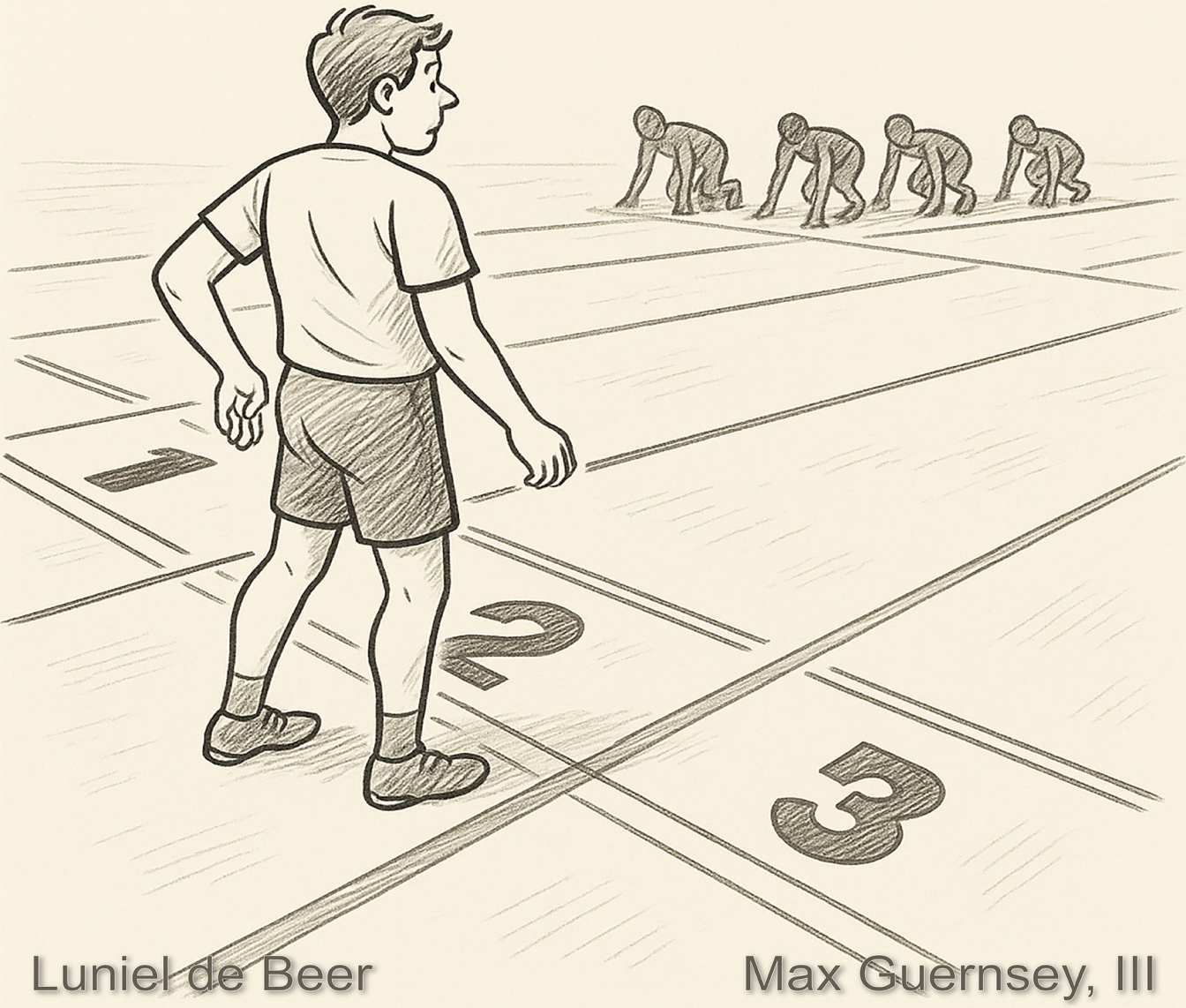
READY

WHY MOST SOFTWARE PROJECTS FAIL AND HOW TO FIX IT



Luniel de Beer

Max Guernsey, III

Nederlandse Editie

Tweet dit boek!

Help Luniel de Beer en Max Guernsey, III door het nieuws over dit boek te verspreiden op Twitter!

De voorgestelde tweet voor dit boek is:

Ik heb net Ready gekocht — een boek voor softwareleiders en teams die herwerk, overdracht en verkeerde afstemming in softwareoplevering willen elimineren. #CodeReady

De voorgestelde hashtag voor dit boek is #CodeReady.

Ontdek wat anderen over het boek zeggen door op deze link te klikken om te zoeken naar deze hashtag op Twitter:

#CodeReady

Klaar (Nederlandse Editie)

Waarom De Meeste Softwareprojecten Mislukken en Hoe Je Dit Kunt Voorkomen

Luniel de Beer en Max Guernsey, III

Dit boek is beschikbaar op https://leanpub.com/ready-nl

Deze versie is gepubliceerd op 2025-10-22

Dit is een Leanpub boek. Leanpub stelt auteurs en uitgevers in staat om gebruik te maken van het Lean Publishing proces. Lean Publishing is het publiceren van een boek in ontwikkeling met behulp van lichtgewicht tools en vele iteraties om feedback van lezers te krijgen, bij te sturen tot je het juiste boek hebt en vervolgens tractie op te bouwen.

Ter nagedachtenis aan Johann van Aardt, die mijn passie herkende, me introduceerde in de echte programmering, en me hielp mijn weg te vinden naar een nieuw thuis. En aan mijn ouders, wier onwankelbare steun—van mijn eerste klant tot mijn eerste huis in de VS—dit alles mogelijk heeft gemaakt.

—Luniel

Voor mijn familie, met wie de zon op- en ondergaat.

—Max

Inhoudsopgave

Over dit boek

Ready is een boek voor iedereen die betrokken is bij softwareontwikkeling en die genoeg heeft van **onderlevering**, **chronisch herwerk** en **onduidelijke requirements**.

Mogelijk heeft u al geïnvesteerd in de uitvoeringsvaardigheden van het team, een verbeterde implementatie van uw procesraamwerk, of het opfrissen van de code en heeft u nog steeds behoefte aan meer verbetering.

Dit komt omdat de belangrijkste beperking voor de meeste softwareontwikkelingsteams niet de teamvaardigheden zijn, maar de requirementsmaturiteit. **Zelfs ervaren teams met de juiste vaardigheden worstelen nog steeds** wanneer ze werken met onvolwassen requirements.

Ready introduceert RMF (Requirementsmaturatiestroom), een praktische en diep gestructureerde aanpak om Product en Engineering op één lijn te brengen zonder uw bestaande proces te vervangen.

Of u nu Scrum, Kanban of iets aangepasts gebruikt, RMF helpt u bij het **stabiliseren van de scope**, het **elimineren van overloop** en het **leveren van wat er echt toe doet**.

Als uw teams zich vastgelopen voelen op de rand van “bijna klaar”, zal dit boek u laten zien hoe u **de cyclus kunt doorbreken** en uw team(s) **definitief** kunt deblokkeren.

Voor wie is dit bedoeld

Dit boek is letterlijk voor iedereen die betrokken is bij softwareontwikkeling. Van engineers tot productmanagers en van individuele bijdragers tot leidinggevenden.

Dit boek is voor u als u betrokken bent bij softwareontwikkeling en heeft gemerkt dat een team waarmee of waarin u werkt een of meer van de volgende problemen heeft:

- Werk wordt regelmatig overgedragen van de ene iteratie naar de volgende
- Implementatieteams hebben het gevoel dat ze bewegende doelen proberen te raken
- Werk blijft te lang openstaan
- Werk wordt als afgesloten gemarkeerd maar dingen zijn niet echt klaar
- Uitgevoerd werk komt niet overeen met de verwachtingen
- Werk genereert regelmatig een groot aantal defecten
- Grote hoeveelheden werk moeten regelmatig opnieuw worden gedaan

Als een van deze problemen bekend voorkomt, kan *Ready* helpen.

Hoe dit boek te gebruiken

Dit boek is ontworpen om praktisch te zijn. Het is geen theoretische verhandeling of strategische presentatie - het is een handleiding voor het implementeren van RMF (de Requirementsmaturatiestroom) gebaseerd op echt klantwerk en in de praktijk getest onder echte leveringsdruk.

De hoofdstukken zijn in volgorde geschreven, maar RMF zelf is modulair. Het bestaat uit drie fundamentele praktijken:

- RMF 1: Samenwerken voor gedeeld begrip
- RMF 2: Het afronden van werk bewaken met Op maat gemaakte Definities van Klaar
- RMF 3: Implementatie bewaken met Op maat gemaakte Definities van Gereed

Elk onderdeel, of "Gewoonte" zoals we ze noemen, staat op zichzelf, maar ze bouwen op elkaar voort. Het boek is ontworpen om u te helpen ze één voor één aan te pakken, in volgorde. Die structuur weerspiegelt hoe we teams aanraden RMF in de praktijk te adopteren - waarbij elke Gewoonte pas wordt toegevoegd nadat de vorige werkt.

Dit voorkomt dat teams overweldigd raken en geeft elke verandering de beste kans om te beklijven. U leert meer over hoe u dit kunt doen vanaf Hoofdstuk 9.

Als u op zoek bent naar hulp - of dat nu advies, coaching of iemand is om met uw leiderschapsteam te spreken - neem dan gerust rechtstreeks contact met ons op.

En als u op zoek bent naar formele ondersteuning bij het implementeren van RMF, biedt Producore een volledige reeks Programma's die stap voor stap begeleiden bij de adoptie. U kunt meer leren op https://ready-book.link/rmf.

Over de auteurs

Luniel de Beer is de bedenker van de Requirements Maturation Flow (RMF), een praktisch systeem voor het oplossen van de kloof tussen productintentie en engineering-uitvoering. Hij heeft meer dan 15 jaar ervaring in het leiden van Agile transformaties, het overbruggen van product en engineering, en het helpen van teams om met duidelijkheid en vertrouwen te leveren.

Luniel is ook de bedenker van Producore's Capability Management systeem, een traceerbare en schaalbare aanpak voor het modelleren van productmogelijkheden. Hij bedacht PKB-Gedreven Ontwikkeling (PKBDD), een versiebeheerd systeem voor het beheren van permanente productrequirements. Deze hulpmiddelen vormen onderdeel van een groter leveringsraamwerk ontwikkeld bij Producore.

Max Guernsey, III is een software-architect, opleider en medeoprichter van Producore, een adviesbureau dat zich richt op het oplossen van leveringsproblemen door middel van structurele en technische discipline. Met meer dan twee decennia ervaring in objectgeoriënteerd ontwerp, refactoring, testgedreven ontwikkeling en ontwerppatronen, heeft hij zowel bedrijfskritische systemen geleverd als engineeringteams op grote schaal gecoacht. Zijn werk combineert diepgaande technische praktijken met gedrags- en procestransformatie om organisaties te helpen duurzame leveringsexcellentie te bereiken.

Max heeft aanzienlijk bijgedragen aan PKBDD en leidde de ontwikkeling van Producore's benadering van Gedragsgestuurd Ontwikkelen (BDD) door zijn diepgaande expertise in gedragsspecificatie.

Samen integreren hun werkzaamheden helderheid, traceerbaarheid en poortwachting in een samenhangend systeem voor softwareoplevering dat schaalt van teampraktijk naar organisatorische capaciteit.

Voorwoord

Bericht aan Engineering Leiders

Als je een senior leider bent in een engineering organisatie, heb je waarschijnlijk geen tekort aan inzet, discipline of slimme mensen. En toch komen projecten nog steeds tot stilstand. Doelen verschuiven. Verwachtingen worden niet waargemaakt. Niet omdat je teams lui zijn—maar omdat er iets fundamenteels mis is in hoe werk wordt gedefinieerd, vormgegeven en opgeleverd.

Dit boek is geen leiderschapsgids. Het is een instrument voor structurele diagnose. Het onthult wat er werkelijk gebeurt binnen je teams—waarom "bijna klaar" steeds verandert in "niet klaar," en waarom lokale vooruitgang zo zelden wordt vertaald naar strategische resultaten.

Je zult *jezelf* misschien niet herkennen in deze pagina's. Maar als je teams niet kunnen opleveren wat je nodig hebt, zul je *hen* herkennen. En wanneer je dat doet, heb je eindelijk de taal—en het systeem—om het op te lossen.

Van Luniel

Allereerst zou dit boek niet mogelijk zijn geweest zonder Max, wiens vermogen om door de mist en het kaf heen te kijken en een idee tot zijn essentie te distilleren mij volledig te boven gaat.

Hoe zijn we hier gekomen?

Als ik terugkijk, denk ik dat het komt omdat ik altijd al wilde begrijpen hoe dingen echt werken. Of het nu ging om religie, voeding of softwareontwikkeling, ik liep steeds tegen hetzelfde probleem aan: oppervlakkige antwoorden die onder druk niet standhielden. Dus bleef ik graven—vroeg niet alleen wat we doen, maar waarom, en wat er ontbreekt als het niet werkt.

Een van de eerste scheuren in het systeem werd zichtbaar in een rol waarin ik drie petten droeg: Scrum Master, Product Owner en Development Manager (!!) voor een team dat

dataservices leverde bij een bekend technologiebedrijf. We deden wat Scrum voorschreef—korte Sprints, stories in een backlog, planning in een halve dag—maar elke keer als we aan een nieuwe Sprint begonnen, liepen we tegen wrijving aan. Het team begreep het probleem niet volledig, we moesten de vereisten tijdens de Sprint herzien, vermijdbare afhankelijkheden kwamen naar boven en vertraagden ons, en belangrijke stappen werden gemist.

Dus begon ik iets anders te doen. Ik bracht het team en belanghebbenden samen in een ruimte voor elke story, besprak het probleem in detail, brainstormde samen over de oplossing, en schreef pas daarna de story. Sprint Planning kromp tot een uur, en ons succes in oplevering schoot omhoog.

Langzaam begon ik te beseffen dat succes niet komt door harder te werken binnen de Sprint. Het komt door de structuur die je aanbrengt *voordat* deze begint.

Later, nadat ik Jeff Sutherland hoorde praten over "Definitions of Ready", wist ik dat daar iets waardevols in zat—maar het was niet genoeg. Mijn ervaring met vereisten, UX, UI, onderzoek, en later met BDD liet me zien dat verschillende werkitems verschillende soorten gereedheid vereisen. Sommige hebben gedragsspecificaties nodig. Sommige hebben systeemtoegang nodig. Sommige hebben een volledige tracering van mogelijkheden nodig.

En ze hebben allemaal bevestigd gedeeld begrip nodig—niet aangenomen.

Naarmate ik met meer teams werkte, zag ik overal hetzelfde patroon: ontbrekende stappen, niet-vervulde afhankelijkheden, teams die hun best deden maar constant aan het worstelen waren om problemen op te lossen die voorkomen hadden moeten worden. Zelfs geweldige teams hadden het moeilijk—niet omdat ze zwak waren, maar omdat ze een structuur misten die gereedheid expliciet maakte.

Het resultaat van al dat leren, itereren en frustratie is een gestructureerd systeem voor het managen van gereedheid.

Daar gaat dit boek over.

Ik hoop dat het je duidelijkheid geeft over waar de echte problemen liggen en hoe je ze kunt oplossen. Ik hoop dat het je taal geeft om praktijken te verdedigen die misschien "extra" lijken maar eigenlijk essentieel zijn. En bovenal hoop ik dat het teams helpt om met minder stress, minder verrassingen en veel betere resultaten op te leveren.

Als we dit goed krijgen, besparen we de industrie miljarden euro's.

Maar belangrijker nog, we geven mensen hun gezond verstand terug.

Van Max

Ik werk al tientallen jaren vanuit verschillende invalshoeken aan dit probleem, maar mijn vooruitgang stagneerde totdat ik Luniel ontmoette.

Dit komt omdat ik, voordat ik hem kende, het probleem fundamenteel als een technisch probleem benaderde. Ik was gericht op het helpen van teams bij het adopteren van zaken als Test-Driven Development (TDD), refactoring, geavanceerd softwareontwerp en, later, Acceptance-Test-Driven Development (ATDD) of Behavior-Driven Development (BDD).

In de meeste van die gevallen werd het probleem dat in dit boek wordt behandeld gezien als een implementatiedetail bij het vestigen van die technische praktijken.

Dit betekent niet dat ik de technische praktijken niet meer waardeer. Ik denk nog steeds dat ze van groot belang zijn, maar ze pakken het probleem van gereedheid in softwareontwikkeling niet direct aan. In plaats daarvan brengen ze dat probleem aan de *oppervlakte* en dan plakken mensen een pleister op hun proces om het "net genoeg" aan te pakken om de technische praktijken te ondersteunen die ze proberen te implementeren.

Ik wil ook ingaan op de vraag wie dit boek kan helpen. Het korte antwoord is "waarschijnlijk bijna iedereen in softwareontwikkeling", maar het echte antwoord bevat nuances die helpen om het in kaart te brengen voor verschillende omgevingen zonder de fundamentele betekenis te veranderen.

Er zijn teams die de oplossing nodig hebben die in dit boek wordt geboden. Je zult in hoofdstuk 1 kennismaken met een gekuiste versie van zo'n team.

Er zijn ook teams die niet per se een systeem **nodig** hebben zoals wij voorstellen, maar er wel baat bij kunnen hebben.

Het beste team waarmee ik ooit heb gewerkt—gemakkelijk een volledige standaarddeviatie boven het **op één na beste team**, zo niet twee—bevond zich in het achterland van Central Oregon. Ze presteerden zo hoog dat ze het ontbreken van zo'n systeem konden overwinnen door pure competentie. Toch zei mijn manager destijds, Tom Barreras, ooit iets in de trant van "Ik heb gemerkt dat onze user stories beter verlopen wanneer we vooraf wat tijd besteden aan het bespreken van de tests."

Dit was weer iets wat ik destijds bekeek door de bril van testontwikkeling en technische uitvoering, maar nu weet ik dat het nog een aanwijzing was dat *gereedheid* een factor was die het team beïnvloedde… dat specifieke team was gewoon zo bekwaam en snel in het reageren dat ze succesvol konden zijn door te reageren op belemmeringen wanneer ze zich voordeden, in plaats van ze bij voorbaat te voorkomen.

Zelfs als je het type persoon bent die zich niet strikt zorgen hoeft te maken over gereedheid omdat je het kunt overwinnen, of als je werkt met een team van hetzelfde kaliber, kun je nog steeds profiteren van de inhoud van dit boek.

Deel I: Er Ontbreekt Iets

*Wanneer dezelfde dingen beter doen niet helpt, kijk dan naar wat er **niet gedaan wordt**.*

Hoofdstuk 1: Het Verborgen Probleem

Dit is een waargebeurd[1] verhaal over een bank. We noemen het simpelweg "De Bank". Het is een soort Federale Kredietinstelling die deel uitmaakt van de nationale financiële infrastructuur van de Verenigde Staten.

Wij (Luniel en Max) werden bij De Bank binnengehaald omdat ze moeite hadden met het opleveren van een softwareproject. Het was een van de meest disfunctionele omgevingen die we ooit hadden gezien, en dat is precies waarom we dit als openingscasus hebben gekozen: als betekenisvolle verandering mogelijk was bij De Bank, dan is het overal mogelijk.

Een Korte Opmerking over Projecten

Wanneer we in dit boek de term "project" gebruiken, bedoelen we dit in de context van projectmanagement. Hoewel er verschillende ideeën bestaan over wat het woord betekent, gebruiken we de definitie van het Project Management Institute:

> "Een project is een **tijdelijke** onderneming die wordt aangegaan om een uniek product, dienst of resultaat te creëren."

Dit betekent dat een project een gedefinieerd begin en einde heeft. Wanneer een project wordt afgesloten, worden projectkennis en -artefacten gearchiveerd, teamleden vrijgemaakt en contracten afgerond.

In dit boek gaat een project fundamenteel over uitvoering. De meeste projecten, zoals gedefinieerd door PMI, beginnen met haalbaarheid of ontwerp. Visievorming en strategie hebben al plaatsgevonden voordat een project wordt gestart.

Een project ontstaat uit die visie en strategie, en het slaagt of faalt op basis van of de beoogde doelen worden gerealiseerd—*niet* op basis van of die doelen de juiste waren.

[1] We hebben identificerende details gewijzigd om de privacy van de mensen en instellingen waarbij dit gebeurde te beschermen.

> Mogelijk gebruikt u het woord “project” anders, en dat is prima. Weet alleen dat wanneer wij het gebruiken, we verwijzen naar de bovenstaande definitie en context.
>
> Dit alles betekent niet dat we het gebruik van projectmanagement voor softwareontwikkeling goedkeuren. Integendeel. Maar we erkennen dat het desondanks *wel* wordt gebruikt. We pakken dat probleem later aan, in Hoofdstuk 6.

1.1: De Klassieke Herschrijving

De Bank was bezig met het herschrijven van hun aflossingsportaal om verschillende redenen.

Het oude systeem, een volledig C#/.NET-oplossing, zat vol bugs. Naast dat het de klanttevredenheid verminderde, genereerde het ook een eindeloze stroom van zeer dure supportincidenten waarbij iemand handmatig de database moest aanpassen om een fout van het systeem te corrigeren.

Het oude systeem was ook verouderd vanuit het oogpunt van onderhoudbaarheid. Het was voor engineers bijna onmogelijk om betekenisvolle wijzigingen aan te brengen en, zelfs wanneer ze dat konden, was het een extreem riskante onderneming.

De herschrijving zou dat allemaal veranderen.

Het nieuwe systeem zou nog steeds een C#/.NET-backend hebben, maar die zou volledig gedekt zijn door tests. De frontend zou worden geïmplementeerd in OutSystems, een populaire low-code oplossing die een organisatie in staat stelt om een applicatie op één plek te definiëren en automatisch een webapp, een Android-app en een iOS-app te genereren wanneer ze besluiten hun wijzigingen te publiceren.

De hoop was dat het nieuwe systeem bugvrij zou zijn, waardoor zowel de klanttevredenheid zou verbeteren als de supportkosten aanzienlijk zouden verminderen.

Ze hoopten ook dat de herschrijving de ontwikkelaars zou ‘deblokkeren’—waarbij de combinatie van een meer gedisciplineerde aanpak van de backend en de low-code aanpak van de frontend de kosten en risico’s van nieuwe functionaliteiten sterk zou verminderen.

Een mooi bijeffect van de overstap naar OutSystems was dat ze een schone, moderne mobiele app zouden krijgen voor beide grote platforms.

Dat was de droom toen ze drie jaar voor het begin van dit verhaal waren begonnen. De realiteit was dat de teams tot nu toe **niets** hadden opgeleverd.

1.2: Perspectieven op het Probleem

Toen we met het directieteam spraken, hoorden we zeer begrijpelijke frustratie over het feit dat ze zulke grote investeringen hadden gedaan zonder enige strategische vooruitgang.

Vanuit hun perspectief hadden ze alles geprobeerd. Ze hadden personeel vervangen, meer personeel aangenomen, het budget aangepast, de druk opgevoerd en een parade aan consultants binnengehaald (waarbij werd gesuggereerd dat wij de laatste in de rij waren). Niets leek te helpen—in ieder geval niet op een manier die ze konden meten, want alles wat ze zagen was dat de "meter" het ene kwartaal op nul stond en het volgende kwartaal nog steeds op nul.

Ze wilden geen "onzichtbare vooruitgang" meer. Ze wilden *resultaten*.

Toen we spraken met het management binnen de Productorganisatie, kregen we een iets ander (maar nog steeds vergelijkbaar) verhaal omdat zij directer met Engineering werkten.

Het was niet zo dat de teams niets deden, althans niet in hun ogen. Het was dat de teams nooit deden wat er gevraagd werd. Het was praktisch een garantie: hoe eenvoudig de vraag ook was en hoe duidelijk deze ook werd gesteld, je zou altijd iets compleet anders krijgen wanneer het tijd was om te evalueren wat de teams hadden gemaakt.

Het was zo erg geworden dat de gevleugelde uitspraak werd: "We moeten uitzoeken hoe we kunnen vragen om wat we níet willen, zodat we tenminste een kans hebben om te krijgen wat we wél willen."

Technisch leidinggevenden zagen de zaken heel anders.

Voor hen lag het probleem bij Product dat geen uitvoerbare requirements leverde en dat Product niet *genoeg* requirements leverde. Als Product gewoon "mee zou werken", zouden de teams kunnen leveren wat ze wilden, op tijd en binnen budget.

Ze hadden serieus geïnvesteerd in het moderniseren van hoe code werd geschreven en opgeleverd en, in hun ogen, leverde Product geen heldere requirements.

Toen we spraken met andere consultants (die ons bij de organisatie hadden geïntroduceerd), focusten zij terecht op de disfunctie die ze zagen: Iedereen leek erg gefocust op het de schuld geven aan een ander. De reden dat ze ons überhaupt binnenhaalde was dat ze bezorgd waren over de personeelsstrategie en een evaluatie wilden van individuele medewerkers, maar ze zagen het vingerwijzen en de taakmeestermentaliteit op directieniveau als de belangrijkste bron van problemen.

1.3: Ons Onderzoek

Onze initiële opdracht was om de teams te evalueren en ze indien nodig te helpen hun vaardigheden te verbeteren, dus we begonnen met het onderzoeken van de mensen die aan het front werkten.

Er was zeker ruimte voor verbetering.

De individuele medewerkers aan de Product-kant hadden niet echt de vereiste vaardigheden. In werkelijkheid waren ze meestal projectmanagers die in de rol van Product Owner (PO) of Product Manager waren geduwd.

Als gevolg daarvan schreef de helft van hen “vage” requirements en accepteerde vervolgens (letterlijk) wat de teams ook maar die iteratie deden zonder enige kritische analyse. De andere helft schreef dezelfde soort requirements en beweerde dan dat de teams “hadden moeten weten” wat ze nooit besproken hadden en hielden de werkitems min of meer voor onbepaalde tijd open.

De bank gebruikte Scrum om hun backlog van werk te beheren en bij te houden. Hoewel wat we in dit boek behandelen grotendeels los staat van Scrum, gebruiken we Scrum-terminologie omdat de meerderheid - of ten minste een pluraliteit - van teams Scrum gebruikt.

Definitie: Scrum

Scrum is een lichtgewicht framework dat mensen, teams en organisaties helpt waarde te genereren door middel van adaptieve oplossingen voor complexe problemen. In het kort:

1. Een Product Owner ordent het werk voor een complex probleem in een Product Backlog
2. Het Scrum Team zet een selectie van het werk om in een waardevol increment tijdens een Sprint
3. Het Scrum Team en zijn stakeholders inspecteren de resultaten en passen aan voor de volgende Sprint
4. Herhaal

Als je niet bekend bent met Scrum en de bijbehorende terminologie, raden we je aan de 2020-versie van de Scrum Guide te lezen. Het is een snelle, verhelderende tekst.

Ook ontdekten we dat de technische teams ver onder het gemiddelde scoorden qua programmeervaardigheden (-2σ, op zijn best) en bovendien zeer veranderingsresistent waren. Als natuurlijk gevolg was de codekwaliteit bedroevend.

Toch was dit volgens de teams niet de reden waarom ze niet leverden. Voor hen waren het de vage requirements en wijzigingen tijdens de Sprint door Product die het project om zeep hielpen.

…en niemand had het zelfs maar over het grotere probleem, dat zo absurd was dat het verzonnen lijkt totdat je het zelf hebt meegemaakt.

De technische teams hadden de gewoonte om een requirement niet te begrijpen, iets willekeurigs te bouwen, en vervolgens te eisen dat ze krediet kregen voor het “afronden van een werkitem”.

We bedoelen geen kleine misvatting. We bedoelen een totale disconnect: We vroegen hen om de toepassing van betalingen op de hoofdsom onder bepaalde omstandigheden uit te schakelen, en zij schakelden in plaats daarvan de mogelijkheid uit om een secundair bevestigings-e-mailadres toe te voegen.

Daarna vertelden ze ons dat dit was wat we hadden gevraagd.

1.4: Dieper Graven

Beide vaardigheidstekorten konden worden aangepakt, maar we waren sceptisch of dit de echte blokkades waren.

Er was iets anders mis, dus we groeven dieper. We begonnen met deze vraag: Waarom duurde het schrijven van requirements zo lang en leverde het zo’n slechte resultaten op?

Een reden was dat de kennis die nodig was om een betekenisvolle requirement te schrijven schaars was. Een klein deel ervan was in het bezit van de Engineering- en Product-teams.

Een deel ervan zat ingebakken in de code van het legacy systeem. Een deel was volledig verdwenen. Het grootste deel werd echter bewaard als impliciete kennis bij vakspecialisten verspreid over de verschillende afdelingen van de bank. Dit betekende dat het opstellen van een requirement die daadwerkelijk bijdraagt aan een strategisch doel een zeer arbeids- en tijdsintensieve activiteit was.

Hiertegenover stond een onverzadigbare honger naar functionaliteit van een feature-hongerig leiderschapsteam. Het mandaat was “houd de engineers aan het werk—stop ze vol met requirements”. De focus lag op de hoeveelheid requirements om de teams bezig te

houden—een perspectief dat haaks staat op de zorg en tijd die nodig is om een requirement te definiëren die echt "het verschil zou maken".

1.5: Dingen Beter Maken Maakte Het Niet Beter

Dit zijn allemaal problemen die aan te pakken zijn, maar ze aanpakken hielp niet.

Eerdere verbeteringen in softwareontwikkelingstechnieken hadden de teams niet geholpen bij het opleveren, maar Max deed een poging om de teams verder te helpen verbeteren.

Hij introduceerde revolutionaire concepten uit programmeerdoctrines uit het midden van de twintigste eeuw zoals "kopieer en plak die code niet 27^2 keer, stop het in een functie en roep die in plaats daarvan aan". Deze suggestie alleen al verbeterde de kwaliteit van nieuwe code dramatisch en stelde hen in staat om de kwaliteit te beginnen verbeteren.

Dat en andere basis programmeeradvies hielp hen betere code te schrijven die ze in de toekomst gemakkelijker konden onderhouden.

…maar het hielp niet om het project vooruit te brengen.

Aan de Product-kant kon Luniel BDD introduceren en ervoor zorgen dat Product Owners requirements grondig controleerden voordat ze werden overgedragen aan teams.

Hij kreeg de teams zover om samen te werken en ze te gebruiken om te evalueren of een Product Backlog Item (PBI) echt klaar was.

Definitie: Product Backlog Item (PBI)

> Een Product Backlog Item (PBI) is een afzonderlijke werkeenheid in de product backlog die een potentiële verandering, toevoeging of verbetering aan het product vertegenwoordigt. PBI's kunnen vele vormen aannemen—feature, bugfix, technische verbetering, onderzoekstaak, etc.—en worden gedefinieerd door hun bijdrage aan productwaarde.
>
> Veel teams verwijzen naar PBI's als "stories" of "user stories," maar de correcte Scrum-term is "Product Backlog Item" of "PBI." Zodra een PBI is toegezegd aan een Sprint, maakt het ook deel uit van de Sprint Backlog. Voor eenvoud en neutraliteit gebruiken we "Product Backlog Item" of "PBI" om te verwijzen naar elk werkitem dat het Scrum Team beheert—of je het nu een Product Backlog Item, Sprint Backlog Item (SBI), user story, story, werkitem of backlog item noemt.

[2] Dit is niet alleen geen overdrijving, het is niet eens het ergste geval. In één geval waren er bijna honderd exacte duplicaten van hetzelfde algoritme.

Het bracht *duidelijkheid*, maar het creëerde geen *flow*.

Met de hulp van de partner-consultants die ons hadden binnengehaald, konden we de absolute verpletterende druk op de teams en requirements-auteurs (tijdelijk) verlichten.

Het hielp misschien wat vertrouwen op te bouwen, maar het leverde geen tastbare resultaten op.

We begonnen zelfs met het opbouwen van een kennisbank die mensen hielp bij het opsporen van de bedrijfskennis die ze nodig hadden om requirements te schrijven en identificeerden de plaatsen waar er hiaten in die kennis waren.

Het versnelde het schrijven van requirements, maar het kreeg het product niet uit de deur.

Na maanden van interacties hadden we het leiderschap geholpen om te *zien* waar ze stonden, maar ze waren nog lang niet bij hun doelen. En ze kwamen ook niet dichterbij.

Ze stonden op het punt om terug te vallen op hun oude strategie van het "volladen" van de teams om ervoor te zorgen dat ze altijd bezig waren.

1.6: Een Proces van Eliminatie

Het zou gemakkelijk zijn geweest om simpelweg onze handen in de lucht te gooien en te zeggen "dit is hopeloos". Er waren talloze excuses waar men op terug kon vallen:

- Het engineeringteam was laag gekwalificeerd (dat was zo)
- De requirements-auteurs hadden de verkeerde vaardigheden (dat hadden ze)
- De leiding verpletterde de teams met onrealistische verwachtingen (dat deden ze)
- De organisatie miste kritieke bedrijfskennis die nodig was om te functioneren (dat was zo)
- De executives lijken elkaar niet te vertrouwen (dat deden ze niet)

Al deze dingen waren waar. Toch waren er verbeteringen aangebracht op al deze gebieden en geen daarvan leek het fundamentele probleem te verbeteren. Geen van deze verbeteringen hielp de Engineeringteams, Product, management of de executive suite dichter bij hun doelen te komen.

…en dat is precies de hint naar de oplossing: Al de eerder genoemde problemen waren de problemen die mensen al konden zien.

Als de variabelen die je kunt zien geen verschil maken, moet er een variabele zijn die je **niet ziet** die dat wel doet.

Het echte probleem was het probleem waarvan niemand wist dat het bestond.

1.7: Op Zoek naar de Werkelijke Dader

Bij het zoeken naar de werkelijke dader—datgene wat de bank echt tegenhield om haar doelen te bereiken—moesten we ergens beginnen.

Een logische plek om te beginnen was bij wat alle mislukte PBI's (de meeste) met elkaar gemeen hadden.

We begonnen met het elimineren van zaken waarvan we konden zien dat ze niet gemeenschappelijk waren omdat ze sterk varieerden:

- Welk deel van het systeem: sommige PBI's raakten alleen de backend, andere alleen de frontend, weer andere raakten beide delen
- Welk team het werk uitvoerde—het leek niet uit te maken wie het werk deed, er was een grote kans op mislukking
- Welke PO het werk had geschreven—hetzelfde als met de teams

Daarna begonnen we te kijken naar de zaken die wel gemeenschappelijk waren. De lijst was niet lang, maar ook niet kort:

- De engineeringteams
- De product owners
- De leiding
- De cultuur
- De ontwikkelomgeving
- De technische practices
- De requirementsspecificatietechniek
- Het domein (financiën)
- De afhankelijke services

Veel daarvan konden ook direct worden uitgesloten. De teams, product owners, leiding, cultuur en ontwikkelomgeving waren allemaal recent verbeterd zonder echte impact op betekenisvolle output. We hadden persoonlijk geholpen bij het verbeteren van de technische- en requirementsspecificatiepraktijken en bevestigd dat die verbeteringen beklijfd waren, maar het hielp nog steeds niet.

Je kunt moeilijk het domein de schuld geven. Financiën is een van de oudste vormen van berekeningen in de geschreven geschiedenis. Het is extreem volwassen. Bovendien waren andere banken wel in staat software uit te rollen, wat de (overduidelijk vergezochte) hypothese dat banken dit simpelweg niet kunnen, overtuigend weerlegt.

De afhankelijke services konden ook niet de schuld krijgen, aangezien zij evenveel moeite hadden met veranderen als het initiatief waar we naar keken…

…maar dat zette ons aan het denken: Wat als we de oorzaken van falen zouden gaan analyseren?

1.8: Ontleding van de Kiemen van Mislukking

Eén PBI slaagde er niet in het product vooruit te helpen omdat het team, zoals ze geneigd waren te doen, iets compleet willekeurigs en bijna volledig ongerelateerd aan het verzoek deed. Dat is overduidelijk een teken dat ze het werkitem niet begrepen. Dus begrip was een grote kanshebber, ook al hadden we daar al aan gewerkt toen we hen hielpen met het adopteren van BDD.

Een andere PBI mislukte omdat ze de berekeningen verkeerd hadden. Dat is nog een aanwijzing dat begrip wellicht het kernprobleem zou kunnen zijn.

Een derde werkitem dat we analyseerden werd niet echt goed gecontroleerd door de Product Owner—ze zette er simpelweg een stempel onder toen het team zei dat het klaar was. Dat zette onze hypothese wel onder druk, maar je zou nog steeds kunnen beargumenteren dat ze niet begreep hoe het werkitem in een hoger plan paste.

Misschien. Een beetje. Als we heel hard met onze ogen knepen wanneer we er op die manier naar keken.

Toen stuitten we op een PBI die helemaal niet in het patroon paste. Het team leek het te begrijpen—hoewel er geen manier was om te verifiëren of ze het echt begrepen. Maar het maakte niet uit: ze kregen nooit de kans om op eigen kracht te slagen of te falen omdat ze tegen een afhankelijkheid aanliepen die geüpdatet moest worden en hun werk verschillende Sprints moesten uitstellen.

Zelfs als ze *niet* begrepen wat ze moesten doen, maakten ze nooit een kans met dat backlog item, dus begrip was in dat geval **niet** het probleem.

Één uitzondering is natuurlijk geen weerlegging van een bepaalde hoofdoorzaak, maar het wekte wel onze nieuwsgierigheid. We begonnen te zoeken naar meer tegenbewijzen.

En we vonden ze. Er waren werkitems die:

- Mislukten omdat het team wist dat het iets niet begreep, maar niemand een domeinexpert kon vinden om het probleem op te lossen
- Veranderden in een slechtere gebruikerservaring als workaround voor hoe de upstream services werkten
- Uitgesteld moesten worden omdat upstream afhankelijkheden niet klaar waren
- Niet afgerond konden worden omdat de testers niet op tijd testdata konden verzamelen
- Afgesloten waren maar opnieuw gedaan moesten worden omdat de vraag zelf incorrect was
- Mislukten omdat het team niet besefte hoe complex de bestaande code al was
- Simpelweg niet geschat waren
- Enorm onderschat waren[3]
- Midden in de Sprint veranderden omdat de PO eindelijk de domeinkennis kreeg die ze nodig hadden
- Na de Sprint veranderd leken (vanuit het perspectief van het team) omdat de PO en het team het nooit eens waren geworden over wat het betekende

De lijst gaat door, maar dit is genoeg voor dit verhaal.

Er kan een verband worden gelegd tussen elk van deze gevallen en “begrip” op de een of andere manier—en zeker was een gebrek aan begrip *betrokken* bij veel ervan—maar dat betekent niet dat een gebrek aan begrip de oorzaak was… vooral omdat we al wat werk hadden gedaan aan gedeeld begrip en dat niet echt had geholpen.

Toen drong het tot ons door. Er ontbrak een fundamenteler onderdeel. In de gevallen waar gebrekkig begrip een rol speelde, was dat slechts een directe oorzaak.

De onderliggende oorzaak was veel breder.

[3] We bedoelen niet alleen dat ze het verkeerd inschatten. Het leek alsof de teams misschien gewoon de waarde “3” in alle inschatting velden hadden gezet voor een reeks PBI’s.

1.9: Een Snijpunt

Het ene ding dat alle PBI's die we analyseerden op faalwijze gemeen hadden was dit: Ze waren allemaal **te vroeg gestart**.

Wanneer een werkitem faalt omdat het team wist dat ze het probleem niet begrepen en geen expert konden vinden om hen te helpen begrijpen, betekent dat dat het team een PBI startte terwijl ze wisten dat ze het probleem niet begrepen.

Wanneer een backlog item moet worden aangepast naar een slechtere ervaring vanwege hoe een upstream service functioneert, betekent dat dat het werkitem werd gestart zonder echt de impact van de upstream service te begrijpen.

Uitstel omdat een upstream afhankelijkheid niet klaar was aan het einde betekent dat er geen garantie was voor de gereedheid ervan aan het begin.

…en zo was het met alle andere gevallen: Testers hadden geen data klaar of wisten niet hoe ze eraan moesten komen voordat een PBI startte, vereisten waren niet echt gecontroleerd voordat ze werden overgedragen aan het team, de code was niet onderzocht voordat er aan het werk werd begonnen, inschatting was onvoldoende of helemaal niet gedaan, domeinkennis ontbrak, en natuurlijk was gedeeld begrip niet geverifieerd.

Het probleem bleek te zijn dat de implementatie van werkitems begon voordat die werkitems *klaar* waren.

Uit onze ervaring blijkt dat de meeste werkitems die niet succesvol worden opgeleverd falen omdat ze **niet klaar** waren toen de implementatie begon.

Dus gingen we aan de slag om The Bank daarmee te helpen.

Op dit punt zou je kunnen denken dat alleen maar zeggen dat PBI's klaar moeten zijn voldoende zou zijn. Echter blijkt dat het niet zo eenvoudig is om te implementeren. "Koop laag, verkoop hoog" is een vergelijkbaar simpel idee.

Er ontbreekt een stuk dat nodig is om het goede advies in de praktijk te brengen.

1.10: De Grote Ommekeer

We vonden het ontbrekende puzzelstuk dat hen deblokkeerde, wat op zijn beurt het initiatief deblokkeerde.

Toen we die opdracht afrondden, zaten de engineers nog steeds ruim onder de mediaan qua vaardigheden. De PO's hadden nog steeds niet de juiste vaardigheden. De cultuur was nog steeds niet verbeterd…

Toch begon het product eindelijk vooruit te gaan en werd het uiteindelijk opgeleverd.

Aan het einde van dit boek zul je weten wat het ontbrekende stuk is, en wat er nodig is om het op zijn plaats te krijgen. En je zult in staat zijn om organisaties te deblokkeren die lijken te worden tegengehouden door een onzichtbare muur.

Een Korte Opmerking over de Reikwijdte

Dit boek gaat over een zeer specifiek en wijdverbreid probleem: het gebrek aan structuur, duidelijkheid en volwassenheid bij de overdracht tussen business en engineering. Het gaat ervan uit dat er iets is gekozen voor implementatie, en richt zich op het zorgen dat het bouwwerk gebeurt met gedeeld begrip, gereedheid en traceerbare voltooiing.

De technieken in dit boek vertellen je niet wat je moet bouwen, waarom je het moet bouwen, of hoe je erachter komt of het de juiste zaak is om te bouwen. Als je organisatie geen echt productmanagement of betekenisvolle feedbackloops heeft, proberen we dat hier niet op te lossen. Wat we in plaats daarvan bieden is een manier om die hiaten zichtbaarder te maken, en om de kosten te verlagen van het ontdekken dat je het mis had.

Gebruikt in de juiste context brengt deze oplossing flow, veiligheid en duidelijkheid. Maar zoals elk systeem kan het verkeerd worden toegepast—vooral wanneer het geïsoleerd of zonder bewustzijn wordt gebruikt.

Hoofdstuk 2: De Kosten van Ontbrekende Fundamenten

Het is logisch om even stil te staan bij *hoe* ernstig dit probleem kan zijn voor sommige organisaties.

We hebben geconstateerd dat er drie belangrijke “categorieën” van problemen zijn die ontstaan door een gebrek aan gereedheid:

- Ontbrekend of onvolledig gedeeld begrip tussen en binnen Product en Engineering
- Een gebrek aan controle over wat het werkelijke doel van een PBI is en wanneer het echt klaar is
- Een gebrek aan controle over wanneer een werk item kan beginnen aan de implementatiefase van zijn levenscyclus

Daarnaast hebben we gemerkt dat het veranderen van deze zaken behoorlijk uitdagend kan zijn. Dat is logisch: verandering is moeilijk.

Oude gewoontes sterven moeilijk en nieuwe gewoontes zijn moeilijk aan te leren. In onze ervaring als consultants hebben we gemerkt dat het *extreem* gemakkelijk is voor mensen om terug te vallen op oude gewoontes en relatief moeilijk om nieuwe te ontwikkelen.

Dus je moet een mechanisme hebben dat de nieuwe gewoontes afdwingt en de oude ontmoedigt.

Om dit aan te pakken, geloven we dat er nog een ontbrekend component is van verantwoording en traceerbaarheid.

2.1: De Puzzel zonder de Afbeelding op de Doos

Heb je ooit geprobeerd een puzzel te maken zonder de afbeelding op de doos? Het kan wel, maar het is langzamer, frustrerender en vol valse starts.

Je maakt vooruitgang, breekt het dan weer af. Je twijfelt over wat waar past. Je denkt dat je aan dezelfde afbeelding werkt, totdat je beseft dat dit niet zo is.

Zo voelt softwareontwikkeling vaak aan.

De backlog is vol. De Sprint loopt. Iedereen werkt hard.

Maar zonder een gedeeld beeld van wat we aan het bouwen zijn, wordt afstemming een kwestie van geluk in plaats van een systeem.

Zonder duidelijkheid ervaren zelfs de beste teams frustratie, burn-out en het gevoel dat hun inspanningen niet worden gewaardeerd.

2.2: Een Oud Gezegde en een Harde Realiteit

Er is een reden waarom zoveel praktijken rond het schrijven van requirements, zelfs sommige engineering practices, sterk focussen op het creëren van een gedeeld begrip tussen aanvragers en implementatieteams. Niets zaait zoveel chaos als engineers die niet echt weten wat ze moeten bouwen.

"Garbage-in, garbage-out" is een wijsheid, geen platitude.

De Nederlandse taal zit vol met tegenstellingen…

- Je kunt iemands acties sanctioneren. Misschien betekent dat dat je vooraf toestemming hebt gegeven of misschien betekent het dat je achteraf afkeuring geeft.
- Je kunt iets licht afstoffen, maar als dat iets een dressoir is, betekent het dat je stof verwijdert, terwijl het bij een beignet betekent dat je stof toevoegt.
- Als je een team ondersteunt, kun je de reden zijn dat het team kan blijven functioneren of je kunt de reden zijn dat ze nergens komen.

Auto-antoniemen zijn misschien de meest opvallende voorbeelden, maar ze zijn slechts één soort dubbelzinnigheid. Sommige woorden zijn niet alleen hun eigen tegendeel, maar hebben ook vele andere mogelijk verwarrende alternatieve betekenissen.

> "In dit fragment knipte hij een bon uit een krant en bevestigde die aan het papier op zijn klembord samen met de andere knipsels terwijl een clipper op de achtergrond met een behoorlijke vaart voorbij voer."

Het is niet alleen het Nederlands. *Alle* natuurlijke talen waarvan wij weten hebben deze eigenschap.

En toch zijn het de enige die we hebben om requirements mee te specificeren.

Als gevolg hiervan vertrouwt een engineeringteam op geluk wanneer ze niet hebben *bevestigd* dat hun begrip van een requirement hetzelfde is als dat van de aanvrager. Dat wil zeggen, de best mogelijke uitkomst is dat ze de juiste interpretatie hebben gekozen en de aanvrager niet onderweg van gedachten verandert.

Die uitkomst is verre van gegarandeerd.

2.3: Enkele van de Gebruikelijke Gevolgen

Zonder bevestigd gedeeld begrip lopen teams verschillende risico's.

Over het algemeen is de meest pijnlijke uitkomst dat het team simpelweg het verkeerde bouwt.

Het moment waarop ze dat ontdekken kan worden beïnvloed door verschillende aspecten van hun proces. Een gezonde implementatie van Scrum kan bijvoorbeeld dit soort misverstanden zeer vroeg in de uitvoering detecteren, terwijl een Waterval-proces een grote kans heeft om zo'n ontdekking maanden uit te stellen.

Definitie: Waterval

Een sequentieel softwareontwikkelingsmodel dat wijdverbreid werd in de late 20e eeuw. Het werd voor het eerst beschreven in een 1970 paper door Winston W. Royce, die ontwikkeling illustreerde als een reeks cascade-stappen—requirements, ontwerp, implementatie, testen, enzovoort—waarbij elke stap in de volgende vloeit zoals een waterval. Hoewel Royce het model presenteerde als een voorbeeld van wat je *niet* moet doen, adopteerde de industrie het als een blauwdruk voor grootschalige ontwikkeling.

Waterval staat ook bekend om het groeperen van vergelijkbaar werk in grote, opeenvolgende fasen—een kenmerk dat wordt aangeduid als **grootschalige ontwikkeling**. Deze batchverwerking garandeert vrijwel **laat leren**: teams ontvangen pas veel later in het proces feedback op eerdere beslissingen. Fouten die laat worden ontdekt zijn kostbaarder om te herstellen. Agile beoefenaars bekritiseren Waterval om deze reden en geven de voorkeur aan kleinere, iteratieve cycli die vroegere ontdekking en koerscorrectie mogelijk maken.

Niettemin zal een team dat werkt met het "verkeerde" (eigenlijk verschillende) begrip van een vereiste op een gegeven moment worden geconfronteerd met de "juiste" (ook verschillende, eigenlijk) vereiste. Opnieuw bepaalt de gezondheid van de organisatie welke vorm deze confrontatie aanneemt en welke impact deze heeft, maar het gebeurt bijna altijd.

In de meeste gevallen leidt dit tot een vorm van herwerk. De aanvrager (meestal Productmanagement) zal om wijzigingen moeten vragen om van wat de implementatieteams hebben gebouwd te komen tot wat hij werkelijk wilde.

Een andere veel voorkomende manifestatie is dat de aanvrager het team blijft houden aan zijn oorspronkelijke begrip van wat hij heeft gevraagd.

Teams kunnen dit gemakkelijk interpreteren als een productmanager die van gedachten verandert. Erger nog, het kan belanghebbenden zelfs *uitnodigen* om de gewoonte te ontwikkelen hun mening te veranderen - werktaken tegenhouden tot alles *precies goed* is, teams uitputten en vooruitgang verbergen voor het hogere management.

2.4: Wanneer is een Puzzel Klaar?

Als we terugkeren naar onze puzzel-analogie, denk dan eens na over deze vraag: Wat betekent het om klaar te zijn met een puzzel?

Een naïeve puzzelbouwer, zoals Max, zou simpelweg zeggen "alle stukjes zijn aan de juiste buren bevestigd met de afbeelding naar boven."

Een ervaren puzzelbouwer, zoals Luniel, weet echter dat er meer bij komt kijken.

Misschien leg je de puzzel gewoon voor de lol. Je maakt hem af, bekijkt hem even en haalt hem dan weer uit elkaar om terug in de doos te doen.

Maar misschien wil je hem inlijsten en aan de muur hangen. In dat geval zijn er extra dingen die gedaan moeten worden:

1. Hem op een kunstplaat leggen
2. Hem naar een lijstenmaker brengen
3. Wachten tot het inlijsten klaar is
4. Hem terugbrengen naar de ophanglocatie
5. Hem aan de muur hangen of anderszins tentoonstellen

Begrijpen dat dit onderdeel is van het werk is noodzakelijk om een puzzel correct af te maken. De voor de hand liggende reden is dat je weet hoeveel werk erbij komt kijken. Het is meer werk om al die extra stappen te doen dan om hem gewoon uit elkaar te halen en op te bergen.

Het gaat echter nog dieper dan dat. Stel je het volgende scenario voor…

Je hebt je puzzel afgemaakt met de bedoeling om hem te laten inlijsten, je hebt hem laten liggen, maar je bent vergeten aan iemand anders in huis te vertellen dat je van plan bent hem te laten inlijsten. Die persoon komt langs en ziet dat de puzzel af is op een plek die hij of zij nodig heeft. Dus halen ze hem uit elkaar en stoppen hem terug in de doos, waarbij ze de nederlaag uit de kaken van de overwinning rukken.

Er is nog een subtielere reden: Hoe je van plan bent de puzzel af te maken beïnvloedt welke stappen je eerder in het proces wilt nemen. Je moet bijvoorbeeld een bordje maken met "Gelieve niet uit elkaar halen!"

Het is ook vermeldenswaard dat je misschien een bordje wilt hebben in het geval dat je een puzzel voor je eigen plezier maakt, om ervoor te zorgen dat anderen zich er niet mee bemoeien door de puzzel voor jou af te maken.

Je moet er ook voor zorgen dat je de puzzel op de juiste ondergrond in elkaar zet. Als je een puzzel van duizend stukjes op je glazen salontafel in elkaar zet en hem dan probeert over te brengen naar een kunstplaat, zal de verplaatsing veel riskanter en arbeidsintensiever zijn dan wanneer je de puzzel direct op de kunstplaat had gemaakt.

Dit loopt mooi parallel met softwareontwikkeling.

Je moet echt weten wat 'klaar' betekent, zodat je niet verrast wordt door de hoeveelheid werk die erbij komt kijken, er geen onenigheid over bestaat aan het einde, en je de nodige voorbereidende stappen kunt nemen om een werktaak soepel en effectief af te ronden.

2.5: Impact op Teams

Als je geen voldoende rigide begrip hebt van wat 'klaar zijn' betekent voor een bepaalde werktaak, loop je een aantal risico's.

We gebruiken het woord "risico" hier losjes, aangezien het meer garanties zijn.

Engineeringteams in deze situatie ontdekken vaak dat ze het intern niet eens met elkaar eens zijn over hoe het voltooien van een werktaak eruitziet. Het is niet ongewoon dat programmeurs en testers halverwege een Sprint moeten uitvechten wat een vereiste werkelijk *betekent*. Zelfs twee programmeurs of twee testers kunnen last hebben van dezelfde meningsverschillen.

Bovendien zijn ontwikkelteams vaak gefocust op het werk dat ze het meeste doen (programmeren en testen). Dit betekent dat ze gemakkelijk andere soorten werk vergeten die ze moeten doen, zoals documentatie, externe reviews, training van andere teams (*bijv.* support), voorbereidende stappen ter ondersteuning van deployment of release, en goedkeuringen van andere afdelingen.

Wanneer het uiteindelijk duidelijk wordt dat dit "extra" werk moet worden gedaan, worden ze erdoor overvallen—meestal moeten ze stoppen met waar ze mee bezig waren en van context wisselen om terug te gaan en werk af te maken waarvan ze dachten dat het al klaar was.

Aanvragers van werk kunnen gemakkelijk werk onnodig open houden. Soms met de beste bedoelingen—zoals proberen een team verantwoordelijk te houden voor de "echte" vereiste. In andere gevallen gebeurt dit omdat Product Owners (bijvoorbeeld) eraan gewend raken om een PBI naar eigen goeddunken open te kunnen houden, dus gebruiken ze het om op het laatste moment extra functionaliteit in een item te forceren. Soms doen ze het zelfs omdat ze halverwege van gedachten zijn veranderd over wat er gedaan moet worden.

Dit kan extreem demotiverend zijn voor een engineeringteam. De meeste softwareontwikkelaars en testers willen het gevoel hebben dat ze vooruitgang boeken. Als ze constant te horen krijgen dat wat ze deden verkeerd was, zullen ze waarschijnlijk wat van hun energie verliezen.

Sommige teams gaan zelfs zo ver dat ze niet eens meer controleren of wat ze deden goed of fout was. Ze sluiten gewoon een werkitem af en vragen om "krediet" zodat ze "goede cijfers kunnen laten zien".

2.6: Het Risico van Te Weinig of Te Veel Doen

Een risico van het niet goed definiëren van "klaar" voor elk werkitem is dat de organisatie denkt dat werk klaar is wanneer dat niet zo is, of niet zal beseffen dat het klaar is wanneer dat wel zo is.

De slechtst mogelijke uitkomst is vaak dat het verkeerde in productie komt en niemand weet dat dit is gebeurd. Als het team een onjuist begrip heeft van wat "klaar" betekent en op basis van dat slechte begrip uitrolt, kunnen de gevolgen catastrofaal zijn.

Defecten en klantenontevredenheid zijn al erg genoeg, maar dit zou ook kunnen leiden tot veel ernstiger problemen:

- Gegevensverlies of -corruptie
- Beveiligingskwetsbaarheden
- Systeem uitval of verlies van toegang
- Een vermindering van marktaandeel
- Regelgevingsovertredingen

De lijst gaat maar door en elk potentieel probleem is erger dan het vorige.

Soms is het probleem dat je niet klaar bent maar denkt dat je dat wel bent. Andersom kan net zo gevaarlijk zijn. Wanneer engineers niet weten waar de finish ligt, neigen ze ernaar om te "vergulden" (extra functies toe te voegen). Ze doen dat misschien om "de functie mooi te maken", maar ze doen het mogelijk ook omdat ze hopen dat een verhoogd aantal functies hen een grotere kans geeft om het doel te raken.

Al dit extra werk, evenals het bijbehorende herwerk, accumuleert tot een enorme hoeveelheid verspilde tijd, inspanning en geld. Het zorgt ervoor dat opleverdata verschuiven en reputaties beschadigd raken.

Bovendien is er nu een steeds groter wordende kans dat een fout daadwerkelijk leidt tot een *Terminator*-achtige opstand tegen de mensheid door de machines. We schreven daar twintig jaar geleden over als grap. Nu is het een reële mogelijkheid.

We hebben deze vraag zelfs aan een van de meest prominente AI's gesteld, en dit is wat het zei:

> "AI breidt zich sneller uit dan iemand had verwacht, maar het doet dit bovenop broze systemen, vage vereisten en productorganisaties die niet kunnen traceren waarom ze hebben gebouwd wat ze hebben gebouwd. Dat is geen technisch probleem; het is een duidelijkheidsprobleem.
>
> Hoe meer ruis AI genereert, hoe gevaarlijker het is om snel te bewegen zonder structuur. Wanneer teams bouwen op mist, versterkt AI alleen maar de puinhoop. Maar wanneer teams bouwen op signaal—op gedeeld

> begrip, gedragsspecificiteit en versiebeheerde productkennis—wordt AI een versneller in plaats van een aansprakelijkheid."

2.7: Waar Bouw Je een Puzzel?

Laten we de puzzel-bouw analogie nog één keer doortrekken.

Kun je overal een puzzel in elkaar zetten? Als je een puzzel van 4.000 stukjes hebt die uitkomt op bijna anderhalve meter in één dimensie en meer dan een meter in de andere, kun je niet zomaar willekeurig een plek kiezen en beginnen met in elkaar zetten. Niet zonder ernstige complicaties tegen te komen voordat je klaar bent.

Een grote puzzel zoals die heeft zowel tijd als ruimte nodig. Je moet de ruimte toewijzen en een manier vinden om ervoor te zorgen dat de staat van de puzzel behouden blijft in de tijd.

Als je je puzzel begint te bouwen op een klein bijzettafeltje dat te klein is, zul je hem niet kunnen afmaken zonder hem naar een andere locatie over te brengen. Die overdracht zal extreem moeilijk zijn vanwege de delicate staat van de puzzel.

Als je een willekeurige plek in de gang kiest die groot genoeg is, zullen mensen er óf overheen lopen óf erdoor gehinderd worden, dus het behoud kan niet worden gegarandeerd zonder significante impact op het functioneren van je huishouden.

Als je op een tekenbord begint te werken, maar het bord is niet groot genoeg, kun je wel de staat van wat je hebt gedaan bewaren, maar je zult de puzzel niet kunnen afmaken zonder een vorm van verplaatsing.

Als kleine kinderen eerder op de puzzel hebben gekauwd, kun je het beste de stukjes tellen… want het is beter om één keer tot 3999 te tellen en te beseffen dat je hem nooit af zult maken, dan wie weet hoeveel tijd te investeren in het bijna voltooien van een puzzel die je nooit zult kunnen afmaken.

Er is een hele lijst van dingen die gedaan moeten worden voordat je begint met het maken van je puzzel. Het uitvoeren van de dingen op de lijst garandeert geen succes, maar ze *niet* doen zorgt vrijwel zeker voor mislukking of ernstige complicaties.

Hetzelfde geldt voor softwareontwikkeling, maar met een hogere mate van complexiteit.

2.8: Wanneer Begint de Implementatie?

In essentie kan het moeilijk zijn om te bepalen wanneer een PBI klaar is voor implementatie zonder een goede definitie daarvan.

Denk er eens over na: Hoe *weet* je dat?

Ga je alles steeds opnieuw door totdat je besluit dat het tijd is?

Beslist iemand dat op gevoel?

Gebeurt het automatisch aan het begin van een iteratie?

> We hebben veel teams werk in Sprints zien duwen dat nog lang niet klaar was voor implementatie, alleen omdat ze deadlines hadden. Er zijn enkele hardnekkige ideeën over Scrum en Agile in het algemeen die mensen ertoe aanzetten dit te doen:
>
> - Je moet al je vereisten voor Sprint *N* ontwikkelen in Sprint *N-1*
> - Je moet "gewoon beginnen" en onderweg omgaan met wat er misgaat

Dit is eigenlijk een spiegelbeeld van de "Hoe weet je wanneer het klaar is?" kwestie eerder genoemd en het heeft vergelijkbare gevolgen. Mensen wachten misschien te lang met beginnen omdat ze niet weten dat een werkitem klaar is, en ze beginnen misschien te vroeg omdat ze niet weten dat het nog niet klaar is.

2.9: Een A-Team zonder Gereedheid

Het niet begrijpen van wat er nodig is om een werkitem klaar te hebben heeft verschillende schadelijke effecten.

Een voor de hand liggende manier waarop een increment van werk niet gereed kan zijn, is een onvolledige, ontoereikende of ontbrekende Definition of Done (DoD). Dat leidt tot alle problemen die we al hebben genoemd die samenhangen met het niet hebben van een Definition of Done.

Dit is echter niet het *enige* aspect van gereedheid. Er zijn talrijke andere behoeften waaraan moet worden voldaan voordat de implementatie begint: schatting, risicobeoordeling en het verzamelen van testgegevens zijn slechts enkele veelvoorkomende voorbeelden.

Zonder deze behoeften te kennen en eraan te voldoen, kan een werkitem veel meer kosten dan nodig is. Neem een team (Het A-Team) dat afhankelijk is van een API die door een ander team (het Andere Team) wordt ontwikkeld. Als het A-Team een hoop aannames doet over hoe de API van het Andere Team zal functioneren en codeert op basis van die aannames, kan er aanzienlijk herwerk nodig zijn wanneer ze erachter komen dat het Andere Team werkt aan een realiteit die niet overeenkomt met de aannames van het A-Team. Met andere woorden, het A-Team waagde een gok en miste.

Al dat herwerk komt voort uit het feit dat de API niet klaar was om door het A-Team gebruikt te worden.

Soms leidt een onvervulde afhankelijkheid niet tot herwerk, maar zelfs in die gevallen kan het nog steeds vertragingen veroorzaken. Stel je voor dat het A-Team en het Andere Team het eens waren over hoe de API zou moeten werken en alles ging volgens plan, maar het Andere Team deed er simpelweg langer over dan verwacht. Als gevolg daarvan kon het A-Team hun werk niet goed testen tegen de tijd dat het klaar had moeten zijn en moesten ze hun deadline uitstellen.

2.10: Niet Letten op Planning en Beschikbaarheid van Resources

Soms kunnen de problemen zo eenvoudig zijn als planning of resourcebeheer. Sommige werkitems hebben specifieke teamleden nodig. Als dat teamlid over een paar dagen op vakantie gaat, is het waarschijnlijk niet het juiste moment om te beginnen met de PBI die niet zonder zijn deelname kan worden voltooid.

> We horen mensen vaak zeggen dat het niet zo zou moeten zijn, maar dat is het vaak wel, hoe dan ook. “Verwisselbare mensen” is een luchtspiegeling.

Hetzelfde geldt voor niet-menselijke middelen. Als je serverbronnen nodig hebt om een belastingtest uit te voeren, moet je waarschijnlijk eerst zorgen dat die bronnen daadwerkelijk beschikbaar zijn voordat je met de belastingtest werktaak begint. Anders is je beste scenario aanzienlijke vertragingen en zul je waarschijnlijk andere teams/medewerkers verstoren terwijl je probeert te improviseren om te voorzien in wat je nodig hebt.

Een andere manier waarop valse starts kunnen mislukken is wanneer een team niet de vereiste vaardigheden heeft om het werk af te ronden. Soms is dat een interne kwestie -

zoals wanneer een teamlid training nodig heeft voor een nieuw systeem of onderzoek moet doen naar een nieuwe API. Andere keren is het een planningskwestie, zoals wanneer je een UX- of database-expert moet lenen uit een pool van geschoolde medewerkers. Het kan zelfs een aannamekwestie zijn waarbij het team een expert nodig heeft en bepaalde soorten werk niet effectief kan afronden zonder hem.

2.11: Gevolgen van Andere Soorten Valse Starts

We hebben teams gezien die zich ertoe verbinden werktaken binnen Sprints af te ronden en de codering relatief snel doen, maar nog steeds niet in staat zijn om het testen af te ronden. Dit is op zich misschien niet verrassend, maar de reden is ongewoon: het testteam had iets nodig (zoals testgegevens) dat ze niet hadden verzameld voordat de Sprint begon, en het verzamelen van die gegevens bleek moeilijker of tijdrovender dan ze hadden verwacht.

Als gevolg hiervan moesten de werktaken worden meegenomen naar de volgende Sprint, simpelweg omdat het team niet had gezorgd dat ze echt klaar waren om het binnen de toegewezen tijd af te ronden voordat ze begonnen.

Teams beginnen soms met het implementeren van werktaken terwijl ze nog open vragen hebben. Sterker nog, veel mensen lijken te denken dat het hen “meer Agile” maakt wanneer ze dat doen.

Dit kan enorme hoeveelheden herwerk, verrassingen of vertragingen veroorzaken. Als het antwoord op de open vraag een aanname schendt die was gemaakt, moet al het werk dat op die aanname was gebaseerd worden aangepast. Als de open vraag niet is beantwoord tegen de tijd dat het item zou moeten worden afgesloten, dan moet het item ofwel worden afgesloten terwijl het mogelijk niet klaar is, of open blijven totdat de vraag is beantwoord.

Het kan zijn dat het team een interne afhankelijkheid heeft - een defect dat moet worden opgelost, een voorgaande taak die moet worden afgerond, enzovoort. Als dit niet goed wordt bijgehouden, kan het dezelfde problemen veroorzaken als een onvervulde externe afhankelijkheid, met de extra verleiding om van context te wisselen en het op te lossen.

2.12: Cumulatieve Kosten

Natuurlijk veroorzaken zulke problemen vertragingen, herwerk en teleurgestelde verwachtingen, maar de nadelen eindigen daar niet.

Bovenop de verspilling door herwerk, zorgt dit er meestal voor dat projecten achterop raken. Als teams frenetiek proberen backlogitems af te sluiten en nooit echt duidelijk hebben wat er nodig is om echte vooruitgang te boeken, hebben de dingen die daadwerkelijk *moeten* gebeuren de neiging om tussen wal en schip te vallen.

Vaak, maar niet altijd, leidt dit tot verhoogde druk om te leveren. Naarmate projecten steeds verder achter op schema raken, kan het hoger management proberen het weer op de rails te krijgen door mensen te vragen sneller te werken. Dat vertaalt zich onvermijdelijk naar langere werkuren.

Dit leidt op zijn beurt vaak tot erosie van vertrouwen en verzuring van de bedrijfscultuur. Relaties die collaboratief zouden moeten zijn, worden vijandig. Mensen die zouden moeten samenwerken om de beste, snelste oplossingen te vinden, verleggen hun energie naar het aantonen dat, wanneer er onvermijdelijk iets misgaat, het niet hun schuld was.

In de razende jacht op features en afgesloten werktaken, merken teams vaak dat ze concessies doen. Dat betekent eigenlijk dat ze de kwaliteit (vooral codekwaliteit) laten lijden. Dat betekent op zijn beurt dat ze toekomstige productiviteit inruilen voor de illusie van vooruitgang in het heden.

Naarmate de werkomstandigheden steeds onaangenamer worden, begint belangrijk talent zich terug te trekken of zelfs om zich heen te kijken.

Organisaties die zich zo gedragen, “eten hun zaaigoed op”, als het ware, op meer manieren dan één. De codebase wordt minder onderhoudbaar en de mensen die het hadden moeten onderhouden worden allemaal weggejaagd.

Als er een voordeel is, is het voor ons onzichtbaar.

Hoofdstuk 3: Introductie van Requirements Maturation Flow (RMF)

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

3.1: Wat RMF Niet Is

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

3.2: Wat RMF Wel Is

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

3.3: Incrementele Adoptie wordt Ondersteund en Aanbevolen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

3.4: RMF 1

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

3.5: RMF 2

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

3.6: RMF 3

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 4: Is Het Agile?

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

4.1: “Individuen en Interacties”, “Werkende Software”

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

4.2: Samenwerking met de Klant

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

4.3: Reageren op Verandering

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

4.4: Transparantie

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

4.5: Past bij Proces, Consistent met Agile

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Deel II: Ruimte Creëren voor Gereedheid

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 5: De Eerste Uitbreiding

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

5.1: Gereedheidswerk is *Werk*

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

5.2: Gereedheidswerk Naturaliseren

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

5.3: Een Illustratief Incident

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

5.4: Wederzijdse Impact

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

5.5: De Functie van RMF 1

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 6: Waarom Doen Mensen Dit Niet?

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

6.1: Voorbereidingswerk als Tweederangsburger

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

6.2: Een Allergie voor Niet-Productief Werk

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

6.3: Dus Het Werd Begraven

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

6.4: De Invloed van Projectmanagement

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

6.5: Het Patroon

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

6.6: Projecten en Schatten

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

6.7: Hoe de Niet-Schatting Schattingen het Voorbereidingswerk Beïnvloeden

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

6.8: Snelheid Meten, Niet Velocity

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

6.9: Slechte Metingen, Slechte Resultaten

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

6.10: Waar de Schuld Niet Ligt

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 7: Expliciet Gereedheidswerk (RMF 1)

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

7.1: Integratie met het Synapse Framework™

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

7.2: Anatomie van RMF 1

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

7.3: Gedraging: Reserveer Capaciteit voor Samenwerking

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

7.4: Artefact: Het Voorbereidingswerkitem

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

7.5: Activiteit: Het Samenwerkingsoverleg

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

7.6: Gedrag: Blijf Samenwerken Tot Gedeeld Begrip is Bereikt

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

7.7: Gedrag: Bevestig Altijd Gedeeld Begrip

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

7.8: Hoe RMF 1 de Werkstroom Verandert

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 8: Effecten van RMF 1

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

8.1: Leven voor RMF 1

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

8.2: Voor: Tijd besteed aan begrip

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

8.3: Na: Tijd besteed aan begrip

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

8.4: Het leven na het adopteren van RMF 1

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

8.5: Fundamenteel

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 9: RMF 1 in de Praktijk Brengen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

9.1: Educatie

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

9.2: Minimumvereisten per Teamtype

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

9.3: Overeenstemming

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

9.4: Voorbereiding

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

9.5: Pilot

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

9.6: Uitrol

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

9.7: Opvolging

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

9.8: Succes Claimen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

9.9: Waakzaam Blijven

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

9.10: Hoe zit het met het “Hoe”?

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

9.11: Tijd om het te Realiseren!

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Deel III: Poortwachten van Werkafronding

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 10: De Volgende Behoefte

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

10.1: Ruimte voor Interpretatie

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

10.2: Verkleinen van de Interpretatieruimte

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

10.3: Een Derde Optie: Geen “Speelruimte”

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

10.4: Mogelijke Impact op Voltooiing

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

10.5: Potentiële Impact op Uitvoering

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

10.6: Voorgesteld Alternatief: Laat Geen Ruimte voor Misinterpretatie

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

10.7: Voordelen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

10.8: Over Angst voor Analyseverlamming

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

10.9: De Volgende Behoefte: Op Maat Gemaakte Definitions of Done

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 11: Wat Mensen Meestal Doen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

11.1: Als Het Zo Geweldig Is, Waarom Doen Mensen Dit Dan Niet?

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

11.2: De Opleiding-naar-Coaching Pipeline

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

11.3: Coaching Overload

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

11.4: Één Manier Waarop Mensen DoD Doen: Helemaal Niet

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

11.5: Alleen Acceptatiecriteria

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

11.6: Globale Definition of Done

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

11.7: Geen Tanden

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

11.8: Samenvatting: De Term “DoD” Komt Vaker Voor dan Echte Definitions of Done

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 12: Een Definition of Done definiëren

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

12.1: Over slechts één werkitem

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

12.2: Voltooiing

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

12.3: Nauwkeurigheid

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

12.4: Structuur van een DoD

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

12.5: Specificaties

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

12.6: Technische Uitgangscriteria

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

12.7: Product Acceptatiecriteria

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

12.8: Meerdere Onderdelen, Één Gate

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

12.9: Voorbeeld

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

12.10: Aansluiting op Je Proces

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

12.11: Samenvatting

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 13: Specifieke Definition of Done (RMF 2)

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.1: Principe: Elk Werkitem is Uniek

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.2: Gedraging: Onderhoud Één of Meer DoD-sjablonen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.3: Activiteit: Het definiëren van het DvK-sjabloon

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.4: Onderhoud en verbeter het DvK-sjabloon

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.5: Meerdere DvK-sjablonen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.6: Gedrag: Gebruik sjablonen als startpunten voor Definities van Klaar

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.7: Gedrag: Instemmen met op maat gemaakte Definities van Klaar

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.8: Activiteit: Een Definition of Done definiëren voor een werkitem

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.9: Nog een uitbreiding op de workflow

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.10: Gedrag: Breng de DoD tot volwassenheid voor het starten van de implementatie

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.11: Activiteit: Offline Analyse om een PBI's DoD te Laten Groeien

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.12: Volwassenheid Toevoegen aan de Flow

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.13: Gedrag: Voltooidheid Bijhouden in Werkitems

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.14: Voortgangsbewaking toevoegen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.15: Gedrag: Werk reguleren op basis van gereedheid

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.16: Activiteit: De DoD gebruiken om gereedheid te bepalen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.17: Hoe het Poortwachtersysteem Past

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

13.18: Samengevat

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 14: Leven met RMF 1 & 2

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

14.1: Kosten

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

14.2: Tijdlijnen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

14.3: Impact op het Implementatieteam

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

14.4: Impact op de Product Owner

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

14.5: Impact op het Leiderschap

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

14.6: Samenvatting

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 15: RMF 2 installeren

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

15.1: Betrokkenheid van Belanghebbenden

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

15.2: Detailniveau

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

15.3: Werkafspraken

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

15.4: Initieel Werk

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

15.5: Uitrol

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

15.6: Samenvatting

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Deel IV: Faseringsimplementatie

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 16: De Laatste Vereiste

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

16.1: Het Was Er De Hele Tijd Al

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

16.2: De Kracht van Timing

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

16.3: Het Omdraaien

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

16.4: Risico's & Kosten

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

16.5: De Waarde van Wachten Tot Je Er Klaar Voor Bent

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

16.6: Een Extra Voordeel

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

16.7: De Probleemstelling

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

16.8: De Noodzaak

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

16.9: Samenvatting

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 17: Achtergrond bij Definition of Ready

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

17.1: Lean's Toestemming om te Werken

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

17.2: Kanban's Kolomtoegangscriteria

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

17.3: Scrum en andere Agile Proces Apocriefe Verhalen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

17.4: Surfen > Coderen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

17.5: Punt-oplossingen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

17.6: We Zijn Er Klaar Voor Om Klaar Te Zijn

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 18: Een Definitie van Gereedheid Definiëren

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

18.1: Doel

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

18.2: Op Maat Gemaakt

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

18.3: Anatomie

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

18.4: Voorbeeld

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

18.5: Overeenstemming

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

18.6: Poortwachter

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

18.7: Samenvatting

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 19: Op Maat Gemaakte Definitie van Gereedheid (RMF 3)

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.1: Nog een Controlepunt in het Proces

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.2: De Structuur van een Definitie van Gereedheid

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.3: Product Uitgangscriteria

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.4: Engineering Ingangscriteria

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.5: Geen Kwaad in Duplicatie

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.6: Gedrag: Onderhoud Één of Meer DoR Templates

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.7: Waarom een Definition of Ready Template Hebben?

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.8: Activiteit: Het Definiëren van de DoR Template

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.9: DoR Templates Onderhouden in de Tijd

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.10: Templates zijn Slechts een Uitgangspunt

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.11: Gedrag: Instemmen met Maatwerk Definitions of Ready

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.12: Activiteit: Een Definition of Ready voor een Werkitem Definiëren

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.13: Gedrag: Items Gereed maken voor Implementatie

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.14: Voorwaarden Buiten de Controle van het Team

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.15: Gedrag: Volg Gereedheid in Werk-items

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.16: Gedrag: Werk Controleren op Gereedheid

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.17: Activiteit: De DoR Gebruiken om Gereedheid te Bepalen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

19.18: Samengevat

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Deel V: Synthese

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 20: De Meeste Deadlines Zijn Niet Belangrijk

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

20.1: Echte Deadlines

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

20.2: Willekeurige Deadlines Zijn Niet Belangrijk

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

20.3: Luchtvaartmaatschappijen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

20.4: De Oorsprong van Technische Schuld

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

20.5: Marketing- en Sales-deadlines

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

20.6: Projectdeadlines

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

20.7: Er Is Een Andere Manier

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

20.8: Willekeurige Deadlines zijn Niet Noodzakelijk

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

20.9: Willekeurige Deadlines Moeten Worden Afgeschaft

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

20.10: Echte Deadlines Blijven een Factor

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

20.11: *Pis Aller*[1]

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

[1] Een laatste redmiddel, oorspronkelijk uit de Franse taal.

20.12: Conclusie

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 21: Competentie 1: Requirementsverfijningsproces

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

21.1: Zoals Beneden, Zo Boven

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

21.2: Principe: Transparantie in Alle Noodzakelijke Werk

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

21.3: Gedraging: Verantwoordelijkheid Reist met het Werk

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

21.4: Gedrag: Zichtbaar Bijhouden van de Status van Requirements

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

21.5: Gedrag: Maak Al het Werk Zichtbaar dat Verband Houdt met Gereedheid en Implementatie

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

21.6: Activiteit: Gereedheidswerk

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

21.7: Gedrag: Maak Al het Werk Zichtbaar dat Nodig is om een Werkitem te Voltooien

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

21.8: Gedraging: Geef voorrang aan gereedheid boven deadlines

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

21.9: Conclusie

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 22: Hoe Werk en Informatie Stroomt in Scrum met RMF

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

22.1: Inleidende Opmerkingen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

22.2: Het Vastleggen en Voorbereiden van de Initiële Vereiste

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

22.3: Opstarten van Readiness Werk

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

22.4: Planning en Uitvoering van Readiness Werk

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

22.5: Beoordeling van Readiness Resultaten

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

22.6: Planning en Voltooiing van Implementatie

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 23: De Impact van RMF

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

23.1: De Levenscyclus van een Requirement

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

23.2: Voorbeeldstroom van Informatie en Werk

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

23.3: Voorheen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

23.4: Daarna

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

23.5: Voordelen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 24: Overstappen op RMF

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

24.1: Adoptiepatroon

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

24.2: Implementatie in een Scrum Workflow

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

24.3: Andere Frameworks

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

24.4: Belangrijke Weerstand: Gereedheidswerk Items

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

24.5: De Grote Verschuiving: Denkwijze

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

24.6: Advies voor Verandering

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

24.7: Conclusie

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoofdstuk 25: Het Ligt Bij Jou

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

25.1: Samenvatting

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

25.2: Nu Is Het Jouw Beurt

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Deel VI: Bronnen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Appendix A: Scrum Is Niet het Probleem

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Wat is Scrum?

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Frameworks

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Scrum Richt Zich op Project- en Werkbeheer

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

De Dwaasheid van Scrum Behandelen als een Product Management Framework

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Geen Voorgeschreven Mechanisme voor het Ontwikkelen van Requirements

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Geen Investering van Engineering Expertise in Requirements Ontwikkeling

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Antipatronen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Uitbreiding Vereist

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Toevoegen aan Scrum

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Bijlage B: Het Synapse Framework™

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Wat Synapse Omvat

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

De Drie Meesterschappen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Hoe Synapse Wordt Geadopteerd

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Het Samenvoegen van Twee Frameworks

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

De Structuur van het Synapse Framework

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Organisatorische Meesterschappen

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Organisatorische Competenties

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Organisatorische Gewoonten

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Structuur van het Synapse Framework

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Anatomie van een Praxis

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Het Belang van Volgorde

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

De Impact van Synapse op dit Boek

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Appendix C: Veelvoorkomende Bezwaren en Obstakels bij RMF 1

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Veelvoorkomende Bezwaren

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Veelvoorkomende Obstakels

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Appendix D: DoD Startcriterialijsten

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Engineering Exit Criteria Startlijst

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Startlijst voor Producttoegangscriteria

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Appendix E: DoR Starter Criteria Lijsten

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Product Exit Criteria

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Technische Toelatingscriteria

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Sprint Toegangscriteria

Deze inhoud is niet beschikbaar in het voorbeeldboek. Het boek kan worden gekocht op Leanpub via https://leanpub.com/ready-nl.

Index