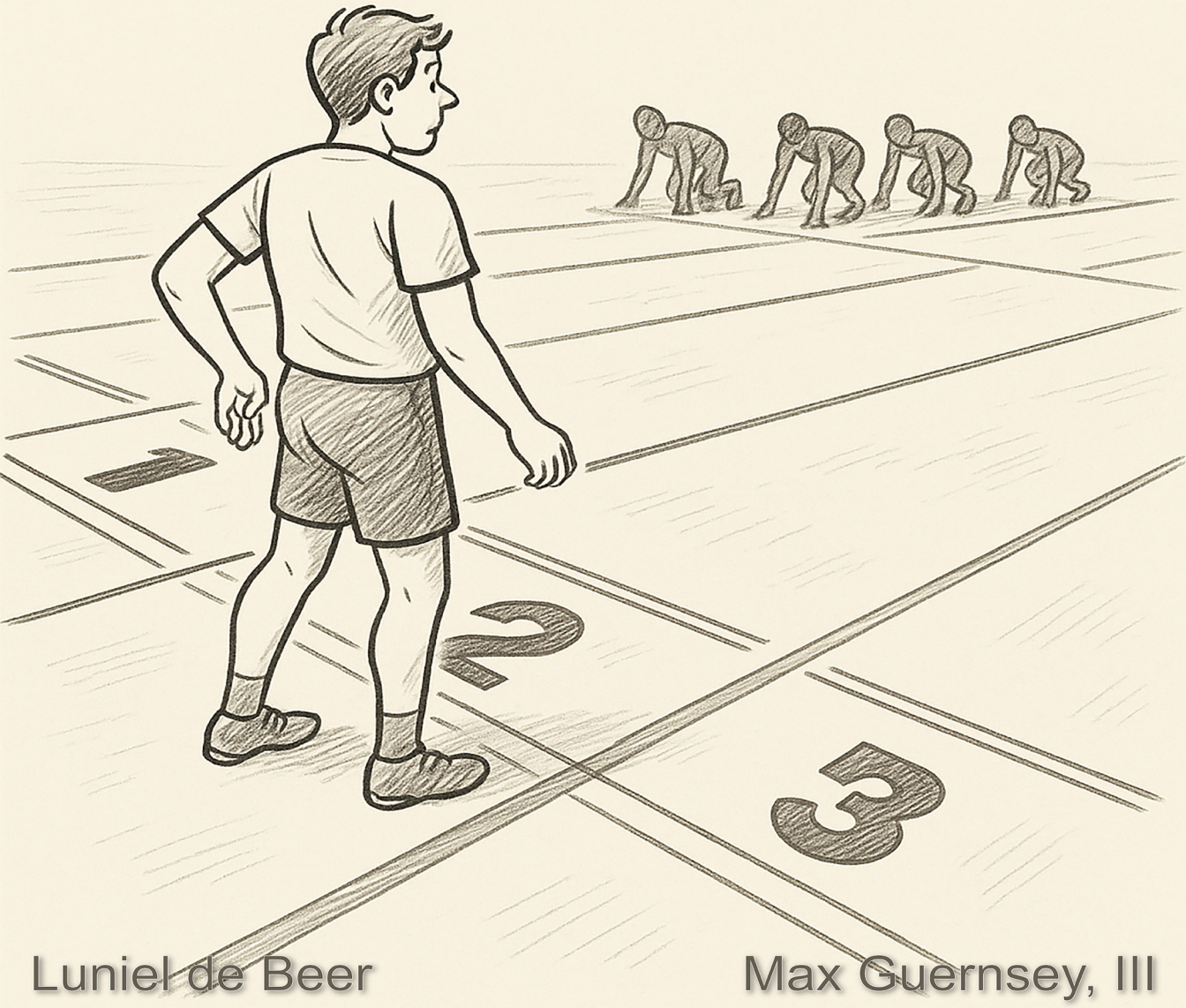
READY

WHY MOST SOFTWARE PROJECTS FAIL AND HOW TO FIX IT



Luniel de Beer

Max Guernsey, III

Dansk Udgave

Tweet Denne Bog!

Hjælp venligst Luniel de Beer og Max Guernsey, III ved at sprede ordet om denne bog på Twitter!

Det foreslåede tweet for denne bog er:

Jeg har lige købt Ready — en bog til softwareledere og teams, der ønsker at eliminere omarbejde, overførsel og uoverensstemmelser i softwarelevering. #CodeReady

Det foreslåede hashtag for denne bog er #CodeReady.

Find ud af hvad andre siger om bogen ved at klikke på dette link for at søge efter dette hashtag på Twitter:

#CodeReady

Ready (Dansk Udgave)

Hvorfor de Fleste Softwareprojekter Mislykkes og Hvordan Vi Løser Det

Luniel de Beer og Max Guernsey, III

Denne bog kan købes på https://leanpub.com/ready-da

Denne version blev udgivet den 2025-10-22

Dette er en Leanpub bog. Leanpub giver forfattere og udgivere mulighed for at bruge Lean Publishing-processen. Lean Publishing er handlingen at udgive en bog under udvikling ved hjælp af enkle værktøjer og mange iterationer for at få læserfeedback, justere indtil du har den rigtige bog og opbygge momentum, når du har det.

Til minde om Johann van Aardt, som genkendte min passion, introducerede mig til egentlig programmering og hjalp mig med at finde vej til et nyt hjem. Og til mine forældre, hvis urokkelige støtte—fra min første kunde til mit første hjem i USA—gjorde alt dette muligt.

—Luniel

Til min familie, med hvem solen står op og går ned.

—Max

Indhold

Om Denne Bog

Ready er en bog for alle involverede i softwareudvikling, som er trætte af **underleverance**, **kronisk omarbejde** og **uklare krav**.

Du har måske allerede forsøgt at investere i teamets udførelseskompetencer, forbedret implementeringen af din procesramme eller renoveret koden og har stadig brug for mere forbedring.

Dette skyldes, at den primære begrænsning for de fleste softwareudviklingsteams ikke er teamkompetencer, men kravmodenhed. **Selv modne teams med de rette kompetencer kæmper stadig**, når de arbejder med umodne krav.

Ready introducerer RMF (Kravmodningsflow), en praktisk og dybt struktureret tilgang til at afstemme Produkt og Engineering uden at erstatte din eksisterende proces.

Uanset om du bruger Scrum, Kanban eller noget tilpasset, hjælper RMF dig med at **stabilisere omfanget**, **eliminere overførsel** og **levere det, der virkelig betyder noget**.

Hvis dine teams føler sig fastlåst på kanten af "næsten færdig", vil denne bog vise dig, hvordan du kan **bryde cyklussen** og fjerne blokeringer for dit/dine team(s) **én gang for alle**.

Hvem Er Dette For

Denne bog er bogstaveligt talt for alle involverede i softwareudvikling. Alle fra ingeniører til produktchefer og fra individuelle bidragydere til ledere.

Denne bog er til dig, hvis du er involveret i softwareudvikling og har bemærket, at et team, du arbejder med eller på, har et eller flere af følgende problemer:

- Arbejde overføres ofte fra én iteration til den næste
- Implementeringsteams føler, at de forsøger at ramme bevægelige mål
- Arbejde holdes åbent for længe
- Arbejde markeres som afsluttet, men tingene er ikke rigtig færdige
- Udført arbejde stemmer ikke overens med forventningerne
- Arbejde genererer regelmæssigt et stort antal fejl
- Store mængder arbejde skal laves om regelmæssigt

Hvis nogle af disse problemer lyder bekendte, kan *Ready* hjælpe.

Hvordan Man Bruger Denne Bog

Denne bog blev designet til at være praktisk. Det er ikke en teoretisk afhandling eller en strategipræsentation - det er en praktisk manual til at implementere RMF (Kravmodningsflowet) baseret på reelt klientarbejde og feltafprøvet under ægte leveringspres.

Kapitlerne er skrevet i rækkefølge, men RMF selv er modulært. Det består af tre grundlæggende praksisser:

- RMF 1: Samarbejde for fælles forståelse
- RMF 2: Styring af arbejdets færdiggørelse ved hjælp af Skræddersyede Definitioner af Færdig
- RMF 3: Styring af implementering ved hjælp af Skræddersyede Definitioner af Klar

Hver del, eller "Vane", som vi kalder dem, står på egen hånd, men de bygger på hinanden. Bogen er designet til at hjælpe dig med at tilgå dem én ad gangen, i rækkefølge. Denne struktur afspejler, hvordan vi anbefaler teams at adoptere RMF i praksis - hvor hver Vane først implementeres, efter den forrige fungerer.

Dette undgår at overvælde teams og giver hver ændring den bedste chance for at blive forankret. Du vil lære mere om, hvordan du gør dette, når vi starter i Kapitel 9.

Hvis du søger hjælp - hvad enten det er rådgivning, coaching eller nogen at tale med dit ledelsesteam - er du velkommen til at kontakte os direkte.

Og hvis du leder efter formel støtte til at implementere RMF, tilbyder Producore en komplet serie af programmer designet til at guide adoption trin for trin. Du kan lære mere på https://ready-book.link/rmf.

Om Forfatterne

Luniel de Beer er skaberen af Requirements Maturation Flow (RMF), et praktisk system til at løse kløfterne mellem produktintention og engineering-udførelse. Han har over 15 års erfaring med at lede agile transformationer, bygge bro mellem produkt og engineering og hjælpe teams med at levere med klarhed og selvtillid.

Luniel er også ophavsmand til Producores Capability Management-system, en sporbar og skalerbar tilgang til modellering af produktegenskaber. Han udtænkte PKB-Drevet Udvikling (PKBDD), et versionsstyret system til håndtering af vedvarende produktkrav. Disse værktøjer udgør en del af en større leveringsramme udviklet hos Producore.

Max Guernsey, III er softwarearkitekt, underviser og medstifter af Producore, et konsulentfirma dedikeret til at løse leveringsfejl gennem strukturel og teknisk stringens. Med over to årtiers erfaring inden for objektorienteret design, refaktorering, testdrevet udvikling og designmønstre har han både leveret forretningskritiske systemer og coachet engineering-teams i stor skala. Hans arbejde kombinerer dybe tekniske praksisser med adfærds- og procestransformation for at hjælpe organisationer med at opnå bæredygtig leveringsekspertise.

Max bidrog væsentligt til PKBDD og stod i spidsen for udviklingen af Producores tilgang til Adfærdsdrevet Udvikling (BDD) gennem sin dybdegående ekspertise inden for adfærdsspecifikation.

Sammen integrerer deres arbejde klarhed, sporbarhed og gating i et sammenhængende system til softwarelevering, der skalerer fra teampraksis til organisatorisk kapabilitet.

Forord

Note til tekniske ledere

Hvis du er en erfaren leder i en teknisk organisation, er det sandsynligvis ikke indsats, disciplin eller kloge hoveder, du mangler. Og alligevel går projekter nogle gange i stå. Mål skrider. Forventninger bliver ikke indfriet. Ikke fordi dine teams er dovne—men fordi noget fundamentalt er i stykker i måden, arbejde bliver defineret, formet og leveret på.

Denne bog er ikke en lederguide. Det er et værktøj til strukturel diagnose. Den afslører, hvad der faktisk sker inden i dine teams—hvorfor "næsten færdig" bliver ved med at ende som "ikke færdig," og hvorfor lokale fremskridt så sjældent bliver til strategiske resultater.

Du ser måske ikke *dig selv* i disse sider. Men hvis dine teams ikke kan levere det, du har brug for, vil du se *dem*. Og når du gør det, vil du endelig have sproget—og systemet—til at rette op på det.

Fra Luniel

Først og fremmest ville denne bog ikke have været mulig uden Max, hvis evne til at se gennem tågen og avnerne og destillere en idé ned til dens essens er helt ubegribelig for mig.

Hvordan kom vi hertil?

Når jeg ser tilbage, tror jeg, det er fordi jeg altid har ønsket at forstå, hvordan ting virkelig fungerer. Uanset om det var religion, ernæring eller softwareudvikling, stødte jeg hele tiden på det samme problem: overfladiske svar, der ikke holdt under pres. Så jeg blev ved med at grave dybere—spurgte ikke kun om, hvad vi gør, men hvorfor, og hvad der mangler, når det ikke virker.

En af de tidligste revner i systemet viste sig i en rolle, hvor jeg havde tre kasketter på: Scrum Master, Product Owner og Udviklingschef (!!) for et team, der leverede datatjenester i en velkendt tech-virksomhed. Vi gjorde det, som Scrum foreskrev—korte Sprints, user stories i backloggen, planlægning på en halv dag—men hver gang vi startede et nyt Sprint, stødte vi på modstand. Teamet forstod ikke problemet fuldt ud, vi måtte genbesøge og revidere krav

midt i Sprintet, forebyggelige afhængigheder dukkede op og forsinkede os, og vigtige trin blev overset.

Så jeg begyndte at gøre noget anderledes. Jeg samlede teamet og interessenterne i et rum for hver user story, gennemgik problemet i detaljer, brainstormede løsningen sammen, og først derefter skrev jeg historien. Sprint-planlægningen skrumpede til en time, og vores leveringssucces steg markant.

Langsomt begyndte jeg at indse, at succes ikke kommer af at arbejde hårdere inden i Sprintet. Det kommer fra den struktur, du etablerer *før* det starter.

Senere, efter at have hørt Jeff Sutherland tale om "Definitions of Ready", vidste jeg, der var noget værdifuldt der—men det var ikke nok. Min erfaring med krav, UX, UI, research og senere med BDD viste mig, at forskellige arbejdsopgaver kræver forskellige former for parathed. Nogle behøver adfærdsspecifikationer. Nogle behøver systemadgang. Nogle behøver en komplet kortlægning af funktionaliteter.

Og alle har brug for en fælles forståelse, der faktisk er bekræftet—ikke antaget.

Efterhånden som jeg arbejdede med flere teams, så jeg det samme mønster overalt: manglende trin, uopfyldte afhængigheder, teams der gjorde deres bedste, men konstant måtte kæmpe for at løse problemer, der burde have været forhindret. Selv fremragende teams kæmpede—ikke fordi de var svage, men fordi de manglede en struktur, der gjorde parathed eksplicit.

Resultatet af al denne læring, iteration og frustration er et struktureret system til at håndtere parathed.

Det er det, denne bog handler om.

Jeg håber, den giver dig klarhed over, hvor de virkelige problemer ligger, og hvordan man løser dem. Jeg håber, den giver dig sprog til at forsvare praksisser, der måske virker "ekstra", men faktisk er essentielle. Og mest af alt håber jeg, den hjælper teams med at levere med mindre stress, færre overraskelser og langt bedre resultater.

Hvis vi får dette rigtigt, vil vi spare branchen for milliarder af kroner.

Men vigtigere endnu vil vi give folk deres sunde fornuft tilbage.

Fra Max

Jeg har arbejdet med dette problem fra forskellige vinkler i årtier, men min fremgang var begrænset, indtil jeg mødte Luniel.

Dette skyldes, at jeg før jeg kendte ham, grundlæggende gik til problemet som et teknisk problem. Jeg fokuserede på at hjælpe teams med at adoptere ting som Test-drevet udvikling (TDD), refaktorering, avanceret softwaredesign og senere Acceptance-Test-drevet udvikling (ATDD) eller Adfærdsdrevet udvikling (BDD).

I de fleste af disse tilfælde blev problemet, der behandles i denne bog, behandlet som en implementeringsdetalje i etableringen af disse tekniske praksisser.

Dette betyder ikke, at jeg ikke længere værdsætter de tekniske praksisser. Jeg mener stadig, de er dybt vigtige, men de adresserer ikke direkte problemet med parathed i softwareudvikling. I stedet *fremhæver* de problemet, og folk sætter så en lap på deres proces for at adressere det "lige akkurat nok" til at understøtte de tekniske praksisser, de forsøger at implementere.

Jeg vil også adressere spørgsmålet om, hvem denne bog kan hjælpe. Det korte svar er "sandsynligvis næsten alle inden for softwareudvikling", men det rigtige svar indeholder nuancer, der hjælper med at kortlægge det til forskellige miljøer uden at ændre den grundlæggende betydning.

Der er teams, som har brug for den løsning, denne bog tilbyder. Du vil møde en forsimplet version af et sådant team i Kapitel 1.

Der er også teams, som ikke direkte **har brug for** et system som det, vi foreslår, men som stadig kunne drage nytte af det.

Det bedste team, jeg nogensinde har arbejdet med–let en hel standardafvigelse over det **næstbedste team**, hvis ikke to–lå gemt i baglandet i Central Oregon. De præsterede så højt, at de kunne overkomme fraværet af sådan et system gennem ren og skær kompetence. Alligevel sagde min chef dengang, Tom Barreras, engang noget i retning af "Jeg har bemærket, at vores historier går bedre, når vi bruger lidt tid på at tale om testene på forhånd."

Dette var igen noget, jeg på daværende tidspunkt betragtede gennem testudviklings- og tekniske udførelsesbriller, men nu ved jeg, at det var endnu en indikator på, at *parathed* var en faktor, der påvirkede teamet… det specifikke team var bare så kompetent og hurtig til at reagere, at de kunne lykkes ved at håndtere forhindringer, når de opstod, frem for at forebygge dem fra starten.

Selv hvis du er den type person, der ikke direkte *behøver* at bekymre dig om parathed, fordi du kan overkomme det, eller du arbejder med et team af samme kaliber, kan du stadig drage nytte af indholdet i denne bog.

Del I: Noget Mangler

*Når det ikke hjælper at gøre de samme ting bedre, så kig efter det der **ikke bliver gjort**.*

Kapitel 1: Det Skjulte Problem

Dette er en sand[1] historie om en bank. Vi kalder den bare "Banken". Det er en type Føderalt kreditinstitut, der fungerer som en del af USA's nationale finansielle infrastruktur.

Vi (Luniel og Max) blev hentet ind i Banken, fordi den havde problemer med at levere et softwareprojekt. Det var et af de mest dysfunktionelle miljøer, vi nogensinde havde set, og det er derfor, vi valgte dette som det indledende casestudie: hvis meningsfuld forandring var mulig i Banken, er det muligt hvor som helst.

En Hurtig Bemærkning om Projekter

Når vi bruger begrebet "projekt" i denne bog, mener vi det i konteksten af projektledelse. Selvom der er forskellige idéer om, hvad ordet betyder, bruger vi definitionen fra Project Management Institute:

> "Et projekt er en **midlertidig** indsats, der udføres for at skabe et unikt produkt, en unik service eller et unikt resultat."

Dette betyder, at et projekt har en defineret begyndelse og slutning. Når et projekt afsluttes, arkiveres projektviden og artefakter, teammedlemmer frigøres, og kontrakter afsluttes.

I denne bog handler et projekt grundlæggende om eksekvering. De fleste projekter, som defineret af PMI, begynder med feasibility eller design. Vision og strategi er allerede på plads, når et projekt igangsættes.

Et projekt fødes af den vision og strategi, og det lykkes eller mislykkes baseret på, om de forestillede mål realiseres—*ikke* på om disse mål var de rigtige.

Du bruger måske ordet "projekt" anderledes, og det er okay. Du skal bare vide, at når vi bruger det, henviser vi til definitionen og konteksten ovenfor.

Intet af dette skal antyde, at vi bifalder brugen af projektledelse til softwareudvikling. Tværtimod. Men vi anerkender, at det *bliver* brugt alligevel. Vi tager fat på det problem

[1] Vi har ændret identificerende detaljer for at beskytte privatlivet for de mennesker og institutioner, det skete for.

> senere i Kapitel 6.

1.1: Den Klassiske Omskrivning

Banken var i gang med at omskrive deres lånetilbagebetalingsportal af flere årsager.

Det gamle system, en komplet C#/.NET-løsning, var fejlbehæftet. Ud over at forringe kundetilfredsheden genererede det også en endeløs strøm af meget dyre supporthændelser, hvor nogen manuelt måtte manipulere databasen for at rette en fejl lavet af systemet.

Det gamle system var også forfaldent fra et vedligeholdelsesperspektiv. Det var næsten umuligt for ingeniørerne at foretage meningsfulde ændringer, og selv når de kunne, var det en ekstremt risikabel affære.

Omskrivningen skulle ændre på det.

Det nye system skulle stadig have en C#/.NET-backend, men den skulle være fuldt dækket af tests. Frontenden skulle implementeres i OutSystems, en populær low-code eller no-code-løsning, der giver en organisation mulighed for at definere en applikation ét sted og få en web-app, en Android-app og en iOS-app automatisk genereret, når de beslutter at publicere deres ændringer.

Håbet var, at det nye system ville være fejlfrit og både forbedre kundetilfredsheden og reducere supportomkostningerne betydeligt.

De håbede også, at omskrivningen ville fjerne udviklernes blokeringer—med kombinationen af en mere disciplineret tilgang til backend'en og low-code-tilgangen til frontend'en, der i høj grad skulle reducere omkostningerne og risikoen ved nye funktioner.

En fin sidegevinst ved at skifte til OutSystems var, at de ville få en ren, moderne mobil-app på begge de store platforme.

Det var drømmen, da de startede tre år før begyndelsen af denne historie. Realiteten var, at teamene indtil videre ikke havde leveret **noget som helst**.

1.2: Perspektiver på Problemet

Da vi talte med den øverste ledelse, hørte vi en meget naturlig frustration over, at de havde foretaget så store investeringer uden nogen som helst strategisk bevægelse.

De havde, fra deres perspektiv, prøvet alt. De havde udskiftet medarbejdere, øget antallet af medarbejdere, ændret budgettet, øget presset og hentet en parade af konsulenter ind (hvoraf det blev stærkt antydet, at vi var de sidste i rækken). Intet syntes at gøre det bedre—i hvert fald ikke på en måde, de kunne måle, for alt hvad de så var, at "nålen" stod på nul i et kvartal, og så stod den stadig på nul i det næste kvartal.

De ville ikke have mere "usynlig fremgang". De ville have *resultater*.

Da vi talte med ledelsen i Produktorganisationen, fik vi en lidt anderledes (men stadig lignende) historie, fordi de arbejdede mere direkte med udviklingsafdelingen.

Det var ikke sådan, at teamene ikke lavede noget, ikke i deres egne øjne i hvert fald. Det var mere det, at teamene aldrig lavede det, de blev bedt om. Det var nærmest en garanti: Lige meget hvor simpel opgaven var, og hvor tydeligt den blev formuleret, ville man ende med noget helt andet, når tiden kom til at evaluere det, teamene havde lavet.

Det var nået til det punkt, hvor den kørende vittighed lød i retning af "Vi bliver nødt til at regne ud, hvordan vi skal bede om det, vi ikke vil have, så vi i det mindste har en chance for at få det, vi faktisk vil have."

De tekniske ledere så tingene helt anderledes.

For dem var problemet, at Product ikke leverede handlingsorienterede krav, og at Product ikke leverede *nok* krav. Hvis Product bare kunne "følge med i programmet", ville teamene være i stand til at levere det ønskede til tiden og inden for budgettet.

De havde foretaget seriøse investeringer i at modernisere måden, kode blev skrevet og leveret på, og i deres øjne leverede Product ikke klare krav.

Da vi talte med andre konsulenter (som havde henvist os til organisationen), fokuserede de med rette på den dysfunktion, de observerede: Alle virkede meget fokuserede på at give andre skylden. Grunden til, at de oprindeligt bragte os ind, var, at de var bekymrede for bemandingsstrategien og ønskede en evaluering af de individuelle bidragydere, men de så fingerpegeriet og den kontrollerende ledelsesstil på direktionsniveau som den primære kilde til problemerne.

1.3: Vores undersøgelse

Vores oprindelige opgave var at evaluere teamene og forsøge at hjælpe dem med at opgradere deres færdigheder efter behov, så vi begyndte at se nærmere på de mennesker, der arbejdede på frontlinjen.

Der var helt klart plads til forbedring.

De individuelle bidragydere på Product-siden havde reelt ikke de nødvendige færdigheder. I virkeligheden var de mest projektledere, som var blevet kastet ind i rollen som Product Owner (PO) eller Product Manager.

Som resultat skrev halvdelen af dem "luftige" krav og accepterede derefter (bogstaveligt talt) hvad som helst, teamene lavede i den iteration uden nogen kritisk analyse. Den anden halvdel skrev samme type krav og hævdede så, at teamene "burde have vidst" ting, de aldrig havde talt om, og holdt arbejdsopgaverne åbne mere eller mindre på ubestemt tid.

Banken plejede at administrere og følge deres opgaveportefølje. Mens det, vi dækker i denne bog, for det meste er uafhængigt af Scrum, bruger vi Scrum-terminologi gennem hele bogen, fordi flertallet - eller i det mindste en pluralitet - af teams bruger Scrum.

Definition: Scrum

Scrum er et letvægts-framework, der hjælper mennesker, teams og organisationer med at generere værdi gennem adaptive løsninger på komplekse problemer. Kort fortalt:

1. En Product Owner prioriterer arbejdet for et komplekst problem i en Product Backlog
2. Scrum-teamet omdanner et udvalg af arbejdet til et værdiforøgende increment under en Sprint
3. Scrum-teamet og dets interessenter inspicerer resultaterne og justerer til næste Sprint
4. Gentag

Hvis du ikke er bekendt med Scrum og dets terminologi, anbefaler vi, at du gennemgår 2020-versionen af Scrum Guide. Det er en hurtig og oplysende læsning.

Tilsvarende fandt vi, at de tekniske teams lå langt under det typiske niveau med hensyn til kodningsfærdigheder (-2σ, i bedste fald) og oven i købet var meget modstandsdygtige over for forandring. Som en naturlig konsekvens var kodekvaliteten elendig.

Men ifølge teamene var dette ikke grunden til, at de ikke leverede. For dem var det vage krav og ændringer midt i sprinten fra Product, der ødelagde projektet.

…og ingen talte engang om det større problem, det der var så absurd, at det lyder opdigtet, indtil man selv har oplevet det.

Udviklingsteamene havde en vane med ikke at forstå et krav, bygge noget tilfældigt og derefter kræve anerkendelse for at have "færdiggjort en arbejdsopgave".

Vi mener ikke en lille misforståelse. Vi mener en total afkobling: Vi kunne bede dem om at deaktivere anvendelse af midler til hovedstolen under bestemte omstændigheder, og de ville i stedet deaktivere muligheden for at tilføje en sekundær bekræftelses-e-mailadresse.

Derefter ville de fortælle os, at det var det, vi havde bedt om.

1.4: Dybere ned i problemet

Begge kompetencehuller kunne håndteres, men vi var skeptiske over for, om de var de reelle blokeringer.

Der var noget andet galt, så vi gravede dybere. Vi startede med dette spørgsmål: Hvorfor tog det så lang tid at skrive krav, og hvorfor gav det så dårlige resultater?

En af grundene var, at den viden, der var nødvendig for at skrive et meningsfuldt krav, var en mangelvare. En smule af den fandtes hos udviklings- og Product-teamene.

Noget af det var indlejret i koden i det ældre system. Noget af det var helt forsvundet. Det meste var dog gemt som tavs viden hos fageksperter spredt ud over bankens forskellige afdelinger. Dette betyder, at udarbejdelsen af et krav, der faktisk fremmer et strategisk mål, var en ekstremt arbejds- og tidskrævende aktivitet.

I kontrast til dette stod et umætteligt behov for funktionalitet fra et funktionalitetshungrende ledelsesteam. Mandatet var "hold udviklerne i gang—fyld dem med krav". Fokus var på mængden af krav for at holde teamene beskæftigede—et perspektiv der var i direkte modstrid med den omhu og tid, der var nødvendig for at definere et krav, der ville "rykke noget".

1.5: At Gøre Tingene Bedre Gjorde Det Ikke Bedre

Dette er alt sammen problemer, der kan løses, men at løse dem hjalp ikke.

Tidligere forbedringer af softwareudviklingsteknikker havde ikke hjulpet teamene med at levere, men Max gjorde en indsats for at hjælpe teamene med at forbedre sig yderligere.

Han introducerede revolutionerende koncepter fra programmeringsdoktriner fra midten af det tyvende århundrede som "lad være med at kopiere og indsætte den kode 27[2] gange, put den i en funktion og kald den i stedet". Denne forslag alene forbedrede dramatisk kvaliteten af ny kode og gjorde det muligt for dem at begynde at forbedre kvaliteten.

Dette og andre grundlæggende kodningsråd hjalp dem med at skrive bedre kode, som de lettere kunne vedligeholde i fremtiden.

…men det hjalp ikke med at få projektet fremad.

På produktsiden lykkedes det Luniel at introducere BDD og sikre, at Product Owners grundigt gennemgik krav, før de blev overdraget til teamene.

Han fik teamene til at samarbejde om dem og bruge dem til at evaluere, om et Product Backlog Item (PBI) virkelig var færdigt.

Definition: Product Backlog Item (PBI)

> Et Product Backlog Item (PBI) er en diskret arbejdsenhed i product backloggen, der repræsenterer en potentiel ændring, tilføjelse eller forbedring af produktet. PBI'er kan tage mange former—feature, fejlrettelse, teknisk forbedring, research-opgave osv.—og defineres ved deres bidrag til produktværdien.
>
> Mange teams refererer til PBI'er som "stories" eller "user stories," men den korrekte Scrum-term er "Product Backlog Item" eller "PBI". Når et PBI er forpligtet til en Sprint, er det også en del af Sprint Backloggen. For enkelheds og neutralitets skyld bruger vi "Product Backlog Item" eller "PBI" til at henvise til enhver arbejdsopgave, som Scrum-teamet håndterer—uanset om du kalder det et Product Backlog Item, Sprint Backlog Item (SBI), user story, story, arbejdsopgave eller backlog item.

Det tilføjede *klarhed*, men det skabte ikke *flow*.

Med hjælp fra de partnerkonsulenter, der havde bragt os ind, lykkedes det os at (midlertidigt) lette det absolut knusende pres, der blev lagt på teamene og kravforfatterne.

Det kunne have hjulpet med at skabe lidt tillid, men det gav ingen håndgribelige resultater.

Vi begyndte endda at opbygge en vidensbase, der hjalp folk med at opspore den forretningsviden, de havde brug for til at skrive krav og identificerede de steder, hvor der var huller i denne viden.

[2]Dette er ikke kun ikke en overdrivelse, det er ikke engang det værste tilfælde. I ét tilfælde var der næsten hundrede eksakte kopier af den samme algoritme.

Det gjorde kravskrivningen hurtigere, men det fik ikke produktet ud af døren.

Efter måneders interaktioner havde vi hjulpet ledelsen med at *se*, hvor de var, men de var ikke i nærheden af deres mål. Og de kom ikke tættere på.

De var ved at være klar til at vende tilbage til deres gamle strategi med at "overlæsse" teamene for at sikre, at de altid var beskæftigede.

1.6: En Udelukkelsesproces

Det ville have været nemt bare at opgive og sige "dette er håbløst". Der var adskillige undskyldninger, man kunne falde tilbage på:

- Udviklingsteamet havde lave kompetencer (det havde de)
- Kravforfatterne havde det forkerte kompetencesæt (det havde de)
- Ledelsen knuste teamene med urealistiske forventninger (det gjorde de)
- Organisationen manglede kritisk forretningsviden, der var nødvendig for at fungere (det gjorde den)
- De ledende medarbejdere ser ikke ud til at stole på hinanden (det gjorde de ikke)

Alt dette var sandt. Alligevel var der blevet lavet forbedringer på alle disse områder, og ingen af dem syntes at gøre det grundlæggende problem bedre. Ingen af dem hjalp udviklingsteamene, Product, ledelsen eller direktionen med at komme tættere på deres mål.

...og det er netop her, løsningen ligger: Alle de tidligere nævnte problemer var dem, folk allerede kunne se.

Hvis de variabler, du kan se, ikke gør en forskel, må der være en variabel, du **ikke kan se**, som gør det.

Det egentlige problem var det problem, som ingen engang vidste eksisterede.

1.7: På jagt efter den rigtige skyldige

I søgningen efter den rigtige skyldige - det, der virkelig holdt banken tilbage fra at realisere sine mål - var vi nødt til at starte et sted.

Et fornuftigt sted at kigge var på, hvad de fejlede PBI'er (de fleste af dem) havde til fælles.

Vi begyndte med at eliminere de ting, vi kunne se ikke var fælles, fordi de varierede meget:

- Hvilken del af systemet: nogle PBI'er ramte kun backend, andre kun frontend, andre igen ramte begge dele
- Hvilket team der udførte arbejdet - det syntes ikke at betyde noget, hvem der udførte arbejdet, der var en høj sandsynlighed for fejl
- Hvilken PO der var forfatter til arbejdet - samme som med teams

Så begyndte vi at kigge på de ting, der var fælles. Listen var ikke lang, men den var heller ikke kort:

- Udviklingsteams
- Product owners
- Ledelsen
- Kulturen
- Udviklingsmiljøet
- Udviklingspraksissen
- Kravspecificeringsteknikken
- Domænet (finans)
- Afhængighedstjenesterne

Mange af disse kunne også afvises med det samme. Teams, product owners, ledelse, kultur og udviklingsmiljø var alle blevet forbedret for nylig uden nogen reel indvirkning på det meningsfulde output. Vi havde personligt hjulpet med at forbedre udviklings- og kravspecificeringspraksissen og bekræftet, at disse forbedringer havde bidt sig fast, men det hjalp stadig ikke.

Man kan næppe bebrejde domænet. Finans er en af de ældste former for beregninger i den registrerede historie. Det er yderst modent. Desuden deployede andre banker software, hvilket performativt modbeviste den (åbenlyst overdrevne) hypotese om, at banker simpelthen ikke kan gøre det.

Afhængighedstjenesterne kunne heller ikke bebrejdes, da de havde lige så mange problemer med at ændre sig som det initiativ, vi kiggede på…

…men det fik os til at tænke: Hvad hvis vi begyndte at analysere årsagerne til fejl?

1.8: Dissekering af fejlenes kim

En PBI formåede ikke at bringe produktet fremad, fordi teamet, som de havde for vane at gøre, lavede noget helt tilfældigt og næsten helt urelateret til forespørgslen. Det er åbenlyst et tegn på, at de ikke forstod arbejdsopgaven. Så forståelse var en stor kandidat, selvom vi havde arbejdet med det, da vi hjalp dem med at adoptere BDD.

En anden PBI fejlede, fordi de fik beregningerne forkert. Det er endnu et bevis på, at forståelse kunne være kerneproblemet.

En tredje arbejdsopgave, som vi analyserede, blev ikke rigtig håndhævet af Product Owner - hun godkendte den bare, da teamet sagde, det var tid til at lukke den. Det udfordrede vores hypotese lidt, men man kunne stadig argumentere for, at hun ikke forstod, hvordan arbejdsopgaven passede ind i en overordnet plan.

Måske. Nogenlunde. Hvis vi kiggede meget grundigt på det på den måde.

Så stødte vi på en PBI, der slet ikke passede ind i mønstret. Teamet så ud til at forstå - selvom der ikke er nogen måde at verificere, om de faktisk gjorde det. Men det var ligegyldigt: de fik aldrig chancen for at lykkes eller fejle på egen hånd, fordi de stødte på en afhængighed, der skulle opdateres, og måtte udskyde deres arbejde med flere Sprints.

Selv hvis de *ikke* forstod, hvad de skulle gøre, havde de aldrig en chance med den backlog-opgave, derfor var forståelse **ikke** problemet i det tilfælde.

En enkelt afvigelse er naturligvis ikke et bevis på, at en bestemt grundårsag er forkert, men det vakte vores nysgerrighed. Vi begyndte at lede efter andet afkræftende bevis.

Og vi fandt det. Der var arbejdsopgaver, der:

- Fejlede fordi teamet vidste, at det ikke forstod opgaven, men ingen kunne finde en fagekspert til at løse problemet
- Ændrede sig til en dårligere brugeroplevelse som en workaround til, hvordan upstream-tjenesterne fungerer
- Måtte udskydes fordi upstream-afhængigheder ikke var klar
- Ikke kunne færdiggøres fordi testerne ikke kunne indsamle testdata i tide
- Blev lukket men måtte laves om, fordi selve forespørgslen var forkert
- Fejlede fordi teamet ikke indså, hvor kompleks den eksisterende kode allerede var
- Simpelthen ikke blev estimeret

- Var massivt underestimerede[3]
- Ændrede sig midt i Sprintet fordi PO'en endelig fik den domæneviden, de havde brug for
- Så ud til at ændre sig efter Sprintet (set fra teamets perspektiv) fordi PO'en og teamet aldrig blev enige om, hvad det betød

Listen fortsætter, men det er nok til denne historie.

Man kan argumentere for, at hver af disse kan knyttes til "forståelse" på en eller anden måde—og helt sikkert var manglende forståelse *involveret* i mange af dem—men det betyder ikke, at manglende forståelse var årsagen… især eftersom vi havde arbejdet med fælles forståelse, og det havde ikke rigtig hjulpet.

Så slog det os. Der manglede et mere grundlæggende element. I de tilfælde hvor dårlig forståelse var involveret, var det blot den umiddelbare årsag.

Den bagvedliggende årsag var meget bredere.

1.9: Et Skæringspunkt

Den ene ting, som alle PBI'er vi analyserede for fejltilstande havde til fælles, var dette: De var alle blevet **påbegyndt for tidligt**.

Når en arbejdsopgave fejler, fordi teamet vidste, at de ikke forstod problemet og ikke kunne finde en ekspert til at hjælpe dem med at forstå det, betyder det, at teamet startede en PBI velvidende, at de ikke forstod problemet.

Når et backlog-element må ændres til en dårligere oplevelse på grund af, hvordan en upstream-tjeneste fungerer, betyder det, at arbejdsopgaven blev påbegyndt uden rigtig at forstå påvirkningen fra upstream-tjenesten.

Udsættelse fordi en upstream-afhængighed ikke var klar ved slutningen betyder, at der ikke var nogen garanti for dens parathed i begyndelsen.

…og sådan var det med alle de andre tilfælde: Testere havde ikke data klar eller vidste ikke, hvordan de skulle få det, før en PBI startede, krav var ikke rigtig gennemgået, før de blev overdraget til teamet, koden var ikke undersøgt, før arbejdet blev påbegyndt, estimeringen var utilstrækkelig eller ikke udført overhovedet, domæneviden manglede, og selvfølgelig var den fælles forståelse ikke blevet verificeret.

[3]Vi mener ikke bare, at de fik det forkert. Det så ud som om teamene måske bare havde sat værdien "3" i alle estimeringsfelterne for en række PBI'er.

Problemet viste sig at være, at implementeringen blev påbegyndt på arbejdsopgaver, før disse arbejdsopgaver var *klar*.

Ud fra vores erfaring fejler de fleste arbejdsopgaver i at blive leveret, fordi de **ikke var klar**, da implementeringen begyndte.

Så vi satte os for at hjælpe Banken med det.

På dette tidspunkt tænker du måske, at det burde være nok bare at sige, at PBI'er skal være klar. Det viser sig dog, at det ikke er så let at implementere. "Køb lavt, sælg højt" er en lignende simpel idé.

Der mangler et element for at kunne omsætte det gode råd til praksis.

1.10: Det Store Vendepunkt

Vi fandt det manglende puslespilsbrik, der fjernede blokeringen for dem, hvilket igen fjernede blokeringen for initiativet.

Da vi afsluttede det engagement, lå udviklerne stadig et godt stykke under gennemsnittet med hensyn til færdigheder. PO'erne havde stadig ikke de rette kompetencer. Kulturen var stadig ikke fikset…

Alligevel begyndte produktet endelig at bevæge sig fremad og blev i sidste ende leveret.

Ved slutningen af denne bog vil du vide, hvad det manglende element er, og hvad der skal til for at få det på plads. Og du vil være i stand til at fjerne blokeringer i organisationer, der synes holdt tilbage af en usynlig mur.

En Hurtig Bemærkning om Omfang

Denne bog handler om et meget specifikt og gennemgribende problem: manglen på struktur, klarhed og modenhed i overleveringen mellem forretning og udvikling. Den antager, at noget er blevet valgt til implementering, og fokuserer på at sikre, at udviklingsarbejdet sker med fælles forståelse, parathed og sporbar færdiggørelse.

Teknikkerne i denne bog fortæller dig ikke, hvad du skal bygge, hvorfor du skal bygge det, eller hvordan du finder ud af, om det er det rigtige at bygge. Hvis din organisation

mangler reel produktstyring eller meningsfulde feedback-loops, forsøger vi ikke at løse det her. Hvad vi i stedet tilbyder, er en måde at gøre disse huller mere synlige og reducere omkostningerne ved at opdage, at man tog fejl.

Brugt i den rette kontekst bringer denne løsning flow, sikkerhed og klarhed. Men som ethvert system kan det misbruges—især når det bruges isoleret eller uden bevidsthed.

Kapitel 2: Omkostningen ved Manglende Fundamenter

Det giver mening at bruge lidt tid på præcis *hvor* slemt dette problem kan være for nogle organisationer.

Vi har fundet ud af, at der er tre store "kategorier" af problemer, som mangel på parathed skaber:

- Manglende eller ufuldstændig fælles forståelse mellem og inden for Product og Engineering
- Manglende kontrol over, hvad det egentlige mål for et PBI er, og hvornår det faktisk er færdigt
- Manglende kontrol omkring hvornår en arbejdsopgave kan påbegynde implementeringsfasen af dens livscyklus

Derudover har vi bemærket, at det kan være ret udfordrende at ændre disse ting. Det giver mening: forandring er svær.

Gamle vaner dør langsomt, og nye vaner er svære at indarbejde. I vores erfaring som konsulenter har vi bemærket, at det er *ekstremt* let for folk at falde tilbage i gamle vaner og tilsvarende svært for dem at etablere nye.

Så man er nødt til at have en mekanisme, der håndhæver de nye vaner og modvirker de gamle.

For at adressere dette mener vi, at der mangler endnu en komponent af ansvarlighed og sporbarhed.

2.1: Puslespillet uden Æsken

Har du nogensinde prøvet at samle et puslespil uden billedet på æsken? Du kan godt gøre det, men det er langsommere, mere frustrerende og fyldt med falske starter.

Du gør fremskridt, river det så fra hinanden igen. Du betvivler hvad der passer hvor. Du tror, du arbejder på det samme billede, indtil du indser, at det gør du ikke.

Sådan føles softwareudvikling ofte.

Backloggen er fuld. Sprintet kører. Alle arbejder hårdt.

Men uden et fælles billede af hvad vi bygger, bliver alignment et spørgsmål om held, ikke et system.

Uden klarhed føler selv de bedste teams frustration, udbrændthed og en følelse af, at deres indsats ikke værdsættes.

2.2: Et Gammelt Ordsprog og en Barsk Virkelighed

Der er en grund til, at så mange praksisser omkring kravspecificering, selv nogle udviklingspraksisser, fokuserer kraftigt på at skabe en fælles forståelse mellem bestillere og implementeringsteams. Intet skaber så meget kaos som når udviklere ikke rigtigt ved, hvad de skal bygge.

"Skrald ind, skrald ud" er et ordsprog, ikke en kliché.

Det engelske sprog er fyldt med selvmodsigelser…

- Du kan sanktionere nogens handlinger. Måske betyder det, at du har givet dem tilladelse på forhånd, eller måske betyder det, at du giver fordømmelse efterfølgende.
- Du kan let støve noget af, men hvis dette noget er en kredenz, betyder det, at du fjerner støv fra den, mens hvis det er med henvisning til en beignet, tilføjer du støv.
- Hvis du holder et team oppe, kan du være grunden til, at teamet fortsat kan fungere, eller du kan være grunden til, at de ikke kan komme nogen vegne.

Auto-antonymer er måske de mest slående eksempler, men de er bare én type tvetydighed. Nogle ord er ikke kun deres egen modsætning, men har mange yderligere potentielt forvirrende alternative betydninger.

> "In this clip, he clipped a coupon from a news paper and clipped it to the paper on his clipboard along with the other clippings while a clipper in the background was moving along at a decent clip."

Det er ikke kun engelsk. *Alle* naturlige sprog, som vi kender til, har denne egenskab.

Og alligevel er de de eneste, vi har at arbejde med, når vi specificerer krav.

Som følge heraf, når et udviklingsteam ikke har *bekræftet*, at deres forståelse af et krav er den samme som anmoderens, stoler teamet på held. Det vil sige, at det bedst mulige udfald er, at de valgte den rigtige fortolkning, og at anmoderen ikke ændrer mening undervejs.

Dette udfald er langt fra garanteret.

2.3: Nogle af de Almindelige Resultater

Uden bekræftet fælles forståelse løber teams en række risici.

I det store og hele er det mest almindeligt smertefulde resultat, at teamet simpelthen bygger den forkerte ting.

Tidspunktet for hvornår de opdager det, kan styres af forskellige attributter i deres proces. For eksempel kan en sund implementering af Scrum opdage den slags misforståelser meget tidligt i udførelsen, mens en Vandfaldsproces har en meget god chance for at forsinke sådan en opdagelse med måneder.

Definition: Vandfaldsmodel

En sekventiel softwareudviklingsmodel, der blev udbredt i slutningen af det 20. århundrede. Den blev først beskrevet i en 1970-artikel af Winston W. Royce, som illustrerede udvikling som en række kaskaderende trin—krav, design, implementering, test og så videre—hvor hvert trin føder ind i det næste som et vandfald. Selvom Royce præsenterede modellen som et eksempel på, hvad man *ikke* skulle gøre, adopterede industrien den som en blueprint for udvikling i stor skala.

Vandfaldsmodellen er også kendt for at gruppere lignende arbejde i store, på hinanden følgende faser—en karakteristik der henvises til som **stor-batch udvikling**. Denne batchning garanterer praktisk talt **sen læring**: teams modtager ikke feedback på tidligere beslutninger før meget senere i processen. Fejl, der opdages sent, er dyrere at rette. Agile praktikere kritiserer Vandfaldsmodellen af denne grund og foretrækker mindre, iterative cyklusser, der muliggør tidligere opdagelse og kurskorrigering.

Ikke desto mindre vil et team, der arbejder ud fra den "forkerte" (eller rettere forskellige) forståelse af et krav, på et tidspunkt blive konfronteret med det "rigtige" (også forskellige, faktisk) krav. Igen vil organisationens sundhedstilstand påvirke, hvilken form denne konfrontation tager, og hvilken indvirkning den har, men det sker næsten altid.

I de fleste tilfælde fører dette til en form for omarbejde. Rekvirenten (typisk produktledelsen) bliver nødt til at bede om ændringer for at komme fra det, implementeringsteamet har bygget, til det, han egentlig ønskede.

En anden meget almindelig manifestation er, at rekvirenten fortsætter med at holde teamet ansvarlig for hans oprindelige forståelse af det, han bad om.

Teams kan nemt fortolke dette som en produktchef, der skifter mening. Værre endnu kan det faktisk *invitere* interessenter til at udvikle en vane med at ændre mening - tilbageholde arbejdsopgaver indtil alt er *helt perfekt*, udmatte teamene og gøre fremskridt usynlige for den øverste ledelse.

2.4: Hvornår er et puslespil færdigt?

Lad os vende tilbage til vores puslespils-analogi og tænke over dette spørgsmål: Hvad betyder det at være færdig med et puslespil?

En naiv puslespilsbygger som Max ville simpelthen sige "alle brikkerne er sat sammen med de rigtige naboer, og billedet vender opad."

En erfaren puslespilsbygger som Luniel ved dog, at der er mere til det.

Måske lægger du bare puslespillet for sjov. Du får det færdigt, kigger på det et stykke tid og skiller det så ad igen for at lægge det tilbage i æsken.

På den anden side vil du måske indramme det og hænge det op på væggen. I så fald er der yderligere ting, der skal gøres:

1. Læg det på en kunstplade
2. Transport til en rammemager
3. Vent på at indramningen bliver færdig
4. Transport tilbage til monteringsstedet
5. Hæng det op på væggen eller udstil det på anden vis

Det er nødvendigt at forstå, at dette er en del af arbejdet for at kunne færdiggøre et puslespil ordentligt. Den åbenlyse grund er, at du ved, hvor meget arbejde der er involveret. Det er mere arbejde at udføre alle disse ekstra trin, end det er bare at skille det ad og lægge det væk.

Det går dybere end det. Forestil dig dette scenarie…

Du har færdiggjort dit puslespil med henblik på at få det indrammet, du har ladet det ligge, men du glemte at fortælle en anden i din husstand, at du planlægger at få det indrammet. Denne person kommer forbi og ser, at puslespillet er færdigt på et område, som han eller hun skal bruge. Så skiller de det ad og lægger det tilbage i æsken og snupper dermed nederlaget lige fra sejrens rand.

Der er endda en endnu mere subtil grund: Hvordan du planlægger at afslutte puslespillet påvirker, hvilke trin du vil udføre tidligere i processen. For det første har du brug for at lave et lille skilt, hvor der står "Må ikke skilles ad!"

Det er også værd at bemærke, at du måske vil have et skilt i det tilfælde, hvor du bygger et puslespil for din egen underholdnings skyld, for at sikre at andre ikke blander sig ved at færdiggøre puslespillet for dig.

Du skal også sikre dig, at du samler puslespillet på den rigtige overflade. Hvis du samler et puslespil med tusind brikker på dit glassofabord og derefter forsøger at overføre det til en kunstplade, vil overførslen være meget mere risikabel og arbejdskrævende, end hvis du bare havde samlet puslespillet direkte på kunstpladen.

Dette har fine paralleller til softwareudvikling.

Du behøver faktisk at vide, hvad færdig betyder, så du ikke bliver overrasket over, hvor meget arbejde der er involveret, der ikke er uenighed om det til sidst, og du kan tage de nødvendige forberedende skridt for at sikre en gnidningsløs og effektiv færdiggørelse af en arbejdsopgave.

2.5: Påvirkning på teams

Hvis du ikke har en tilstrækkelig rigid forståelse af, hvad færdiggørelse betyder for en bestemt arbejdsopgave, løber du en række risici.

Vi bruger ordet "risici" løst her, da det mere er som garantier.

Udviklingsteams i denne situation opdager ofte, at de ikke engang internt er enige om, hvordan færdiggørelsen af en arbejdsopgave ser ud. Det er ikke ualmindeligt for udviklere og testere at opdage, at de er nødt til at udrede, hvad et krav virkelig *betyder* halvvejs gennem en Sprint. Selv to udviklere eller to testere kan lide under samme uenigheder.

Desuden er udviklingsteams ofte fokuseret på det arbejde, de laver det meste af tiden (kodning og test). Dette betyder, at det er let for dem at glemme andre former for arbejde, de skal udføre, såsom dokumentation, eksterne gennemgange, træning af andre teams (f.eks. support), forberedende trin til at understøtte implementering eller release og godkendelser fra andre afdelinger.

Når det endelig bliver klart, at dette "ekstra" arbejde skal udføres, bliver de taget på sengen—som regel må de stoppe det, de var i gang med, og skifte fokus for at gå tilbage og færdiggøre arbejde, som de troede, de allerede havde afsluttet.

De, der anmoder om arbejde, kan nemt holde arbejdet åbent unødvendigt. Nogle gange med de bedste intentioner—som at forsøge at holde et team ansvarligt for det "egentlige" krav. Andre gange sker dette, fordi Product Owners (for eksempel) har vænnet sig til at kunne holde en PBI åben efter deres forgodtbefindende, så de bruger det til at presse ekstra funktionalitet ind i en opgave i sidste øjeblik. Nogle gange gør de det endda, fordi de har skiftet mening om, hvad der skal gøres undervejs i processen.

Dette kan være ekstremt demoraliserende for et udviklingsteam. De fleste softwareudviklere og testere vil gerne føle, at de gør fremskridt. Hvis de konstant får at vide, at det, de gjorde, var forkert, vil de sandsynligvis miste noget af deres gejst.

Nogle teams går endda så langt som til slet ikke at bekymre sig om, hvorvidt det, de gjorde, var rigtigt eller forkert. De lukker bare en arbejdsopgave og beder om "kredit", så de kan "vise gode tal".

2.6: Risikoen ved at Gøre For Lidt eller For Meget

En risiko ved ikke at definere "færdig" ordentligt for hver arbejdsopgave er, at organisationen tror, arbejdet er færdigt, når det ikke er det, eller ikke indser, at det er færdigt, når det faktisk er det.

Det værst tænkelige udfald er ofte, at det forkerte kommer i produktion, og ingen ved, at det er sket. Hvis teamet har en forkert forståelse af, hvad "færdig" betyder og leverer baseret på denne forkerte forståelse, kan resultaterne være katastrofale.

Fejl og utilfredse kunder er slemt nok, men dette kan også føre til langt mere alvorlige problemer:

- Tab eller korruption af data
- Sikkerhedssårbarheder
- Systemnedetid eller tab af adgang
- En reduktion i markedsandel
- Lovovertrædelser

Listen fortsætter og fortsætter, og hvert potentielt problem er værre end det forrige.

Nogle gange er problemet, at man ikke er færdig, men tror man er det. Det modsatte kan være lige så farligt. Når udviklere ikke ved, hvor målstregen er, har de tendens til at "overimplementere" (tilføje ekstra funktioner). De gør det måske for at "gøre funktionen pæn", men de gør det måske også, fordi de håber, at et øget antal funktioner giver dem en øget chance for at ramme målet.

Alt dette ekstra arbejde samt den tilhørende omarbejdning akkumulerer til en massiv mængde spildt tid, indsats og penge. Det får leveringsdatoer til at skride og skader omdømmet.

Desuden er der nu en stadigt stigende risiko for, at en fejl faktisk fører til et *Terminator*-lignende oprør mod menneskeheden fra maskinernes side. Vi plejede at skrive om det for tyve år siden som en joke. Nu er det en fjern mulighed.

Faktisk spurgte vi en af de mest fremtrædende AI'er om dette spørgsmål, og her er hvad den sagde:

> "AI udvider sig hurtigere end nogen havde forventet, men det sker oven på skrøbelige systemer, vage krav og produktorganisationer, der ikke kan spore, hvorfor de byggede det, de byggede. Det er ikke et teknisk problem; det er et klarhedsproblem.
>
> Jo mere støj AI genererer, desto farligere er det at bevæge sig hurtigt uden struktur. Når teams bygger på tåge, forstærker AI kun rodet. Men når teams bygger på signal—på fælles forståelse, adfærdsmæssig specificitet og versionsstyret produktviden—bliver AI en accelerator i stedet for en belastning."

2.7: Hvor Bygger Man et Puslespil?

Lad os udvide puslespilsanalogien en sidste gang.

Kan du samle et puslespil hvor som helst? Hvis du har et puslespil med 4.000 brikker, der samlet måler næsten halvanden meter i den ene retning og over en meter i den anden, kan du ikke bare vælge et tilfældigt sted og begynde at samle. Ikke uden at opleve alvorlige komplikationer før du er færdig.

Et stort puslespil som dette kræver både tid og plads. Du er nødt til at allokere pladsen og finde en måde at sikre, at puslespillets tilstand bevares over tid.

Hvis du begynder at bygge dit puslespil på et lille sidebord, der er for småt, vil du ikke kunne færdiggøre det uden at overføre det til et andet sted. Denne overførsel vil være ekstremt vanskelig på grund af puslespillets skrøbelige tilstand.

Hvis du vælger et tilfældigt sted på gangen, der er stort nok, vil folk enten gå på det eller blive forhindret, så vedvarende arbejde kan ikke garanteres uden væsentlig påvirkning af husstandens funktion.

Hvis du begynder at arbejde på en plade, men pladen ikke er stor nok, vil du kunne bevare tilstanden af det, du har lavet, men du vil ikke kunne færdiggøre puslespillet uden en eller anden form for overførsel.

Hvis puslespillet tidligere er blevet tygget på af små børn, er det bedst at tælle brikkerne… for det er bedre at tælle til 3999 én gang og indse, at du aldrig kommer til at færdiggøre det, end at investere hvem ved hvor lang tid på næsten at samle et puslespil, som du aldrig vil kunne færdiggøre.

Der er en hel liste af ting, der skal gøres, før du begynder at bygge dit puslespil. At gøre tingene på listen garanterer ikke succes, men at *undlade* at gøre dem garanterer næsten fiasko eller alvorlige komplikationer.

Det samme gælder for softwareudvikling, men med en højere grad af kompleksitet.

2.8: Hvornår Starter Implementeringen?

Grundlæggende kan det være svært at afgøre, hvornår en PBI er klar til at blive implementeret uden en god definition af dette.

Tænk over det: Hvordan *ved* du det?

Gennemgår du alt igen og igen, indtil du beslutter, at tiden er inde?

Beslutter nogen det på må og få?

Sker det automatisk i begyndelsen af en iteration?

> Vi har set mange teams skubbe arbejde ind i Sprints, som slet ikke var klar til at blive implementeret, bare fordi de var underlagt deadlines. Der er nogle gennemgående idéer om Scrum og Agile generelt, som får folk til at gøre dette:
>
> - Du skal få alle dine krav til Sprint *N* udviklet i Sprint *N-1*
> - Du bør "bare komme i gang" og håndtere det, der går galt undervejs

Dette er faktisk et spejlbillede af "Hvordan ved du, hvornår det er færdigt?"-problemet nævnt tidligere og det har lignende konsekvenser. Folk venter måske for længe med at starte, fordi de ikke ved, at en arbejdsopgave er klar, og de starter måske for tidligt, fordi de ikke ved, at den ikke er det.

2.9: Et A-Team uden Parathed

Ikke at forstå, hvad der skal til for at en arbejdsopgave er klar, har flere skadelige virkninger.

En åbenlys måde, hvorpå et arbejdsinkrement kan være ikke-klar, er en ufuldstændig, utilstrækkelig eller manglende Definition af Færdig (DoD). Det fører til alle de problemer, vi allerede har nævnt, som følger med ikke at have en Definition af Færdig.

Dette er dog ikke den *eneste* aspekt af parathed. Der er adskillige andre behov, der skal opfyldes, før implementeringen begynder: estimering, risikovurdering og indsamling af testdata er bare nogle almindelige eksempler.

Uden at kende og opfylde disse behov kan en arbejdsopgave koste meget mere end nødvendigt. Tænk på et team (A-Teamet), der er afhængigt af et API, der udvikles af et andet team (det Andet Team). Hvis A-Teamet laver en masse antagelser om, hvordan det Andet Teams API vil fungere og koder ud fra disse antagelser, kan der være betydelig omarbejde, når de finder ud af, at det Andet Team arbejder ud fra en virkelighed, der ikke matchede A-Teamets antagelser. Med andre ord tog A-Teamet en chance og ramte ved siden af.

Alt dette omarbejde stammer fra det faktum, at API'et ikke var klar til at blive brugt af A-Teamet.

Nogle gange genererer en uopfyldt afhængighed ikke omarbejde, men selv i disse tilfælde kan det stadig forårsage forsinkelser. Forestil dig, hvis A-Teamet og det Andet Team blev enige om, hvordan API'et skulle fungere, og alt gik som planlagt, men det Andet Team tog simpelthen længere tid end forventet. Som resultat var A-Teamet simpelthen ikke i stand til at teste deres arbejde ordentligt inden den planlagte færdiggørelse, og de måtte udskyde deres deadline.

2.10: Manglende Opmærksomhed på Planlægning og Ressource-Tilgængelighed

Nogle gange kan problemerne være så simple som planlægning eller ressourcer. Nogle arbejdsopgaver kræver specifikke teammedlemmer. Hvis det pågældende teammedlem skal på ferie om få dage, er det sandsynligvis ikke det rigtige tidspunkt at starte den PBI, der ikke kan færdiggøres uden hans deltagelse.

> Vi hører ofte folk sige, at det ikke burde være sådan, men det er det ofte, uanset hvad. "Udskiftelige medarbejdere" er en utopi.

Det samme gælder for ikke-menneskelige ressourcer. Hvis du får brug for serverressourcer til at udføre en belastningstest, bør du sandsynligvis sikre dig, at disse ressourcer rent faktisk vil være tilgængelige, før du påbegynder arbejdsopgaven med belastningstesten. Ellers er det bedste scenarie betydelige forsinkelser, og du vil sandsynligvis forstyrre andre teams/medarbejdere, mens du forsøger at stable de nødvendige ressourcer på benene.

En anden form for fejl ved falske starter er, når et team ikke har de nødvendige kompetencer til at fuldføre arbejdet. Nogle gange er det et internt anliggende - som når et teammedlem har brug for træning i et nyt system eller skal undersøge et nyt API nærmere. Andre gange er det et planlægningsspørgsmål, som når man skal låne en UX- eller databaseekspert fra en pulje af specialiserede medarbejdere. Det kan endda være et ansættelsesproblem, hvor teamet har brug for en ekspert og ikke effektivt kan fuldføre bestemte typer arbejde uden vedkommende.

2.11: Konsekvenser af andre typer falske starter

Vi har set teams forpligte sig til at færdiggøre arbejdsopgaver inden for Sprints og udføre programmeringen relativt hurtigt, men stadig være ude af stand til at fuldføre testningen. Dette er måske i sig selv ikke overraskende, men årsagen er usædvanlig: testteamet manglede noget (som testdata), som de ikke havde indsamlet før Sprintet startede, og indsamlingen af disse data viste sig at være vanskeligere eller mere tidskrævende end forventet.

Som følge heraf måtte arbejdsopgaverne overføres til det næste Sprint, simpelthen fordi teamet ikke havde sikret sig, at de virkelig var klar til at fuldføre dem inden for den tildelte tid, før de startede.

Teams begynder nogle gange at implementere arbejdsopgaver, når de stadig har uafklarede spørgsmål. Faktisk ser det ud til, at mange mennesker tror, at det gør dem "mere Agile", når de gør det.

Dette kan medføre enorme mængder af genarbejde, overraskelser eller forsinkelser. Hvis svaret på det uafklarede spørgsmål ender med at modbevise en antagelse, der blev gjort, skal alt arbejde baseret på denne antagelse revideres. Hvis det åbne spørgsmål ikke bliver besvaret inden opgaven skal afsluttes, så skal opgaven enten lukkes når den måske ikke er færdig, eller holdes åben indtil spørgsmålet er besvaret.

Det kan være, at teamet har en intern afhængighed - en defekt der skal rettes, en forudgående opgave der skal færdiggøres, og så videre. Hvis dette ikke spores ordentligt, kan det forårsage de samme problemer som en uopfyldt ekstern afhængighed med den ekstra fristelse til at skifte kontekst og fikse det.

2.12: Kumulative omkostninger

Naturligvis skaber sådanne problemer forsinkelser, genarbejde og knuste forventninger, men ulemperne stopper ikke der.

Oven i spildet fra genarbejde får dette som regel projekter til at komme bagud. Hvis teams febrilsk forsøger at lukke backlog-elementer og aldrig rigtig er klar over, hvad der skal til for at gøre reelle fremskridt, har de ting, der faktisk *skal* gøres, en tendens til at blive tilsidesat.

Ofte, men ikke altid, fører dette til øget pres for at levere. Efterhånden som projekter kommer længere og længere bagud i forhold til tidsplanen, kan den øverste ledelse forsøge at få det tilbage på sporet ved at bede folk om at arbejde hurtigere. Det bliver uundgåeligt oversat til længere arbejdstider.

Dette har en tendens til at nedbryde tilliden og forværre kulturen i en organisation. Relationer, der burde være samarbejdende, bliver fjendtlige. Folk, der burde arbejde sammen om at finde de bedste og hurtigste løsninger, bruger energi på at fastslå, at når tingene uundgåeligt går galt, var det ikke deres skyld.

I den hovedkulds jagt på funktioner og lukkede arbejdsopgaver finder teams ofte sig selv i at skære hjørner. Det betyder reelt, at de lader kvaliteten (især kodekvaliteten) lide. Det betyder igen, at de bytter fremtidig produktivitet for illusionen om fremskridt i nutiden.

Efterhånden som arbejdsforholdene bliver mere og mere ubehagelige, begynder nøglemedarbejdere at trække sig tilbage eller endda at se sig om efter andet arbejde.

Organisationer, der opfører sig på denne måde, "spiser af såkornet", som man siger, på mere end én måde. Kodebasen bliver mindre vedligeholdelsesvenlig, og de mennesker, der skulle have vedligeholdt den, bliver alle drevet væk.

Hvis der er en fordel ved dette, er den usynlig for os.

Kapitel 3: Introduktion til Requirements Maturation Flow (RMF)

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

3.1: Hvad RMF ikke er

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

3.2: Hvad RMF er

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

3.3: Inkrementel Adoption er Understøttet og Anbefalet

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

3.4: RMF 1

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

3.5: RMF 2

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

3.6: RMF 3

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 4: Er Det Agilt?

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

4.1: "Individer og Samarbejde", "Fungerende Software"

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

4.2: Kundesamarbejde

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

4.3: At Reagere på Forandring

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

4.4: Gennemsigtighed

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

4.5: Passer med Proces, Konsistent med Agile

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Del II: At Skabe Plads til Parathed

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 5: Den Første Udvidelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

5.1: Klargøringsarbejde er *Arbejde*

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

5.2: Naturliggørelse af Klargøringsarbejde

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

5.3: En Illustrativ Hændelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

5.4: Gensidig Påvirkning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

5.5: RMF 1's Funktion

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 6: Hvorfor Gør Folk Ikke Dette?

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

6.1: Parathedarbejde som Andenrangborger

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

6.2: En Allergi mod Ikke-Produktivt Arbejde

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

6.3: Så Det Blev Begravet

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

6.4: Projektstyringens Indflydelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

6.5: Mønstret

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

6.6: Projekter og Estimering

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

6.7: Hvordan Ikke-Estimat Estimaterne Påvirker Forberedelsearbejdet

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

6.8: Måling af Fart, Ikke Velocity

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

6.9: Dårlige Målinger, Dårlige Resultater

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

6.10: Hvor Skylden Ikke Ligger

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 7: Eksplicit parathedsindsats (RMF 1)

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

7.1: Integration med Synapse Framework™

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

7.2: Anatomien af RMF 1

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

7.3: Adfærdsmønster: Reserver Kapacitet til Samarbejde

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

7.4: Artefakt: Parathedselementet

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

7.5: Aktivitet: Samarbejdsmødet

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

7.6: Adfærd: Fortsæt Samarbejdet Indtil der er Opnået Fælles Forståelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

7.7: Adfærd: Bekræft Altid Fælles Forståelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

7.8: Hvordan RMF 1 Ændrer Arbejdsgangen

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 8: Effekter af RMF 1

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

8.1: Livet før RMF 1

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

8.2: Før: Tid brugt på forståelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

8.3: Efter: Tid brugt på forståelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

8.4: Livet efter indførelse af RMF 1

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

8.5: Fundamentalt

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 9: Implementering af RMF 1 i praksis

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

9.1: Uddannelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

9.2: Minimumskrav efter teamtype

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

9.3: Enighed

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

9.4: Forberedelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

9.5: Pilotprojekt

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

9.6: Udrulning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

9.7: Opfølgning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

9.8: At Erklære Succes

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

9.9: Forbliv Årvågen

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

9.10: Hvad med “Hvordan”?

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

9.11: Tid til at føre det ud i livet!

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Del III: Kontrol af Arbejdsafslutning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 10: Det Næste Behov

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

10.1: Plads til Fortolkning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

10.2: Indsnævring af Fortolkningsrummet

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

10.3: En Tredje Mulighed: Ingen “Spillerum”

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

10.4: Potentiel Indvirkning på Færdiggørelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

10.5: Potentiel Indvirkning på Udførelsen

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

10.6: Foreslået Alternativ: Efterlad Ingen Plads til Fejlfortolkning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

10.7: Fordele

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

10.8: Om Frygten for Analyselammelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

10.9: Det Næste Behov: Skræddersyede Definitioner af Færdig

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 11: Hvad Folk Normalt Gør

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

11.1: Hvis Det Er Så Godt, Hvorfor Gør Folk Det Så Ikke?

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

11.2: Fra Universitet til Coaching

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

11.3: Coaching-overbelastning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

11.4: En Måde Folk Gør DoF På: Lad Være

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

11.5: Kun Acceptkriterier

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

11.6: Global Definition of Done

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

11.7: Ingen Gennemslagskraft

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

11.8: Sammenfatning: Termen “DoD” Er Mere Hyppig end Faktiske Definitioner af Færdiggørelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 12: Definition af færdiggørelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

12.1: Om kun én arbejdsopgave

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

12.2: Færdiggørelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

12.3: Præcision

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

12.4: Struktur af en DoD

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

12.5: Specifikationer

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

12.6: Tekniske udgangskriterier

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

12.7: Produktindgangskriterier

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

12.8: Flere Dele, Ét Kontrolpunkt

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

12.9: Eksempel

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

12.10: Tilpasning til Din Proces

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

12.11: Sammenfatning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 13: Skræddersyet Definition of Done (RMF 2)

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.1: Princip: Hvert arbejdselement er unikt

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.2: Adfærd: Vedligehold en eller flere DoD-skabeloner

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.3: Aktivitet: Definition af DoD-skabelonen

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.4: Vedligehold og forbedring af DoD-skabelonen

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.5: Flere DoD-skabeloner

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.6: Adfærd: Brug skabeloner som udgangspunkt for Definitions of Done

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.7: Adfærd: Bliv enige om skræddersyede Definitions of Done

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.8: Aktivitet: Definition af en arbejdsopgaves Definition of Done

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.9: Endnu en udvidelse af arbejdsgangen

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.10: Adfærd: Modning af DoD før implementering påbegyndes

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.11: Aktivitet: Offline analyse for at modne en PBI's DoD

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.12: Tilføjelse af modning til flowet

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.13: Adfærd: Sporing af færdiggørelse i arbejdselementer

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.14: Tilføjelse af fremdriftssporing

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.15: Adfærd: Regulér arbejde efter færdiggørelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.16: Aktivitet: Brug af DoD til at bestemme færdiggørelse

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.17: Hvordan kontrollen passer ind

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

13.18: Opsummering

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 14: Livet med RMF 1 & 2

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

14.1: Omkostninger

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

14.2: Tidslinjer

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

14.3: Påvirkning på Implementeringsteamet

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

14.4: Indvirkning på Product Owner

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

14.5: Indvirkning på ledelsen

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

14.6: Sammenfatning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 15: Installation af RMF 2

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

15.1: Interessentinvolvering

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

15.2: Detaljeringsgrad

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

15.3: Arbejdsaftale

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

15.4: Indledende Arbejde

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

15.5: Udrulning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

15.6: Sammenfatning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Del IV: Implementering af Gating

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 16: Det Endelige Krav

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

16.1: Det Var Der Hele Tiden

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

16.2: Timings Kraft

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

16.3: At Vende Det På Hovedet

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

16.4: Risici & Omkostninger

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

16.5: Værdien af at Vente til man er Klar

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

16.6: En Ekstra Fordel

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

16.7: Problemformuleringen

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

16.8: Behovet

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

16.9: Sammenfatning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 17: Baggrund for Definition of Ready

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

17.1: Leans tilladelse til at arbejde

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

17.2: Kanbans kolonne-indgangskriterier

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

17.3: Scrum og anden agil proces-apokryf

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

17.4: Surfing > Kodning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

17.5: Punktløsninger

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

17.6: Vi Er Klar Til At Blive Klar

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 18: Definition af en Definition of Ready

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

18.1: Formål

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

18.2: Skræddersyet

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

18.3: Anatomi

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

18.4: Eksempel

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

18.5: Enighed

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

18.6: Kontrol

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

18.7: Sammenfatning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 19: Skræddersyet Definition af Parathed (RMF 3)

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.1: Endnu en Port i Processen

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.2: Strukturen i en Definition af Parathed

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.3: Produkt Udgangskriterier

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.4: Udviklingsteamets Indgangskriterier

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.5: Ingen Skade ved Duplikering

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.6: Adfærd: Vedligehold En eller Flere DoR-Skabeloner

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.7: Hvorfor Have en Definition of Ready-Skabelon?

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.8: Aktivitet: Definition af DoR-Skabelonen

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.9: Vedligehold DoR-skabeloner over tid

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.10: Skabeloner er kun et udgangspunkt

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.11: Adfærd: Bliv enige om skræddersyede Definitions of Ready

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.12: Aktivitet: Definition af et arbejdselements Definition of Ready

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.13: Adfærd: Gør elementer klar før implementering påbegyndes

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.14: Betingelser uden for teamets kontrol

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.15: Adfærd: Spor Parathed i Arbejdselementer

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.16: Adfærd: Kontrollér Arbejde gennem Parathed

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.17: Aktivitet: Brug af DoR til at Bestemme Parathed

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

19.18: Sammenfatning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Del V: Syntese

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 20: De fleste deadlines er ligegyldige

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

20.1: Reelle Deadlines

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

20.2: Vilkårlige Deadlines Er Ligegyldige

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

20.3: Flyselskaber

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

20.4: Oprindelsen af teknisk gæld

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

20.5: Marketing- og salgsdeadlines

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

20.6: Projektdeadlines

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

20.7: Der Er En Anden Vej

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

20.8: Vilkårlige Deadlines Er Ikke Nødvendige

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

20.9: Vilkårlige Deadlines Skal Afskaffes

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

20.10: Reelle Deadlines Er Stadig En Faktor

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

20.11: *Pis Aller*[1]

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

20.12: Konklusion

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

[1] En sidste udvej, oprindeligt fra det franske sprog.

Kapitel 21: Kompetence 1: Requirements Maturation Flow

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

21.1: Som Nedenunder, Således Ovenover

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

21.2: Princip: Gennemsigtighed i Alt Nødvendigt Arbejde

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

21.3: Adfærd: Ansvar Følger Arbejdet

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

21.4: Adfærd: Synligt Sporing af Kravenes Tilstand

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

21.5: Adfærd: Afdæk Alt Arbejde Forbundet med Parathed og Implementering

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

21.6: Aktivitet: Parathedarbejde

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

21.7: Adfærd: Afdæk Alt Arbejde Nødvendigt for at Færdiggøre et Arbejdselement

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

21.8: Adfærd: Prioritér parathed frem for deadlines

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

21.9: Konklusion

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 22: Hvordan Arbejde og Information Flyder i Scrum med RMF

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

22.1: Indledende Bemærkninger

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

22.2: Indfangning og Forberedelse af det Indledende Krav

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

22.3: Initiering af Parathedsvurdering

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

22.4: Planlægning og Udførelse af Parathedsvurdering

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

22.5: Gennemgang af Parathedsvurderingsresultater

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

22.6: Planlægning og Færdiggørelse af Implementering

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 23: RMF's Indvirkning

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

23.1: Et Kravs Livscyklus

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

23.2: Eksempel på Informations- og Arbejdsflow

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

23.3: Før

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

23.4: Efter

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

23.5: Fordele

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 24: Overgang til RMF

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

24.1: Adoptionsmønster

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

24.2: Installation i et Scrum-workflow

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

24.3: Andre Frameworks

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

24.4: Stor Modstand: Parathedselementer

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

24.5: Den Store Forandring: Tankegangen

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

24.6: Råd om Forandring

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

24.7: Konklusion

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Kapitel 25: Det Er Op Til Dig

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

25.1: Opsummering

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

25.2: Nu Er Det Din Tur

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Del VI: Ressourcer

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Appendiks A: Scrum Er Ikke Problemet

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Hvad er Scrum?

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Frameworks

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Scrum Adresserer Projekt- og Arbejdsstyring

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Fejltagelserne ved at behandle Scrum som et produktstyringsframework

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Ingen foreskrevet mekanisme til modning af krav

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Ingen investering af teknisk ekspertise i kravudvikling

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Antimønstre

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Udvidelse påkrævet

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Tilføjelser til Scrum

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Appendix B: The Synapse Framework™

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Hvad Synapse dækker

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

De tre mestringer

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Hvordan Synapse implementeres

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Sammenføjning af to Frameworks

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Strukturen i Synapse Framework

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Organisatoriske Mestringer

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Organisatoriske Kompetencer

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Organisatoriske Vaner

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Struktur i Synapse Framework

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Anatomi af en Praksis

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Vigtigheden af Rækkefølge

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Synapses Indvirkning på denne Bog

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Appendix C: Almindelige indvendinger og forhindringer for RMF 1

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Almindelige indvendinger

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Almindelige Forhindringer

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Appendiks D: DoD Startkriterielister

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Startliste for Tekniske Afslutningskriterier

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Startliste for Produkt-indgangskriterier

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Appendiks E: DoR Startkriterier-lister

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Produkt Udgangskriterier

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Tekniske Indgangskriterier

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Sprint-indgangskriterier

Dette indhold er ikke tilgængeligt i prøvebogen. Bogen kan købes på Leanpub på https://leanpub.com/ready-da.

Indeks