worden afgemaakt of gestart voordat WIP is gedaan. Andere taken zijn afhankelijk van de Developers die bezet zijn, omdat ze eraan werken.
2. **Cyclustijd (Cycle time)**: De hoeveelheid tijd, verstreken tussen het moment waarop een werkitem begint en het moment waarop een werkitem klaar is.
3. **Werkitem leeftijd (Work item age)**: De hoeveelheid tijd tussen het moment waarop een werkitem begon en de huidige tijd. Dit geldt alleen voor items die in uitvoering zijn.

[11] https://nkdagility.com/the-kanban-guide-for-scrum-teams/

4. **Doorvoer (Throughput)**: Het aantal afgewerkte werkitems per tijdseenheid.
5. **Service Level Verwachting (service level expectation , SLE):** Voorspelt hoelang het zal duren dat een bepaald item van begin tot eind door de werkstroom van het Scrum-team gaat.

Voor degenen die ervaring hebben met Value Stream Mapping (waardestroomdiagram, VSM), zullen deze meetwaarden bekend zijn. Het is zeer raadzaam om, als Scrum Practitioner, Lean te bestuderen. Dit maakt ons beter in Scrum en helpt om ons begrip te vergroten voor de specifieke werkwijzen in Scrum.

Waarom zijn deze meetwaarden belangrijk?

10.6.1 De wet van Little

Om flow te begrijpen, moet de Wet van Little (Little's Law) bekend zijn, die stelt:

$$average\ cycle\ time = \frac{average\ work\ in\ progress}{average\ throughput}$$

Gemiddelde cyclustijd is: gemiddelde WIP, gedeeld door gemiddelde doorvoer.

Wat betekent dit?

Eenvoudig gezegd: aan hoe meer zaken er op een bepaald moment gewerkt wordt, hoe langer het zal duren om de hele klus te klaren.

Dit is logisch. Wanneer er voortdurend tussen taken geschakeld wordt, dus naarmate er meer werkitems in uitvoering zijn, duurt het langer om een taak af te ronden (werkitem leeftijd). Bovendien gaat er focus verloren, met als gevolg dat er niet alleen minder gedaan gekregen wordt (doorvoer), maar ook dat het werk minder goed gedaan wordt.

Kanban helpt Scrum-teams om de cyclustijden te verkorten door WIP beter te beheren.

Als we van het voorbeeld-scrumbord afleiden dat alleen Sanjay kan testen en alleen Aviaja de implementaties kan doen, dan hebben beide kolommen een WIP-limiet van één (1).

De kolom 'Ontwikkeling' heeft een WIP-limiet van vier (4) Mubai, Sponono, Yuri en Maria. Maar zelfs als vier mensen code kunnen ontwikkelen, mag de hoeveelheid ontwikkelde code nooit groter zijn dan wat getest en uitgerold kan worden. Dat is wat de knelpunten aankunnen en door dit te doen wordt het hele systeem sneller.

We kunnen aanvoeren dat het ontwikkelingswerk normaal veel langer duurt dan het testen en samen met historische gegevens, kan dit worden gebruikt om de theoretische WIP-limiet die aan de kolom Ontwikkeling is opgelegd, aan te passen. In werkelijkheid kan het betekenen dat de WIP-limiet voor ontwikkeling drie (3) bedraagt (één minder dan het aantal mensen dat op enig moment ontwikkeling kan doen), maar het zal nooit meer zijn dan het aantal mensen dat beschikbaar is om het werk te doen.

Als Sanjay en Aviaja het tempo bepalen van het werk dat kan worden gedaan, wordt dit een pull-systeem genoemd. In pull-systemen wordt zo gewerkt dat het volgende werkstation in de lijn werk kan aannemen als dit station klaar is.

Pull-systemen voorkomen de opeenhoping van werk vóór een bepaalde stap in de werkstroom.

Wat het team idealiter wil, is wat in Lean single-piece-flow (één tegelijk) wordt genoemd. Het team wil ervoor zorgen dat het werk doorstroomt zonder oponthoud of wachttijd voor een bepaalde stap. Dit impliceert dan ook dat het plannen van de werkstroom in een sprint onderdeel is van de sprint planning.

Door WIP-limieten toe te voegen en de flow te plannen wordt het scrumbord in een Kanban veranderd.

Kanban geeft het Scrum-team ook duidelijk zicht op vaardigheden en afhankelijkheden en hoe de nodige vaardigheden het best in evenwicht kunnen worden gebracht en het werk kan worden geoptimaliseerd. Het gebruik van limieten voor werk in uitvoering op het scrum- of Kanban bord zal snel potentiële knelpunten in het systeem aan het licht brengen en het team in staat stellen het werk zodanig aan te passen dat de flow wordt geoptimaliseerd.[12]

Wat normaal gebeurt in Scrum-teams en wat deel uitmaakt van de nodige aanpassing, is dat teamleden crossfunctionele vaardigheden ontwikkelen naarmate ze van elkaar leren. Developers worden T-shaped professionals door in crossfunctionele teams te werken.[13] Dit betekent dat in de toekomst een Developer een tester zou kunnen worden en knelpunten in het systeem kan wegnemen door van rol te veranderen. Maar vergeet niet dat een Developer nooit de eigen code of het eigen werk kan testen.

Dit is echter een oplossing voor lange termijn. Een meer directe oplossing kan zijn om het aantal testers uit te breiden, of om te kijken of meer van het testen kan worden geautomatiseerd en onderdeel worden van de CI/CD-pijplijn. Zo kunnen Sanjay en Aviaja meer doen op een dag, zonder van taak te wisselen.

Een andere oplossing voor de korte termijn is wat zwermen (swarming) wordt genoemd, wanneer één of meer teamleden stoppen met hun taak om een belemmering in de flow weg te nemen.

---

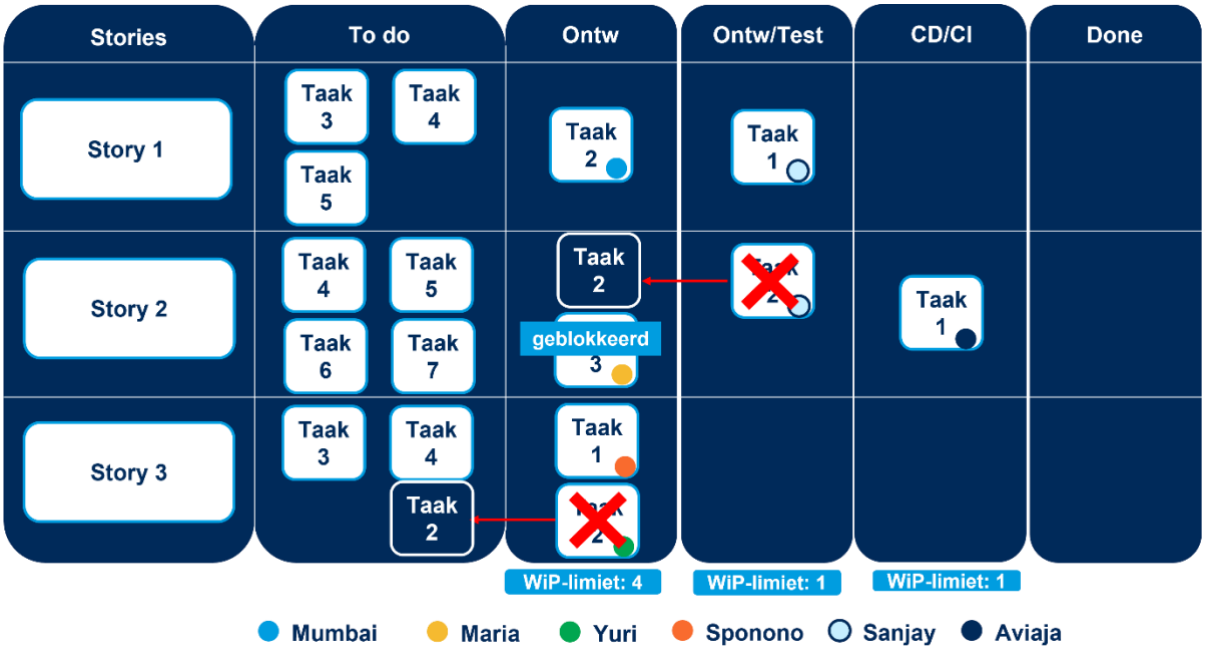
[12] Meer over de theorie van beperkingen is te lezen in Dr. Eliyahu Goldratt's baanbrekende boek *The Goal*.
[13] Meer information over T-shaped professionals is te vinden op https://www.exin.com/about-career-path-certifications

10.7 Hoe wijzigt het scrumbord als er Kanbantechnieken gebruikt worden?

Het belangrijkste verschil tussen een scrumbord en een Kanban bord is de focus. Een scrumbord is een visuele voorstelling van het werk dat gedaan wordt en een Kanban bord is een hulpmiddel om het werk zo te beheren dat de flow maximaal is.

In meest basale vorm, door alleen WIP-limieten op het scrumbord in te voeren, worden de kernwaarde van Kanban al geïmplementeerd.

Figuur 18 Hoe work in progress (WIP) limieten helpen blokkades te vermijden en een pull systeem te creëren



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us.

Laten we aannemen dat de eerder besproken WIP-limieten de juiste maatregel zijn, dan zal het bord er nu ongeveer zo uitzien (zie Figuur 18).

Het is te zien dat Taak 2 van Story Twee niet wordt overgedragen aan testen (Sanjay) en als gevolg daarvan terug zal verhuizen naar de kolom Ontwikkeling. Aangezien het nu een plaats inneemt in Ontwikkeling, kan één van de andere Ontwikkeling taken niet starten aangezien de WIP-limiet voor Ontwikkeling vier (4) is.

Yuri kan niet beginnen met het ontwikkelen van de code die hem is toegewezen. Dus wat kan hij doen, niets?

Het juiste antwoord is dat Yuri kijkt hoe hij Sanjay kan helpen zijn taken sneller af te krijgen, met behoud van kwaliteit. Ze gaan samenwerken en richten zich op de toezeggingen die het team heeft gedaan. Als hij niet over de vaardigheden beschikt om dat te doen, werkt het systeem beter door niets te doen.

In dit voorbeeld is er niets anders dat Yuri kan doen; maar in de meeste sprints is het mogelijk voor Yuri om een taak te vinden die niet afhankelijk is van het knelpunt.

Scrum beveelt aan dat Developers ook werken aan het verbeteren van het systeem dat ze gebruiken om incrementen op te leveren; in essentie maakt continue verbetering deel uit van het werk dat gepland is voor elke sprint. Dus in dit voorbeeld, als Yuri niet kan helpen met testen, kan hij beginnen aan een activiteit die hoort bij een story van verbetering, die niet afhankelijk is van de testvaardigheden van Sanjay.

Door dit te doen, zorgt het Scrum-team ervoor dat werkitems in de werkstroom worden getrokken met ongeveer dezelfde snelheid als waarmee ze de werkstroom verlaten, met als gevolg dat werkitems niet onnodig oud worden.

Wanneer zich een knelpunt voordoet, kunnen teamleden snel reageren op geblokkeerde werkitems of werkitems in de wachtrij, alsook op werkitems die de verwachte cyclustijd van het team overschrijden. Dit wordt de ***service level expectation***, of SLE, genoemd.

Het Scrum-team gebruikt SLE om problemen met de flow te vinden, ze te inspecteren en aan te passen als ze onder het verwachte niveau raken.

De SLE bestaat uit twee delen: de verstreken tijd, meestal dagen; en de waarschijnlijkheid van de voorspelde periode (bijvoorbeeld, 70% van de taken is klaar op sprint dag zeven).

De SLE is gebaseerd op de historische cyclustijd van het Scrum-team. Is deze berekend, dan maakt het Scrum-team deze zichtbaar voor iedereen. Het is aan te raden om een plek op het scrumbord vrij te laten voor de SLE en voor de product goal, sprint goal en DoD.

Is het team nieuw en zijn er geen gegevens uit het verleden om dit op te baseren, dan doet het Scrum-team een gok.

10.8 Wat zijn geblokkeerde items?

We hebben gezien dat taak 3 van story 2 in bovenstaand overzicht **geblokkeerd** is.

Hoewel het een goed idee is om op een scrumbord problemen aan te geven die verhinderen dat werkitems doorgaan, is het vooral van belang dit te gebruiken wanneer Scrum en Kanban worden gecombineerd.

In het voorbeeld hierboven, hindert een afhankelijkheid of probleem Maria in het voltooien van haar taak. Aangezien de taak begonnen is, kan deze niet terug naar de Te Doen kolom. Hier moet het team bekijken hoe ze Maria kunnen helpen met de afhankelijkheid om te gaan. Als de afhankelijkheid intern is, kan het team besluiten de werkvolgorde aan te passen, zodat Maria's taak wordt voltooid, of teamleden gaan zwermen (swarming) op het probleem om Maria te helpen de oorzaak te begrijpen en te verhelpen.

Als de afhankelijkheid extern is, zal de Scrum Master de kwestie met de externe partij opnemen om het probleem op te lossen.

Hoewel eerder is gesteld dat werkitems met een geblokkeerde status (een blokker ticket) niet terug kunnen naar de vorige kolom, is de realiteit dat als er niets aan te doen is, het team in overleg met de Product Owner kan besluiten de taak van de sprint backlog terug naar de product backlog te plaatsen. Dit mag als de sprint goal nog kan worden bereikt. Of, als laatste redmiddel, kan de taak worden teruggezet in de te doen kolom (niet in één van de andere) in de hoop dat deze later in de sprint kan worden opgepakt.

Als het team niets kan doen aan een afhankelijkheid of de oorzaak van een probleem en dit betekent dat de sprint goal en de definition of done (DoD) niet kan worden bereikt, mislukt de sprint en worden work items teruggebracht naar de product backlog.

10.9 Scrum en Kanban vergelijken

De onderstaande tabel is een voorbeeld van hoe een planning met Scrum van Kanban kan verschillen.

Tabel 2 – Hoe het gebruik van een traditioneel scrumbord kan verschillen van Kanban

Gebruik van een scrumbord	Gebruik van een Kanban bord
Het werk wordt uitgevoerd op basis van prioriteit	Het werk wordt uitgevoerd op basis van optimalisatie van de flow
Taken kunnen niet midden in een sprint worden toegevoegd, maar veel taken kunnen wel in uitvoering zijn.	Het vermogen om werk gedaan te krijgen, bepaalt of er aan een taak wordt gewerkt.
PUSH systeem is mogelijk	Alleen PULL systeem

10.10 Gebruik van de Kanban-methode

De Kanban-methode gebruiken en niet alleen het bord, om waarde te leveren in de vorm van producten of diensten is niet hetzelfde als Scrum. Er zijn enkele fundamentele verschillen tussen de twee methoden.

Naast de bovengenoemde verschillen tussen de twee verschilt de Kanban-methode ook in:

- aanpak;
- rollen;
- rapportage;
- manier van werken.

In de volgende tabel worden enkele van de bijkomende verschillen tussen de methoden weergegeven.

Tabel 3 – Verschillen tussen Scrum en Kanban in meer detail

Scrum gebruiken	Kanban gebruiken (methode)
Time-boxed gebeurtenissen (sprint, daily scrum)	Geen time-boxes, het team werkt aan taken zoals ze komen
Zeer specifieke verantwoordelijkheden (Product Owner, Scrum Master, Developers)	Geen vaste rollen of verantwoordelijkheden, deze worden naar behoefte toegewezen.
Het werk in een sprint is beperkt tot de sprint backlog	Er is geen aparte iteratie-backlog
Tijdens een sprint worden geen taken toegevoegd	Nieuwe taken kunnen voortdurend worden gegenereerd uit nieuwe vereisten van de klant, dus taken kunnen elk moment veranderen
Het team meet prestaties aan de hand van hoeveel er gedaan is en hoeveel er nog gedaan gaat worden (velocity en burn-down chart).	Omdat het werk niet is vastgelegd in een tijdsbestek, kan beter worden gemeten wat er is gedaan (cumulatieve flow).
Het scrum team houdt een retrospective aan het einde van de sprint om te bespreken wat goed ging, wat niet goed ging en hoe ze kunnen verbeteren.	Er is geen specifieke gebeurtenis na een iteratie om dit te bespreken. Maar, teams in een Lean-omgeving doen iets soortgelijks (Kaizen).

10.11 Hoe veranderen sprinttaken bij gebruik van Kanban?

10.11.1 De sprint

Wanneer Kanban met Scrum wordt gebruikt, heeft Scrum de overhand. Het betekent dat de sprintcadans dezelfde blijft. Wordt er met sprints van twee weken gewerkt, dan blijft dat het geval; wordt er met sprints van vier weken gewerkt, dan wordt die cadans aangehouden.

10.11.2 Sprint planning

Een flow-gebaseerde sprint planningsmeeting gebruikt flow meetwaarden als een hulpmiddel voor het ontwikkelen van de sprint backlog. Verwerkingscapaciteit van het Scrum-team in voorbije sprints wordt gebruikt om de capaciteit van een team te plannen voor werk dat in de volgende sprint wordt opgenomen.

Dit klinkt als een snelheidsmeting en dat is het ook, maar in plaats daarvan wordt de formule voor service level expectation (SLE) gebruikt.

10.11.3 Daily scrum

De daily scrum is de bijeenkomst van de Developers. Hoewel de Product Owner en Scrum Master meestal aanwezig zijn, hebben ze geen invloed op de discussie in de bijeenkomst, tenzij ze geraadpleegd worden en dit blijft zo wanneer Kanban wordt gebruikt.

In de daily scrum verschuift de focus: van dingen gedaan krijgen naar het laten stromen van werk door het systeem. Knelpunten zijn een belangrijke categorie van belemmeringen die worden aangepakt in de daily scrum.

Scrum-teams moeten niet alleen begrijpen waaraan gewerkt wordt en wat er gedurende de dag zal worden afgerond, maar praten ook over:

- **Zijn er WIP-limieten overschreden** en zo ja, hoe wordt dit gecorrigeerd? Meer specifiek, hoe kunnen geblokkeerde werkitems worden gedeblokkeerd?
- **Welk werk stroomt trager door dan verwacht?** Hier kijkt het team naar wat de werkitem leeftijd is van elk item in uitvoering, welke werkitems hun SLE hebben overschreden of op het punt staan te overschrijden en wat het Scrum-team kan doen om dat werk af te krijgen.
- **Zijn er factoren of werkitems die niet op het bord staan** en die impact hebben op het vermogen van het Scrum-team om het werk vandaag af te maken?
- Zal wat we vandaag hebben geleerd, gevolgen hebben voor waar morgen aan wordt gewerkt, mocht de volgorde of prioriteit van de geplande werkzaamheden worden gewijzigd?
- **Is er iets nieuws geleerd** dat invloed heeft op waar het Scrum-team van plan is aan te gaan werken?

10.11.4 Sprint review

Sprint reviews veranderen niet bij gebruik van Kanban methodes.

Hoewel een overzicht van Kanban flow meetwaarden, vooral doorvoer, gelegenheid kan geven voor nieuwe gesprekken, is het beter om details te bespreken tijdens de sprint retrospective.

10.11.5 Sprint retrospective

Het inspecteren van flow meetwaarden en de analyses daarvan, helpen om te bepalen welke verbeteringen het Scrum-team kan aanbrengen in hun processen.

Het Scrum-team inspecteert ook de definitie van ‘workflow’ en past deze aan om de flow in de volgende sprint te optimaliseren.

Het team zou een cumulatief stroomdiagram kunnen gebruiken om de WIP van het team, de gemiddelde cyclustijd en de gemiddelde doorvoer te visualiseren. Deze kan worden gebruikt om processen te verbeteren of bij het plannen van de hoeveelheid werk die in de volgende sprint door het team zal worden gedaan.

Let wel: de beschreven activiteiten maken deel uit van de inspectie in de loop van de sprint en niet alleen van de retrospective.

10.11.6 Increment

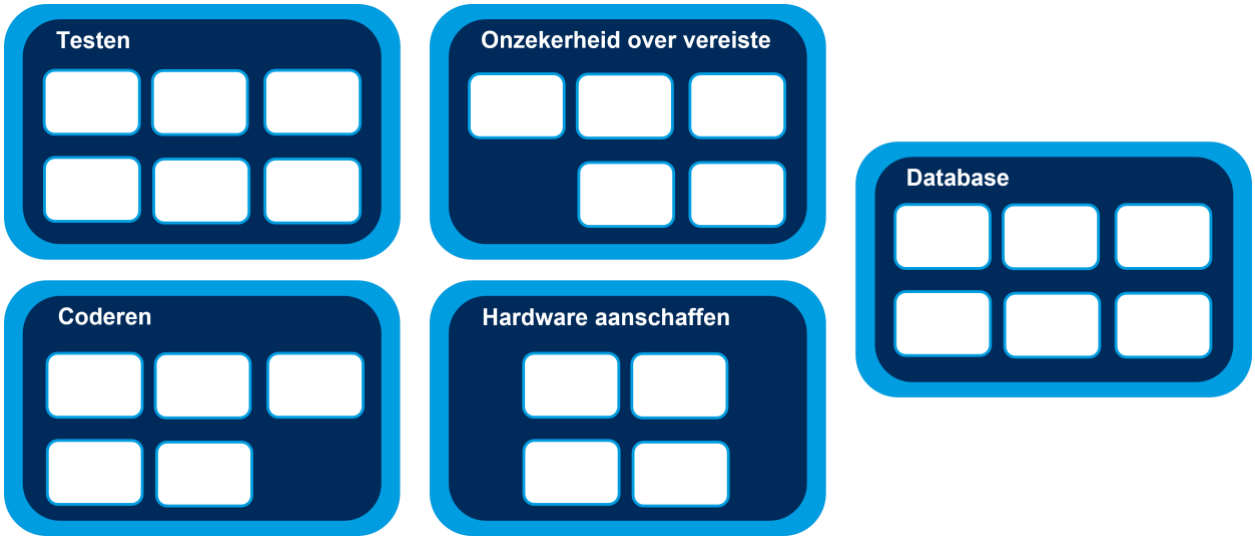
Scrum vereist dat het team elke sprint ten minste één potentieel implementeerbaar increment van ‘Done’ creëert. Scrum’s empirisme stimuleert om meerdere incrementen te creëren tijdens een sprint en om deze incrementen te implementeren zodra het werk door het team is voltooid en niet pas aan het einde van de sprint.

Het bereiken van een constante flow betekent dus dat klanten of gebruikers voortdurend profiteren van het werk van het team.

10.12 Wat staat er nog meer op een goed sprintbord?

Het sprint- of Kanban bord is slechts één van de information radiators (informatieverspreider) die te vinden is waar Scrum-teams elkaar ontmoeten. Naast het sprint- of Kanban bord kunnen andere tools worden gebruikt om de artefacten en het bijbehorend werk zichtbaar te maken, wat een goede communicatie, coördinatie en uitvoering van een sprint zal bevorderen. Sommige van deze artefacten zijn al besproken, zoals de sprint goal, de incrementen waaraan tijdens de sprint wordt gewerkt, het bijbehorende werk en records zoals definitions of done en blokker tickets. Maar dat is misschien niet de enige informatie die nuttig is voor het team om zichtbaar te maken.

Figuur 19 De KJ methode gebruiken om blokker tickets te groeperen



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us.

Ook als blokkades zijn verwijderd, is het nuttig voor teams om blokker tickets niet weg te gooien. Het bijhouden van opgeloste issues of hindernissen is heel nuttig bij de sprint retrospectives. Sommige Scrum-teams hebben een plek om opgeloste blokker tickets te 'parkeren' voor latere referentie (tijdens de retrospective) en sommige teams gebruiken zelfs affiniteitsdiagrammen om ze te organiseren in logisch gerelateerde thema's, als er in een sprint veel van zijn.

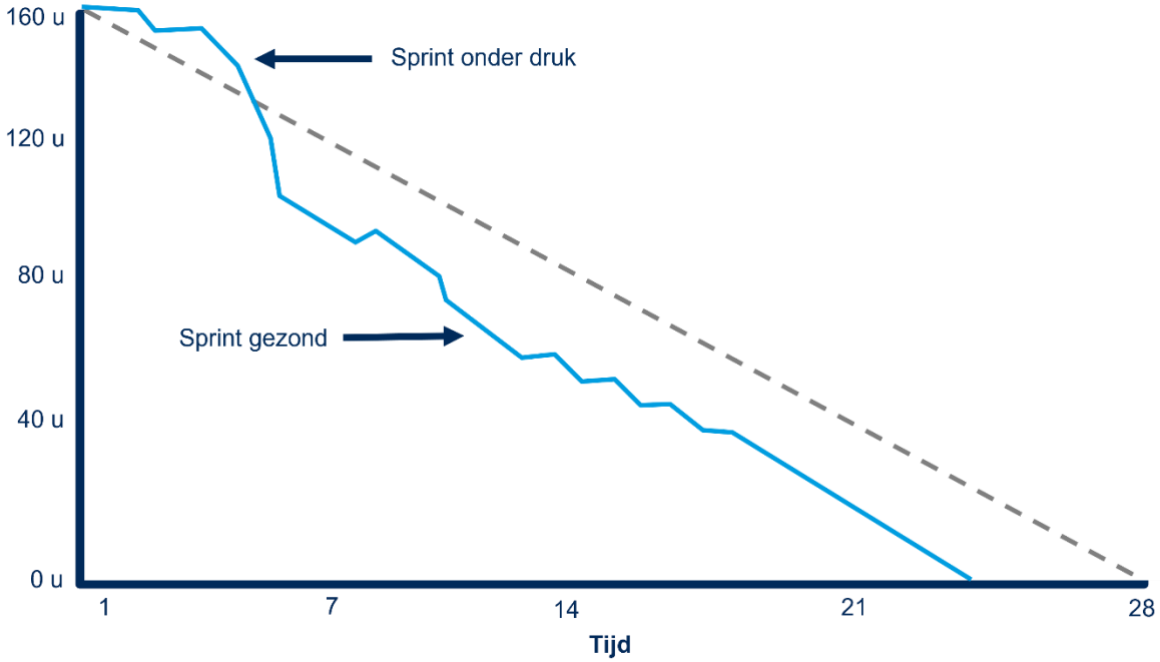
Een affiniteitsdiagram, ook bekend als de KJ Methode naar de uitvinder ervan, de Japanse antropoloog Jiro Kawakita, is een grafisch hulpmiddel dat is ontworpen om losse, ongestructureerde gegevens, in dit geval blokker tickets, te helpen organiseren. Blokker tickets worden gegroepeerd in betekenisvolle categorieën die affiniteitssets worden genoemd. Deze sets brengen verschillende tickets samen rond een onderliggend thema, verduidelijken problemen en bieden een structuur voor een systematische zoektocht naar één of meer oplossingen.

10.12.1 Burn-down/burn-up chart

Burn-down of burn-up charts zijn visuele managementtools, bedoeld om de voortgang bij te houden, maar ze kunnen ook snelheid meten, de snelheid waarmee dingen worden gedaan in een sprint.

Sprints worden zo gepland dat ze kunnen worden voltooid in een vaste time-box, meestal twee tot zes weken.[14] In Agile, wordt vastgehouden aan het budget en de toegewezen tijd (time-box) voor de sprint. De variabele is hoeveel waarde er wordt opgeleverd. Aangezien de definition of done dit in overweging neemt, moet een sprint minstens één definition of done opleveren.

Figuur 20 Een typische burn-down chart



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us.

Naarmate taken worden voltooid en gecontroleerd aan de hand van schattingen, wordt de resterende inspanning berekend en weergegeven in een grafiek. De kromming kan naar beneden zijn, naar nul resterende inspanning, wat een burn-down chart wordt genoemd; of de kromming kan naar boven zijn, naar verschillende taken die binnen een tijdsbestek van twee tot zes weken zijn voltooid, wat een burn-up chart wordt genoemd.

De meeste klanten geven de voorkeur aan burn-down charts, omdat zij meer geïnteresseerd zijn in hoeveel werk er nog moet worden gedaan volgens de toegewezen time box dan hoeveel werk er is voltooid (het Kanban bord laat dat in ieder geval zien).

Tijdens de planning wordt het grafieksjabloon gemaakt met een rechte lijn van dag 0 tot 28, gebruikmakend van het aantal taken. Het gegeven voorbeeld is een sprint

[14] In de praktijk zien we wel langere sprints. Maar, volgens de Scrum Guide is de maximale lengte van een sprint een maand.

van vier weken – met nul taken over. Deze lijn vertegenwoordigt de ideale snelheid waarmee taken worden voltooid (stippellijn = ideale snelheid).

Volg en breng de vooruitgang dagelijks in kaart. Als de lijn boven de stippellijn komt, betekent dit dat de sprint niet vordert zoals gepland en dat het team moet ingrijpen. Blijft de dagelijkse plot onder de stippellijn, dan is de sprint op tijd af en levert op wat gepland was.

10.12.2 Wat is snelheid?

Snelheid (velocity) is een maatstaf voor de hoeveelheid werk die een Scrum-team tijdens een enkele sprint kan afronden. Er is geen industriestandaard voor. Snelheid wordt berekend aan het einde van de sprint door het optellen van de punten voor alle volledig afgeronde user story's en deze vervolgens te vergelijken met eerdere registraties. Onafgewerkt werk kan niet worden meegeteld of in aanmerking worden genomen bij het meten van de snelheid.

De snelheid weten van een team, is belangrijke informatie voor de sprint planning, zodat teams zich niet vastleggen op werk dat ze niet in een sprint kunnen afmaken.

Let wel: de snelheid van een team in een bepaalde sprint hangt ook af van het werk dat ze doen en hoe goed het werk is afgestemd op de vaardigheden en capaciteiten van het team.

Het evalueren van trends in snelheid van teams kan waardevolle input zijn voor de sprint retrospective en kan gebruikt worden om verbeterpunten te identificeren.

In het algemeen zal de snelheid van een team met de tijd toenemen en een goede metriek om hier te gebruiken, is een verbetering van 10% in elke sprint. In onervaren Scrum-teams komt het vaak voor dat tijdens de sprint duidelijk wordt dat de snelheid lager is dan verwacht. In zulke gevallen is het goed om manieren te vinden om de snelheid te verhogen, of wanneer dit niet mogelijk is, de snelheid opnieuw te schatten.

OPMERKING: Product Owners kunnen besluiten na een sprint de volgorde van product backlog items te wijzigen, rekening houdend met veranderingen in snelheid, verhoogde voorspelbaarheid en flexibiliteit.

10.12.3 Wat is het verschil tussen snelheid (velocity) en SLE?

Het concept service level expectation (SLE) werd eerder geïntroduceerd als een voorspellende maatstaf, vergelijkbaar met de ideale snelheid die in kaart wordt gebracht op een burn-down of burn-up chart.

Wat SLE's anders maakt, is dat de voorspelling niet alleen gebaseerd is op hoeveel er gedaan moet worden in een hele sprint, maar ook rekening houdt met het complexiteitsniveau van wat er gedaan moet worden en hoelang het zal duren om taken gedaan te krijgen. Vergeet niet dat op de burn-down chart een activiteit als gedaan gemarkeerd kan worden als deze af is. Als een taak 5 dagen duurt en deze is voor 80% klaar, dan staat deze nog steeds niet op de burn-down chart en dat vertekent het beeld van de voortgang en de snelheid.

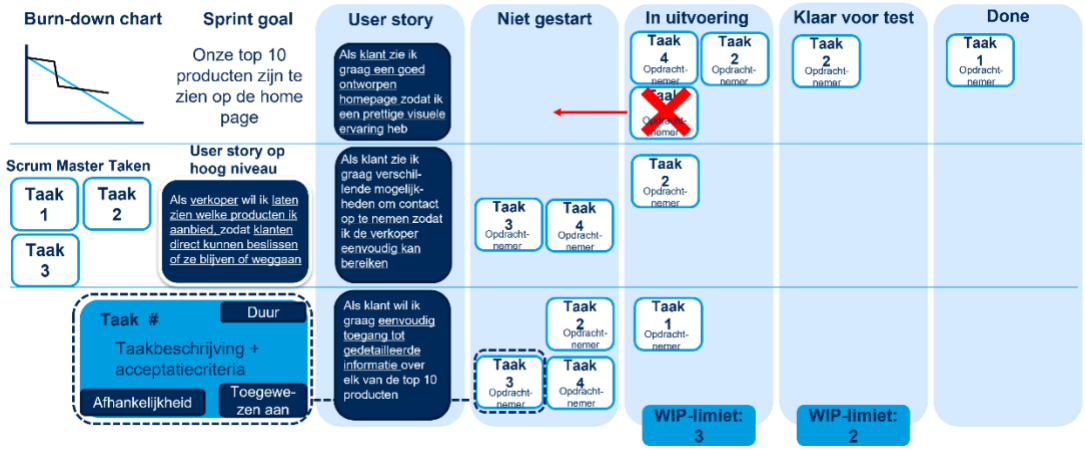
Sommige teams gebruiken SLE's om de ideale progressielijn uit te zetten op hun burn-down chart, omdat die de voortgang nauwkeuriger weergeeft.

10.13 Information radiators en ontmoetingsplaatsen

De information radiators hebben een vaste plaats en alle teamactiviteiten worden eromheen georganiseerd, zodat ze voor alle teamleden steeds goed zichtbaar zijn.

Een typische muur waar een Scrum-team samenkomt, kan er ongeveer zo uitzien:

Figuur 21 Een typische Scrum-teamruimte



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2020). *Scrum Lego-game workbook [Courseware]*. GetITright.

Een typische Scrum-team ruimte zal naast het scrumbord ook andere information radiators bevatten. Hier is vaak ook een burn-down chart te zien en het begin van een verzameling blokkers die tijdens de sprint retrospective geanalyseerd kunnen worden met behulp van de JK-methode.

11 Principes sturen patronen en anti-patronen

In de vorige versies van het boek was dit hoofdstuk gewijd aan traditionele benaderingen van schalen, die vaak werden gestimuleerd door andere Agile-technieken dan Scrum.

In het volgende hoofdstuk zullen we de Scrum-manier van schalen beschrijven.

Maar wat bedoelen organisaties als ze zeggen dat ze Agile of Scrum schalen?

Er zijn over het algemeen twee manieren waarop het woord schalen wordt gebruikt.

Soms wordt de term schalen gebruikt wanneer een organisatie grote en complexe projecten heeft en ze deze ook willen uitvoeren als grote en complexe projecten. De enige manier om dit te doen is door technieken te gebruiken die worden gebruikt in traditioneel projectmanagement. Maar het probleem van die aanpak is: we beginnen hiermee eigenlijk dingen te doen die niet agile zijn!

Op andere momenten betekent schalen dat het management van een organisatie wil dat de hele organisatie agile is. Dit is een goede ambitie, maar de aanpak die hiervoor wordt gebruikt, is het vervangen van de huidige processen door 'Agile-processen' en huidige structuren door 'Agile-structuren'. Dit wordt aangedreven door een groot verandermanagement-initiatief om de organisatie agile te laten worden. Deze aanpak zal nooit werken omdat we niet agile kunnen doen; we moeten agile zijn. Deze aanpak versterkt niet-agile gedrag en laat mensen in de organisatie zien dat ze hun gedachten en gedrag niet hoeven te veranderen; het is immers gewoon een nieuwe manier waarop we oude dingen zullen doen!

Beide bovenstaande voorbeelden zijn anti-patronen. Het schalen van Scrum en daarbij de twaalf Agile principes negeren of tegenwerken, vermindert de potentie om succesvol agile te zijn.

We hebben de verwijzing naar dit specifieke anti-patroon van schalen in deze versie van het boek verwijderd, maar het is alsnog goed om een hoofdstuk te besteden aan veelvoorkomende anti-patronen en hoe men Agile-patronen kan volgen die de organisatie ten goede komen en de potentie van de organisatie om behendig te zijn, vergroten.

Er is veel geschreven over anti-patronen, maar pas sinds de publicatie van het boek 'Sooner Safer Happier' van Jon Smart is er een definitieve referentie. Dit hoofdstuk is gebaseerd op het boek van Smart en we raden aan om meer te lezen over dit belangrijke onderwerp dan alleen dit hoofdstuk!

11.1 Wat zijn anti-patronen?

Simpel gezegd: het zijn de dingen die organisaties doen - meestal omdat ze denken dat ze een probleem zullen oplossen - die de situatie eigenlijk erger maken of het

moeilijker maken om een Agile mindset ontwikkelen en bijbehorend gedrag te faciliteren.

Laat duidelijk zijn dat anti-patronen geen domme ideeën zijn van organisaties; ze zijn vaak gebaseerd op traditioneel denken en traditionele managementpraktijken en er is veel bewijs dat deze werken in een industriële context. Ze zijn problematisch omdat ze niet werken in een Agile-context en in veel gevallen de potentie van organisaties om agile te zijn, verminderen of wegnemen!

Van sommige dingen die we identificeren als anti-patronen zouden we kunnen zeggen dat ze binnen conventionele wijsheid als een slim idee worden gezien, maar helaas is in een Agile-context bewezen dat ze heel slechte ideeën zijn!

Wat het extra moeilijk maakt om van anti-patronen af te komen, is dat ze vaak jaren en jaren van leren en conditionering tegenspreken.

Als we echt willen dat Agile in de organisatie werkt, moeten we moedig en meedogenloos zijn tegenover anti-patronen.

11.2 De 'goede ideeën' die dat... niet zijn!

Dit hoofdstuk is geen definitieve leidraad voor anti-patronen (en patronen - de goede dingen om te doen). Het is een introductie van het concept.

We zullen zes veelvoorkomende voorbeelden bespreken, maar er zijn er veel meer.

Hoe zouden we weten of iets een anti-patroon is?

De gemakkelijkste manier om ze te herkennen is dat ze de twaalf Agile principes tegenspreken. In het boek van Smart worden extra principes gedefinieerd op basis van ervaringen van organisaties die hebben bewezen dat Agile goed voor hen werkt.

11.2.1 Een agile implementatie of transformatie uitvoeren

Ja, een agile implementatie uitvoeren betekent meestal het uitvoeren van een verandermanagementproject; de middelen die hiervoor worden gebruikt, zijn precies de middelen die we willen vermijden - dat is onzinnig!

In plaats daarvan is het belangrijk om te zorgen voor begrip en training, zoeken naar vrijwilligers om nieuwe manieren van werken uit te proberen, en hen te beschermen tegen het organisatorische immuunsysteem (oude regels), omdat veel van de nieuwe manieren van werken oude regels zullen overtreden. We zeggen niet dat er geen regels zijn; we hebben een paar regels nodig die zijn gebaseerd op de nieuwe manier van denken!

De kunst is om te focussen op de gewenste resultaten en daarbij nieuwe maatstaven (succescriteria) te definiëren op basis van wat we in de toekomst willen in plaats van het gebruiken van de huidige maatstaven.

Wanneer niet wordt aangepast hoe we succes meten (en, als gevolg daarvan, hoe mensen worden gemeten) falen we gegarandeerd.

Peter Drucker zei: "Wat gemeten wordt, wordt gedaan". Wanneer we nieuw gedrag verwachten, maar oud gedrag meten, is dus duidelijk wat de uitkomst is!

Als we willen dat het werkt, hebben we allereerst bereidwillige deelnemers nodig die het kunnen laten werken. Dit betekent dat we klein moeten beginnen.

11.2.2 Agile schalen

Welk van de twee voorbeelden van schalen uit de vorige paragraaf is een anti-patroon?

Het antwoord is beide.

Een agile transformatie bestaat grotendeels uit schalen, wat inhoudt dat de hele organisatie agile wordt zonder gedrag te veranderen.

We zullen nu behandelen waarom organisatiebrede projecten een anti-patroon zijn.

Grote, complexe projecten zijn vaak grootse plannen van mensen met een kristallen bol. Deze projecten kunnen werken in een ideale wereld, maar in de huidige, snel veranderende wereld verouderen zelfs grote ideeën van de ene op de andere dag. Een gedetailleerde langetermijnplanning is een *contradictio in terminis* en is één van de redenen waarom organisaties overstappen van Waterval- naar een Agile-benadering voor het leveren van waarde.

Mensen met een Agile mindset zullen herkennen dat het enige wat we kunnen doen is het geven van richting en het spreken over algemene resultaten die op lange termijn bereikt moeten worden (waarbij een jaar al een lange termijn is).

Het is het beste om werknemers te laten uitzoeken hoe ze die resultaten kunnen bereiken en ze daarbij stap voor stap te laten zetten. Deze aanpak heeft als bijkomend voordeel dat er vaak correcties kunnen worden aangebracht en er altijd kan worden geleverd in de context waarin de organisatie zich op dat moment bevindt. Agile zijn stelt men in staat om de toekomst te creëren terwijl het gebeurt en niet constant de werkelijkheid te moeten inhalen.

Elke manier van schalen waarbij deze als groot en complex project met vooraf bepaalde zekerheden wordt benaderd, zal methoden gebruiken die niet agile zijn.

Er kan worden gesteld dat dit soort projecten de realiteit van de huidige organisatie zijn en dat dit de enige manier is waarop we kunnen opereren in onze organisatie en industrie. Echter, verrassend genoeg is dit niet de manier waarop Google, Cisco en Microsoft werken. En niet enkel organisaties in de IT-sector werken zo, maar ook Barclays, Boeing, John Deere, Airbus, Amazon, de Amerikaanse luchtmacht, Panera Bakeries, Sony, Lego, KPMG enzovoort!

Een laatste opmerking: we zeggen niet dat Agile-methoden een wondermiddel zijn, omdat in sommige scenario's Watervalmethoden beter werken. We zeggen dat agile zijn, agile denken en zich agile gedragen een evolutionair proces is.

Dit leidt tot het volgende anti-patroon.

11.2.3 Veronderstellen dat één aanpak voor alle organisaties werkt

In snel veranderende omgevingen (zoals de huidige wereld) is het proberen om dé Agile Way te definiëren een grote fout. In Star Trek: Discovery zei kapitein Lorca: “Universele wetten zijn voor lakeien, context is voor koningen”. Wij zijn het daarmee eens.

Dit hangt ook samen met de wens van sommige leiders om organisaties van de ene dag op de andere te veranderen.

Wat is de beste actie in de context van onze organisatie? Dat hangt ervan af en wanneer we de organisatie opsplitsen, verschilt het zelfs op elk niveau. Daarom zijn zelfmanagement en echt eigenaarschap twee belangrijke Agile-principes!

Ja, we begrijpen de aantrekkingskracht van het opzetten van een systeem voor behendigheid en het bedrijfsbreed toepassen van dit systeem, maar dat is industrieel denken. Dit werkt niet, hoe hard we het ook proberen en hoe slim we ook zijn, omdat het niet past binnen Agile. Agile kan niet wordt opgelegd aan mensen in een organisatie, omdat iemand iets opleggen per definitie niet agile is!

Maar hoe binden we alles aan elkaar?

Smart stelt voor om VOICE te gebruiken:

- Focus op waarden (**v**alues) en principes.
- Doelgericht beheren en resultaten (**o**utcomes) meten.
- Dit betekent dat leiderschap gebaseerd is op **i**ntentie, een voorbeeld is en iedereen ruimte geeft om dingen op hun eigen manier te doen.
- Het **c**oachen en ondersteunen van initiatieven, experimenten en leren is de nieuwe norm voor het management.
- Dit leidt naar **e**xperimenteren. Experimenteren is de enige manier om te ontdekken of de nieuwe manier werkt! Dit betekent dat we anders gaan meten; als de metingen leiden tot straf bij mislukking, kan niet worden verwacht dat mensen experimenteren!

11.2.4 Commando en controle

De nieuwe rol van een manager in een agile organisatie is die van een stimulator en een coach. Doe-wat-ik-zeg is verleden tijd!

Het meeste werk is tegenwoordig kenniswerk; de mensen die aan ons rapporteren weten waarschijnlijk meer over het werk dan wij en hen vertellen wat ze moeten doen, is niet logisch. Mensen kunnen niet eindverantwoordelijk worden gehouden, wanneer ze worden verteld wat ze moeten doen.

Is het opgevallen dat de anti-patronen elkaar voeden? Hoe meer anti-patronen we behandelen, hoe waarschijnlijker het is dat we ook iets moeten doen aan alle voorgaande anti-patronen.

Psychologische veiligheid op de werkplek is van vitaal belang; mensen maken fouten en het proberen van nieuwe dingen zal waarschijnlijk leiden tot meer fouten. Dit impliceert dat het de taak van het management is om een omgeving te creëren

waar het veilig is om fouten te maken en waar mensen veilig kunnen leren en beloond worden voor dit leren. Ze moeten hierbij niet bestraft worden. Let op: ‘belonen voor falen’ is niet wat we zeggen.

Al het management is agile, wat dienend leiderschap inhoudt!

11.2.5 Ingewikkelde controles vanuit het gevoel geen controle te hebben

Een conventionele denkwijze van management zal waarschijnlijk betekenen dat een manager in een agile organisatie zich ongemakkelijker voelt naarmate de besluitvorming wordt overgedragen. De eerste reactie is dan om meer controles in te stellen om het gevoel van controle te behouden.

Dit is precies wat we niet moeten doen!

Het management moet kijken hoe het controleren kan worden vereenvoudigd en tegelijkertijd transparant kan worden gemaakt. Deze transparantie is de juiste reactie op de eerdergenoemde ongemakkelijkheid.

Zorg ervoor dat de controles passend zijn - de meeste controles die nuttig waren in een industriële omgeving zijn op zichzelf al anti-patronen!

Focus op flow, waarde en resultaten, en niet op (geïsoleerde) kwaliteit, procesnaleving en activiteiten.

Eigenlijk is de ultieme manier om te controleren of goed werk wordt geleverd, klanten horen zeggen dat we hen de beste waarde ooit bieden!

11.2.6 Focussen op de snelheid van levering

Voor veel organisaties geldt dat wanneer ze hun agile transitie beginnen, ze dit doen omdat het hen in staat zal stellen sneller te leveren.

Snelheid is echter de verkeerde maatstaf, omdat dit slecht gedrag bevordert; snelheid wordt het doel in plaats van het leveren van veel klantwaarde.

De laatste tijd wordt er meer gesproken over een increment dat klaar, maar niet helemaal klaar is. Dit betekent dat ofwel de kwaliteit lager is of technische schuld is opgebouwd om deadlines te kunnen halen.

Dit is zeer gevaarlijk!

Onthoud dat het concept agility is afgeleid van Lean en in Lean worden defecten niet doorgegeven in Lean! Doe het de eerste keer goed, zelfs als dat betekent dat we een deadline missen, want anders zullen we na verloop van tijd zoveel schuld hebben dat we gegarandeerd deadlines zullen missen en geen waarde zullen toevoegen voor klanten.

We stelden eerder dat het management zich moet richten op resultaten en dit is een perfect voorbeeld daarvan! Als een levering ondermaats is, wat is dan uiteindelijk het resultaat?

Zijn we overtuigd dat wat we leveren perfect is?

Wat doen we in een scenario waarin hetgeen wat we leveren niet perfect is?

We doen het juiste en maken het perfect en anders leveren, verkopen of implementeren we niet. Dat is precies wat een Andon doet in een Toyota-onderneming!

12 Nexus en schaalvergroting

12.1 Wat is een Nexus?

Nexus is een framework om geschaalde productontwikkeling op een duurzame manier uit te voeren en het gebruikt Scrum als basisbouwsteen.

Geen van de definities verandert en ook het gebruik van de Scrum verantwoordelijkheden, gebeurtenissen, artefacten of regels verandert niet – Nexus voegt eerder iets toe aan Scrum dan dat het Scrum verandert.

In die zin is Nexus een geschaald ontwikkelingsframework. Een Nexus-sprint is een ontwikkelingseenheid die bestaat uit vele iteraties die samen een ‘increment’ opleveren, die waarde levert aan klanten.

In een iteratie of sprint zijn alle inspanningen gericht op het resultaat van die sprint en dat wordt bepaald door de doelen en de definition of done (DoD) van de Nexus sprint.

In de praktijk is het zo dat, hoewel elke sprint een potentieel werkend product zou moeten opleveren, dit door de complexiteit van de meeste ontwikkelomgevingen vaak niet mogelijk is.

Het werk wordt in een bepaalde volgorde uitgevoerd en opgeleverd, waarbij rekening wordt gehouden met onderlinge afhankelijkheden en de vaardigheden in de verschillende teams, om te kunnen voldoen aan de werkelijke vraag van de klant. De DoD van een sprint is daarom vaak niet bruikbaar; het increment is niet potentieel werkend of zelfs begrijpelijk voor klanten in het licht van wat ze nodig hebben voor het faciliteren van een product of dienst.

Wat betekent dit in werkelijkheid?

Vaak leveren meerdere Scrum-teams waarde vanuit dezelfde product backlog. In softwareontwikkeling en in vele andere producten, betekent dit gebruik maken van dezelfde bouwstenen (codebase) en geconfronteerd worden met dezelfde afhankelijkheden die ontstaan tussen het werk van deze verschillende teams. De incrementen die door verschillende teams worden opgeleverd, worden vaak ook samen getest en uitgerold, willen klanten er waarde uit kunnen halen.

Wat het nog complexer maakt, is dat Scrum-teams zich steeds vaker niet eens in hetzelfde gebouw, dezelfde stad of hetzelfde land bevinden. Ongecoördineerd werken loopt uit op een ramp.

Teams moeten zich bewust zijn van de impact van hun werk op anderen, van hoe andere teams afhankelijk zijn van hun resultaten en de gevolgen van werk dat niet af is. Verder is een gemeenschappelijk begrip van de vereisten belangrijk, dat het gebruik en de toepassing van wat door elk team wordt ontwikkeld, door de anderen worden begrepen.

Hoe meer teams betrokken zijn, hoe complexer het wordt.

Vanwege deze afhankelijkheden is er dus, als Scrum moet opschalen, een methode nodig om het werk te coördineren, de eisen te begrijpen en te beheren,

Deze methode moet specifiek gericht zijn op:

- **Vereisten.** Ervoor zorgen dat een algemeen begrip ontstaat onder de leden van alle betrokken teams en hoe het werken aan de ene eis het werk aan de andere zal beïnvloeden.
- **Domeinkennis.** Gedaan of te doen werk wordt toegewezen aan teams op basis van zowel hun technische als business vaardigheden, hun kennis en capaciteiten.
- **Product-, test-, integratie- en kwaliteitsborgingsartefacten.** Zorgen voor een uniforme aanpak voor het testen van de samenstellende delen als één oplevering.

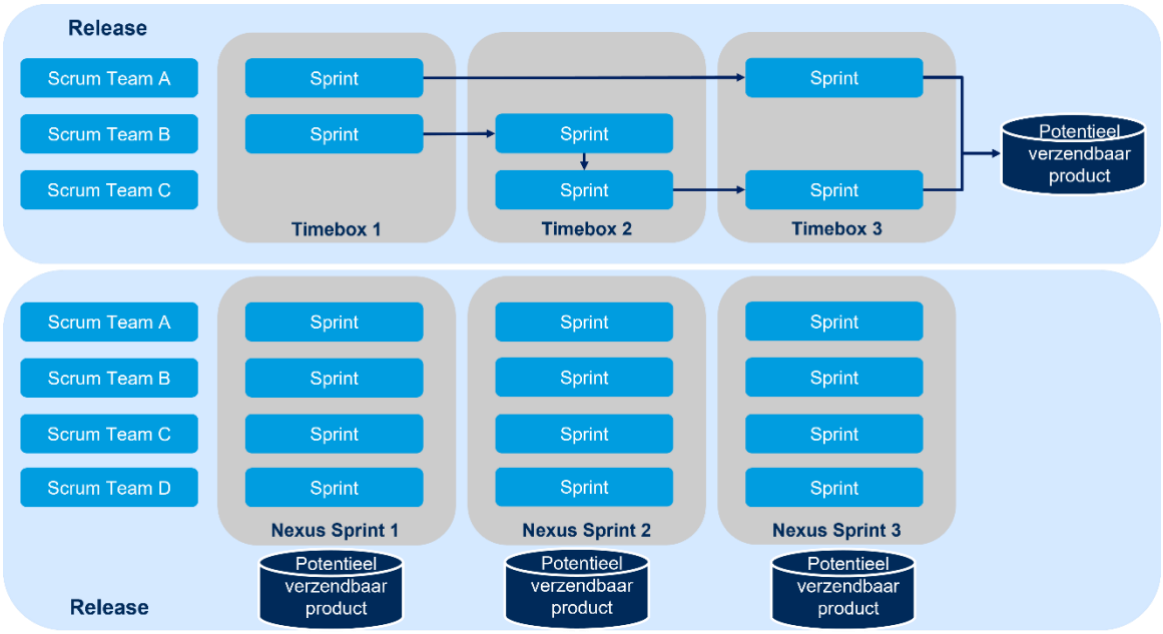
Een aanpak als Nexus zorgt ervoor dat werk niet als een pingpongbal tussen de teams rond stuitert en dat bij de werkverdeling niet alleen rekening wordt gehouden met domeinkennis, maar ook met de ervaring die teams hebben opgedaan bij de ontwikkeling van verwante of afhankelijke opleveringen. Er is niet alleen een geïntegreerde aanpak nodig, ook een geoptimaliseerde aanpak.

Nexus bereikt dit alles door een kader te bieden waarin teams de lege plekken kunnen invullen.

12.2 Het verschil tussen Nexus en een release-aanpak

Hoewel Nexus erg lijkt op wat vroeger bekend stond als Agile releases, verschilt het fundamenteel in die zin dat Nexus zich slechts op één sprint of time-box tegelijk richt. Dit is een belangrijk verschil en uitgangspunt ten opzichte van de release-aanpak, omdat specifiek wordt erkend dat het plannen van releases aan het begin van een project gebaseerd is op onvolledige kennis – en het was dit specifieke feit dat de aanleiding was voor het verlaten van de Waterval-aanpak.

Figuur 22 Nexus versus release aanpak



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us.

Velen zullen beweren dat releaseplanning op hoog niveau gebeurt, niet gedetailleerd, dat deze verandert in de loop van een project en wordt aangepast naarmate de eisen en het inzicht veranderen. Dit is zeker de bedoeling en zo hoort het te zijn. De realiteit is echter dat releaseplanningen veel minder veranderen dan wat voorstanders veronderstellen of bepleiten, omdat de aanpak zo lijkt op wat de meeste deelnemers in projecten gewend zijn te doen in Waterval projecten.

Veel releases omvatten opeenvolgende iteraties of sprints, terwijl een Nexus zich altijd richt op slechts één iteratie. Er kunnen voordelen van beide benaderingen beargumenteerd worden.

Vergeet niet dat de release-aanpak niet essentieel is in Scrum en werd overgenomen van andere methoden, terwijl Nexus is ontworpen voor Scrum.

In de bovenstaande grafiek zien we dat een release verdacht veel lijkt op een Gantt-grafiek die in Waterval-projecten wordt gebruikt.

Sprints die deel uitmaken van een Nexus-sprint zijn incrementen van één iteratie en de gecombineerde deliverable van de iteraties die deel uitmaken van de Nexus-sprint is gedefinieerd in de DoD voor de gehele Nexus-sprint.

12.3 De nieuwe manier van schalen met het Nexus framework

We kunnen Nexus zien als een framework dat bestaande Scrum methoden, verantwoordelijkheden, artefacten, enzovoort gebruikt, maar dat de creatie van een

geïntegreerd increment van waarde voor de business, aan Scrum toevoegt. De focus van een Nexus team ligt daarom meer op afhankelijkheden, interoperabiliteit, bedrijfsresultaten en de waarde die het creëert, dan het geval zou zijn als alleen Scrum, of Scrum met een intern ontwikkelde release-aanpak, zou worden gebruikt.

Het Nexus-team levert één *done* voor de hele Nexus en doet dat voor elke iteratie in de Nexus.

Om dit te kunnen doen, worden Scrum verantwoordelijkheden uitgebreid. Een nieuwe verantwoordelijkheid wordt gecreëerd: het Nexus Integration Team. De rol van het Nexus Integration Team is het coördineren, coachen en begeleiden van de toepassing van Nexus en Scrum, om waarde voor de klant te maximaliseren. Let wel: zij houden geen toezicht op het werk van de Developers, omdat dat in strijd zou zijn met het principe van zelfsturing.

Het Nexus Integration Team (Nexus integratieteam, NIT) is een nieuwe verantwoordelijkheid voor een nieuw team, met een Product Owner, een Scrum Master en vertegenwoordigers van elk Scrum-team. De vertegenwoordigers zijn leden van het integratieteam, die de Developers van een Scrum-team vertegenwoordigen.

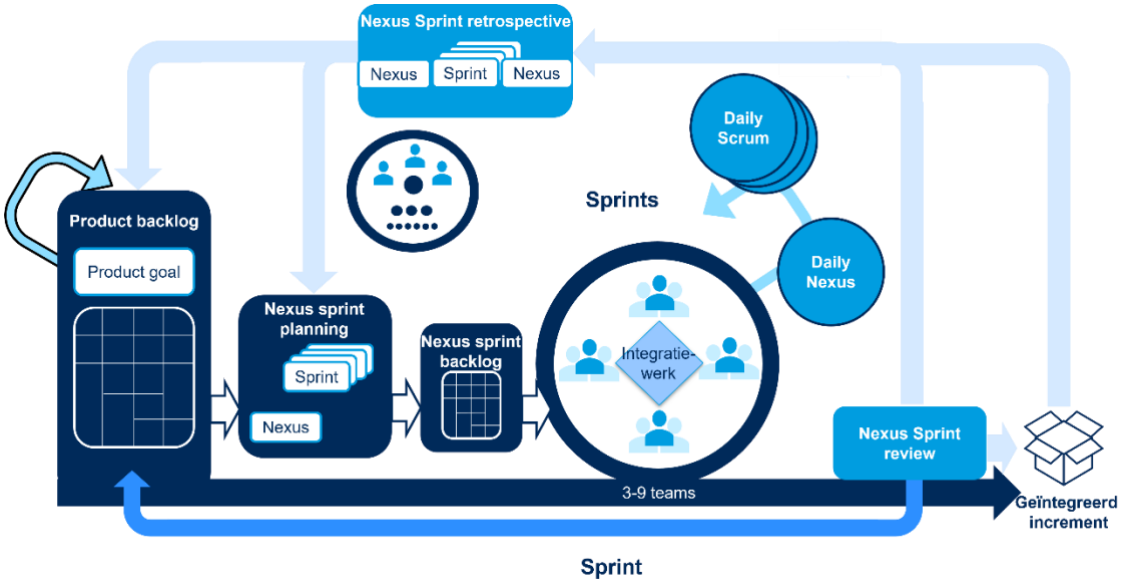
Alle Scrum-teams werken op basis van dezelfde product backlog en verfijning (refinement) zorgt ervoor dat alle teamleden een goed zicht hebben op en begrip van welke items als volgende aan de beurt zijn. Het is ook goed om ervoor te zorgen dat voor de volgorde van items een visueel middel wordt gebruikt om zo de urgentie voor de business te begrijpen.

Zodra Scrum-teams items hebben geselecteerd, voeren ze op de gebruikelijke manier een sprint uit en werken ze aan hun eigen sprint backlog.

Geen van de bestaande Scrum gebeurtenissen wordt vervangen. In plaats daarvan zullen sommige gebeurtenissen nu een gecombineerde activiteit worden van alle Scrum-teams die deelnemen aan de Nexus, of op zijn minst door vertegenwoordigers van de teams.

Onderstaand plaatje schetst het exoskelet dat door het Nexus-framework wordt gecreëerd.

Figuur 23 Nexus gebruiken met Scrum



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us en Schwaber, K. & Scrum.org (2021). *The Nexus™ Guide – The Definitive Guide to Scaling Scrum with Nexus.* Scrum.org. https://www.scrum.org/resources/nexus-guide.

12.4 Het Nexus-proces

Opmerking: Dit gedeelte wordt hier gepresenteerd zoals het in de Nexus Guide staat.

Het werk in een Nexus sprint kan worden gedaan door elk lid van de sprints die deel uitmaken van die Nexus sprint. Dus alle teamleden van Scrum-teams die deel uitmaken van de Nexus kunnen werken als onderdeel van een breder crossfunctioneel team. Teamleden die aan backlog items werken, zullen dus gekozen worden op basis van zowel hun expertise als de onderlinge afhankelijkheid van werk items, om zo het detailleren van werk te minimaliseren.

Tijdens Nexus planning worden backlog items verfijnd en de teams die er waarschijnlijk aan zullen werken worden zo vroeg mogelijk geïdentificeerd. Dat betekent dat dit een nieuwe activiteit is, die wordt toegevoegd tijdens verfijning van de product backlog.

Vervolgens wordt een geïntegreerde planningssessie gehouden, genaamd Nexus sprint planning. Vertegenwoordigers van alle Scrum-teams komen bij elkaar om de verfijnde product backlog te bespreken en te beoordelen, items uit de backlog voor elk team te selecteren en deze in de sprint backlog van elk team te plaatsen. Een gecombineerde Nexus backlog wordt ook bijgehouden, om een geïntegreerde beoordeling en retrospective van Nexus uitkomsten te vereenvoudigen.

De specifieke reden om een geïntegreerde Nexus backlog bij te houden is om afhankelijkheden aan het licht te brengen!

Deze beschrijving maakt duidelijk dat de Scrum Master van een team niet de ideale kandidaat is als vertegenwoordiger, aangezien dit in strijd is met het principe van zelfsturing en de taak van de Scrum Master als dienend leider.

Elk team plant vervolgens de eigen sprint, en, indien nodig, integreert werk met anderen. De teams zorgen ervoor dat sprint goals in lijn zijn met de algemene Nexus-doelen zoals gedefinieerd en overeengekomen tijdens Nexus sprint planning.

Ontwikkeling en testen gebeuren zoals gebruikelijk in de sprint, maar voor sommige activiteiten is coördinatie tussen teams nodig, zodat ook geïntegreerd getest kan worden als er afhankelijkheden zijn.

Daarom verdient het de voorkeur dat een gemeenschappelijke test- en integratieplatform en -aanpak worden overeengekomen, gebruikt en onderhouden.

Teams voeren nog steeds hun daily scrum uit, maar vertegenwoordigers van alle teams komen bovendien dagelijks bijeen om integratieproblemen en belemmeringen te bespreken en er wordt feedback gegeven aan elk team, om de beurt. Let wel: The Nexus Guide zegt niet dat de vertegenwoordiger van het team altijd dezelfde persoon is. Ja, het kan een voordeel geven om dit te doen, vanuit een continuïteitsperspectief. Toch moet de vertegenwoordiger de persoon zijn die het beste de problemen kan aanpakken die op dat moment in de Nexus worden ervaren.

De Nexus daily scrum wordt daarom gedaan vóór de individuele daily scrums, zodat feedback en meer bewustzijn deel uitmaken van het dagplan van elk team.

Sprint reviews worden gedaan als een Nexus. Nexus sprint reviews laten daarom een geïntegreerde Nexus 'Done' zien. De Product Owner kan deze informatie vervolgens gebruiken om de product backlog bij te werken en aan te passen.

Een Nexus sprint review wordt gedaan met vertegenwoordigers van elk team. Uitdagingen, problemen en geleerde lessen worden meegenomen naar de teams, die op hun beurt de eigen sprint retrospective doen.

Vertegenwoordigers van elk team komen na de team retrospective opnieuw bijeen om de gemeenschappelijke lessen en de door alle teams te nemen maatregelen te bespreken.

Let wel: De complexiteit van projecten, wijzigingen of verbeteringsinitiatieven is meestal groot. Het is verstandig om ook na te denken over hoe het werk kan worden geautomatiseerd of hoe complexe workflows optimaal kunnen worden beheerd.

12.5 Nexus-verantwoordelijkheden in meer detail

De verantwoordelijkheid van het Nexus Integration Team is ervoor te zorgen dat in elke iteratie een geïntegreerd increment wordt geproduceerd. Scrum-teams ontwikkelen incrementen die, wanneer samengebracht, voldoen aan de Nexus goal.

Het Nexus Integration Team is een Scrum-team bestaande uit:

- de Product Owner;
  - denk eraan: één product backlog en één Product Owner.
- een Scrum Master;
  - meestal één uit de deelnemende teams.
- deelnemers in het Nexus Integration Team;
  - die de deelnemende teams vertegenwoordigen, tenminste één per deelnemend team.

Als Nexus Integration Team deelnemers werken in de individuele sprints, wordt prioriteit gegeven aan het werk voor het Nexus Integration Team. Het werk dat zij uitvoeren als onderdeel van het Nexus Integration Team heeft altijd voorrang boven het werk in individuele Scrum-teams. Dit is belangrijk, omdat het oplossen van problemen die te maken hebben met de manier waarop teams samenwerken of het omgaan met problemen die de Nexus uitkomsten kunnen beïnvloeden of die veel teams raken, prioriteit heeft vanwege de impact en het risico voor alle betrokkenen.

Hoewel lidmaatschap van het Nexus Team niet noodzakelijkerwijs hetzelfde is van Nexus tot Nexus, kan het voordelen hebben om het lidmaatschap van het Nexus Integration Team stabiel te houden – bijvoorbeeld om te begrijpen hoe het werk van Nexus naar Nexus gaat. De teamleden moeten echter ook rekening houden met de unieke vereisten van de specifieke Nexus en daarom zal het team waarschijnlijk in de loop van de tijd veranderen.

Het Nexus Integration Team is eigenaar van alle integratiekwesties en het is hun taak ervoor te zorgen dat Scrum-teams hun taak altijd beschouwen als een onderdeel van een groter geheel. Beperkingen en afhankelijkheden die de Nexus kunnen belemmeren zijn een gebied van focus en aandacht.

Het team speelt ook een rol in het coachen van de eigen leden van Scrum-teams op hun rol in een Nexus.

12.6 Product Ownerschap

Teams op basis van Nexus werken met één product backlog, daarom werken ze met één Product Owner die de samenstelling van de product backlog bepaalt.

De traditionele verantwoordelijkheid van Product Owner verandert niet, behalve dat deze deelneemt aan Nexus-activiteiten die voorheen niet bestonden.

Let wel: Het gebruik van een Product Owner Team structuur zoals eerder beschreven, wordt over het algemeen afgekeurd door Scrum puristen – het kan echter van waarde zijn om de twee benaderingen te combineren.

12.7 Scrum Master in het Nexus Integration Team

Zoals eerder vermeld, heeft een Nexus Integration Team tenminste één Scrum Master nodig. Hoewel deze Scrum Master ook Scrum Master kan zijn voor één van

de samenstellende Scrum-teams, is dit waarschijnlijk geen goed idee in complexe omgevingen, vooral wanneer organisaties Nexus voor het eerst gaan gebruiken.

De Nexus Scrum Master is verantwoordelijk voor het coachen, het faciliteren van de communicatie en het mogelijk maken van Nexus uitkomsten, zoals in een sprint het geval is.

12.8 Nexus Integration Team leden

Een geschaalde aanpak van ontwikkeling en levering kan heel anders zijn dan wat Scrum-teams gewend zijn en zal hoogstwaarschijnlijk extra tools en technieken vragen. Kies lidmaatschap van het Nexus Integration Team (NIT) op basis van de bekwaamheid en vaardigheden van iemand op deze gebieden, omdat het NIT, naast het effectieve gebruik van nieuwe tools en technieken, ook verantwoordelijk is voor de implementatie van de tools, de processen en de praktijken die nodig zijn voor succes.

Als specialisten in de manier waarop een Nexus in de context van de organisatie zal werken, worden NIT-leden ook coaches en gidsen in het gebruik van Nexus en de bijbehorende tools en technieken.

Een vaak gestelde vraag is hoe de rol van de architect verandert in Scrum. Architecten in Scrum definiëren alleen architectuurprincipes en niet gedetailleerde operationele modellen die geïmplementeerd worden. In Nexus worden de definitie en implementatie van deze operationele modellen de verantwoordelijkheid van het NIT.

NIT-leden vervullen, vanwege hun zicht op wat er in een Nexus sprint gebeurt, ook de taak van kwaliteitsbewaker – specifiek met betrekking tot de integratie van Nexus elementen tijdens de iteratie.

12.9 Nieuwe gebeurtenissen in Nexus

Als een exoskelet op hoger niveau, voor de sprints die deel uitmaken van de Nexus, worden nieuwe gebeurtenissen geïntroduceerd, die zorgen voor coördinatie en integratie, namelijk:

12.9.1 Nexus sprint planning

Wat gaat er gebeuren in deze Nexus, wie gaat het doen, wat zijn de onderlinge afhankelijkheden, wat zijn de prioriteiten? Dit zijn allemaal belangrijke vragen om te stellen tijdens de Nexus sprint planning.

De Product Owner begeleidt de deelnemers bij het selecteren van product backlog items voor elk Scrum-team, maar schrijft niet voor waaraan gewerkt zal worden door wie.

Maar voordat dit kan gebeuren, werken vertegenwoordigers van verschillende Scrum-teams samen met de Product Owner aan de verfijning van de product backlog, met name het aanpassen van de volgorde van items, door de domeinkennis die zij meebrengen toe te passen. Vaak wordt bediscussieerd wat de

prioriteiten voor de business zijn en wat de afhankelijkheden zijn die eerst moeten worden aangepakt. Product backlog verfijning is een evenwichtsoefening tussen business professionals en technische of praktische vereisten waaraan moet worden voldaan om de business vereisten te bereiken.

Indien mogelijk nemen alle Scrum-teamleden deel aan verfijning, om begrip en communicatie te maximaliseren en misverstanden en veronderstellingen te minimaliseren.

Het doel van de Nexus sprint en de uitkomsten die moeten worden bereikt door de gecombineerde teams, wordt tijdens de Nexus sprint planning vastgelegd in de Nexus sprint goal. Eénmaal definitief gemaakt en begrepen door alle teams, gaat elk Scrum-team verder met de eigen sprint planning.

Dit kan het beste plaatsvinden in een gemeenschappelijke ruimte, zodat teams voortdurend kunnen samenwerken en afhankelijkheden, problemen en risico's kunnen delen naarmate ze worden ontdekt.

Omdat teams items kiezen uit dezelfde backlog, onderhandelen en heroverwegen ze ook vaak wat hun team kan doen en wat beter door een ander Scrum-team gedaan kan worden – zelfs als er afhankelijkheden zijn.

Hoewel een goed gedefinieerde product backlog het ontdekken van nieuwe afhankelijkheden of eisen zal minimaliseren, worden sommige onvermijdelijk ontdekt tijdens Nexus sprint planning.

Nieuw ontdekte vereisten en afhankelijkheden worden onmiddellijk zichtbaar gemaakt en het gebruik van een gemeenschappelijke visuele planningstool kan heel nuttig zijn. Het minimaliseren van afhankelijkheden door backlog items te verplaatsen tussen teams en het herschikken van werk, zijn gemeengoed tijdens Nexus sprint planning.

De Nexus sprint planning verbetert product backlog verfijning en omdat Scrum-teams nu worden blootgesteld aan een bredere context, is het gebruik van Nexus een duidelijke verbetering van product backlog verfijning in de loop van de tijd.

12.9.1 De Nexus daily scrum

Het doel van de Nexus daily scrum is om de voortgang van het huidige geïntegreerde increment te inspecteren en om team overstijgende afhankelijkheden of integratieproblemen te identificeren als ze worden ontdekt.

Elk Scrum-team moet op de juiste manier vertegenwoordigd zijn en hier geeft de bewuste woordkeuze aan dat de persoon die de problemen het beste begrijpt aanwezig moet zijn bij de Nexus daily scrum. Verwar deelname aan de Nexus daily scrum niet met het lidmaatschap van het Nexus Integration Team – het lidmaatschap van het Nexus Integration Team blijft constant, althans voor de Nexus sprint, maar de Nexus daily scrum vertegenwoordiging kan variëren gedurende de Nexus sprint.

12.9.2 Nexus sprint review

De Nexus sprint review wordt aan het eind van de sprint gehouden om feedback te geven op het geïntegreerde increment dat een Nexus tijdens de sprint heeft gebouwd. Alle individuele Scrum-teams komen samen met belanghebbenden om het geïntegreerde increment te beoordelen. De product backlog kan worden aangepast.

12.10 Nexus gebeurtenissen

De duur van een Nexus gebeurtenis wordt bepaald door de duur van de corresponderende gebeurtenis in Scrum. Het zijn timeboxes in aanvulling op de overeenkomstige Scrum gebeurtenis.

12.10.1 Verfijning

Verfijning van de geschaalde product backlog dient een tweeledig doel. Het helpt de Scrum-teams om te voorspellen welk team welke product backlog items zal opleveren en het identificeert de afhankelijkheden tussen die teams. Deze transparantie stelt de teams in staat afhankelijkheden te bewaken en te minimaliseren.

Verfijning van product backlog items door Nexus gaat door totdat de product backlog items voldoende onafhankelijk zijn om aan gewerkt te worden door een enkel Scrum-team, zonder overmatige conflicten.

Het aantal, de frequentie, de duur en de aanwezigheid bij verfijning zijn gebaseerd op de afhankelijkheden en de onzekerheid die inherent zijn aan de product backlog. Product backlog items doorlopen verschillende niveaus van decompositie, van zeer grote en vage verzoeken tot uitvoerbaar werk dat een enkel Scrum-team binnen een sprint zou kunnen opleveren.

Verfijning vindt continu plaats gedurende de sprint, als dat nodig en gepast is. Product backlog verfijning zal doorgaan binnen elk Scrum-team, zodat de product backlog items klaar zijn voor selectie in een Nexus sprint planning.

12.10.2 Nexus sprint planning

Het doel van de Nexus sprint planning is het coördineren van de activiteiten van alle Scrum-teams voor een enkele sprint. De Product Owner levert domeinkennis en begeleidt beslissingen over selectie en prioriteit. De product backlog is voldoende verfijnd, afhankelijkheden zijn geïdentificeerd en verwijderd of geminimaliseerd vóór de Nexus sprint planning.

Effectieve Nexus sprint planning bestaat uit drie stappen:

1. Zoals in Scrum, **selecteren teams in de Nexus hun eigen werk voor de sprint** die ze gaan doen. Dit is bij voorkeur een gezamenlijke activiteit met vertegenwoordiging van andere Scrum-teams, om dubbel werk te voorkomen.
2. **Ieder team volgt het eigen sprint planning proces**. Dit kan in parallel plaatsvinden.

3. **Teams verfijnen werk wanneer afhankelijkheden worden ontdekt.** Werk wordt mogelijk herverdeeld tussen teams om onnodige onderbrekingen en afhankelijkheden tussen teams te voorkomen.

Tijdens de Nexus sprint planning, valideren de vertegenwoordigers van elk Scrum-team de volgorde van het werk zoals die is vastgesteld tijdens verfijning en passen deze zo nodig aan. Aan verfijning nemen bij voorkeur alle leden van Scrum-teams in de Nexus deel, om problemen in de communicatie te minimaliseren.

De Product Owner bespreekt de Nexus sprint goal tijdens de Nexus sprint planning. De Nexus sprint goal beschrijft het doel dat zal worden bereikt door de Scrum-teams tijdens de sprint. Zodra het totale werk voor de Nexus is begrepen, volgt de eigen sprint planning van elk Scrum-team. De Scrum-teams delen nieuwe afhankelijkheden die ze vinden met andere Scrum-teams in de Nexus. Nexus sprint planning is afgerond wanneer alle Scrum-teams klaar zijn met hun sprint planning.

Tijdens de Nexus sprint planning kunnen nieuwe afhankelijkheden boven komen. Deze worden inzichtelijk gemaakt en geminimaliseerd. De volgorde van het werk in de teams kan ook worden aangepast. Een voldoende verfijnde product backlog zal het ontstaan van nieuwe afhankelijkheden tijdens de Nexus sprint planning beperken. Alle product backlog items die geselecteerd zijn voor de sprint en hun afhankelijkheden worden transparant gemaakt in de Nexus sprint backlog.

12.10.3 Nexus sprint goal

De Nexus sprint goal is een doelstelling voor de sprint. Het is de som van alle werkzaamheden en sprint goals van de Scrum-teams binnen de Nexus. De Nexus laat de functionaliteit zien die is ontwikkeld (Done) om de Nexus sprint goal te bereiken, om zo feedback van de belanghebbenden te krijgen.

12.10.4 Nexus daily scrum

De Nexus daily scrum is een gebeurtenis voor geschikte vertegenwoordigers (individuele Developers) om de huidige staat van het geïntegreerde increment te inspecteren en om integratieproblemen of nieuw ontdekte team overschrijdende afhankelijkheden of team overschrijdende effecten te identificeren.

Tijdens de dagelijkse Nexus scrum, richten de aanwezigen zich op de impact van elk team op het geïntegreerde increment en bespreken:

- Is het werk van de vorige dag met succes geïntegreerd? Zo niet, waarom niet?
- Welke nieuwe afhankelijkheden of effecten zijn vastgesteld?
- Welke informatie moet worden gedeeld tussen de teams in de Nexus?

De teams gebruiken de Nexus daily scrum om de voortgang naar het Nexus sprint goal te inspecteren. Minimaal tijdens elke Nexus daily scrum wordt de Nexus sprint backlog aangepast aan het huidige inzicht in het werk van de Scrum-teams binnen de Nexus.

De individuele Scrum-teams nemen dan kwesties en werk die werden geïdentificeerd tijdens de Nexus daily scrum, terug naar hun individuele Scrum-teams voor planning binnen hun individuele daily scrums.

12.10.5 Nexus sprint review

De Nexus sprint review wordt gehouden aan het einde van de sprint om feedback te geven op het geïntegreerde increment dat de Nexus heeft gebouwd gedurende de sprint en om, indien nodig, de product backlog aan te passen.

Een Nexus sprint review vervangt individuele Scrum-team sprint reviews, omdat het gehele geïntegreerde increment de focus is bij het verzamelen van feedback van belanghebbenden. Mogelijk kan niet al het voltooide werk in detail worden getoond. Technieken kunnen nodig zijn om maximale feedback van belanghebbenden te krijgen. Het resultaat van de Nexus sprint review is een herziene product backlog.

12.10.6 Nexus sprint retrospective

De Nexus sprint retrospective is een formele gelegenheid voor een Nexus om zichzelf te inspecteren en aan te passen en een plan te maken voor verbeteringen die tijdens de volgende sprint worden doorgevoerd om continue verbetering te garanderen. De Nexus sprint retrospective vindt plaats na de Nexus sprint review en voor de volgende Nexus sprint planning.

De retrospective bestaat uit drie delen:

1. Het eerste deel is een gelegenheid voor de vertegenwoordigers in een Nexus om samen te komen en **kwesties te identificeren die impact hadden op meer dan één team**. Het doel is om gezamenlijke problemen transparant te maken voor alle Scrum-teams.
2. Het tweede deel bestaat eruit dat **elk Scrum-team de eigen sprint retrospective houdt**, zoals beschreven in het Scrum-framework. Zij kunnen issues die in het eerste deel van de Nexus retrospective naar voren zijn gekomen, gebruiken als input voor hun teamdiscussies. De individuele Scrum-teams stellen acties vast om deze kwesties aan te pakken.
3. Het laatste, derde deel is een gelegenheid voor de vertegenwoordigers van de Scrum-teams om opnieuw samen te komen en **afspraken te maken over hoe de vastgestelde acties te visualiseren en te volgen**. Dit stelt de Nexus als geheel in staat zich aan te passen.

Veel voorkomende valkuilen in het schalen worden in iedere retrospective besproken:

- Is er werk blijven liggen? Heeft de Nexus technische schuld gegenereerd?
- Zijn alle artefacten, met name code, vaak (dagelijks) met succes geïntegreerd?
- Is de software met succes gebouwd, getest en vaak genoeg ingezet om een opeenhoping van onopgeloste afhankelijkheden te voorkomen?

Bij de bovenstaande vragen, bespreek zo nodig:

- Waarom is dit gebeurd?
- Hoe kan technische schuld worden opgeheven?
- Hoe kan herhaling worden voorkomen?

12.11 Nexus artefacten

Artefacten vertegenwoordigen volgens de Scrum Guide werk of waarde. Ze bieden de transparantie die inspectie en aanpassing mogelijk maken.

Product backlog

Er is één product backlog voor de gehele Nexus en alle Scrum-teams die in de Nexus werken. De Product Owner is eindverantwoordelijk voor de product backlog, inclusief de inhoud, beschikbaarheid en de ordening ervan.

Op grotere schaal moet de product backlog duidelijk maken waar afhankelijkheden kunnen worden opgespoord en geminimaliseerd. Om het werk te ondersteunen, is de functionaliteit zoals beschreven in product backlog items vaak granulair of in dunne plakjes. Product backlog items worden ‘ready’ geacht voor de Nexus sprint planning meeting wanneer de Scrum-teams items kunnen selecteren die gedaan worden zonder of met minimale afhankelijkheden van andere Scrum-teams.

Nexus sprint backlog

Een Nexus sprint backlog is de verzameling van product backlog items uit de sprint backlogs van de individuele Scrum-teams. Deze wordt gebruikt om afhankelijkheden en de werkstroom in de sprint te markeren. Deze sprint backlog wordt minstens dagelijks bijgewerkt, vaak als onderdeel van de Nexus daily scrum.

Geïntegreerd increment

Het geïntegreerde increment is de som van alle geïntegreerde werkzaamheden die door een Nexus zijn voltooid. Het geïntegreerde increment is bruikbaar en potentieel inzetbaar, wat betekent dat het voldoet aan de definition of done (DoD). Het geïntegreerde increment wordt geïnspecteerd bij de Nexus sprint review.

Artefact transparantie

Net als Scrum, is Nexus gebaseerd op transparantie. Het Nexus Integration Team werkt samen met de Scrum-teams in een Nexus en met de organisatie om ervoor te zorgen dat er transparantie is in alle artefacten en dat de status van het geïntegreerde increment breed wordt begrepen, door alle teamleden.

De status van Nexus-artefacten is bepalend voor veel beslissingen die tijdens de Nexus worden genomen en is zo effectief zijn als de mate van transparantie over de status van artefacten. Gedeeltelijke of onvolledige informatie zal leiden tot onjuiste of gebrekkige beslissingen, die op hun beurt zullen doorwerken in en vergroten op de schaal van de Nexus.

Incrementen worden zo ontwikkeld dat afhankelijkheden worden opgespoord en opgelost voordat de technische schuld onaanvaardbaar wordt voor de Nexus of de

lopende operatie. Een gebrek aan transparantie maakt het leiden van een Nexus, het minimaliseren van risico's en het maximaliseren van waarde onmogelijk.

Definition of done (DoD)

Het Nexus Integration Team is verantwoordelijk voor een definition of done die kan worden toegepast op het geïntegreerde increment dat in elke sprint wordt ontwikkeld. Alle Scrum-teams van een Nexus houden zich aan deze definition of done. Het increment is pas Done als het geïntegreerd, bruikbaar en inzetbaar is voor de Product Owner.

Individuele Scrum-teams kunnen ervoor kiezen om binnen hun eigen team een strengere definition of done te hanteren, maar kunnen geen minder strenge criteria hanteren dan wat voor het increment is overeengekomen.

12.12 De kern van Nexus is het Nexus Integration Team

De Nexus bestaat uit 3 tot 9 Scrum-teams, met één gezamenlijke Product Owner en één product backlog. Het Nexus Integration Team (NIT) bestaat uit

- ***de*** Product Owner;
- ***een*** Scrum Master;
- en Nexus Integration Teamleden, van teams die deel uitmaken van de Nexus.

Hoewel de Product Owner deel uitmaakt van het NIT, vervult zij of hij ook de verantwoordelijkheid van Product Owner in elk Scrum-team dat deel uitmaakt van de Nexus. De Product Owner zorgt ervoor (eindverantwoordelijk) dat de product backlog wordt geordend en verfijnd door de waarde voor belanghebbenden te maximaliseren.

De Scrum Master in het Nexus Integration Team is er verantwoordelijk voor dat Nexus als methode wordt begrepen en werkt. Zij faciliteren ook typisch Nexus gebeurtenissen zoals Nexus daily scrum, Nexus sprint planning, cross-team verfijning, Nexus sprint review en Nexus retrospective, of op zijn minst ondersteunen zij degenen die deze faciliteren.

De leden van het Nexus Integration Team zijn de Developers van Scrum-teams binnen de Nexus. Het NIT coördineert alle activiteiten binnen een Nexus. Teamleden worden, net als Scrum Masters dat doen in Scrum, coaches in hun eigen teams. Zij coördineren en begeleiden gezamenlijk de implementatie, toepassing en het gebruik van Nexus om tot optimale uitkomsten te komen.

De samenstelling van het Nexus Integration Team kan in de loop van de tijd veranderen, afhankelijk van wat de Nexus op dat moment nodig heeft.

NIT-leden zijn gewoonlijk afkomstig uit Scrum-teams binnen de Nexus en in het algemeen is het NIT-lidmaatschap deeltijds; maar sommige NIT-leden kunnen wel voltijds lid zijn, vooral in complexe omgevingen waar coördinatie van afhankelijk werk van vitaal belang is.

NIT-leden geven hun rol als NIT-lid altijd voorrang boven die als Scrum-teamlid. Het geheel is belangrijker dan de delen.

Eindverantwoordelijkheid

Het Nexus Integration Team heeft de eindverantwoordelijk dat tenminste elke sprint een geïntegreerd increment wordt geproduceerd. Dit zorgt ervoor dat het geïntegreerde product de focus van alle teams heeft, in plaats van alleen de inspanningen van individuele teams.

Geschaalde softwareontwikkeling vraagt om tools en praktijken die individuele Scrum-teams misschien niet vaak gebruiken. Daarom moet het Nexus Integration Team bestaan uit professionals die technisch onderlegd zijn. Integratie is echter breder dan code schrijven. Hoe teams samenwerken is een vitaal onderdeel van de integratie. Bijvoorbeeld, hoe teams reageren op het afbreken van builds of hoe zij collectief code-eigenaar zijn, maken allemaal deel uit van een succesvolle integratie. Het Nexus Integration Team heeft leden met zowel 'harde' (technische) als 'zachte' (menselijke) vaardigheden.

Activiteit

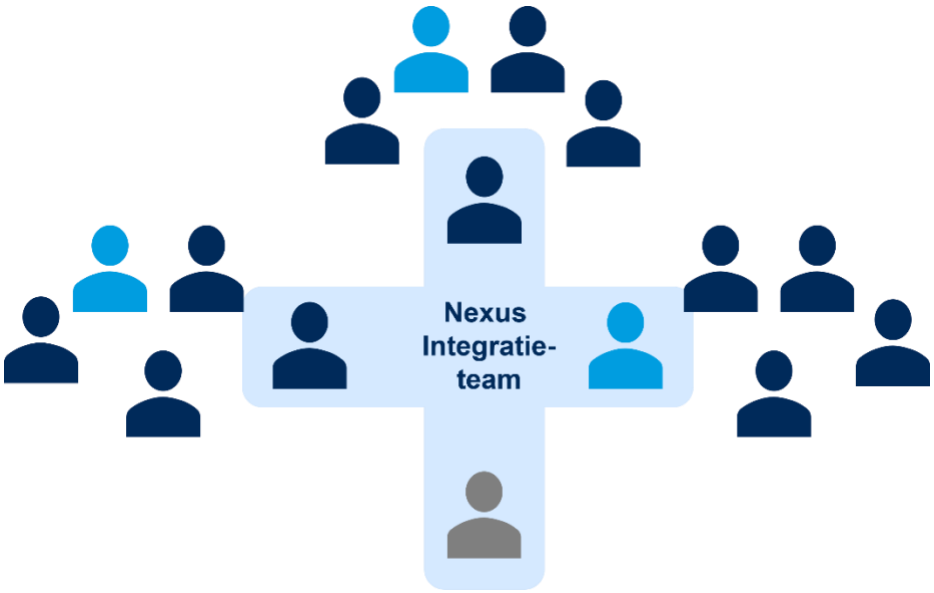
Nexus Integration Teamleden fungeren als coaches van de aangesloten Scrum-teams. Als zodanig bezitten Nexus Integration Teamleden de nodige vaardigheden en eigenschappen om de Scrum-teams binnen de Nexus in staat te stellen de staat van het geïntegreerde increment voortdurend te verbeteren.

Activiteiten die het Nexus Integration Team dagelijks kan uitvoeren zijn onder meer:

- helpen bij het coördineren van het werk tussen de teams;
- zo vroeg mogelijk bewust maken van afhankelijkheden;
- ervoor zorgen dat integratietools en -praktijken bekend zijn en worden gebruikt;
- fungeren als adviseur, coach en link in de communicatie;
- soms assisteren bij het werk, wanneer dat nodig of passend is;
- faciliteren van gedeelde architectuur/infrastructuur;
- voortdurend zorgen voor transparantie van de integratie.

Het is belangrijk dat de mensen die deel uitmaken van het Nexus Integration Team dienende leiders zijn en een coachende benadering hebben van verbetering. Dit is niet het 'all-star' team. Door het Nexus Integration Team te bemensen met leden uit de Scrum-teams binnen de Nexus, zoals weergegeven in onderstaande figuur, wordt een ‘wij en zij’ tweedeling vermeden.

Figuur 24 Een Nexus Integratieteam samenstellen



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us.

Het Nexus Integration Team bestaat uit leden van de Scrum-teams. Het Nexus Integration Team werkt typisch niet als een Scrum-team samen, items trekkend uit de product backlog. In plaats daarvan zijn zij een gemeenschap van coaches en begeleiders die service verlenen aan de Scrum-teams binnen de Nexus. Omdat zij ook lid zijn van die Scrum-teams, levert de Nexus inherent eigen leiderschap.

Het Nexus Integration Team als Scrum-team

Als er ernstige integratie- of andere problemen zijn, kan het Nexus Integration Team als laatste redmiddel besluiten dat ze werk uit de backlog nemen en als Scrum-team eraan werken.

In dit geval verhuizen ze fulltime van hun bestaande Scrum-teams naar het Nexus Integration Team, totdat de problemen zijn opgelost.

Zij kunnen dit doen als:

- Het werk om vaardigheden vraagt die alleen leden van het Nexus Integration Team hebben.
- Het product in een onbekende staat is.
- De Scrum-teams binnen de Nexus tijdelijk niet in staat zijn succesvol te integreren.

Dit is geen duurzaam werkmodel, want het betekent dat de Nexus er niet in is geslaagd Scrum te schalen en dat het werk vertraagt tot het tempo van één enkel team – het Nexus Integration Team. Uiteindelijk, als alleen het Nexus Integration Team het werk kan doen, is de schaalvergroting mislukt.

De beslissing om op deze manier te werken wordt genomen door het Nexus Integration Team, inclusief de Product Owner, als een tijdelijk, laatste redmiddel om aan hun eindverantwoordelijkheid te voldoen.

Het Nexus Integration Team laten werken als een Scrum-team wordt beschouwd als een foutmodus. Dit zou een tactische keuze voor de korte termijn moeten zijn. In het algemeen is het goed voor leden van het Nexus Integration Team om samen te werken met de Scrum-teams binnen de Nexus en zo schaarse vaardigheden en kennis over te dragen door pairing en coaching.

Lidmaatschap

Anders dan de verantwoordelijkheid van de Product Owner, hoeft lidmaatschap van het Nexus Integratie Team niet permanent te zijn. In feite zou het situationeel moeten zijn. Het soort vaardigheden dat nodig is, verandert in de loop van de tijd, wat leidt tot veranderingen in de samenstelling van het team.

De Nexus kan besluiten mensen van buiten het Nexus uit te nodigen om deel uit te maken van het Nexus Integration Team. Daartoe kunnen vertegenwoordigers behoren van gebieden die van cruciaal belang zijn voor het welslagen van de Nexus. Als zij dezelfde integratieverantwoordelijkheid delen, kan hun deelname aan het Nexus Integration Team tot positieve resultaten leiden.

Het laatste woord over Nexus Integration Teams

Het Nexus Integration Team is geen managementteam en het is ook geen team van mensen met heel specifieke expertise en ervaring. Het is een team van professionals, meestal uit de individuele Scrum-teams binnen de Nexus, die ervoor zorgen dat werkwijzen en tools worden geïmplementeerd, begrepen en gebruikt om afhankelijkheden op te sporen. Zij zijn ervoor verantwoordelijk dat tenminste elke sprint een geïntegreerd increment wordt geproduceerd. Hun focus op integratie omvat zowel technische als niet-technische kwesties binnen de Nexus.

Het Nexus Integration Team handelt precies zoals een Scrum Master handelt in een Scrum-team, niet als een hiërarchische manager die leidt maar door dienend leiderschap en coaching. Zij gebruiken bottom-up intelligentie van teams om problemen op te lossen en om te verbeteren. Dat elk team in staat is voortdurend bij te dragen aan toenemende productontwikkeling, leidt tot succes in schalen.

12.13 Visualiseren van de Nexus sprint backlog

Het doel van Nexus sprint planning is het coördineren van de activiteiten van alle Scrum-teams die deelnemen aan een Nexus voor een enkele Nexus sprint. De Nexus sprint backlog wordt gemaakt tijdens de Nexus sprint planning en is een visualisatie van het werk in de Nexus. De backlog benadrukt met name afhankelijkheden.

De gecombineerde Nexus-sprint planning vindt op een gestructureerde manier plaats, waarbij ervoor wordt gezorgd dat het werk goed wordt gecoördineerd en dat rekening wordt gehouden met alle afhankelijkheden.

In een complexe omgeving is voortdurende verfijning nodig, aangezien nieuwe afhankelijkheden vaak pas tijdens een Nexus worden ontdekt.

Een gezamenlijk Nexus-verfijningsbord is een nuttige tool om het werk van Scrum-teams in de Nexus te valideren en bij te houden.

12.14 Team overstijgende verfijning in Nexus

Product backlog verfijning is een doorlopende activiteit in Scrum; het is echter geen verplichte gebeurtenis. Vanwege de complexiteit die voortvloeit uit het samenwerken van veel teams aan één product (product backlog) in een Nexus, wordt verfijning een officiële en verplichte gebeurtenis.

Gezamenlijke backlog verfijning is gericht op het voldoende ontleden van product backlog items (PBI's) zodat de teams begrijpen welk werk ze kunnen doen en in welke volgorde ze dat werk kunnen opleveren in de komende sprints. Deze gebeurtenis stelt de teams ook in staat om zich te richten op het minimaliseren en verwijderen van afhankelijkheden tussen teams.

Let wel: Voor degenen die zich afvragen wat releaseplanning vervangt in Nexus, het antwoord is: niets. Echter, gedeelde backlog verfijning creëert een wederzijds begrip van het belang, de afhankelijkheden en de volgorde van PBI's die worden opgeleverd in meerdere incrementen of sprints. Zie de vragen hieronder.

In een Nexus zijn er meerdere Scrum-teams die werk uit één product backlog halen; daarom zijn er nieuwe vragen over het verfijnen van de backlog:

- Welke teams doen welk werk?
- Hoe kunnen we het werk het beste verdelen, over sprints en teams, om een evenwicht te vinden tussen snelle levering van waarde, tegen welk risico en met welke complexiteit?

Bij het opschalen van Scrum wordt aanbevolen dat deze vragen worden beantwoord door de Developers zelf, in team overstijgende verfijning.

Vertegenwoordigers van elk team wonen een team overstijgende verfijningsgebeurtenis bij. Vertegenwoordigers worden geselecteerd en wonen de workshop bij op basis van het werk dat wordt verfijnd en niet op basis van de functie die zij binnen hun team hebben. Het kan zijn dat verschillende mensen deze workshops bijwonen op basis van de vaardigheden die nodig zijn.

Maar wordt bepaald welke teams welk werk doen?

Product backlog items ontleden en rangschikken

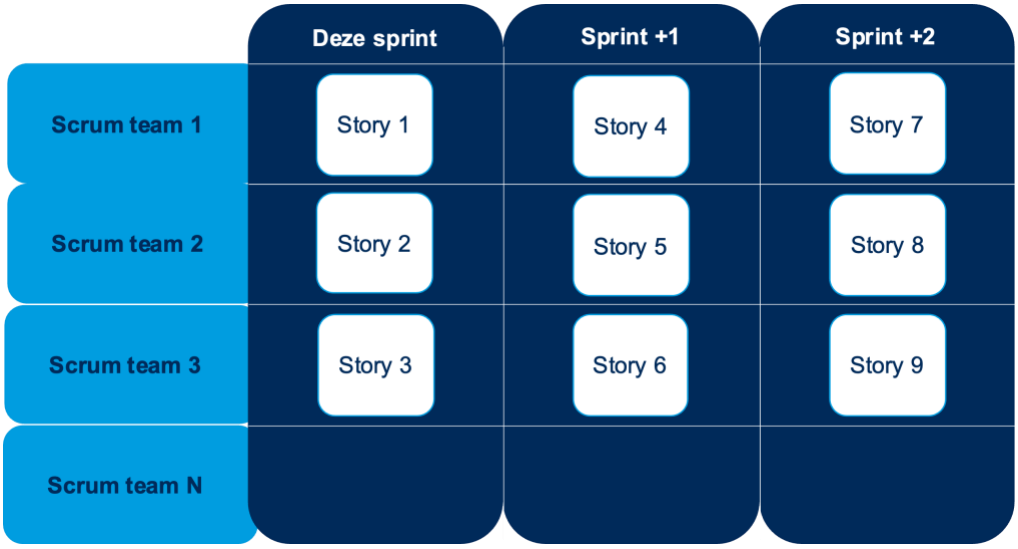
De Product Owner legt uit welke product backlog items (PBI's) moeten worden verfijnd. Dit zijn over het algemeen grote items die nog niet zijn uitgewerkt of teruggebracht tot een omvang die past binnen een sprint. Dit wijkt af van de conventionele wijsheid dat PBI's bovenaan de product backlog altijd worden verfijnd voordat de sprint planning begint. Het uitstellen van dit detailleren geeft de

mogelijkheid om de items uit te werken op basis van vaardigheden en afhankelijkheden die naar voren komen tijdens het verfijnen.

De eerste gesprekken zullen zeer vlot verlopen en gaan vooral over de vraag welke teams over de nodige vaardigheden beschikken om het werk uit te voeren. Bestaande beperkingen, vanwege specialistische vaardigheden die niet in elk team beschikbaar zijn, worden zorgvuldig overwogen.

Zodra de vertegenwoordigers van de Scrum-teams de PBI's beginnen te begrijpen, kunnen ze deze ontleden in kleinere items, die dan kunnen worden teruggebracht naar hun teams voor de normale product backlog verfijning en sprint planning.

Figuur 25 Een Nexusbord creëren



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us.

Merk op dat in de eerste verfijning ook afhankelijkheden worden opgespoord en hoewel vergelijkbaar met de traditionele releaseplanning, zal vastgesteld worden wat in deze sprint kan worden gedaan en wat moet wachten tot de volgende sprint. Afhankelijkheden sturen het gesprek en niet wat maanden eerder aan klanten is beloofd. Het is ook geen goed idee om afhankelijkheden pas te zoeken na drie sprints.

Het begrijpen van de werkstroom voor de komende sprints en het visualiseren ervan per team maakt daarom deel uit van de initiële verfijningsactiviteit, maar zodra dit is gedaan, vragen de NIT-leden zich af: “Hoe kunnen we het werk het beste rangschikken over de sprints en teams om een evenwicht te vinden tussen vroege levering van waarde enerzijds en risico en complexiteit anderzijds?”

Dit wordt gedaan door afhankelijkheden te visualiseren op het verfijningsbord en vervolgens te overwegen hoe de afhankelijkheden het beste kunnen worden beheerd.

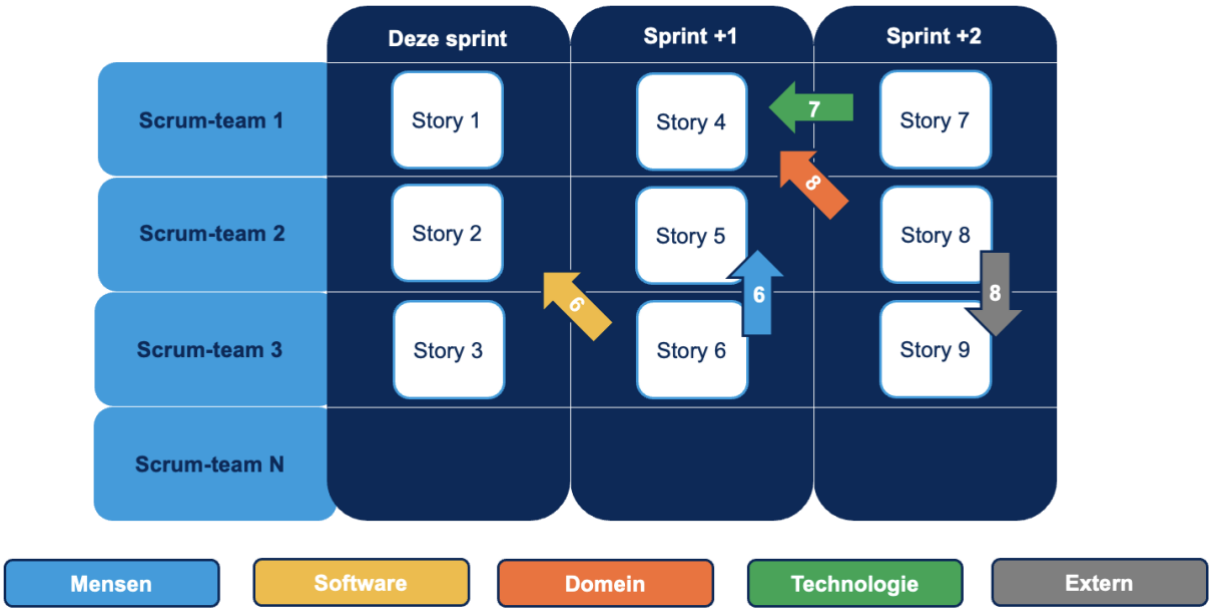
Niet alle afhankelijkheden zijn van dezelfde aard en teams moeten bij het definiëren de afhankelijkheden in categorieën indelen.

In het algemeen kunnen de volgende categorieën en subcategorieën als uitgangspunt worden gebruikt:

- **Volgorde van bouwen** – Een item kan niet worden voltooid voordat het bovenliggende item is voltooid (kan technologie, domein of software zijn)
- **Mensen/Vaardigheden** – Alleen bepaalde mensen/teams kunnen een item voltooien
- **Extern** – Het bovenliggende item wordt geleverd van buiten de Nexus

Afhankelijkheden worden aangegeven met pijlen en de kleur van de pijl helpt om onderscheid te maken tussen soorten afhankelijkheden. De richting van de pijl geeft de ouder-kind relatie aan en voor de duidelijkheid kan ook het nummer van de bovenliggende story in de pijl worden opgenomen.

Figuur 26 Omgaan met afhankelijkheden in een taak die is verspreid over Nexus scrum teams



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us.

Overweeg in ieder geval de externe afhankelijkheden te identificeren. Misschien kunnen er kleuren toegevoegd worden als dat in de context zinvol is.

Hoe meer pijlen, hoe hoger het risico en hoe belangrijker het is dat er duidelijke planning en coördinatie plaatsvindt. Afhankelijkheden markeren het kritieke pad van het werk dat nog gaat komen en de verfijning, die op haar beurt de afhankelijkheden of in ieder geval de impact van de afhankelijkheden minimaliseert.

De volgende algemene conventie kan worden gebruikt:

- ← Een horizontale pijl duidt op afhankelijkheden in één Scrum-team (over het algemeen laag risico).
- ↖ Een diagonale pijl omhoog wijst op afhankelijkheden over tijd en teams heen (over het algemeen een middelgroot risico).
- ↑ Een verticale pijl wijst op een team overstijgende afhankelijkheid, maar tijdens dezelfde sprintcyclus (dit wordt als een hoog risico beschouwd, aangezien de vertraging in één team de hele Nexus bedreigt).
- ↙ Een neerwaartse diagonale pijl geeft een externe afhankelijkheid in de tijd aan (dit wordt als een middelgroot risico beschouwd)
- ↓ Een neerwaartse verticale pijl geeft een andere afhankelijkheid in de sprintcyclus aan, maar buiten het Nexus-team (dit wordt als een afhankelijkheid met hoog risico beschouwd).

Het visualiseren van afhankelijkheden helpt het Nexus Team de complexiteit te begrijpen en het NIT kan dan proberen werkitems te herverdelen tussen teams, waarbij afhankelijk werk in één team wordt gehouden en zo team overstijgende afhankelijkheden en risico's worden opgeheven of tot een minimum worden beperkt.

Een uitzondering op deze praktijk is als het NIT besluit het werk aan een specifiek item, dat te groot is voor een enkel team, over meer teams te verdelen, om zo de oplevering tijdens deze sprint mogelijk te maken.

Hier volgen enkele manieren waarop afhankelijkheden en risico's tot een minimum kunnen worden beperkt:

- Verplaatsen van werk tussen teams om overdracht en team overstijgende afhankelijkheden te minimaliseren.
- Mensen van het ene team naar het andere overplaatsen, zodat er minder team overstijgende afhankelijkheden zijn. Dit kan het risico aanzienlijk verminderen, als bepaalde vaardigheden voor een sprint of twee worden herverdeeld over teams om de afhankelijkheid van vaardigheden in teams te minimaliseren.
- Door items op verschillende manieren op te splitsen of door het uit te voeren werk een andere vorm te geven, kan het mogelijk zijn afhankelijkheden weg te werken.
- Probeer bij het plotten van het werk op het bord alle risico's zo veel mogelijk naar voren te halen in de planning en probeer de afhankelijkheden tussen de teams zo vroeg mogelijk zichtbaar te maken.

Het verfijnen en opsporen van afhankelijkheden is geen eenmalige gebeurtenis en wordt gedurende een Nexus en over sprints heen telkens opnieuw bekeken.

12.15 Meer over Nexus sprint planning en Nexus daily scrum

Team overschrijdende informatie over verfijning levert input voor Nexus sprint planning; Nexus sprint planning houdt rekening met de huidige afhankelijkheden in de sprint.

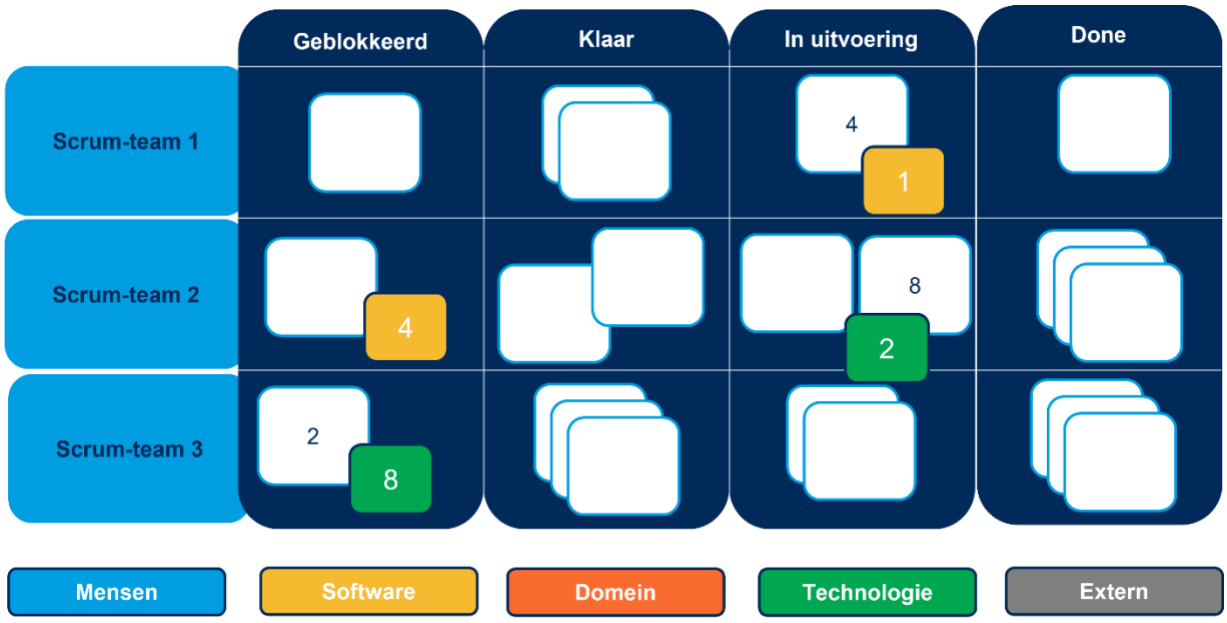
Informatie over afhankelijkheden is ook het aandachtspunt van de dagelijkse team overschrijdende synchronisatie, het beheer van risico's en de voortgang, tijdens de dagelijkse Nexus scrum.

Nexus sprint backlog

Een manier om een sprint backlog te visualiseren is het maken van een scrumbord. Afhankelijkheden in de sprint tussen de Nexus teams kunnen worden beheerd door een Nexus Scrumbord te maken zoals in het voorbeeld hieronder.

Hier wordt product backlog item PBI 1 opgeleverd door Scrum-team 2, maar het hangt af van PBI 4 die wordt opgeleverd door Scrum-team 1 en PBI 2 die wordt opgeleverd door Scrum-team 3 hangt af van PBI 8 die wordt opgeleverd door Scrum-team 2.

Figuur 27 Werken met afhankelijkheden tussen teams



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us.

Met behulp van een Geblokkeerd kolom op het Nexus scrumbord, is het mogelijk om afhankelijkheden tussen teams in de sprint weer te geven.

Deze visualisering is belangrijk, omdat ze afhankelijkheden en bijbehorende risico's binnen de sprint belicht; dit moedigt ook coördinatie aan en aandacht voor de juiste volgorde van werk tussen teams.

Tijdens de dagelijkse Nexus Scrum, is de Nexus sprint backlog (scrumbord) normaal gesproken een centraal punt en bepaalt de gesprekken in de daily scrum.

Als een Nexus geen afhankelijkheden binnen de sprint heeft, kunnen teams er ook voor kiezen de Nexus sprint backlog weer te geven via de huidige sprint kolom op het team overstijgende verfijningsbord.

In dat geval wordt het verfijningsbord het middelpunt van de daily scrum.

13 Implementeren van en slagen met Agile Scrum

Onderzoek wijst uit dat, hoewel in bijna alle organisaties Agile-initiatieven bestaan, de grote meerderheid hun eigen niveau van Agile-volwassenheid als ondermaats beschouwt.

Toch zijn de voordelen van Agile duidelijk: snellere oplevering van software, beter kunnen inspelen op veranderende prioriteiten, hogere productiviteit en betere afstemming tussen business en IT.

Scrum is veruit de populairste Agile-methode die door organisaties wordt gebruikt, dus we bevinden ons in goed gezelschap.

Er is echter ook een somber kantje aan de onderzoeksresultaten.

De uitdagingen voor het invoeren en schalen van Agile die boven aan de lijst staan, blijven de uitdagingen gerelateerd aan organisatiecultuur. Deze zijn: algemene weerstand tegen verandering, ontoereikende steun en sponsoring van het management, gebrek aan participatie van leiderschap en een organisatiecultuur die op gespannen voet staat met de Agile-waarden.

Ondanks deze uitdagingen is het echter duidelijk dat de toekomst nog wendbaarder (agile) zal zijn – en dat degenen die er niet in slagen wendbaarheid van de business te omarmen, achterop zullen raken.

13.1 Het gaat allemaal om verandering in de organisatie

McKinsey stelt in een artikel ‘The journey to an Agile organization’ dat de overstap naar een Agile-besturingsmodel moeilijk is, vooral voor gevestigde organisaties. De vijand van Agile en veel andere vooruitstrevende praktijken is het oude silo model en de verouderde managementbenadering die ermee gepaard gaat.

De verschuiving naar Agile is het moeilijkst voor het management. Zij moeten loslaten en erop vertrouwen dat hun teams het juiste doen. Het decentraliseren van besluitvorming naar het laagst mogelijke niveau is de kern van het succes van Agile en dit goed te doen, is de meest uitdagende onderneming voor een bedrijf uit het industriële tijdperk.

Experts zijn unaniem in hun oordeel. Een organisatiebrede Agile-transformatie is nodig, deze moet veelomvattend en iteratief zijn om de voordelen van Agile te realiseren.

In hun artikel benadrukt McKinsey de belangrijkste gebieden van verandering als volgt:

- mensen;
- structuur;
- processen;
- systemen en tools.

Mensen

De sleutel tot succes in de categorie Mensen is dat het leiderschap van de organisatie haar veranderde rol aanvaardt. Alle leidinggevenden moeten worden getraind om hun focus te verleggen van het toezicht houden op werk en mensen naar het bieden van een visie die hun team kan volgen en gebruiken om dingen gedaan te krijgen in multidisciplinaire, zelfsturende teams. De taak van het management is ook inspireren en coachen, eerder dan leiden. Managers gaan helpen het werk te modelleren (WAT), maar niet dicteren HOE het werk zal worden gedaan.

Dit impliceert dat alle teamleden bijdragen om dingen gedaan te krijgen en ook collectief beslissen hoe ze gedaan zullen worden. Als er nagedacht wordt over wat er in een sprint gebeurt, is dit precies wat Scrum voorstelt.

Als onderdeel van coaching richten managers zich veel meer op het helpen van mensen om te groeien en nieuwe vaardigheden en competenties te verwerven. Agile betekent dat de taak van een manager verandert, van het managen van activiteiten, processen en procedures, naar het versterken van mensen en het bouwen van teams. Een goede manier om nu naar werk te kijken is de waardestromen herkennen en hoe mensen en processen bijdragen aan het creëren van waarde voor de klant.

Hoewel dit eenvoudig lijkt, is het voor managers heel moeilijk om Tayloristisch managementgedrag, dat deel uitmaakte van hun ontwikkeling als managers, van zich af te schudden en hun nieuwe rol in de organisatie, van dienend leider, stimulator, coach en mentor te omarmen.

Het betekent ook dat invloed in organisaties veel belangrijker wordt dan status of positie en de natuurlijke uitwerking daarvan zal in het volgende hoofdstuk te zien zijn.

Structuur

Plattere organisaties en functie-overschrijdend getrainde werknemers zijn de natuurlijke uitvloeisels van deze andere manier van werken.

Nu zal een meer missie- of waarde gedreven benadering bepalen hoe teams er uit zien en uiteindelijk hoe de organisatie er uit ziet. En dit zal invloed hebben op de omvang van het personeelsbestand en op de locatie.

Omdat teams werken als multidisciplinaire eenheden en vaak tijdelijke structuren zijn, betekent dit dat organisaties hun rapportagestructuren sterk gaan

vereenvoudigen en ontdoen van lagen. De Agile-organisatie is standaard meer gedecentraliseerd en grote hoofdkantoren worden eerder een belemmering dan een hulp.

Centralisatie bemoeilijkt besturen en als gevolg daarvan worden bestuursmodellen vereenvoudigd en gestroomlijnd. Wij kunnen niet langer onszelf beschermen, omdat een andere organisatie dat zinvol vond. Wij moeten risico's grondig begrijpen en de bestuursstructuur, -methoden en -controles opbouwen en voortdurend verfijnen, met name die welke gebaseerd zijn op het risico van de organisatie.

Processen

Rigide proces- en conformiteitsregimes hinderen organisaties vandaag vaak meer dan dat ze helpen. Processen horen op een hoog niveau te worden gedefinieerd en veel vrijheid te laten voor teams om op hun eigen manier te conformeren aan de richtlijnen. Dit houdt in dat elk team een eigen manier mag hebben om hetzelfde (proces) te doen (procedure) – en, zolang de gewenste resultaten, de uitkomsten en de essentiële regels duidelijk zijn, zou dit moeten volstaan.

Organisaties gebruiken vaak processen en procedures als middel om prestatie te meten, maar als het proces of de procedure niet dezelfde zijn, kunnen prestatiemetingen bijna volledig op uitkomsten worden gebaseerd.

Systemen en tools

Het management zorgt ervoor dat de organisatie over de juiste tools en systemen beschikt en dat deze zijn gestroomlijnd voor een Agile manier van werken.

Dit houdt ook in dat alleen het principe van de architectuur van de onderneming centraal wordt gedefinieerd, zodat de architectuur in de loop van de tijd naar behoefte verder kan worden ontworpen en ontwikkeld.

Het is zinloos waardevolle incrementen te leveren als zij niet onmiddellijk kunnen worden gebruikt. Automatisering, vooral voor de realisatie en levering van producten en diensten, moet een hoge prioriteit krijgen.

In een softwareomgeving betekent dit een geautomatiseerde leveringspijplijn, die de test- en integratieprocessen automatiseert om snelle en continue levering mogelijk te maken, gebruikmakend van bijvoorbeeld de drie DevOps-manieren (Three DevOps Ways).

Gezien vanuit IT-infrastructuur betekent dit het inbouwen van flexibiliteit en schaalbaarheid. De toepassing van Cloud methoden en -architecturen wordt haast onvermijdelijk.

13.2 Verandering faciliteren

We hebben de ADAPT- en ADKAR-modellen beschreven, die als basis dienen om verandering te vereenvoudigen.

Lean-management is ook een manier om evolutionaire organisatieveranderingen te faciliteren die inclusief zijn en minder gevoelig voor tegenwerking.

Een voordeel van Lean-management is dat het begint bij het management en wordt aangestuurd door het management in hun nieuwe rol als dienend leider.

Volgens het ‘Prosci Best Practices in Change Management Benchmarking’ rapport, is de belangrijkste reden voor managers om zich te verzetten tegen verandering, de angst om controle en autoriteit te verliezen. Als managers zich de nieuwe manier van werken eigen maken, is het veel eenvoudiger om verandering in de rest van de organisatie door te voeren. Onder één voorwaarde: managers moeten bereid zijn hun gedrag te veranderen en hun medewerkers laten zien dat het veilig is om hun eigen gedrag te veranderen.

In het rapport worden voor zowel werknemers als managers de volgende redenen als eerste genoemd:

Werknemers	Managers
Gebrek aan bewustzijn	Angst om controle en gezag te verliezen
Angst voor het onbekende	Gebrek aan tijd
Gebrek aan baanzekerheid	Tevreden met de status quo
Gebrek aan sponsoring	Wat zit erin voor mij?
Geen betrokkenheid bij het ontwerp	Geen betrokkenheid bij het ontwerp

We kunnen de grootste weerstand verwachten van degenen die het meest te verliezen hebben (vermeend verlies, niet werkelijk) en er valt een vorm van alliantie van partijen tegen het veranderingsinitiatief te verwachten.

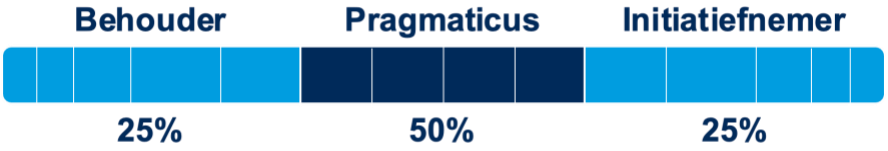
Maar er zullen mensen zijn in de organisatie die de verandering van ganser harte omarmen, meestal omdat zij ontevreden zijn met de status quo. Een zeer kleine groep zal de verandering steunen, omdat zij van verandering houden en het spannend vinden.

Chris Musselwhite en Robyn Ingram noemen dit spectrum van individuele reacties op verandering het continuüm van verandering en zij stelden de Change Style Indicator op die mensen positioneert in één van de drie hoofdcategorieën die aangeven hoe een individu verandering ervaart.

De drie categorieën in het continuüm zijn:

- behouder;
- pragmaticus;
- initiatiefnemer (originator).

Figuur 28 Schaal van drie typen gebruikers tijdens een verandering



Afbeelding gecreëerd door EXIN gebaseerd op: Underwood, J. (2013). *Musselwhite and Ingram's Change Style Indicator.* https://innovategov.org/2013/08/15/musslewhite-and-ingrams-change-style-indicator/.

- De **behouder** verzet zich tegen verandering uit angst voor het onbekende en de onzekerheid die verandering met zich meebrengt.
- De **pragmaticus** bevindt zich in het midden van het continuüm en verandert wanneer het absoluut noodzakelijk is.
- **Initiatiefnemers** voelen zich volledig op hun gemak bij het doorvoeren van verandering en hebben een 'laten we het uitproberen en zien wat er gebeurt'-mentaliteit.

Hoewel hun perceptie aanzienlijk verschilt, draagt elk van hen waardevolle inzichten en eigenschappen bij die het veranderingsinitiatief ten goede kunnen komen.

De voorzichtige aard van de behouder kan worden gebruikt in de planningsrol. Zij verkiezen incrementele (evolutionaire) verandering en zullen verandering op een manier implementeren die de organisatie verandert zonder significante verstoring van de business te veroorzaken.

De pragmaticus faciliteert, werkt samen en bemiddelt, terwijl de initiatiefnemer zorgt voor visie, energie en nieuwigheden.

De pragmaticus is teamgericht, bekijkt beide kanten en fungeert vaak als bemiddelaar om een aanvaardbare gemeenschappelijke basis te vinden. Een goede opleiding of een proefproject (pilot) kan pragmatici helpen de verandering te omarmen.

De initiatiefnemers zijn de visionairs in veranderingsinitiatieven. Het nadeel is dat de praktische aspecten van de uitvoering voor hen een bijzaak kunnen zijn, omdat zij degenen zijn die de grote lijnen uitzetten.

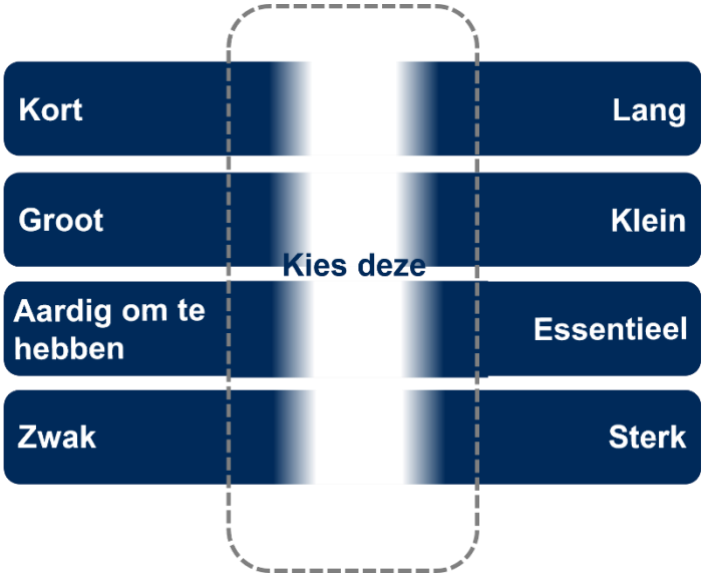
Als organisaties de bereidheid aanboren van initiatiefnemers om nieuwe ideeën toe te passen, kan een organisatie nieuwe business genereren en de efficiëntie en effectiviteit stroomlijnen, wat leidt tot winstgevender organisaties en meer waarde voor klanten.

13.3 Implementeren door middel van proefprojecten

Zodra bekend wordt dat de organisatie Agile Scrum gaat invoeren, zullen alle ogen op de projecten gericht zijn. Sommigen hopen dat ze zullen slagen, anderen zullen wensen dat ze mislukken, maar de meerderheid zal een oordeel tot later bewaren.

Het is van groot belang dat de eerste projecten slagen en daarom moet het proefproject zorgvuldig worden gekozen. Het is raadzaam om bij de keuze van het eerste proefproject het goudlokje-principe te volgen met betrekking tot vier belangrijke criteria.

Figuur 29 Het kiezen van een goed proefproject



Afbeelding gecreëerd door EXIN gebaseerd op: Cohn, M. (2009). *Four Attributes of the Ideal Pilot Project.* https://www.mountaingoatsoftware.com/blog/four-attributes-of-the-ideal-pilot-project.

Looptijd

Het kiezen van heel korte projecten zal ertoe leiden dat tegenstanders van Scrum beweren dat Scrum alleen werkt bij korte projecten.

Als er echter een project gekozen wordt dat te lang duurt, bestaat het risico dat succes pas geclaimd kan worden als het project is afgelopen.

Veel traditioneel beheerde projecten beweren op schema te liggen, maar overschrijden uiteindelijk het budget en lopen vertraging op, dus een Scrum-project dat hetzelfde beweert, is misschien niet erg overtuigend.

Het beste is een project te kiezen met een duur die ongeveer normaal is voor de organisatie. Als het project langer duurt dan zes maanden, verdeel het dan in deelprojecten met duidelijk omschreven resultaatdoelen en doe de eerste drie maanden als showcase of proefproject.

Dit geeft een team genoeg tijd om goed te worden in het werken met sprints, om er plezier aan te beleven en om de voordelen voor zowel het product als het team te zien. Een project van drie tot vier maanden is meestal ook genoeg om aan te tonen dat Scrum tot succes zal leiden in langere en grotere projecten.

Omvang

Kies indien mogelijk een project waarbij gestart kan worden met één team waarvan alle leden op dezelfde locatie zitten.

Begin altijd met één team, ook al zal het proefproject uiteindelijk tot meer teams uitgroeien en probeer een proefproject te kiezen dat niet verder zal groeien dan vijf teams, ook al zijn dergelijke projecten gebruikelijk in de organisatie.

Kies in eerste instantie geen project waarbij het werk tussen veel Scrum-teams gecoördineerd moet worden, want dat is een kansloze missie. Er is waarschijnlijk toch geen tijd om van één team naar meer dan vijf te groeien.

Belang

Het kan verleidelijk zijn om een project met weinig belang en weinig risico als proefproject te kiezen. Het risico is klein als het slecht gaat; mensen merken misschien niet eens dat het mislukt.

Critici houden echter nauwlettend in de gaten houden wat er gedaan wordt en de pogingen terzijde schuiven, omdat het project toch te eenvoudig was.

Kies in plaats daarvan een belangrijk project met een relatief laag risico. Bedenk ook dat teams sommige moeilijke aspecten goed moeten doen om over te kunnen stappen naar Scrum. Als het project niet belangrijk is, doen mensen misschien niet alles wat nodig is.

Ga de dialoog aan met de business sponsor(s)

Scrum adopteren vraagt veranderingen in het hele bedrijf, niet alleen technische resources. Het is van cruciaal belang om een business-sponsor te hebben die belangstelling heeft voor Agile en de tijd en de wil heeft om met het team samen te werken.

Een geëngageerde en betrokken business-sponsor kan het team helpen om conflicten op te lossen en om vastgeroeste processen, praktijken en regels die worden afgedwongen door afdelingen of individuen, ter discussie te stellen. Product Owners spelen, vooral tijdens proefprojecten, een cruciale rol in het faciliteren van de communicatie met de business en tussen de business en het Scrum-team. Het is niet ongewoon om in deze fase een toename in de betrokkenheid van de Product Owner te zien.

Evenzo toont goede sponsoring de betrokkenheid van het management aan en is een nuttige tool om het succes van het project bekend te maken, vooral als zij zeggen dat het beter is gegaan dan verwacht.

13.4 Verspreiden van Agile Scrum in de organisatie

Wat gebeurt er na een succesvol proefproject? Hoe kan Agile Scrum in de organisatie geschaald worden?

Wel, de organisatie wil adoptie van Agile in de rest van de organisatie. Er kan gekozen worden voor een big-bang aanpak of voor een gefaseerde implementatie.

Hoewel veel organisaties zeggen dat big-bang voor hen goed werkte, wordt een gefaseerde aanpak aanbevolen.

Soms zijn meerdere proefprojecten nuttig, nadat succes in één gebied aantoont dat het in een ander gebied zal werken. Vaak zeggen mensen: ‘het is prima voor afdeling XYZ, maar het zal niet werken in ons gebied omdat dat anders is’.

We kunnen een verticale en horizontale benadering gebruiken voor het schalen.

- **Verticaal schalen** schalen binnen een business unit
- **Horizontaal schalen** schalen dwars door de organisatie

Verticaal schalen is eenvoudiger dan horizontaal schalen. Het is veel gemakkelijker om andere teams in dezelfde business unit te leren hoe ze Agile kunnen gebruiken, omdat de omgeving vergelijkbaar is en veel van de geleerde lessen kunnen worden toegepast.

Een proefproject helpt om de waarde van de bijdragen aan en het gebruiken van Scrum door de verschillende teams te laten zien. Aangezien Scrum-teams functie overschrijdend zijn, zijn leden van verschillende business-teams waarschijnlijk bij het proefproject betrokken.

Split-and-seed

Eén of meer teamleden van het proefproject team gebruiken als leden van nieuwe Scrum-teams, betekent dat iemand met een positieve ervaring in het gebruik van Scrum beschikbaar is voor andere Scrum-teamleden, die misschien problemen hebben, twijfels of gewoon iets willen weten of leren over de nieuwe manier van werken. Deze aanpak wordt splijt en zaai (split-and-seed) genoemd.

Grow-and-split

Een tussentijdse strategie kan zijn: kweek en splijt (grow-and-split). Het team dat betrokken is bij het proefproject wordt uitgebreid. Ervaren leden helpen nieuwe leden gedurende twee of drie sprints en vervolgens wordt de nieuwe, uitgebreide groep gesplitst om nieuwe teams te zaaien. Gebruik in dit scenario zowel meer als minder ervaren leden bij het zaaien van een nieuw team.

Coachen

Coachen is ook een doeltreffend mechanisme om het Agile-programma uit te breiden en zowel interne als externe coaches kunnen worden gebruikt. Coaches brengen een paar uur per week door met hun teams, late zien hoe Scrum het beste kan worden gebruikt en delen de lessen die zijn geleerd binnen de bredere context van de organisatie. Interne coaches zijn dus geen vaste leden van het Scrum-team dat ze coachen.

Interne coaches hebben een beter begrip van de context; externe coaches kennen Scrum misschien beter en hebben vaak meer ervaring die zij kunnen inbrengen.

Het is raadzaam een combinatie van de hier beschreven technieken te gebruiken om het gebruik van Scrum door de hele organisatie te verspreiden.

13.5 Omgaan met mensen

Mensen verzetten zich om vele redenen tegen de overstap naar Scrum. Sommigen gebruiken goed onderbouwde logica en sterke argumenten; anderen verzetten zich door de verandering stilletjes te saboteren.

Mike Cohn deelt in ‘Success with Agile’ enkele mooie inzichten in dit verband; enkele van deze inzichten zullen hier worden verkend.

We zullen vaak verkeerd begrepen argumenten horen van mensen of dat ze handelen op grond van hun verkeerd vertaald begrip. Hun weerstand maakt het hun onmogelijk de juiste boodschap te horen.

Iemand kan bijvoorbeeld zeggen of denken: ‘Jij vindt niet documenteren een goed idee? Ik zal je laten zien wat niet documenteren is. ‘Het was natuurlijk nooit de bedoeling, toen de Agile-principes werden uitgelegd, om te impliceren dat documentatie niet belangrijk is. Maar mensen die zich verzetten tegen verandering kunnen zich vastklampen aan zo’n punt en overgaan tot niets opschrijven, zelfs als het team heeft afgesproken dat er iets gedaan moet worden.

Anderen verzetten zich door de verandering stilletjes te negeren, zoveel mogelijk op de oude manier te werken en te wachten tot de volgende verandering zich aandient en Scrum wegvaagt.

Dus hoe verzetten mensen zich tegen het veranderen van hun gedrag en hoe kan daarmee omgegaan worden?

Eerst moet vastgesteld worden hoe mensen zich verzetten, actief of passief.

- **Actieve weerstand** betekent dat er acties worden ondernomen om de implementatie van Agile Scrum te belemmeren of te doen ontsporen.
- **Passieve weerstand**, aan de andere kant, is duidelijk wanneer iemand nalaat actie te ondernemen of iets verwaarloost. Ze hebben het idee dat als het maar lang genoeg wordt genegeerd, het wel weg zal gaan.

Ten tweede moet vastgesteld worden waarom mensen zich verzetten tegen verandering. Er zijn meestal twee belangrijke redenen:

- **De status quo is prima**, angst voor het onbekende
- **Ze houden niet van Scrum**, een oprechte afkeer van Agile en Scrum

Ze hebben mogelijk een slechte ervaring als referentie (dat hoeft niet eens hun eigen ervaring te zijn) of de ideeën die in Agile worden gepropageerd zijn gewoon te vreemd voor hen en daarom onacceptabel.

Er zijn dus twee manieren waarop mensen zich verzetten en twee redenen waarom zij zich verzetten. Op basis daarvan kan een 2x2 matrix worden gemaakt, zoals hier.

Figuur 30 Hoe om te gaan met weerstand tegen verandering

Afbeelding gecreëerd door EXIN gebaseerd op: Cohn, M. (2010). *Succeeding with Agile*. Addison-Wesley.

Elk kwadrant krijgt een naam die de persoon beschrijft die zich verzet op de manier die wordt aangegeven in de labels op de assen.

13.5.1 Sceptici

Sceptici zijn het niet eens met de principes of praktijken van Scrum, maar verzetten zich alleen passief tegen de transitie. Sceptici argumenteren op een beleefde manier tegen Scrum en vergeten veel te vaak om de daily scrum bij te wonen. Het zijn individuen die de transitie proberen tegen te houden, in tegenstelling tot diegenen die het een beetje vreemd vinden maar bereid zijn om het te proberen.

Middelen om de weerstand van **sceptici** te overwinnen zijn:

- laat de tijd het werk doen;
- geef training;
- vraag anekdotes van collega's;
- benoem een kampioen scepticus;
- voer de druk op;
- bouw bewustzijn op.

13.5.2 Saboteurs

Net als sceptici verzetten saboteurs zich meer tegen de transitie vanuit een afkeer van Scrum; een saboteur biedt echter actieve weerstand door te proberen de overgang te ondermijnen, misschien door door te gaan met het schrijven van ellenlange ontwerpdocumenten vooraf, enzovoort.

Aan de linkerkant staan degenen die zich comfortabel voelen met de status quo. Zij zijn gewend dingen op een bepaalde manier te doen en hun werk leidt vaak tot prestige in de ogen van collega's. Ze zijn misschien niet tegen Scrum als zodanig, maar ze verzetten zich tegen elke verandering die hun huidige positie in gevaar kan brengen.

Enkele tools in de omgang met saboteurs:

- succes;
- herhaal en versterk de commitment;
- verplaats ze;
- ontsla ze (ja, goed gelezen, soms is dat de enige oplossing, omdat ze ‘de put vergiftigen!’);
- zorg ervoor dat de juiste mensen praten.

13.5.3 Diehards

Diehards houden van de status quo en verzetten zich actief tegen verandering. Zij doen moeite de transitie te verhinderen door anderen voor hun zaak te werven.

Zo kan er omgegaan worden met diehards:

- stem beloning af op resultaat;
- creëer ontevredenheid over de status quo;
- erken de angst en ga die aan.

13.5.4 Volgers

Volgers houden van de status quo en verzetten zich passief tegen verandering. Volgers zijn niet kwaad bij het vooruitzicht van verandering, maar ze zullen zo weinig mogelijk doen, in de hoop dat de verandering slechts een bevlieging is. De manier om hen te winnen is te laten zien dat Scrum de nieuwe status quo is.

Tenslotte, de middelen die gebruikt kunnen worden om met volgers om te gaan:

- wijzig de samenstelling van het team;
- beloon het juiste gedrag;
- betrek ze erbij;
- wees zelf voorbeeld van het juiste gedrag;
- identificeer de echte barrière.

13.5.5 Hoe om te gaan met elk type weerstand

Het advies van Cohn is pragmatisch en eenvoudig toe te passen. Alvorens te besluiten hoe er omgegaan wordt met degenen die zich tegen verandering verzetten, zal er rekening gehouden moeten worden met de cultuur van de organisatie, de samenleving en de wettelijke context waarin er gewerkt wordt.

13.6 De juiste omgeving creëren

Eén van de eerste uitdagingen in het overstappen op Agile Scrum is het werken als zelfsturende, crossfunctionele teams.

Gezien de waarden van Scrum, kan verondersteld worden dat Agile-teams voornamelijk op de volgende manier werken. De teamleden

- werken als één Scrum-team;
- doen één taak tegelijk;
- doen het werk in korte iteraties;
- leveren waarde in elke iteratie;
- richten zich op prioriteiten en resultaten voor de business;
- inspecteren en passen aan.

Uit bovenstaande kunnen direct enkele uitdagingen voor traditionele organisatiestructuren, faciliteiten, verantwoordelijkheden, functies of rollen worden afgeleid.

Als Scrum-teams functie overschrijdend zijn, maar nauw samenwerken en regelmatig samenwerken, is een fysieke ruimte waar ze kunnen werken het beste.

Visueel management is de sleutel. Iedereen in het team is in staat de belangrijke artefacten te zien, zoals de Scrum-, Kanban – of Nexus borden. Als zodanig zijn open ruimtes en herconfigureerbare werkstations ideaal. Dit gezegd hebbende, soms moet iemand zich concentreren of met iemand afspreken, dan werken de gemeenschappelijke ruimtes averechts.

Zorg ervoor dat hiervoor faciliteiten beschikbaar zijn, maar zorg er ook voor dat er weinig stilteruimtes zijn, zodat niet iedereen permanent in deze ruimtes kan zitten en werken. De dagelijkse activiteiten vinden plaats in de gemeenschappelijke ruimte. Dit wordt soms aangeduid als ‘caves-and-commons’ (individuele en gemeenschappelijke ruimtes).

Het is vooral belangrijk om een ruimte te hebben waar de daily scrum kan plaatsvinden. Teamartefacten moeten zichtbaar blijven, ook als de daily scrum is gedaan.

Als er software wordt gebruikt voor Scrum/Kanban-borden, zorg er dan voor dat ze permanent worden geprojecteerd op een muur in de ruimte waar de daily scrums plaatsvinden.

13.7 Werken als virtuele teams en teams op afstand

Als er één ding is dat er van 2020 te leren valt, dan is het wel dat virtuele teams blijvend zijn. Zelfs vóór 2020 waren veel Scrum-teams niet op dezelfde locatie gevestigd en vaak verspreid over de hele wereld. Zoals iedereen die deel uitmaakt van een virtueel team weet, brengt dit enkele uitdagingen met zich mee.

Maar wat is een virtueel team?

Virtuele teams zijn teams die bestaan uit mensen die hetzelfde doel nastreven, ook al bevinden zij zich niet op dezelfde fysieke locatie. De verschillende teamleden kunnen vanuit huis werken of vanuit kantoren in verschillende steden of landen. Om virtuele teams succesvol te laten zijn, is het belangrijk om de formule voor succes met virtuele teams te kennen.

Er zijn momenteel veel verschillende visies op de kwestie van virtuele teams en hoe ze zouden moeten werken, maar het goede nieuws is dat een deel van het werken als een succesvol virtueel team al is ingebouwd in het DNA van Agile Scrum. Dezelfde principes en praktijken die multidisciplinaire teams helpen om samen te werken als een samenhangend Scrum-team, helpen ook virtuele teams om samen te werken als een samenhangend Scrum-team.

Laten we dus eens kijken naar virtuele Scrum-teams en hoe ze kunnen werken.

- **Maak een duidelijk proces voor alle Scrum-activiteiten**. Dit hoeft niet gedetailleerd te zijn, maar zo dat iedereen precies weet wanneer en hoe wat gebeurt. Kijk vooral naar:
    - sprint planning;
    - daily scrum;
    - sprint review;
    - sprint retrospective;
    - product backlog verfijning;
    - ad hoc bijeenkomsten om een probleem op te lossen.
- **Gebruik gemeenschappelijke tools voor samenwerking en beheer**. Welke tool ook gebruikt wordt, zorg ervoor dat iedereen hetzelfde overzicht heeft van taken, voortgang en problemen en dat dit overzicht altijd beschikbaar is.
    - Zorg er ook voor dat de tools voor samenwerking goed werken en wees ervan bewust dat **face-to-face contact erg belangrijk is.** Zestig procent van communicatie is non-verbaal.

  Hoewel de beste vorm van communicatie is dat mensen in één ruimte zijn, kan dit vaak niet**; daarom moeten alle virtuele bijeenkomsten video gebaseerd zijn** en niet alleen audio. Met video betekent ook dat mensen niet kunnen multitasken.
    - Wanneer deelnemers verbinding maken, is dat bij voorkeur vanuit een **standaard en vooraf bepaalde plaats** die bevorderlijk is voor productieve bijdrage (niet vanuit bed of vanuit de trein bijvoorbeeld)
    - **Gebruik interactieve vragen en software waarbij mensen tijdens de vergadering kunnen stemmen** over onderwerpen, om de interactie te bevorderen.
    - Er kan ook een teamruimte worden gecreëerd voor **informele en normale dagelijkse interactie** in het team, zoals een WhatsApp-groep.
- Zorg ervoor dat **voor alle activiteiten de verwachtingen vooraf zijn vastgesteld**, zelfs als ze voor de zoveelste keer worden herhaald.
    - Wat verwacht het team van de deelnemers voor deze activiteit?
    - Wat kunnen de deelnemers van het team verwachten?
    - Wat zijn de verwachte output en uitkomsten van deze gebeurtenis?

    - Geef iedereen de kans om een verwachting naar voren te brengen die niet bij de gebeurtenis hoort.
    - Als niemand zich aan de regels van de bijeenkomst houdt, is deze afgelopen en wordt een nieuwe afspraak gemaakt. Laat de regels geen enkele keer glippen.
- **Een goede facilitator is de sleutel**; hier kunnen Scrum Masters een grote bijdrage leveren, maar vergeet niet wiens vergadering het is. Als de Scrum Master niet wordt aangewezen als facilitator is de Scrum Master niet de facilitator. Daily scrum is de bijeenkomst van de Developers, product backlog verfijning is de bijeenkomst van de Product Owner, etc.
    - Vermijd tools zoals PowerPoint; het is slechts een middel voor éénrichtingscommunicatie. Conversatie en deelname van iedereen zijn nog belangrijker bij virtuele vergaderingen!
    - Een goede tip komt van Mike Dwyer – hij noemt het de NOSTUESO-regel – **No One Speaks Twice Until Everyone Speaks Once** Niemand Spreekt Twee keer Totdat Iedereen Eénmaal heeft Gesproken.
    - Dit schept ook ruimte voor mensen om hun zegje te doen. Uit onderzoek is gebleken dat als een deelnemer in de eerste vijf minuten spreekt, de kans drie keer zo groot is dat zij of hij nog een keer het woord neemt.
    - Ook is gebleken dat wanneer een vrouw als eerste spreekt, de deelname van andere vrouwen veel groter is.
    - Zorg voor de totstandkoming van goede relaties en positieve maar openhartige communicatie in het Scrum-team.

14 Eindverantwoordelijkheden en Verantwoordelijkheden – Scrum & Nexus gebeurtenissen en praktijken

Door het boek heen wordt verwezen naar gebeurtenissen of praktijken, samen met een verwijzing naar wie deze doet (bijvoorbeeld de Scrum Master, Developer, enz.). Maar nergens is een definitieve lijst van waar elke ze (eind)verantwoordelijk voor zijn.

Maar het is natuurlijk belangrijk om te begrijpen wat er verwacht wordt van elke verantwoordelijkheid in Scrum. Dit hoofdstuk bestaat uit drie delen, één voor elke verantwoordelijkheid. In elk van deze secties worden de verantwoordelijkheden uiteengezet en gematched met Scrum gebeurtenissen en – praktijk.

14.1 De Product Owner

14.1.1 Sprint planning

	(Eind)Verantwoordelijkheden
Deelnemers	Scrum-team, Developers zijn eigenaar
Voorbereiding van sprint planning	Ervoor zorgen dat de product backlog wordt verfijnd en in lijn is met wat de business nu nodig heeft.
Een sprint goal bepalen	De product goal nogmaals onder de aandacht brengen en ervoor zorgen dat iedereen weet wat de business nu nodig heeft. Akkoord gaan met de sprint goal.
De sprint backlog vaststellen	Geef input waar nodig en zorg ervoor dat wat de business nodig heeft, is weerspiegeld in backlog items. Als er discrepanties zijn, licht deze eruit en onderhandel met de Developers over wijzigingen.
Het doen van schattingen en het uitsplitsen van taken	Aannames valideren en meer duidelijkheid verschaffen over vereisten en wat de business nodig heeft.
Het creëren van de sprint backlog	Geef input indien geraadpleegd.
Product backlog verfijnen	Verzamel input van teamleden, werk de product backlog bij en herschik items op basis van nieuwe inzichten.

14.1.2 Daily scrum

	(Eind)Verantwoordelijkheden
Deelnemers	Developers, Scrum Master en soms andere belanghebbenden inclusief de Product Owner.
Inspecteren en aanpassen	Geef input indien nodig en stel onderzoekende vragen.
Product backlog verfijnen	Verzamel input van teamleden, update de product backlog en herorden items op basis van nieuwe inzichten. Houd rekening met capaciteiten, beperkingen, tijd en budget.

14.1.3 Sprint review

	(Eind)Verantwoordelijkheden
Deelnemers	Het Scrum-team en essentiële belanghebbenden zoals vastgesteld door de Product Owner. Soms voorgezeten door de Product Owner. Het is echter de vergadering van het team.
Kick-off	Zorg ervoor dat de aanwezigen begrijpen waarom ze daar zijn. Omdat er belanghebbenden van buiten het Scrum-team deelnemen, heeft de Product Owner de leiding.
Demonstreren en inspecteren	Help bij het demonstreren van het opgeleverde increment.
Feedback krijgen op wat je doet	Stel onderzoekende vragen.
Feedback krijgen over veranderingen in de omgeving	Noteer veranderingen in de omgeving.
Problemen	Let in het bijzonder op afhankelijkheden die van invloed kunnen zijn op de volgorde van levering.
Product backlog verfijnen	Verzamel input van teamleden, update de product backlog en herorden items op basis van nieuwe inzichten. Houd rekening met capaciteiten, beperkingen, tijd en budget.

14.1.4 Creëren en onderhouden van de product backlog

	(Eind)Verantwoordelijkheden
Verzamelen van vereisten	Accepteer verzamelde vereisten. Soms verzamelen Product Owners zelf vereisten; dit kan echter ook worden gedaan door klanten en gebruikers, analisten, of andere teamleden. Product Owners screenen wel vereisten en gaan vaak terug om eisen te verfijnen voordat ze worden toegevoegd aan de product backlog.
Eerste verfijning van de backlog	Overweeg initiële vereisten, verzamel input van teamleden, vul de product backlog aan op basis van business vereisten en andere inzichten. Houd rekening met mogelijkheden, vereisten, tijd en budget.
Voortdurende verfijning van backlog	Verzamel input van teamleden, update de product backlog en herorden items op basis van nieuwe inzichten en veranderende business vereisten. Houd rekening met mogelijkheden, vereisten, tijd en budget.

14.1.5 Sprint retrospective

	(Eind)Verantwoordelijkheden
Sprint retrospective	
Aanwezigen	Scrum-team; deze gebeurtenis wordt geleid door Developers met input van andere verantwoordelijkheden of rollen.
Inspecteren	Bijdragen maar niet leiden.
Aanpassen	Bijdragen maar niet leiden.
Lessen delen	Hoewel het leren in de eerste plaats teamgericht is, kan de Product Owner ervoor kiezen om lessen te delen met andere Scrum-teams die met dezelfde backlog werken.

14.1.6 Nexus sprint planning

	(Eind)Verantwoordelijkheden
Aanwezigen	Leden van deelnemende Nexus teams en een Product Owner.
Voorbereiding sprint planning	Een Nexus wordt nooit over meerdere product backlogs uitgevoerd, de Product Owner speelt een meer dominante maar nog steeds begeleidende rol in het helpen van betrokken teams bij het plannen van de Nexus. In werkelijkheid is de Product Owner de initiator van de Nexus en speelt dus een belangrijke rol in het samenstellen van de Nexus en het betrekken van de teams bij de Nexus. Hun rol is verder vergelijkbaar met die in de normale sprint planning.
Kick-off	De Product Owner is de Nexus owner en voorzitter; het plannen en uitvoeren van kick-off meetings is dan ook de verantwoordelijkheid van de Product Owner.

14.1.7 Nexus sprint review

	(Eind)Verantwoordelijkheden
Aanwezigen	Leden van deelnemende Nexus teams en een Product Owner.
Uitvoering	Hoewel de inhoud en het format van de Nexus sprint reviews vergelijkbaar zijn met sprint reviews, vragen ze coördinatie en planning. Er zijn geen aparte sprint-reviews, alleen de Nexus sprint review in een Nexus. Eigenaar en voorzitter van deze bijeenkomst is de Product Owner.

14.1.8 Nexus sprint retrospective

Hetzelfde als in sprint retrospectives

	(Eind)Verantwoordelijkheden
Aanwezigen	Scrum-team; deze gebeurtenis wordt geleid door Developers met input van andere verantwoordelijkheden of rollen.
Inspecteren	Bijdragen maar niet leiden.
Aanpassen	Bijdragen maar niet leiden.
Lessen delen	Hoewel het leren in de eerste plaats teamgericht is, kan de Product Owner ervoor kiezen om lessen te delen met andere Scrum-teams die met dezelfde backlog werken.

14.1.9 Nexus product backlog verfijning

Nexus product backlog verfijning is niet alleen een voortdurende activiteit, maar ook gedefinieerd als een Nexus gebeurtenis. De gebeurtenis wordt geleid en gecoördineerd door de Product Owner.

14.2 De Scrum Master

Belangrijk: Scrum Masters faciliteren Scrum-teamleden en sturen nooit een gebeurtenis of actie aan. Zij faciliteren, coachen en assisteren.

14.2.1 Sprint planning

	(Eind)Verantwoordelijkheden
Aanwezigen	Scrum-team, maar de Developers zijn eigenaar.
Voorbereiding sprint planning	Helpen met het coördineren en opzetten van de bijeenkomst.
Een sprint goal bepalen	Acteren als facilitator, bemiddelaar en helpen bij het oplossen van conflicten.
Het bepalen van de sprint backlog	Acteren als facilitator, bemiddelaar en helpen bij het oplossen van conflicten.
Het doen van schattingen en taakverdeling	Acteren als facilitator, bemiddelaar en helpen bij het oplossen van conflicten.
Het creëren van de sprint backlog	Hoewel de sprint backlog toebehoort aan de Developers, vragen zij vaak de Scrum Master om te helpen bij het opstellen, documenteren en bijhouden van de sprint backlog.
Product backlog verfijnen	Acteren als facilitator, bemiddelaar en helpen bij het oplossen van conflicten.

14.2.2 De daily scrum

	(Eind)Verantwoordelijkheden
Aanwezigen	Developers, Scrum Master, soms andere belanghebbenden, inclusief de Product Owner.
Inspecteren en aanpassen	Faciliteer het gesprek, geef coaching indien nodig, bemiddel indien nodig.
Omgaan met belemmeringen	Een belangrijke functie van een Scrum Master is het helpen van Developers met belemmeringen om te gaan; gewoonlijk houdt dit in: het faciliteren van bijeenkomsten, het zoeken van ondersteuning, middelen of vaardigheden en frequente communicatie en follow-up met belanghebbenden.
Vooruitgang volgen	Het actueel houden van information radiators wordt meestal gedaan door de Scrum Master. Het is niet hun verantwoordelijkheid, maar de taak wordt vaker wel dan niet door de Developers aan de Scrum Master uitbesteed.

Coachen van het team	Een belangrijke functie van een Scrum Master is het coachen en instrueren van het team zodat de teamleden vertrouwd zijn met Scrum praktijken en op de hoogte van de nieuwste ontwikkelingen in Scrum.
Product backlog verfijnen	Faciliteer het gesprek, geef coaching indien nodig, bemiddel indien nodig.

14.2.3 Sprint review & retrospectives

	(Eind)Verantwoordelijkheden
Faciliteren	Faciliteer het gesprek, geef coaching indien nodig, bemiddel indien nodig.

14.2.4 Creëren en onderhouden van de product backlog

	(Eind)Verantwoordelijkheden
Verzamelen van vereisten	In deze activiteit biedt de Scrum Master ondersteuning aan de Product Owner.
Eerste verfijning van de backlog	Faciliteer het gesprek, geef coaching indien nodig, bemiddel indien nodig.
Voortdurende verfijning van backlog	Faciliteer het gesprek, geef coaching indien nodig, bemiddel indien nodig.

14.2.5 Nexus

Niet alle Scrum Masters van de deelnemende Scrum-teams zullen deelnemen aan Nexus specifieke gebeurtenissen zoals Nexus planning, de Nexus review en retrospective. Echter, tenminste één Scrum Master is Scrum Master in het Nexus Integration Team. In deze rol wordt veel tijd besteed aan het coördineren en verfijnen van activiteiten tussen teams. Hoewel dit geen fulltimebaan is, is het in een complexe Nexus een permanente en gerichte rol. Het werk van de Nexus heeft altijd voorrang op het Scrum-teamwerk.

14.3 Developers

14.3.1 Sprint planning

	(Eind)Verantwoordelijkheden
Aanwezigen	Het Scrum-team met de Developers als eigenaar.
Voorbereiding sprint planning	Sprint planning is de taak van Developers; Developers delegeren het voorbereiden van de bijeenkomst vaak aan de Scrum Master.
Een sprint goal bepalen	De sprint goal wordt bepaald door de Developers in overleg met de Product Owner.
Het voorbereiden van de sprint backlog	De sprint backlog items worden geselecteerd door Developers, met input van de Product Owner en soms met hulp van de Scrum Master.
Het doen van schattingen en taakverdeling	Developers doen de schatting en taakverdeling met hulp en input van de Product Owner en hulp van de Scrum Master.
Het creëren van de sprint backlog	Developers maken de sprint backlog; ze vragen vaak de Scrum Master om de beheerder te zijn van de sprint backlog, wat het eenvoudiger maakt, omdat dan slechts één persoon belast is met het bijhouden van de sprint backlog.
Product backlog verfijnen	Developers helpen de Product Owner met het verfijnen van de product backlog. Ook al is dit een doorlopende activiteit; vaak wordt in de sprint planning als laatste een snelle verfijning gedaan op basis van wat tijdens de planning werd geleerd.

14.3.2 De daily scrum

	(Eind)Verantwoordelijkheden
Aanwezigen	Developers en Scrum Master – soms andere belanghebbenden, waaronder de Product Owner.
Inspecteren	Developers rapporteren over de voortgang, ontdekken afhankelijkheden en belemmeringen.
Aanpassen	Developers beslissen hoe om te gaan met belemmeringen en afhankelijkheden. Als externe hulp nodig is, kunnen ze de Scrum Master en Product Owner vragen om een positieve uitkomst te faciliteren.
Voortgang volgen	Het bijhouden van de voortgang is de verantwoordelijkheid van de Developers; dit wordt vaak gedelegeerd aan de Scrum Master.
Product backlog verfijnen	Alle nieuwe informatie die een verfijning van de
	gedaan als Developers problemen aan het licht brengen. De Product Owner is eigenaar van de verfijning.

14.3.3 Sprint review

	(Eind)Verantwoordelijkheden
Aanwezigen	Scrum-team en belangrijke belanghebbenden, geïdentificeerd door de Product Owner. Soms voorgezeten door de Product Owner maar het team blijft eigenaar.
Kick-off	Omdat deze gebeurtenis betrekking heeft op belanghebbenden buiten het Scrum-team, neemt de Product Owner de leiding bij de sprint review.
Demonstreren en inspecteren	De Developers laten de andere belanghebbenden zien wat ze hebben gedaan en beantwoorden vragen.
Feedback over wat je doet en vanuit de omgeving	Let wel: de sprint review is geen moment van aftekenen maar een gelegenheid om feedback te krijgen van belanghebbenden, met name klanten en gebruikers. Belanghebbenden laten vaak nieuwe vereisten weten of informeren over veranderingen in de omgeving waar het Scrum-team niet van op de hoogte was! Feedback sessies kunnen ook problemen aan het licht brengen; deze praktijk wordt echter ontmoedigd, omdat het snel uit de hand kan lopen.
Product backlog verfijnen	Met nieuwe informatie, inzichten en feedback is het een goed moment om de product backlog te verfijnen, terwijl deze informatie nog vers in het geheugen van het team zit.

14.3.4 Creëren en onderhouden van de product backlog

	(Eind)Verantwoordelijkheden
Aanwezigen	Product Owner met hulp van de Developers.
Verzamelen van eisen en product backlog verfijning	Developers zijn vaak niet betrokken bij de initiële verzameling van vereisten, maar komen wel nieuwe vereisten op het spoor, als gevolg van hun interacties met gebruikers tijdens een sprint of bij het plannen en uitvoeren van een sprint. Vereisten worden aan de Product Owner gegeven, normaal gesproken als onderdeel van product backlog verfijning.

14.3.5 Sprint retrospective

	(Eind)Verantwoordelijkheden
Aanwezigen	Het Scrum-team.
Inspectie	Er zijn verschillende technieken beschikbaar; één daarvan is vragen wat er goed ging, wat er fout ging en wat verbeterd kan worden. Hier kan analyse van issues en blokker tickets die tijdens de sprint werden verzameld, nuttig zijn.
Aanpassing	Zo kan het team zich ook verbeteren door zich af te vragen wat ze moeten veranderen, verbeteren en vermijden – en dit om te zetten in uitvoerbare plannen.
Lessen delen	Vooral bij geschaalde implementaties kan het nuttig zijn ervaringen met andere Developers te delen.

14.3.6 Nexus sprint planning

	(Eind)Verantwoordelijkheden
Aanwezigen	Leden van deelnemende Nexus teams en een Product Owner
Coördinatie	Nexus sprint planning wordt gedaan door vertegenwoordigers van alle deelnemende Scrum-teams, genaamd het Nexus Integration Team. Vaak zijn dit ervaren Developers die vanuit de deelnemende Scrum-teams zijn gedetacheerd.

14.3.7 Nexus sprint review

Tijdens een Nexus wordt een gecombineerde sprint review gedaan waarbij het werk van de gecombineerde Nexus wordt geëvalueerd, inclusief alle incrementen die deel uitmaakten van de Nexus. Vertegenwoordigers van alle Scrum-teams in de Nexus laten hun werk zien en krijgen feedback. De Nexus sprint review werkt verder hetzelfde als een sprint review.

	(Eind)Verantwoordelijkheden
Aanwezigen	Vertegenwoordigers van Scrum-teams, de Product Owner en geselecteerde belanghebbenden, inclusief klanten en key users (hoofdgebruikers).

14.3.8 Nexus sprint retrospective

Alle Scrum-teams die betrokken zijn bij een Nexus voeren eerst een normale Scrum retrospective uit. Vervolgens herhalen alle leden van de teams deze oefening voor de Nexus als geheel. Dit is waar het uitwisselen van lessen kan leiden tot significante winst, als het geleerde wordt toegepast in alle teams.

	(Eind)Verantwoordelijkheden
Aanwezigen	Alle leden van de samenstellende Scrum-teams.

14.3.9 Nexus product backlog verfijning

Gedeelde backlog verfijning is gericht op het voldoende ontleden van product backlog items (PBI's) zodat de teams begrijpen welk werk ze kunnen opleveren en de volgorde van oplevering van het werk (in komende sprints). Deze gebeurtenis stelt de teams ook in staat zich te richten op het minimaliseren en verwijderen van afhankelijkheden tussen teams.

	(Eind)Verantwoordelijkheden
Aanwezigen	Deze activiteit wordt ten minste bijgewoond door leden van het Nexus Integration Team: een bredere vertegenwoordiging van de samenstellende Scrum-teams wordt echter aangeraden.

Bijlage A – Andere Agile-methoden

Dit is een compilatie van inhoud afkomstig uit verschillende openbare bronnen die zijn aangegeven. Het enige doel van dit document is voorbereiding op de EXIN Agile Scrum Product Owner en EXIN Agile Scrum Master-certificeringen.

Sinds de publicatie van het Agile Manifest in 2001, hebben veel organisaties verschillende Agile-methodologieën aangenomen om hun teams in staat te stellen hun doelen en deliverables te bereiken. De meest bekende methodologie is Scrum, die veel gebruikt wordt in allerlei soorten bedrijven. In de begindagen werden deze Agile-praktijken alleen toegepast door IT-afdelingen. Daarentegen worden Agile-methoden nu op grote schaal gebruikt in een breed scala van verschillende niet-IT business gebieden.

Dit artikel geeft een overzicht en korte uitleg van de meest gebruikte Agile-methodologieën naast Scrum:

- Crystal;
- Extreem Programmeren (XP);
- DSDM (nu ABC genoemd);
- LeSS;
- SAFe;
- Kanban.

Het is belangrijk in gedachten te houden dat, hoewel deze methoden verschillen, ze drie dingen gemeen hebben: open communicatie, saamhorigheid en flexibiliteit. Deze kenmerken worden gedeeld door al deze methodes, omdat ze de kern vormen van het Agile Manifest.

Crystal methodologieën[15]

De Crystal familie is een groep van lichtgewicht Agile-methodologieën. Ze worden als lichtgewicht beschouwd, omdat deze methodologieën het proces van ondergeschikt belang achten en in plaats daarvan de nadruk leggen op andere elementen.

Deze elementen zijn:

- mensen;
- interactie;
- gemeenschap;
- vaardigheden;
- talenten;
- communicatie.

Crystal methodologieën werden ontwikkeld door Alistair Cockburn in de jaren 1990. Ze zijn ontstaan als gevolg van onderzoek van Cockburn waaruit bleek dat Developers formele methodologieën niet gebruikten zoals ze bedoeld waren, maar toch succesvolle projecten opleverden. Cockburn maakt onderscheid tussen enkele belangrijke aspecten:

- **Methodologie**: een geheel van elementen (b.v. praktijken, tools).
- **Technieken**: de vaardigheidsgebieden (b.v. ontwikkelen van use cases).
- **Beleidslijnen**: definities van hoe de organisatie zich hoort te gedragen.

Als resultaat van zijn onderzoek, definieerde Cockburn het gedrag van mensen in teams als volgt:

- Mensen zijn communicerende wezens, die het beste functioneren in persoonlijke aanwezigheid, met directe vraag en antwoord.
- Mensen hebben moeite met consequent handelen in de loop van de tijd.
- Mensen zijn zeer veranderlijk, variërend van dag tot dag en van plaats tot plaats.
- Mensen willen over het algemeen goede burgers zijn, kunnen goed om zich heen kijken, nemen initiatief en doen 'wat nodig is' om het project te laten slagen.

Crystal methodologieën zijn onderverdeeld in 8 verschillende kleuren. De kleuren worden gebruikt om het 'gewicht' van de methodologie aan te geven. De lijst begint met Crystal Clear dat geschikt is voor projecten met maximaal 6 personen en loopt door tot Crystal Diamond of Crystal Sapphire dat geschikt is voor missiekritische projecten met een potentieel risico voor mensenlevens.

[15] Vrij naar: https://en.wikiversity.org/wiki/Crystal_Methods

Een belangrijk punt dat ook aandacht verdient, is dat er zeven gemeenschappelijke eigenschappen zijn die door de hele Crystal familie worden gedeeld. Deze eigenschappen zijn:

- frequente levering;
- reflectieve verbetering;
- nabije of osmotische communicatie;
- persoonlijke veiligheid;
- focus;
- toegang tot deskundige gebruikers.

Frequente levering

Regelmatig uitbrengen van iteraties van de software/het product dat in ontwikkeling is.

Reflectieve verbetering

Het projectteam wordt gevraagd van tijd tot tijd na te denken over hoe de processen kunnen worden verbeterd. Dit zorgt niet alleen voor afwisseling, maar helpt ook de efficiëntie te vergroten.

Nabije of osmotische communicatie

Communicatie moet door het projectteam stromen. Het projectteam is samen in dezelfde ruimte, zodat communicatie die de voortgang van het project ondersteunt, wordt aangemoedigd.

Persoonlijke veiligheid

Iedereen in het team moet het gevoel hebben dat zij zich in een open en veilige omgeving bevinden, vrij om een mening te geven. Negatieve reacties, zoals uitgelachen worden bij het stellen van een vraag of opperen van een idee, worden volledig ontmoedigd. Mensen moeten elkaar immers kunnen vertrouwen en wie het mikpunt is van negativiteit, zal waarschijnlijk niet meer proactief deelnemen.

Focus

Focus is cruciaal voor succes. Wanneer in de Crystal methodologieën naar focus wordt verwezen, verwijst dit naar twee dingen: voortgang en richting. Voortgang betekent voldoende tijd besteden aan een taak in een project zodat vooruitgang wordt geboekt. Richting verwijst naar de richting die het project opgaat.

Toegang tot deskundige gebruikers

Een echte gebruiker van de nieuwe functionaliteit die wordt ontwikkeld, moet bereikbaar zijn voor het projectteam om vragen te beantwoorden en te helpen met oplossingen voor problemen. Deze deskundigen kunnen het project helpen met hun praktijkervaring en idealiter zijn ze vaak beschikbaar, ook tijdens bijeenkomsten of via telefoongesprekken.

Er moet een technische omgeving beschikbaar zijn met geautomatiseerde tests, configuratiebeheer en frequente integratie.

Frequente integratie en testen zijn essentieel zodat problemen (fouten, bugs, enz.) vroegtijdig kunnen worden opgespoord. Is integratie continu dan wordt voorkomen dat problemen groter worden, omdat ze in een vroeg stadium worden opgelost.

Extreme Programming (XP)[16]

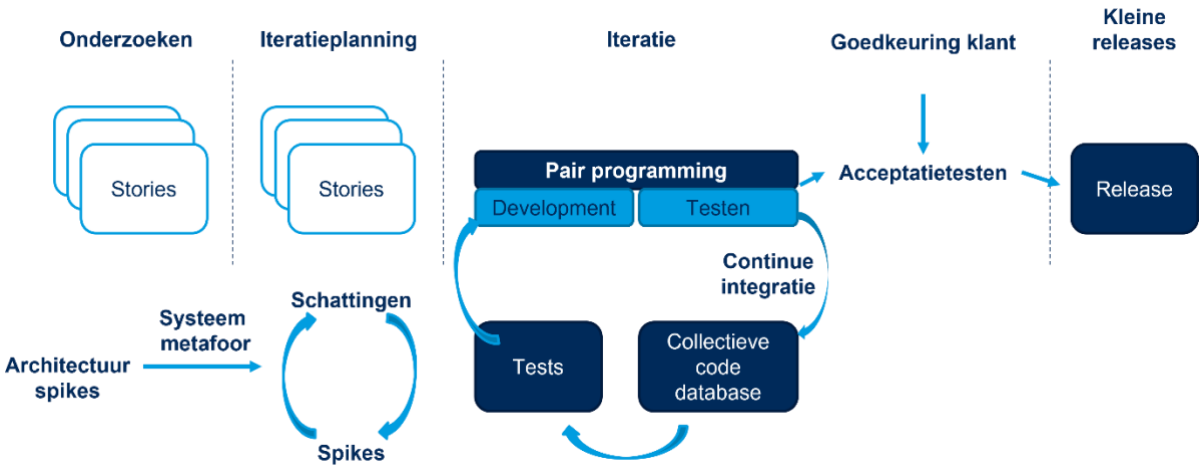
XP is een softwareontwikkelingsmethodologie die de nadruk legt op klanttevredenheid. Dit garandeert dat wat geproduceerd wordt, gebaseerd is op de behoeften van de klant. Eén van deze manieren is door user story's om te zetten in waardevolle bedrijfsmiddelen.

Een bepalend kenmerk van XP is voortdurende en open communicatie. Zonder dit belangrijke aspect zou de praktijk niet in lijn zijn met de verwachtingen. Een belangrijk voordeel van deze stijl van communiceren is een toename in vertrouwen van zowel de klant als het projectteam.

XP werd gecreëerd als antwoord op een behoefte in de markt om tegemoet te komen aan de voortdurende veranderingen waarmee dienstverleners werden geconfronteerd bij de ontwikkeling van een nieuw product of een nieuwe dienst. Het succes ervan kan worden toegeschreven aan de belangrijke communicatie- en samenwerkingsaspecten tussen alle partijen die bij het project betrokken zijn. Dit betekent dat klanten, managers, projectteams en alle andere betrokkenen samenwerken vanaf het moment dat de scope is bepaald en de story's zijn gedefinieerd, tot en met het testen en de uiteindelijke release.

De methodologie werd ontworpen voor kleine projectteams, hoewel soms grote teams XP gebruiken en ook succes hebben.

Figuur 31 Extreme Programming (XP) overzicht



Afbeelding gecreëerd door EXIN gebaseerd op: Meier, J. D. (2019). *Extreme Programming at a Glance.* https://jdmeier.com/extreme-programming-at-a-glance/

Hoe werkt Extreme Programming (XP)?

De eerste stap in XP is het verzamelen van user story's en het uitvoeren van spike oplossingen voor de user story's die een risico kunnen vormen.

[16] Vrij naar: http://www.extremeprogramming.org/

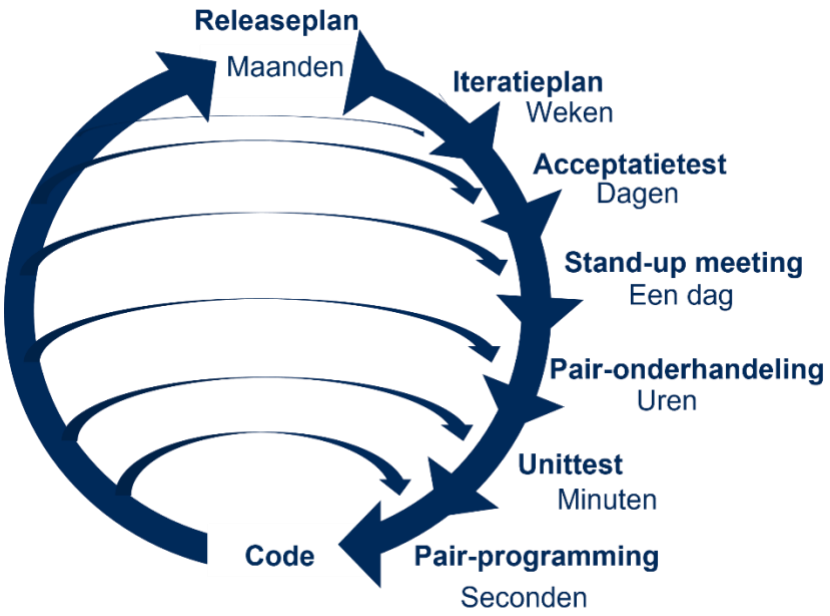
Na de eerste stappen wordt een releaseplanning bijeenkomst gepland. In dit stadium is het belangrijk om alle benodigde mensen uit te nodigen. Dit kunnen klanten, projectteams en managers zijn. Het belangrijke doel in dit stadium is een gemeenschappelijk doel te bepalen dat voor iedereen aanvaardbaar is. Deze bijeenkomst en het vaststellen van het doel worden gevolgd door de iteratieve planningsbijeenkomst om overeenstemming te bereiken over de volgende releases.

Met deze paar stappen gaat het projectteam al in de richting van het ontwikkelen van de gevraagde functionaliteit.

De regels van Extreme Programming

XP is afhankelijk van frequente feedbacklussen zoals geïllustreerd in onderstaande afbeelding.

Figuur 32 Planning loops in XP



Afbeelding gecreëerd door EXIN gebaseerd op: Wells, D. (2000). *Introducing Extreme Programming.* http://www.extremeprogramming.org/introduction.html

Hoewel sommige van deze feedbacklussen nu erg lang lijken, waren ze eind jaren negentig behoorlijk revolutionair.

De kern van XP is het idee van regels. De regels van XP in het kort:

- planning;
- beheer;
- ontwerp;
- coderen;
- testen.

Planning

- User story’s worden geschreven.
- Releaseplanning genereert het release schema.
- Doe regelmatig kleine releases.
- Het project is opgedeeld in iteraties.
- Sprint planning begint bij elke iteratie.

Beheer

- Geef het team een eigen open werkruimte.
- Bepaal een duurzaam tempo.
- Elke dag begint met een stand-up bijeenkomst.
- De projectsnelheid wordt gemeten.
- Verplaats mensen.
- Repareer XP als het kapot gaat.

Ontwerp

- Eenvoud.
- Kies een systeemmetafoor.
- Gebruik CRC (class responsibility collaborator) kaarten voor ontwerpsessies.
- Creëer spike-oplossingen om risico's te verminderen.
- Functionaliteit wordt niet vroegtijdig toegevoegd.
- Herstructureer (refactor) waar en wanneer mogelijk.

Coderen

- De klant is altijd beschikbaar.
- Code wordt geschreven volgens overeengekomen standaarden.
- Codeer eerst de unittest.
- Alle productiecode is pair geprogrammeerd.
- Eén paar per keer integreert code.
- Integreer vaak.
- Zet een speciale integratiecomputer op.
- Gezamenlijk eigendom.

Testen

- Alle codes hebben unittests.
- Alle codes slagen voor alle unittests voordat ze worden vrijgegeven.
- Als een bug wordt gevonden, worden er tests gemaakt.
- Acceptatietests worden vaak uitgevoerd en de score wordt gepubliceerd.

Pair-programming

Pair-programming is een Agile-techniek afkomstig van Extreme Programming (XP) waarbij twee Developers samenwerken en aan één computer werken. De twee mensen werken samen aan het ontwerpen, coderen en testen van user story's. Idealiter zijn de twee mensen even bekwaam en hebben ze evenveel tijd aan het toetsenbord, maar dit kan ook een effectieve manier zijn om vaardigheden over te dragen van een senior naar een junior programmeur.

DSDM[17]

Dynamic Systems Development Method (DSDM) is een framework dat zich richt op de volledige projectcyclus. Hoewel velen nog steeds naar DSDM verwijzen, is de nieuwe naam van de organisatie het Agile Business Consortium.

Deze aanpak schrijft voor dat in elke stap slechts het minimale werk wordt gedaan, om verder te kunnen naar het volgende item. De opvatting hierachter is dat continue verandering een natuurlijk kenmerk is van projecten. Het is een welbekend feit dat business-vereisten in een oogwenk kunnen veranderen. Door alleen het noodzakelijke werk te doen om de vereiste stap als voltooid te beschouwen, bespaart het projectteam moeite, middelen en tijd.

Het DSDM-framework kan worden gebruikt voor verschillende toepassingen in organisaties, van de ontwikkeling van een nieuw product of nieuwe dienst tot financiële afdelingen. Elk onderdeel van de business kan er voordeel uit halen.

Principes van DSDM

1. Actieve deelname van gebruikers is essentieel.
2. DSDM-teams zijn bevoegd om beslissingen te nemen.
3. Vaak en regelmatig opleveren van producten is een prioriteit.
4. Essentieel acceptatiecriterium voor een oplevering is de geschiktheid voor business-doeleinden.
5. Iteratieve en incrementele ontwikkeling is noodzakelijk om tot een accurate business-oplossing te komen.
6. Alle veranderingen tijdens ontwikkeling, kunnen worden teruggedraaid.
7. Initiële eisen zijn algemeen opgesteld.
8. Testen is geïntegreerd in het hele traject.
9. Samenwerking tussen alle projectparticipanten is essentieel.

Bron: Agile Business Consortium

De fasen van het DSDM-framework

Het DSDM-framework bestaat uit 6 fasen. Deze bestaan uit een pre-project en post-projectfase en de projectfase kent vier hoofdfasen: Haalbaarheid, Fundamenten, Evolutionaire Ontwikkeling en Implementatie.

Fase 1 – Pre-project

Onderdeel van de voorbereiding van het project is het bepalen van een duidelijke doelstelling. De pre-projectfase helpt ervoor te zorgen dat projecten correct worden opgezet en dat alleen de juiste projecten worden gestart.

Fase 2 – Haalbaarheid

Zoals de naam van deze fase al doet vermoeden, gaat dit vooral om de haalbaarheid van het project in kwestie, zowel uit technisch oogpunt als uit het

[17] Vrij naar: https://www.agilebusiness.org

oogpunt van kosteneffectiviteit. Deze fase mag niet te veel tijd in beslag nemen – genoeg om te besluiten of het de moeite waard is het project te bestuderen of dat het waarschijnlijk niet levensvatbaar is.

Fase 3 – Fundament

In deze fase is het de bedoeling inzicht te krijgen in de reikwijdte van het werk. Dit omvat hoe, wie, wanneer en waar van het uitvoeren van het eigenlijke werk. De levenscyclus van het project wordt ook bepaald door te kijken naar hoe het DSDM-proces zal worden toegepast.

Fase 4 – Evolutionaire ontwikkeling

Het Solution Development Team past praktijken toe zoals iteratieve ontwikkeling, MoSCoW, of timeboxing om de oplossing te laten ontstaan. Het doel is dat de oplossing verder evolueert zodat deze accuraat is, technisch gezien op de juiste manier is ingebouwd en voldoet aan wat de business nodig heeft. Het team zal iteratief blijven ontwikkelen en testen naarmate het project vordert.

Fase 5 – Uitrol

Deze fase richt zich op drie hoofdactiviteiten: Samenstellen, Review en Uitrol. Deze fase heeft tot doel de volledige oplossing, of een subset ervan, vrij te geven. Het project wordt formeel afgesloten na de laatste release.

Fase 6 – Post-Project

In de post-projectfase wordt nagegaan, in de vorm van een Benefits Assessment, in welke mate aan de verwachtingen van de business is voldaan.

LeSS

LeSS staat voor Large-Scale Scrum. Dit is een methode die kan worden toegepast wanneer meerdere teams samenwerken aan één product of dienst die wordt ontwikkeld.

LeSS kan worden toegepast in gevallen waarin:

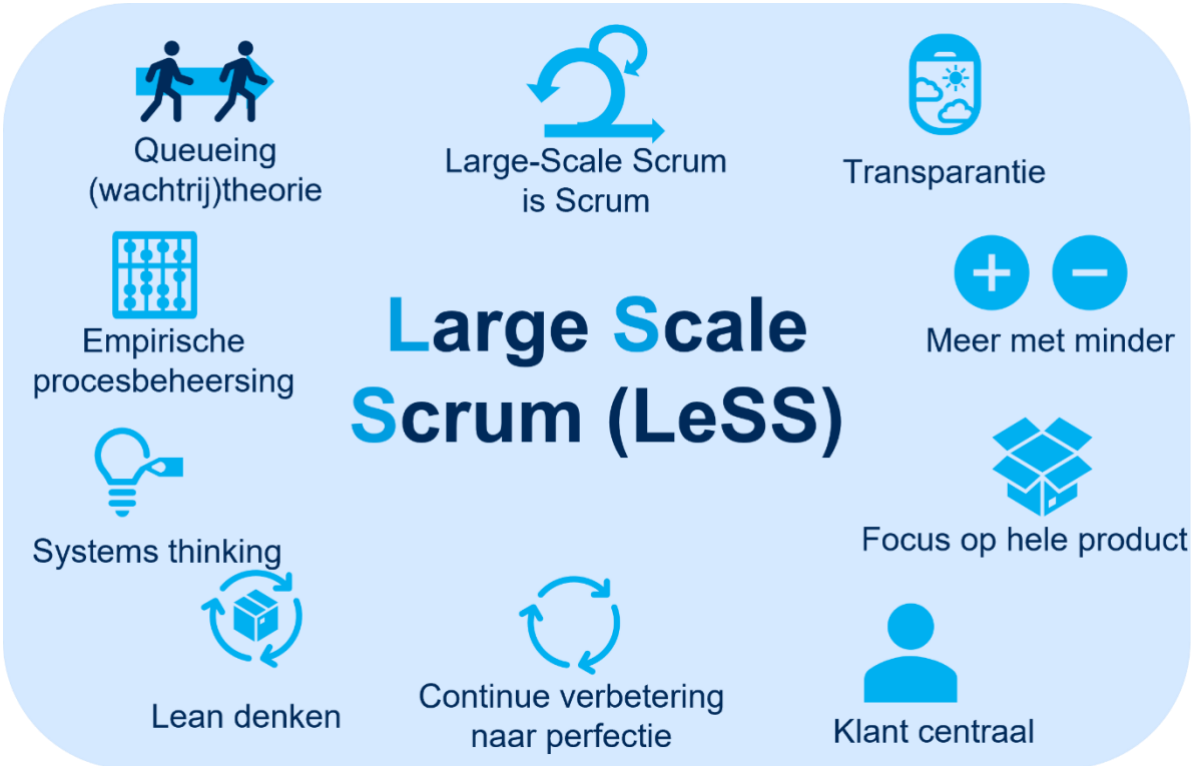
- Meerdere teams nodig zijn voor een specifieke ontwikkeling
- Meerdere teams samen werken aan een gemeenschappelijk doel
- Meerdere teams tegelijkertijd werken aan een product of dienst

Voldoet een project niet aan deze kenmerken, dan kan beter standaard Scrum worden gebruikt.

De 10 Principes van LeSS

De 10 Principes van LeSS zijn samengevat in het onderstaande plaatje.

Figuur 33 Concepten in LeSS



Afbeelding gecreëerd door EXIN gebaseerd op: The LeSS Company B.V. (2014). *Large Scale Scrum is Scrum [Visual].* https://less.works/img/principles/principles.pdf

Geschaalde Scrum is Scrum

LeSS is nog steeds een vorm van Scrum; het is echter geschaald voor de ontwikkeling van grote producten of diensten. LeSS biedt een set van regels en leiding bij het toepassen van Scrum in een multi-team context. Het is geen nieuw Scrum-framework.

Transparantie

De belangrijkste resultaten zijn zichtbaar voor alle partijen. Korte feedback-lussen verhogen transparantie. Transparantie is essentieel om adaptief beheer en verbetering mogelijk te maken.

Meer met LeSS

Dit principe staat centraal in LeSS en erkent dat complexe organisatorische oplossingen nadelig zijn en meer problemen kunnen veroorzaken dan ze oplossen. Dit principe beoogt complexiteit weg te nemen door eenvoudige en verschillende oplossingen voor problemen te kiezen.

Focus op het hele product

In LeSS ligt de nadruk op het volledige product. Dit komt, omdat delen van het product geen werkelijke waarde hebben totdat ze worden samengevoegd om zo het eindproduct te creëren. Klanten kopen immers een heel product, niet de onderdelen ervan. De focus houden op het geheel is één van de grootste uitdagingen in een grote ontwikkelgroep.

Klantgericht

Net als bij productfocus vormt de grootschaligheid van LeSS een risico voor klanttevredenheid. In LeSS verbindt de Product Owner klanten/gebruikers en projectteams. Om ervoor te zorgen dat de klant centraal blijft staan, worden teams georganiseerd op basis van end-to-end functionaliteit en componenten.

Continue verbetering naar perfectie

Als één van de twee pijlers van het Lean denken, behoort continue verbetering naar perfectie ook tot LeSS. In LeSS wordt verandering verwacht en omarmd, continu.

Lean denken

Respect voor mensen en continue verbetering zijn centrale concepten in Lean die door LeSS zijn overgenomen.

Systeemdenken

Dit is het beginsel dat lange-termijn systemische verbeteringen voorkeur verdienen boven snelle oplossingen.

Empirische procesbeheersing

Dit Scrum-concept wordt in LeSS toegepast om ervoor te zorgen dat teams ‘net genoeg proces’ hebben. Op die manier doen ze hun werk in een cyclus van transparantie, inspectie en aanpassing.

Wachtrij-theorie

Deze theorie over hoe items door een systeemwachtrij bewegen is een tool voor het denken over het vermogen grote eenheden (batches) werk aan te kunnen. Dit is vooral relevant voor Scrum op grote schaal.

Scaled Agile Framework® (SAFe®)[18]

Het Scaled Agile Framework is een vrij beschikbare verzameling kennis die is ontwikkeld om problemen aan te pakken die zich voordoen wanneer Agile wordt opgeschaald naar meer dan één team. Deze ontwikkelmethodologie voor de gehele onderneming combineert principes van Lean en Agile. De organisatie- en werkstroompatronen in het framework zijn ontworpen om organisaties te ondersteunen bij het opschalen van hun Lean- en Agile-praktijken.

De negen SAFe Lean-Agile-principes:

1. Kijk economisch.
2. Pas systeemdenken toe.
3. Ga uit van veranderlijkheid; houdt opties open.
4. Ontwikkel incrementeel, met snelle, geïntegreerde leercycli.
5. Baseer mijlpalen op een objectieve evaluatie van werkende systemen.
6. Visualiseer werk; beperk de omvang van batches en wachtrijen.
7. Pas een standaard cadans (timing) toe, synchroniseer gebiedsoverschrijdende planning.
8. Spreek de intrinsieke motivatie van kenniswerkers aan.
9. Decentraliseer besluitvorming.

Het SAFe framework

De Scaled Agile Framework website geeft dit overzicht van SAFe. In SAFe zijn er vier configuraties:

- essential;
- portfolio;
- large solution;
- full.

Essential

Essential SAFe is de meest basale configuratie. Het beschrijft de meest kritische elementen, bedoeld om het merendeel van de voordelen van het framework te bieden. Het omvat team- en programmaniveau, die Agile Release Trains of ART's worden genoemd.

Portfolio

Portfolio SAFe omvat strategische richting, financiering van investeringen en Lean governance.

Large Solution

Large Solution SAFe laat coördinatie en synchronisatie toe over meerdere programma's, maar zonder portfolio-overwegingen. In vroegere versies van SAFe, werd naar dit niveau verwezen als waardestroom.

---

[18] Vrij naar: https://www.scaledagileframework.com/

Full

Full SAFe combineert de andere drie niveaus.

Disciplined Agile Delivery (DAD)

Disciplined Agile Delivery (DAD) is een framework dat context-specifieke leiding biedt, die past bij de behoeften van de onderneming, om sneller producten van hoge kwaliteit te produceren. Het is een hybride model, dat is gevormd uit een verzameling van beproefde Lean- en Agile-methoden, zoals Scrum, Kanban, XP, Agile Modelling, Unified Process en vele andere.

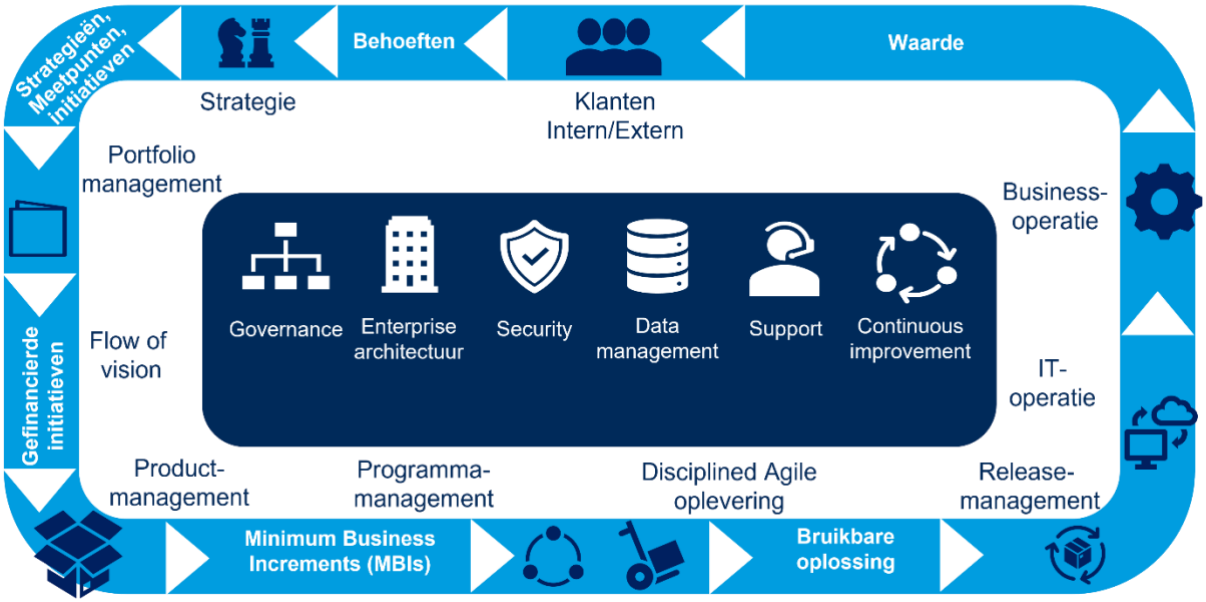
Het framework werd oorspronkelijk ontwikkeld bij IBM Rational van begin 2009 tot juni 2012. Het IBM-team werkte nauw samen met business partners, waaronder Mark Lines en werd geleid door Scott Ambler.

Het intellectuele eigendom van DAD ging in oktober 2012 daadwerkelijk over op het Disciplined Agile Consortium, een feit dat in juni 2014 juridisch werd erkend door IBM.

DAD ontwikkelde zich later tot een uitgebreid instrumentarium en wat nu Disciplined Agile 5.x heet, werd in mei 2020 uitgebracht en gepubliceerd in een boek genaamd 'Choose Your WoW! A Disciplined Agile Delivery Handbook for Optimizing Your Way of Working (WoW).'

DAD is nu slechts een subset van Disciplined Agile, dat een veel breder terrein bestrijkt.

Figuur 34 Hoe partijen samenwerken in DA Flex Workflow



Afbeelding gecreëerd door EXIN gebaseerd op: Project Management Institute, Inc. (2021). *Introduction to Disciplined Agile® (DATM)*. https://www.pmi.org/disciplined-Agile/introduction-to-disciplined-Agile

Kanban

Kanban is een Lean-techniek die populair is in productie-omgevingen en de laatste tijd meer en meer wordt gebruikt in met name IT-projecten en operatie. Wat mensen bedoelen met Kanban in IT-omgevingen is meestal een ontwikkelingsmethode die een combinatie is van verschillende Scrum-elementen met de Kanban-techniek.

Het gebruik van Kanban met Scrum is aan de orde gekomen in hoofdstuk 10.10 Gebruik van de Kanban methode.

Bijlage B – Meer over releases en veel gebruikte release management tools in Agile-omgevingen

Vrijgeven of niet vrijgeven, dat is de vraag

De kritiek op het release concept in Scrum groeit en met goede reden. Het concept van een release drijft de spot met niet ver vooruitplannen, terwijl voorstanders van releases in Scrum zullen zeggen dat het geen detailplanning is! Wel, dat is het wel en dat is het niet.

Let wel, het concept van releases maakt sinds 2011 officieel geen deel meer uit van de Scrum gids. Waarom is releaseplanning eigenlijk verwijderd uit de Scrum gids? Het antwoord is kort en eenvoudig: Het is mogelijk succes te hebben met Scrum zonder releaseplanning.

Het niet nodig hebben van releaseplanning kan een positieve indicator zijn van de gezondheid van een product. Scrum is bedoeld om voortdurend waarde te leveren aan klanten door middel van korte iteraties. Het gebruik van releases kan de levering van waarde, feedback van de klant en return on investment (RoI) vertragen. Met volwassen ontwikkelpraktijken en goede uitrolstrategieën kan de levering van waarde elke sprint plaatsvinden.[19]

Laten we eens onderzoeken waarom een traditionele release managementaanpak geen goed idee is in Scrum, door enkele van de mythes te ontkrachten die voorstanders van release management naar voren brengen, namelijk:

Mythe 1

In complexe en grootschalige omgevingen is planning vooraf noodzakelijk.

Realiteit 1

Ja – maar alle planning die nodig is, kan worden gedaan door ervoor te zorgen dat de product backlog is verfijnd.

Mythe 2

Product backlog verfijning is onvoldoende planning om coördinatie, afhankelijkheden en volgorde te bepalen.

Realiteit 2

Ja – maar het gebruik van een geschaald mechanisme zoals Nexus volstaat om een goede coördinatie te verzekeren tussen Scrum-teams die deel uitmaken van de Nexus en om goed om te gaan met afhankelijkheden, zelfs als die over iteraties heen gaan.

[19] https://www.Scrum.org/resources/gone-are-Release-planning-and-Release-turndown

Mythe 3

Methoden als Nexus houden geen rekening met opeenvolgende sprints en houden dus onvoldoende rekening met afhankelijkheden in de tijd.

Realiteit 3A

Het redelijk gedetailleerd plannen van wat er in opeenvolgende sprints zal gebeuren, zoals mensen doen in releaseplanning, is Scrum terug veranderen in Waterval. Inzicht in volgorde, afhankelijkheden en wat er hoogstwaarschijnlijk gaat gebeuren in de volgende sprint, kan heel goed worden opgevangen door de product backlog goed te beheren. Bovendien, een geschaalde aanpak zoals Nexus, registreert en beheert specifiek de volgorde voor afhankelijkheden over sprints en alleen dat, omdat de rest gebeurt bij het beheren van de product backlog.

Realiteit 3B

Eén van de meest negatieve gevolgen van de release-aanpak is dat teams in de val lopen door te zeggen: 'Nou, in sprint 2 gaan we x doen, dat was al besloten. ' Daarmee schenden ze Agile-principe 2 en mogelijk als gevolg daarvan ook 1, 4, 5, 10 en 11.

Mythe 4

Releases creëren voorspelbaarheid en helpen ons aan business-doelstellingen te committeren, de business wil deadlines.

Realiteit 4

Wat? Dat is WaterScrumVal. Dit is niet Agile.

Release richtlijnen

Als blijkt dat release management nodig is in de organisatie, dan zijn hier een aantal richtlijnen.

- Strek release cycli niet uit over meer dan drie sprints.
- Zorg ervoor dat iedereen weet dat het releaseplan na elke iteratie kan en moet veranderen, tenzij alles in de business hetzelfde blijft, wat hoogst onwaarschijnlijk is.
- Het maken en gebruiken van een release backlog (zoals een sprint backlog maar dan voor alle sprints die deel uitmaken van de release) moet op hoog niveau en zeer flexibel zijn. Dit betekent dat nieuwe items voortdurend hun weg zullen vinden naar de release backlog en als gevolg daarvan zullen items worden geschrapt, tenzij meer teams deelnemen aan de release.
- Gebruik releaseplanning als een leidraad en een middel om te focussen op de product goal; dingen die nog niet bekend waren, worden ontdekt en moeten snel worden aangepakt.
- Wees flexibel en focus op wat juist is voor de klant, houd niet alleen vast aan het plan.
- Leg afhankelijkheden en volgorde vast in de product backlog.

- Gebruik een schalingsmechanisme zoals Nexus. Releaseplanning en management zal veel eenvoudiger en van hoog niveau zijn. Let wel: methoden als Scrum@Scale omvatten het release managementconcept, dus het kan nuttig zijn om de richtlijnen daarvan te onderzoeken.
- Doe geen taakverdeling in de releaseplanning; laat dat voor iedere sprintcyclus. In essentie betekent dit dat als er geen schalingsmechanisme zoals Nexus gebruikt wordt, er in elke sprintcyclus een beetje releaseplanning plaatsvindt.
- Eindig niet met WaterScrumVal.

Burn-down en schatten

Een burn-down chart is een bijzonder nuttige tool om releases te volgen. Het idee verschilt niet van de normale burn-down chart, maar heeft een paar aanpassingen.

De voortgang in Scrum projecten kan worden bijgehouden door middel van een release burn-down chart. De Scrum Master die de release beheert, werkt de release burn-down aan het einde van elke sprint bij.

Net als bij een sprint burn-down chart, toont de verticale as de hoeveelheid werk die overblijft aan het begin van elke sprint en om dit uit te drukken kunnen story points, ideal days, team days, of wat er dan ook gekozen is, worden gebruikt. De horizontale as toont de tijd, maar in dit geval, het aantal sprints voordat de release (project) klaar is. Waarom die moeite doen als een release niet verder weg kan zijn dan drie sprints?

Dit is waar de kicker in beeld komt.

Als er releases gebruikt worden, is de kans groot dat gebeurt, omdat men gelooft in Mythe 4.

Als er voortdurend werk wordt toegevoegd en de klanten willen geen prioriteit verlagen van product backlog items in de release backlog, dan is de enige logische consequentie dat het langer zal duren!

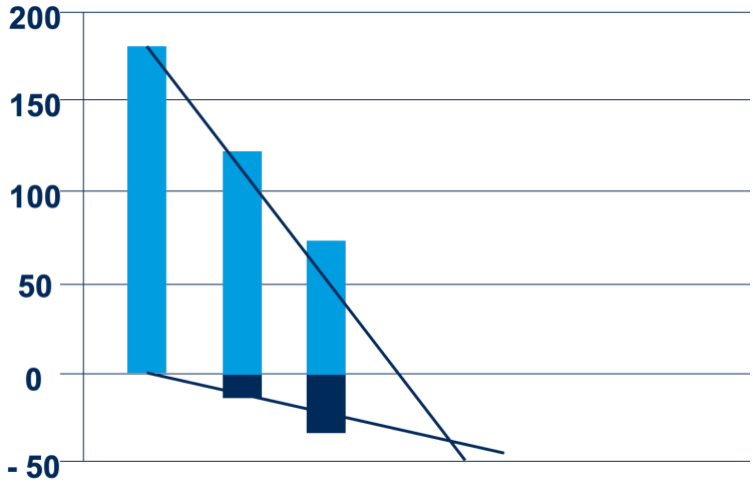
Maar hoe veel langer?

Het staafdiagram voor de release burn-down helpt het antwoord te vinden.

Release burn-down staafdiagrammen gaan niet alleen tot nul, maar staan ook toe negatief te gaan op de verticale balk. Als extra werk wordt toegevoegd tijdens de release, wordt het toegevoegd onder de nullijn, zoals bij elke sprint cyclus.

Dus waarom is dit nuttig? Omdat er op deze manier twee trendlijnen gebruikt kunnen worden om te voorspellen hoeveel sprints het zal duren om het werk in de release backlog af te maken.

Figuur 35 Release burn-down staafdiagram



Afbeelding gecreëerd door EXIN gebaseerd op: Botha, J. (2018). *Scrum Masters and Product Owners [Courseware]*. GetITright.

De release burn-down heeft nu twee trendlijnen: één voor de gegevens boven de nullijn; en één voor de gegevens onder de nullijn. Het snijpunt vertelt de Scrum Master die de release begeleidt dat een extra sprintcyclus nodig is om de release af te maken, met het nieuw toegevoegde werk in sprints 2 en 3. Dit is niet ideaal, gezien de richtlijn van drie sprints.

De realiteit is dat dit er in de praktijk vaak nog slechter uitziet, waarbij vooruitzichten op 10 of meer sprints niet ongewoon zijn. Daar heb is het – een Waterval project.

Gebruik van Gantt-grafieken voor releaseplanning

Gantt-grafieken zijn gangbaar om teams visueel te informeren over wat in de release backlog zal gebeuren in welke sprint. Ze worden ook gebruikt om de onderlinge afhankelijkheden van product backlog items en bijbehorende taken te laten zien.

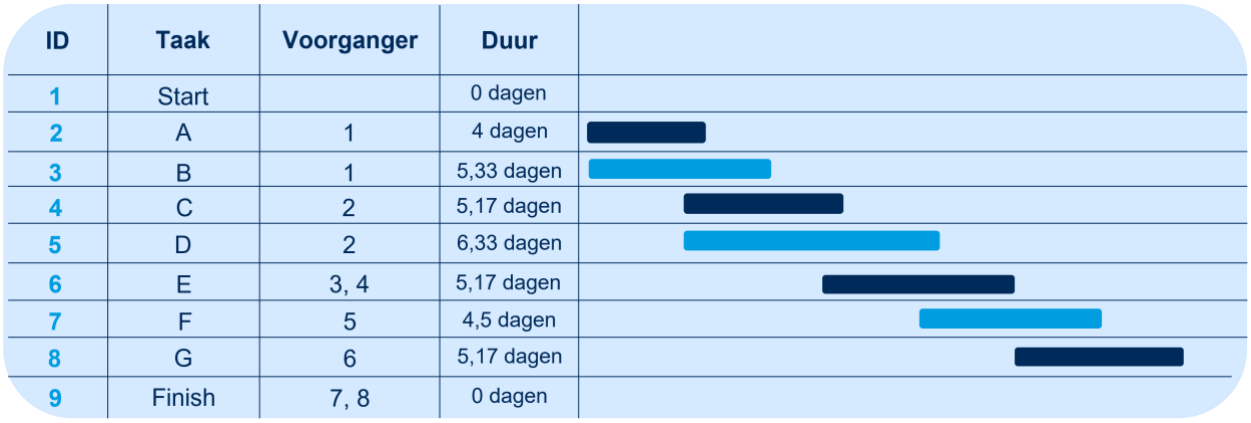
Maar wat is een Gantt-grafiek?

Een Gantt-grafiek is een visueel overzicht van in de tijd geplande taken, hun onderlinge relaties en hun afhankelijkheden.

Het concept werd rond 1910 ontwikkeld door Henry L. Gantt, die samenwerkte met Frederick Taylor, als een methode om productieplanning en het laden van grondstoffen voor fabrieken en werkplaatsen te beschrijven. Eén van de eerste belangrijke toepassingen van Gantt-grafieken was door de Verenigde Staten tijdens de Eerste Wereldoorlog, om de logistiek voor de oorlog te plannen.

We hebben deze grafieken allemaal wel eens gezien tijdens projectvergaderingen en ze zien er ongeveer zo uit:

Figuur 36 Releaseplanning in Gantt-grafiek



ID	Taak	Voorganger	Duur
1	Start		0 dagen
2	A	1	4 dagen
3	B	1	5,33 dagen
4	C	2	5,17 dagen
5	D	2	6,33 dagen
6	E	3, 4	5,17 dagen
7	F	5	4,5 dagen
8	G	6	5,17 dagen
9	Finish	7, 8	0 dagen

Afbeelding gecreëerd door EXIN gebasserd op: Gantt Chart. (2021). Verkregen op 14 februari 2021 van https://en.wikipedia.org/wiki/Gantt_chart.

In principe is er niets mis met het gebruik van Gantt-grafieken als een middel om gegevens te visualiseren; het belangrijkste is de context waarin ze worden gebruikt en wat de informatie impliceert. Als ze worden gebruikt voor gedetailleerde planning, maken ze van Agile en Scrum een aanfluiting.

Bibliografie & referenties

Agile Business Consortium (2014). *The DSDM Agile Project Framework (2014 Onwards).* https://www.agilebusiness.org/page/TheDSDMAgileProjectFramework

Ambler, S. & Lines, M. (2020). *Choose your WoW!: A Disciplined gile Delivery Handbook for Optimizing Your Way of Working (WoW).* Project Management Institute.

Bigelow, S. J. (2020). *7 techniques for better Agile requirements gathering.* https://searchsoftwarequality.techtarget.com/tip/7-techniques-for-better-Agile-requirements-gathering

Botha, J. (2018). *Scrum Masters and Product Owners [Courseware].* GetITright.

Botha, J. (2019). *Agile: A Manager's Guide to Unlocking Business Value.* Amazon Digital Services LLC – Kdp Print Us.

Botha, J. (2020). *Scrum Lego-game workbook [Courseware].* GetITright.

Botha, J. (2021). *[Courseware].* GetITright. Retrieved in personal communication from the author of the book.

Botha, J. (2021). *Lean fundamentals for IT [Courseware].* GetITright.

Cohn, M. (2005). *Agile Estimating and Planning.* Pearson Education.

Cohn, M. (2009). *Four Attributes of the Ideal Pilot Project.* https://www.mountaingoatsoftware.com/blog/four-attributes-of-the-ideal-pilot-project

Cohn, M. (2010). *Succeeding with Agile.* Addison-Wesley.

Crystal Methods. (2021). Retrieved on September 23, 2021 from https://en.wikiversity.org/wiki/Crystal_Methods

EXIN Holding B.V. (2019). *Agile Methodologies.* EXIN Holding B.V. https://dam.exin.com/api/&request=asset.permadownload&id=3144&type=this&token=8659eabedff466d86ac2eba5ed72b33d

Exoskeleton. (2021). Retrieved on September 23, 2021 from https://en.wikipedia.org/wiki/Exoskeleton

Gantt Chart. (2021). Retrieved on September 23, 2021 from https://en.wikipedia.org/wiki/Gantt_chart

Goldratt, E. M. & Cox, J. (2012). *The Goal.* (3rd Edition) North River Press.

Hiatt, J. M. & Creasey, T. J. (2012). *Change Management: The People Side of Change.* Prosci Learning Center Publications.

Kano+ (2020). *The Kano model – Assessing Product Features based on Customer Satisfaction.* Kano+ https://kano.plus/about-kano#analyze-a-study

Maher, R., & Kong, P. (2016). *Cross-Team Refinement in Nexus™.* Scrum.org. https://www.scrum.org/resources/cross-team-refinement-nexus

Maher, R., & Kong, P. (2016). *The Nexus Integration Team.* Scrum.org. https://www.scrum.org/resources/nexus-integration-team

Maher, R., & Kong, P. (2016). *Visualizing the Nexus Sprint Backlog.* Scrum.org. https://www.scrum.org/resources/visualizing-nexus-sprint-backlog

Martin, B. A. (1980). *The goal: to improve credibility in the reporting of engineering progress.* Project Management Quarterly, 11(2), 14–22.

McKinsey & Company (2019). *The journey to an Agile organization*. https://www.mckinsey.com/business-functions/organization/our-insights/the-journey-to-an-agile-organization

Meier, J. D. (2019). *Extreme Programming at a Glance.* https://jdmeier.com/extreme-programming-at-a-glance/
Overeem, B. (2016). *The 11 Advantages of Using a Sprint Goal.* https://www.scrum.org/resources/blog/11-advantages-using-sprint-goal
Pichler, R. (2010). *Agile Product Management with Scrum.* Addison-Wesley.
Project Management Institute, Inc. (2021). Introduction to Disciplined Agile® (DA™). https://www.pmi.org/disciplined-Agile/introduction-to-disciplined-Agile
Prosci (2021). *Best Practices in change management benchmarking report.* https://www.prosci.com/resources/articles/change-management-best-practices
Rad, N. K. (2021). *Agile Scrum Handbook* (3de ed.). Van Haren Publishing.
Rother, M. (2018). *The Toyota Kata Practice Guide: Practicing Scientific Thinking Skills for Superior Results in 20 Minutes a Day.* Mcgraw-Hill Education.
Scaled Agile (2021). *Scaled Agile Framework.* https://www.scaledagileframework.com/
Schwaber, K., & Scrum.org (2021). *The Nexus™ Guide – The Definitive Guide to Scaling Scrum with Nexus.* Scrum.org. https://www.scrum.org/resources/nexus-guide
Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide – The Definitive Guide to Scrum: The Rules of the Game*. Scrumguides.org. https://scrumguides.org/scrum-guide.html
Scrum.org (2021). *Gone Are Releaseplanning and the Release Turndown.* https://www.scrum.org/resources/gone-are-Release-planning-and-Release-turndown
Scrum.org., Vacanti, D., & Yeret, Y. (2021). *The Kanban Guide for Scrum Teams.* Scrum.org. https://www.scrum.org/resources/kanban-guide-scrum-teams
Senge, P. (2006). *The Fifth Discipline.* Bantam Doubleday Dell Publishing Group Inc.
Smart, J. (2020). *Sooner Safer Happier. Antipatterns and Patterns for Business Agility.* IT Revolution Press.
Sutherland, J., & Scrum Inc. (2021). *The Scrum@Scale® Guide – The Definitive Guide to Scrum@Scale: Scaling that Works.* Scrum.org. https://www.scrumatscale.com/wp-content/uploads/2020/12/official-scrum-at-scale-guide.pdf
Technical Debt. (2021). Retrieved on September 23, 2021 from https://en.wikipedia.org/wiki/Technical_debt
The LeSS Company B.V. (2014). *Large Scale Scrum is Scrum [Visual].* https://less.works/img/principles/principles.pdf
Underwood, J. (2013). *Musselwhite and Ingram's Change Style Indicator.* https://innovategov.org/2013/08/15/musslewhite-and-ingrams-change-style-indicator/
Wells, D. (2000). *Introducing Extreme Programming.* http://www.extremeprogramming.org/introduction.html
Wells, D. (2013). *Extreme Programming: A gentle introduction.* http://www.extremeprogramming.org/

Contact EXIN

www.exin.com