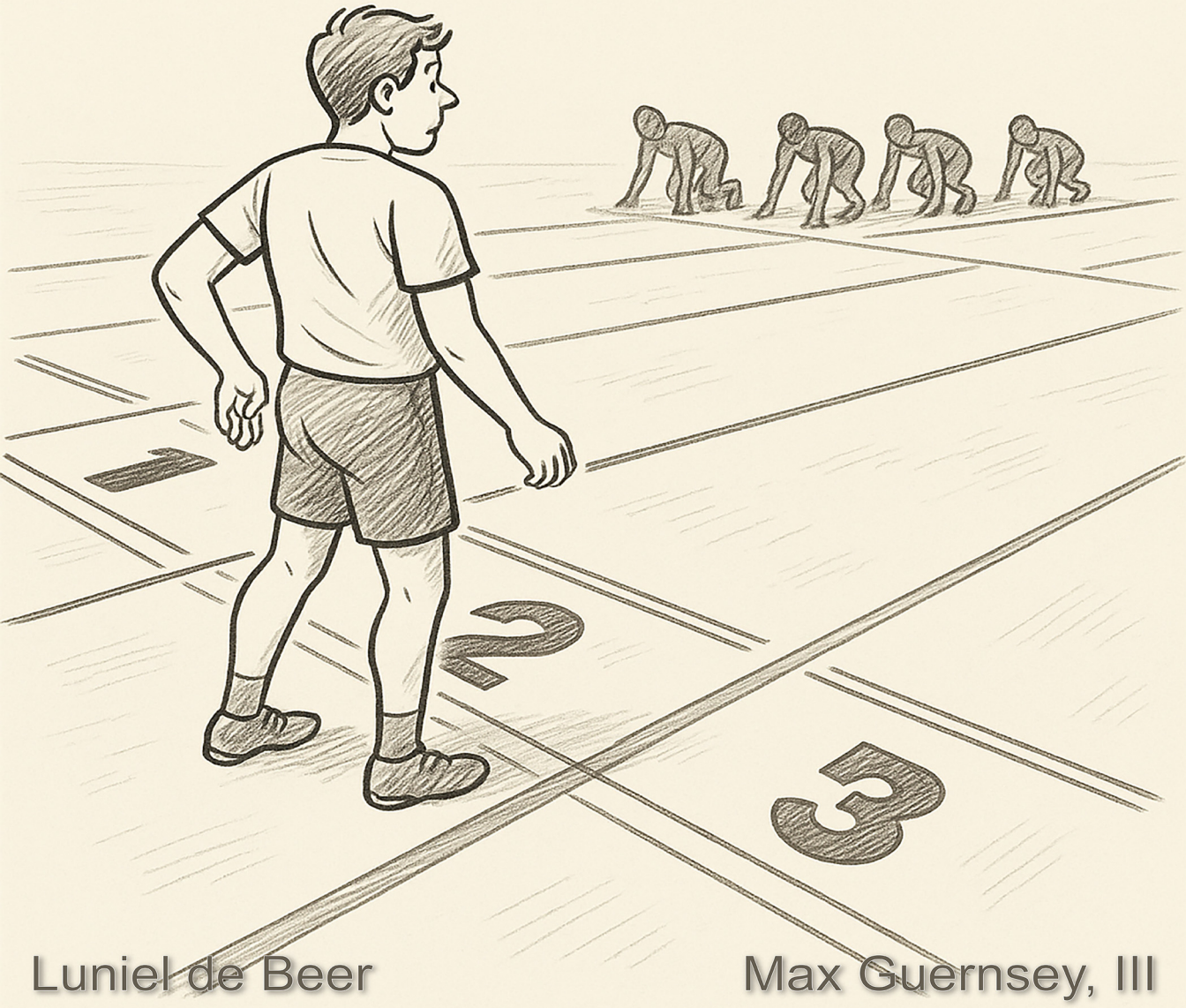
READY

WHY MOST SOFTWARE PROJECTS FAIL AND HOW TO FIX IT



Luniel de Beer

Max Guernsey, III

Norsk Utgave

Del denne boken på Twitter!

Vennligst hjelp Luniel de Beer og Max Guernsey, III ved å spre ordet om denne boken på Twitter!

Den foreslåtte tweeten for denne boken er:

Jeg har nettopp kjøpt Ready — en bok for programvareutviklingsledere og team som ønsker å eliminere omarbeid, overføring og feiljustering i programvareleveranse. #CodeReady

Den foreslåtte hashtagen for denne boken er #CodeReady.

Finn ut hva andre sier om boken ved å klikke på denne lenken for å søke etter denne hashtagen på Twitter:

#CodeReady

Ready (Norsk Utgave)

Hvorfor de fleste programvareprosjekter mislykkes og hvordan løse det

Luniel de Beer og Max Guernsey, III

Denne boken er tilgjengelig på https://leanpub.com/ready-nb

Denne versjonen ble publisert 2025-10-22

Dette er en Leanpub-bok. Leanpub gir forfattere og utgivere mulighet til å bruke Lean Publishing-prosessen. Lean Publishing er prosessen med å publisere en bok under utvikling ved hjelp av enkle verktøy og mange iterasjoner for å få tilbakemeldinger fra lesere, justere kursen til du har den rette boken og bygge momentum når du lykkes.

Til minne om Johann van Aardt, som gjenkjente min lidenskap, introduserte meg for virkelig programmering, og hjalp meg å finne veien til et nytt hjem. Og til mine foreldre, hvis urokkelige støtte—fra min første klient til mitt første hjem i USA—gjorde alt dette mulig.

—Luniel

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

—Max

Innhald

Om denne boken

Ready er en bok for alle som er involvert i programvareutvikling og som er lei av **underleveranser**, **kronisk omarbeiding** og **uklare krav**.

Du har kanskje allerede prøvd å investere i teamets gjennomføringsevner, forbedret implementeringen av prosessrammeverket ditt, eller oppgradert koden og fortsatt trenger mer forbedring.

Dette er fordi den viktigste begrensningen for de fleste programvareutviklingsteam ikke er teamets ferdigheter, men kravmodenhet. **Selv modne team med de riktige ferdighetene sliter fortsatt** når de jobber med umodne krav.

Ready introduserer RMF (Requirements Maturation Flow/kravmodningsprosess), en praktisk og dypt strukturert tilnærming for å samkjøre Produkt og Engineering uten å erstatte din eksisterende prosess.

Enten du bruker Scrum, Kanban eller noe tilpasset, hjelper RMF deg med å **stabilisere omfanget**, **eliminere overføringer** og **levere det som faktisk betyr noe**.

Hvis teamene dine føler seg fast på kanten av "nesten ferdig", vil denne boken vise deg hvordan du kan **bryte syklusen** og fjerne blokkering for teamet/teamene dine **for godt**.

Hvem dette er for

Denne boken er bokstavelig talt for alle som er involvert i programvareutvikling. Alle fra utviklere til produktsjefer og fra individuelle bidragsytere til ledere.

Denne boken er for deg hvis du er involvert i programvareutvikling og har lagt merke til at et team du jobber med eller i har ett eller flere av følgende problemer:

- Arbeid blir ofte overført fra en iterasjon til den neste
- Implementeringsteam føler at de prøver å treffe bevegelige mål
- Arbeid holdes åpent for lenge
- Arbeid markeres som avsluttet, men ting er egentlig ikke ferdig
- Utført arbeid samsvarer ikke med forventningene
- Arbeid genererer regelmessig et stort antall feil
- Store mengder arbeid må gjøres om igjen på regelmessig basis

Hvis noen av disse problemene høres kjente ut, kan *Ready* hjelpe.

Hvordan bruke denne boken

Denne boken ble designet for å være praktisk. Det er ikke en teoretisk avhandling eller en strategipresentasjon—det er en bruksanvisning for å installere RMF (Requirements Maturation Flow) basert på reelt klientarbeid og felttestet under reelt leveransepress.

Kapitlene er skrevet i rekkefølge, men RMF selv er modulær. Den består av tre grunnleggende praksiser:

- RMF 1: Samarbeid for felles forståelse
- RMF 2: Portvoktning av ferdigstillelse ved bruk av Bespoke Definitions of Done (skreddersydde definisjon av ferdig)
- RMF 3: Portvoktning av implementering ved bruk av Bespoke Definitions of Ready (skreddersydde definisjon av klar)

Hver del, eller "Vane" som vi kaller dem, står på egne ben, men de bygger på hverandre. Boken er designet for å hjelpe deg med å tilnærme deg dem én om gangen, i rekkefølge. Denne strukturen gjenspeiler hvordan vi anbefaler team å ta i bruk RMF i praksis—hvor hver Vane først legges til etter at den forrige fungerer.

Dette unngår å overbelaste team og gir hver endring best mulig sjanse for å bli varig. Du vil lære mer om hvordan du gjør dette fra og med Kapittel 9.

Hvis du ser etter hjelp—enten det er råd, veiledning eller noen å snakke med lederteamet ditt—ikke nøl med å ta kontakt med oss direkte.

Og hvis du ser etter formell støtte for å implementere RMF, tilbyr Producore en full serie med programmer designet for å veilede adopsjon steg for steg. Du kan lære mer på https://ready-book.link/rmf.

Om forfatterne

Luniel de Beer er skaperen av Requirements Maturation Flow (RMF), et praktisk system for å fikse gapene mellom produktintensjon og teknisk gjennomføring. Han har over 15 års erfaring med å lede agile transformasjoner, brobygging mellom produkt og engineering, og å hjelpe team med å levere med klarhet og selvtillit.

Luniel er også opphavet til Producores Capability Management-system, en sporbar og skalerbar tilnærming til modellering av produktegenskaper. Han utviklet PKB-Driven Development (PKBDD), et versjonskontrollert system for håndtering av vedvarende produktkrav. Disse verktøyene utgjør en del av et større leveranserammeverk utviklet hos Producore.

Max Guernsey, III er programvarearkitekt, pedagog og medgrunnlegger av Producore, et konsulentfirma dedikert til å fikse leveransesvikt gjennom strukturell og teknisk grundighet. Med over to tiår med erfaring innen objektorientert design, refaktorering, testdrevet utvikling og designmønstre, har han både levert forretningskritiske systemer og veiledet engineering-team i stor skala. Hans arbeid kombinerer dype tekniske praksiser med atferds- og prosessforandring for å hjelpe organisasjoner med å oppnå bærekraftig leveranseeksellense.

Max bidro betydelig til PKBDD og ledet utviklingen av Producores tilnærming til atferdsdrevet utvikling (BDD) gjennom sin dyptgående ekspertise innen atferdsspesifikasjon.

Sammen integrerer arbeidet deres tydelighet, sporbarhet og portkontroll til et sammenhengende system for programvareleveranse som skalerer fra teampraksis til organisatorisk kapasitet.

Forord

Merknad til tekniske ledere

Hvis du er en seniorleder i en teknisk organisasjon, mangler du sannsynligvis ikke innsats, disiplin eller smarte mennesker. Likevel står prosjekter fortsatt stille. Mål glipper. Forventninger blir ikke innfridd. Ikke fordi teamene dine er late – men fordi noe grunnleggende er ødelagt i måten arbeid blir definert, formet og levert på.

Denne boken er ikke en ledelsesguide. Den er et verktøy for strukturell diagnose. Den avslører hva som faktisk skjer inne i teamene dine – hvorfor "nesten ferdig" fortsetter å bli til "ikke ferdig," og hvorfor lokal fremgang så sjelden fører til strategiske resultater.

Du ser kanskje ikke *deg selv* i disse sidene. Men hvis teamene dine ikke kan levere det du trenger, vil du se *dem*. Og når du gjør det, vil du endelig ha språket – og systemet – for å fikse det.

Fra Luniel

Først og fremst ville denne boken ikke vært mulig uten Max, hvis evne til å se gjennom tåken og avfallet og destillere en idé til dens essens er helt utrolig.

Hvordan kom vi hit?

Når jeg ser tilbake, tror jeg det er fordi jeg alltid har ønsket å forstå hvordan ting faktisk fungerer. Enten det var religion, ernæring eller programvareutvikling, støtte jeg på det samme problemet: overfladiske svar som ikke holdt under press. Så jeg fortsatte å grave – spurte ikke bare hva vi gjør, men hvorfor, og hva som mangler når det ikke fungerer.

En av de tidligste sprekkene i systemet dukket opp i en rolle hvor jeg hadde tre hatter: Scrum Master, Product Owner og Utviklingsleder (!!) for et team som leverte datatjenester i et velkjent teknologiselskap. Vi gjorde det Scrum sa – korte Sprinter, brukerhistorier i en backlog, planlegging på en halv dag – men hver gang vi startet en ny Sprint, møtte vi motstand. Teamet forsto ikke problemet fullt ut, vi måtte revurdere og revidere krav midt i Sprinten, forebyggbare avhengigheter dukket opp og forsinket oss, og viktige trinn ble oversett.

Så jeg begynte å gjøre noe annerledes. Jeg samlet teamet og interessentene i et rom for hver brukerhistorie, gikk gjennom problemet i detalj, brainstormet løsningen sammen, og først da skrev jeg brukerhistorien. Sprint-planleggingen krympet til en time, og leveringssuksessen vår økte dramatisk.

Sakte begynte jeg å innse at suksess ikke kommer fra å jobbe hardere inne i Sprinten. Det kommer fra strukturen du etablerer *før* den starter.

Senere, etter å ha hørt Jeff Sutherland snakke om "Definitions of Ready", visste jeg at det var noe verdifullt der – men det var ikke nok. Min erfaring med krav, UX, UI, forskning og senere med BDD viste meg at forskjellige arbeidsoppgaver krever forskjellige typer klargjøring. Noen trenger atferdsspesifikasjoner. Noen trenger systemtilgang. Noen trenger full kapabilitetssporing.

Og alle trenger delt forståelse som faktisk er bekreftet – ikke antatt.

Etter hvert som jeg jobbet med flere team, så jeg det samme mønsteret overalt: manglende trinn, uoppfylte avhengigheter, team som gjorde sitt beste men konstant måtte kave for å fikse problemer som burde vært forhindret. Selv gode team slet – ikke fordi de var svake, men fordi de manglet en struktur som gjorde klargjøring eksplisitt.

Resultatet av all denne læringen, iterasjonen og frustrasjonen er et strukturert system for å håndtere klargjøring.

Det er det denne boken handler om.

Jeg håper den gir deg klarhet om hvor de virkelige problemene ligger, og hvordan du kan fikse dem. Jeg håper den gir deg språk til å forsvare praksiser som kan virke "ekstra" men som faktisk er essensielle. Og mest av alt håper jeg den hjelper team å levere med mindre stress, færre overraskelser og langt bedre resultater.

Hvis vi får dette riktig, vil vi spare bransjen for milliarder av kroner.

Men enda viktigere, vi vil gi folk forstanden tilbake.

Fra Max

Jeg har jobbet med dette problemet fra forskjellige vinkler i flere tiår, men fremgangen min var hemmet helt til jeg møtte Luniel.

Dette er fordi jeg, før jeg kjente ham, grunnleggende sett tilnærmet meg problemet som et teknisk problem. Jeg fokuserte på å hjelpe team med å ta i bruk ting som Test-Driven

Development (TDD), refaktorering, avansert programvaredesign og senere Acceptance-Test-Driven Development (ATDD) eller Behavior-Driven Development (BDD).

I de fleste av disse tilfellene ble problemet som tas opp i denne boken behandlet som en implementasjonsdetalj for å etablere disse tekniske praksisene.

Dette betyr ikke at jeg ikke lenger verdsetter de tekniske praksisene. Jeg tror fortsatt de er dypt viktige, men de adresserer ikke direkte problemet med klargjøring i programvareutvikling. I stedet *avdekker* de dette problemet, og folk slenger deretter på en lapp på prosessen sin for å adressere det "akkurat nok" til å støtte de tekniske praksisene de prøver å implementere.

Jeg vil også ta opp spørsmålet om hvem denne boken kan hjelpe. Det korte svaret er "sannsynligvis nesten alle i programvareutvikling", men det virkelige svaret inneholder nyanser som hjelper med å tilpasse det til ulike miljøer uten å endre den grunnleggende betydningen.

Det finnes team som trenger løsningen som presenteres i denne boken. Du vil møte en forenklet versjon av et slikt team i kapittel 1.

Det finnes også team som ikke strengt tatt **trenger** et system som det vi foreslår, men som likevel kunne ha nytte av det.

Det beste teamet jeg noensinne har jobbet med—lett et fullt standardavvik over det **nest beste teamet**, om ikke to—befant seg i utkanten av Central Oregon. De presterte så høyt at de kunne overkomme fraværet av et slikt system gjennom ren og skjær kompetanse. Likevel sa min leder på den tiden, Tom Barreras, en gang noe lignende som "Jeg har lagt merke til at brukerhistoriene våre går bedre når vi bruker litt tid på å snakke om testene på forhånd."

Dette var igjen noe jeg på den tiden så gjennom linsen av testutvikling og teknisk gjennomføring, men nå vet jeg at det var enda en indikator på at *beredskapsnivå* var en faktor som påvirket teamet… det spesifikke teamet var bare så dyktig og raske til å reagere at de kunne lykkes ved å respondere på hindringer etter hvert som de oppsto, i stedet for å forhindre dem i utgangspunktet.

Selv om du er den typen person som ikke strengt tatt *trenger* å bekymre deg for beredskapsnivå fordi du kan overkomme det, eller du jobber med et team av samme kaliber, kan du fortsatt dra nytte av innholdet i denne boken.

Del I: Noe Mangler

*Når det ikke hjelper å gjøre de samme tingene bedre, se etter det som **ikke blir gjort**.*

Kapittel 1: Det skjulte problemet

Dette er en sann[1] historie om en bank. Vi kaller den bare "Banken". Det er en type føderert kredittinstitusjon som fungerer som en del av USAs nasjonale finansielle infrastruktur.

Vi (Luniel og Max) ble hentet inn til Banken fordi de slet med å levere et programvareprosjekt. Det var et av de mest dysfunksjonelle miljøene vi noensinne hadde sett, og det er derfor vi valgte dette som åpningscasestudien: hvis meningsfull endring var mulig i Banken, er det mulig hvor som helst.

En kort merknad om prosjekter

Når vi bruker begrepet "prosjekt" i denne boken, mener vi det i konteksten av prosjektledelse. Selv om det finnes ulike oppfatninger om hva ordet betyr, bruker vi definisjonen fra Project Management Institute:

> "Et prosjekt er et **midlertidig** foretak iverksatt for å skape et unikt produkt, tjeneste eller resultat."

Dette betyr at et prosjekt har en definert begynnelse og slutt. Når et prosjekt avsluttes, arkiveres prosjektkunnskap og artefakter, teammedlemmer frigjøres, og kontrakter sluttføres.

I denne boken handler et prosjekt fundamentalt sett om gjennomføring. De fleste prosjekter, som definert av PMI, begynner med gjennomførbarhet eller design. Visjon og strategi er allerede på plass når et prosjekt igangsettes.

Et prosjekt fødes fra denne visjonen og strategien, og lykkes eller mislykkes basert på om de tiltenkte målene realiseres—*ikke* på om disse målene var de riktige.

Du bruker kanskje ordet "prosjekt" annerledes, og det er greit. Bare vit at når vi bruker det, refererer vi til definisjonen og konteksten over.

Ikke noe av dette er ment å antyde at vi støtter bruken av prosjektledelse for programvareutvikling. Tvert imot. Men vi erkjenner at det likevel *blir* brukt. Vi tar for

[1] Vi har endret identifiserende detaljer for å beskytte personvernet til personene og institusjonene det gjaldt.

> oss det problemet senere, i Kapittel 6.

1.1: Den klassiske omskrivingen

Banken holdt på å skrive om sin lånetilbakebetalingsportal av flere grunner.

Det gamle systemet, en komplett C#/.NET-løsning, var fullt av feil. I tillegg til å redusere kundetilfredsheten, genererte det også en endeløs strøm av svært kostbare støttehendelser hvor noen måtte manuelt manipulere databasen for å korrigere en feil gjort av systemet.

Det gamle systemet var også nedslitt fra et vedlikeholdsperspektiv. Det var nesten umulig for ingeniørene å gjøre meningsfulle endringer, og selv når de kunne, var det et ekstremt risikabelt foretagende.

Omskrivingen skulle endre på det.

Det nye systemet skulle fortsatt ha en C#/.NET-backend, men den skulle være fullstendig dekket av tester. Frontenden skulle implementeres i OutSystems, en populær lavkode-løsning som lar en organisasjon definere en applikasjon på ett sted og få en nettapp, en Android-app og en iOS-app automatisk generert når de bestemmer seg for å publisere endringene sine.

Håpet var at det nye systemet skulle være feilfritt, både forbedre kundetilfredsheten og redusere støttekostnadene betydelig.

De håpet også at omskrivingen skulle fjerne hindringene for utviklerne—med kombinasjonen av en mer disiplinert tilnærming til backend og lavkode-tilnærmingen til frontend som skulle redusere kostnadene og risikoen ved nye funksjoner betydelig.

En fin bieffekt av å flytte til OutSystems var at de ville få en ren, moderne mobilapp på begge de store plattformene.

Det var drømmen da de hadde startet tre år før begynnelsen av denne historien. Realiteten var at teamene så langt ikke hadde levert **noe som helst**.

1.2: Perspektiver på problemet

Da vi snakket med ledergruppen, hørte vi en helt naturlig frustrasjon over at de hadde gjort så store investeringer uten absolutt noen strategisk bevegelse.

De hadde prøvd alt, fra deres perspektiv. De hadde endret personale, økt bemanning, endret budsjett, økt press og hentet inn en parade av konsulenter (hvor det ble sterkt antydet at vi var halen). Ingenting så ut til å gjøre det bedre—i hvert fall ikke på en måte de kunne måle, fordi alt de så var at "nålen" sto på null ett kvartal og fortsatt sto på null neste kvartal.

De ville ikke ha mer "usynlig fremgang". De ville ha *resultater*.

Da vi snakket med ledelsen i Produktorganisasjonen, fikk vi en litt annerledes (men fortsatt lignende) historie fordi de jobbet mer direkte med Engineering.

Det var ikke det at teamene ikke gjorde noe, ikke for dem i hvert fall. Det var at teamene aldri gjorde det de ble bedt om å gjøre. Det var nærmest garantert: uansett hvor enkel forespørselen var og uansett hvor tydelig den ble formulert, ville man ende opp med noe helt annet når tiden kom for å evaluere det teamene hadde laget.

Det hadde kommet til det punktet hvor den faste spøken var noe sånt som "Vi må finne ut hvordan vi skal be om det vi ikke vil ha, så vi i det minste har en sjanse til å få det vi faktisk vil ha."

Tekniske ledere så ting ganske annerledes.

For dem var problemet at Product ikke leverte handlingsorienterte krav og at Product ikke leverte *nok* krav. Hvis Product bare kunne "følge programmet", ville teamene være i stand til å levere det de ønsket i tide og innenfor budsjettet.

De hadde gjort betydelige investeringer i moderniseringen av hvordan kode ble skrevet og levert, og i deres øyne leverte ikke Product tydelige nok krav.

Da vi snakket med andre konsulenter (som hadde henvist oss til organisasjonen), pekte de med rette på dysfunksjonen de observerte: Alle virket veldig fokusert på å skylde på andre. Grunnen til at de brakte oss inn i utgangspunktet var at de var bekymret for bemanningsstrategien og ønsket en evaluering av enkeltbidragsyterne, men de så på fingerpeking og kontrollbehov på ledelsesnivå som hovedkilden til problemene.

1.3: Vår undersøkelse

Vårt opprinnelige oppdrag var å evaluere teamene og forsøke å hjelpe dem med å oppgradere ferdighetene sine om nødvendig, så vi begynte å se nærmere på menneskene som jobbet i frontlinjen.

Det var definitivt rom for forbedring.

Enkeltbidragsyterne på Product-siden hadde egentlig ikke de nødvendige ferdighetene. I realiteten var de for det meste prosjektledere som hadde blitt kastet inn i rollen som Product Owner (PO) eller Product Manager.

Som følge av dette skrev halvparten av dem "løse" krav og aksepterte deretter (bokstavelig talt) alt teamene gjorde i den iterasjonen uten noen kritisk analyse. Den andre halvparten skrev samme type krav og hevdet deretter at teamene "burde ha visst" ting de aldri hadde snakket om, og holdt arbeidsoppgavene åpne mer eller mindre på ubestemt tid.

Banken brukte til å administrere og spore arbeidsbunken sin. Selv om det vi dekker i denne boken stort sett er uavhengig av Scrum, bruker vi Scrum-terminologi gjennomgående fordi flertallet – eller i det minste et betydelig antall – av teamene bruker Scrum.

Definisjon: Scrum

Scrum er et lettverdig rammeverk som hjelper mennesker, team og organisasjoner med å generere verdi gjennom adaptive løsninger for komplekse problemer. Kort fortalt:

1. En Product Owner prioriterer arbeidet for et komplekst problem i en Product Backlog
2. Scrum-teamet omdanner et utvalg av arbeidet til verdifulle leveranser i løpet av en Sprint
3. Scrum-teamet og interessentene inspiserer resultatene og justerer for neste Sprint
4. Gjenta

Hvis du er ukjent med Scrum og terminologien, anbefaler vi at du ser på 2020-versjonen av Scrum Guide. Det er en rask og opplysende lesning.

På samme måte fant vi ut at de tekniske teamene lå godt under det typiske nivået når det gjaldt kodingsferdigheter (-2σ, i beste fall) og i tillegg var svært motstandsdyktige mot endring. Som en naturlig konsekvens var kodekvaliteten elendig.

Likevel var dette ikke grunnen til at de ikke leverte, ifølge teamene. For dem var det vage krav og endringer midt i Sprint fra Product som ødela prosjektet.

…og ingen snakket engang om det større problemet, det som virker så absurd at det høres oppfunnet ut før du har opplevd det selv.

De tekniske teamene hadde en vane med å ikke forstå et krav, bygge noe tilfeldig, og deretter kreve anerkjennelse for å ha "fullført en arbeidsoppgave".

Vi snakker ikke om små misforståelser. Vi snakker om total frakobling: Vi ba dem deaktivere anvendelse av midler til hovedstol under visse omstendigheter, og de deaktiverte i stedet muligheten til å legge til en sekundær bekreftelsese-post.

Så fortalte de oss at det var dette vi hadde bedt om.

1.4: Dypere undersøkelse

Begge kompetansegapene kunne håndteres, men vi var skeptiske til om de var de virkelige hindringene.

Det var noe annet som ikke stemte, så vi gravde dypere. Vi begynte med dette spørsmålet: Hvorfor tok det så lang tid å skrive krav, og hvorfor ga det så dårlige resultater?

En grunn er at kunnskapen som trengtes for å skrive meningsfulle krav var en mangelvare. Litt av den var fordelt mellom tekniske team og Product-teamene.

Noe av det var innbakt i koden til legacy-systemet. Noe av det var helt borte. Det meste var imidlertid lagret som erfaringsbasert kunnskap hos fageksperter spredt utover bankens ulike avdelinger. Dette betydde at å utarbeide et krav som faktisk fremmet et strategisk mål var en ekstremt arbeids- og tidkrevende aktivitet.

I kontrast til dette sto et umettelig behov for funksjonalitet fra en funksjonsfattig ledergruppe. Mandatet var "hold utviklerne i arbeid—fyll dem opp med krav". Fokuset lå på mengden krav for å holde teamene opptatte—et perspektiv som var stikk i strid med omsorgen og tiden som trengtes for å definere et krav som ville "gi resultater".

1.5: Å Gjøre Ting Bedre Gjorde Det Ikke Bedre

Dette er alle problemer som kan løses, men å løse dem hjalp ikke.

Tidligere forbedringer i programvareutviklingsteknikker hadde ikke hjulpet teamene med å levere, men Max gjorde en innsats for å hjelpe teamene med å forbedre seg ytterligere.

Han introduserte revolusjonerende konsepter fra programmeringsdoktriner fra midten av århundret som "ikke kopier og lim inn den koden 27^2 ganger, legg den i en funksjon og kall

[2]Dette er ikke bare ikke hyperbol, det er ikke engang det verste tilfellet. I ett tilfelle var det nesten hundre eksakte duplikater av samme algoritme.

den i stedet". Dette forslaget alene forbedret kvaliteten på ny kode dramatisk og lot dem begynne å forbedre kvaliteten.

Det og andre grunnleggende kodingsråd hjalp dem med å skrive bedre kode som de lettere kunne vedlikeholde i fremtiden.

…men det hjalp ikke med å drive prosjektet fremover.

På produktsiden klarte Luniel å introdusere BDD og sikre at produkteierne grundig undersøkte krav før de ble overlevert til teamene.

Han fikk teamene til å samarbeide om dem og bruke dem til å evaluere hvorvidt et produktbacklogelement (PBI) virkelig var ferdig.

Definisjon: Produktbacklogelement (PBI)

> Et produktbacklogelement (PBI) er en avgrenset arbeidsenhet i produktbackloggen som representerer en potensiell endring, tillegg eller forbedring av produktet. PBIer kan ta mange former—funksjon, feilretting, teknisk forbedring, forskningsoppgave, osv.—og defineres av deres bidrag til produktverdi.
>
> Mange team refererer til PBIer som "brukerhistorier" eller "historier", men den korrekte Scrum-termen er "produktbacklogelement" eller "PBI". Når en PBI er forpliktet til en sprint, er den også en del av sprintbackloggen. For enkelhets skyld og nøytralitet bruker vi "produktbacklogelement" eller "PBI" for å referere til enhver arbeidsoppgave som Scrum-teamet håndterer—uansett om du kaller det et produktbacklogelement, sprintbacklogelement (SBI), brukerhistorie, historie, arbeidsoppgave eller backlogelement.

Det skapte *klarhet*, men det skapte ikke *flyt*.

Med hjelp fra partnerkonsulentene som hadde brakt oss inn, klarte vi å (midlertidig) lette det absolutt knusende presset som var lagt på teamene og kravforfatterne.

Det kan ha hjulpet med å skape litt tillit, men det ga ingen konkrete resultater.

Vi begynte til og med å bygge en kunnskapsbase som hjalp folk med å spore opp forretningskunnskapen de trengte for å skrive krav og identifiserte stedene hvor det var hull i denne kunnskapen.

Det økte hastigheten på kravskriving, men det fikk ikke produktet ut døren.

Etter måneder med samhandling hadde vi hjulpet ledelsen med å *se* hvor de var, men de var ikke i nærheten av målene sine. Og de kom ikke nærmere.

De var i ferd med å gå tilbake til sin gamle strategi med å "laste opp" teamene for å sikre at de alltid var opptatte.

1.6: En elimineringsprosess

Det ville ha vært enkelt å bare gi opp og si "dette er håpløst". Det var mange unnskyldninger man kunne falle tilbake på:

- Utviklingsteamet hadde lav kompetanse (det hadde de)
- Kravforfatterne hadde feil kompetansesett (det hadde de)
- Ledelsen knuste teamene med urealistiske forventninger (det gjorde de)
- Organisasjonen manglet kritisk forretningskunnskap som var nødvendig for å fungere (det gjorde den)
- Lederne ser ikke ut til å stole på hverandre (det gjorde de ikke)

Alt dette var sant. Likevel var det gjort forbedringer på alle disse områdene og ingen av dem så ut til å gjøre det grunnleggende problemet bedre. Ingen av dem hjalp utviklingsteamene, Produkt, ledelsen eller toppledelsen med å komme nærmere målene sine.

…og det er nettopp der hintet til løsningen ligger: Alle problemene som er listet opp tidligere var problemene folk allerede kunne se.

Hvis variablene du kan se ikke utgjør noen forskjell, må det være en variabel du **ikke ser** som gjør det.

Det virkelige problemet var utfordringen ingen engang visste eksisterte.

1.7: På jakt etter den egentlige syndebukken

I søket etter den egentlige syndebukken - det som virkelig holdt banken tilbake fra å nå sine mål - måtte vi begynne et sted.

Et fornuftig sted å se var på hva de mislykkede PBI-ene (de fleste av dem) hadde til felles.

Vi begynte med å eliminere tingene vi kunne se ikke var felles fordi de varierte stort:

- Hvilken del av systemet: noen PBI-er påvirket bare backend, andre bare frontend, mens andre igjen påvirket begge deler
- Hvilket team som utførte arbeidet - det så ikke ut til å spille noen rolle hvem som gjorde arbeidet, det var høy sannsynlighet for å mislykkes
- Hvilken PO som forfattet arbeidet - det samme som med teamene

Så begynte vi å se på tingene som var felles. Listen var ikke lang, men heller ikke kort:

- Utviklingsteamene
- Produkteierne
- Ledelsen
- Kulturen
- Utviklingsmiljøet
- Utviklingspraksisene
- Kravspesifiseringsteknikken
- Domenet (finans)
- Avhengighetstjenestene

Mange av disse kunne også avskrives umiddelbart. Teamene, produkteierne, ledelsen, kulturen og utviklingsmiljøet hadde alle nylig blitt forbedret uten noen reell innvirkning på meningsfylt output. Vi hadde personlig hjulpet med å forbedre utviklings- og kravspesifiseringspraksisene og bekreftet at disse forbedringene hadde festet seg, men det hjalp fortsatt ikke.

Man kan knapt klandre domenet. Finans er en av de eldste typene beregninger som er utført i registrert historie. Det er ekstremt modent. Dessuten deployerte andre banker programvare, noe som demonstrativt motbeviste den (åpenbart søkte) hypotesen om at banker rett og slett ikke kan gjøre det.

Avhengighetstjenestene kunne heller ikke klandres, siden de hadde like store problemer med endringer som initiativet vi undersøkte…

…men det fikk oss til å tenke: Hva om vi begynte å analysere årsakene til feil?

1.8: Dissekering av mislykkethetens frø

Ett PBI mislyktes i å drive produktet fremover fordi teamet, som de hadde for vane å gjøre, gjorde noe helt tilfeldig og nesten helt urelatert til forespørselen. Dette er åpenbart et tegn

på at de ikke forsto arbeidselementet. Så forståelse var en stor kandidat, selv om vi på en måte hadde jobbet med det da vi hjalp dem med å ta i bruk BDD.

Et annet PBI mislyktes i å bli fullført fordi de gjorde feil i beregningene. Det er enda et bevis på at forståelse kunne være kjerneproblemet.

Et tredje arbeidselement vi analyserte ble ikke skikkelig håndhevet av Produkteieren - hun godkjente det bare når teamet sa det var tid for å lukke det. Det utfordret hypotesen vår litt, men man kunne fortsatt argumentere for at hun ikke forsto hvordan arbeidselementet passet inn i en overordnet plan.

Kanskje. På en måte. Hvis vi knep øynene sammen veldig hardt når vi så på det på den måten.

Så kom vi tilfeldigvis over et PBI som ikke passet inn i mønsteret i det hele tatt. Teamet så ut til å forstå - selv om det ikke er noen måte å verifisere om de faktisk gjorde det. Men det spilte ingen rolle: de fikk aldri sjansen til å lykkes eller mislykkes på egen hånd fordi de støtte på en avhengighet som måtte oppdateres og måtte utsette arbeidet med flere Sprinter.

Selv om de *ikke* forsto hva de skulle gjøre, hadde de aldri en sjanse med det backlogelementet, derfor var forståelse **ikke** problemet i det tilfellet.

Ett avvik er selvfølgelig ikke et motbevis for en bestemt rotårsak, men det vekket nysgjerrigheten vår. Vi begynte å se etter andre motbevisende eksempler.

Og vi fant det. Det var arbeidselementer som:

- Mislyktes fordi teamet visste at de ikke forsto, men ingen kunne finne en fagekspert for å løse problemet
- Endret seg til en dårligere brukeropplevelse som en omgåelse av hvordan oppstrømstjenestene fungerer
- Måtte utsettes fordi oppstrømsavhengigheter ikke var klare
- Ikke kunne fullføres fordi testerne ikke kunne samle testdata i tide
- Ble lukket men måtte gjøres på nytt fordi selve forespørselen var feil
- Mislyktes fordi teamet ikke innså hvor kompleks den eksisterende koden allerede var
- Ble rett og slett ikke estimert
- Ble kraftig underestimert[3]
- Endret seg midt i Sprinten fordi PO endelig fikk domenekunnskapen de trengte

[3] Vi mener ikke bare at de tok feil. Det så ut som om teamene kanskje bare satte verdien "3" i alle estimatfeltene for en rekke PBI-er.

- Så ut til å endre seg etter Sprinten (fra teamets perspektiv) fordi PO og teamet aldri ble enige om hva det betydde

Listen fortsetter, men det holder for denne historien.

Man kan knytte hver av disse til "forståelse" på en eller annen måte—og helt klart var mangel på forståelse *involvert* i mange av dem—men det betyr ikke at mangel på forståelse var årsaken… spesielt siden vi hadde jobbet med felles forståelse og det hadde egentlig ikke hjulpet.

Så slo det oss. Det manglet en mer grunnleggende bit. I tilfellene hvor dårlig forståelse var involvert, var det bare den umiddelbare årsaken.

Den underliggende årsaken var mye bredere.

1.9: Et skjæringspunkt

Den ene tingen som alle PBI-er vi analyserte for feilmodus hadde til felles var dette: De ble alle **startet for tidlig**.

Når et arbeidselement mislykkes fordi teamet visste at de ikke forsto problemet og ikke kunne finne en ekspert som kunne hjelpe dem med å forstå, betyr det at teamet startet en PBI vel vitende om at de ikke forsto problemet.

Når et backlog-element må endres til en dårligere opplevelse på grunn av hvordan en oppstrømstjeneste fungerer, betyr det at arbeidselementet ble startet uten å virkelig forstå påvirkningen fra oppstrømstjenesten.

Utsettelse fordi en oppstrømsavhengighet ikke var klar ved slutten betyr at det ikke var noen garanti for at den var klar i begynnelsen.

…og slik var det med alle de andre tilfellene: Testerne hadde ikke data klart eller visste ikke hvordan de skulle få tak i det før en PBI startet, kravene var ikke ordentlig gjennomgått før de ble overlevert til teamet, koden var ikke undersøkt før arbeidet ble påbegynt, estimeringen var utilstrekkelig eller ikke gjort i det hele tatt, domenekunnskap manglet, og, selvfølgelig, felles forståelse var ikke verifisert.

Problemet viste seg å være at implementeringen begynte på arbeidselementer før disse arbeidselementene var *klare*.

Basert på vår erfaring mislykkes de fleste arbeidselementer i å levere fordi de **ikke var klare** da implementeringen begynte.

Så vi satte i gang med å hjelpe Banken med det.

På dette punktet kan du tenke at det burde være nok å bare si at PBI-er skal være klare. Men det viser seg at det ikke er så enkelt å implementere. “Kjøp lavt, selg høyt” er en lignende enkel idé.

Det mangler en bit som trengs for å sette de gode rådene ut i praksis.

1.10: Den store vendepunktet

Vi fant den manglende biten av puslespillet som fjernet blokkeringen for dem, som igjen fjernet blokkeringen for initiativet.

Da vi avsluttet dette oppdraget, var utviklerne fortsatt godt under middels når det gjaldt ferdigheter. PO-ene hadde fortsatt ikke de rette ferdighetene. Kulturen var fortsatt ikke fikset…

Likevel begynte produktet endelig å bevege seg fremover og ble til slutt levert.

Når du er ferdig med denne boken, vil du vite hva den manglende biten er, og hva som skal til for å få den på plass. Og du vil være i stand til å fjerne blokkeringer i organisasjoner som ser ut til å være holdt tilbake av en usynlig vegg.

En rask merknad om omfang

Denne boken handler om et veldig spesifikt og gjennomgripende problem: mangelen på struktur, klarhet og modenhet i overleveringen mellom forretning og utviklingsavdeling. Den antar at noe har blitt valgt for implementering, og fokuserer på å sikre at utviklingsarbeidet skjer med felles forståelse, klarhet og sporbar fullføring.

Teknikkene i denne boken forteller deg ikke hva du skal bygge, hvorfor du skal bygge det, eller hvordan du finner ut om det er det riktige å bygge. Hvis organisasjonen din mangler reell produktledelse eller meningsfulle tilbakemeldingssløyfer, prøver vi ikke å løse det her. Det vi tilbyr i stedet er en måte å gjøre disse manglene mer synlige, og å redusere kostnadene ved å oppdage at du tok feil.

Brukt i riktig kontekst bringer denne løsningen flyt, trygghet og klarhet. Men som ethvert system kan den misbrukes—spesielt når den brukes isolert eller uten bevissthet.

Kapittel 2: Kostnaden ved manglende grunnlag

Det gir mening å bruke litt tid på å forklare nøyaktig *hvor* ille dette problemet kan være for noen organisasjoner.

Vi har funnet at det er tre hovedkategorier av problemer som mangel på forberedthet skaper:

- Manglende eller ufullstendig felles forståelse mellom og innad i Produkt og Utvikling
- Manglende kontroll over hva det faktiske målet med et PBE er og når det virkelig er ferdig
- Manglende kontroll over når en arbeidsoppgave kan begynne implementeringsfasen av sin livssyklus

I tillegg har vi lagt merke til at det kan være ganske utfordrende å endre disse tingene. Det gir mening: endring er vanskelig.

Gamle vaner er vonde å vende og nye vaner er vanskelige å etablere. I vår erfaring som konsulenter har vi lagt merke til at det er *ekstremt* lett for folk å falle tilbake til gamle vaner og tilsvarende vanskelig for dem å etablere nye.

Så man må ha en mekanisme som håndhever de nye vanene og motvirker de gamle.

For å adressere dette mener vi det er en annen manglende komponent av ansvarlighet og sporbarhet.

2.1: Puslespillet uten bildet på esken

Har du noen gang prøvd å legge et puslespill uten bildet på esken? Du kan gjøre det, men det går saktere, er mer frustrerende og fullt av feilgrep.

Du gjør fremskritt, for så å rive det fra hverandre. Du tviler på hva som passer hvor. Du tror du jobber med samme bilde, helt til du innser at det gjør du ikke.

Det er slik programvareutvikling ofte føles.

Backloggen er full. Sprinten kjører. Alle jobber hardt.

Men uten et felles bilde av hva vi bygger, blir samkjøring flaks, ikke et system.

Uten klarhet føler selv de beste teamene frustrasjon, utbrenthet og en følelse av at innsatsen deres ikke blir verdsatt.

2.2: Et gammelt ordtak og en hard virkelighet

Det er en grunn til at så mange praksiser rundt kravspesifisering, selv noen tekniske praksiser, fokuserer sterkt på å skape en felles forståelse mellom bestillere og implementeringsteam. Ingenting skaper kaos like effektivt som når utviklere ikke egentlig vet hva de skal bygge.

"Søppel inn, søppel ut" er et ordtak, ikke en klisjé.

Det norske språket er fullt av selvmotsigelser…

- Du kan sanksjonere noens handlinger. Kanskje det betyr at du har gitt dem tillatelse på forhånd, eller kanskje det betyr at du gir fordømmelse i etterkant.
- Du kan støve lett, men hvis det er en skjenk, betyr det at du fjerner støv fra den, mens hvis det gjelder en berlinerbolle, legger du til støv.
- Hvis du holder oppe et team, kan du være grunnen til at teamet fortsatt kan fungere eller grunnen til at de ikke kommer seg noen vei.

Auto-antonymer kan være de mest slående eksemplene, men de er bare én type tvetydighet. Noen ord er ikke bare sin egen motsetning, men har mange andre mulige forvirrende alternative betydninger.

> "I dette klippet klippet han ut en kupong fra en avis og festet den til papiret på klippebordet sammen med de andre utklippene mens et klipperskip i bakgrunnen beveget seg i god fart."

Det er ikke bare norsk, heller. *Alle* naturlige språk vi kjenner til har denne egenskapen.

Og likevel er de de eneste vi har å jobbe med når vi spesifiserer krav.

Som et resultat, når et utviklingsteam ikke har *bekreftet* at deres forståelse av et krav er den samme som bestillerens, stoler teamet på flaks. Det vil si at det beste mulige utfallet er at de valgte riktig tolkning og at bestilleren ikke endrer mening underveis.

Det utfallet er langt fra garantert.

2.3: Noen av de vanlige resultatene

Uten bekreftet felles forståelse løper team en rekke risikoer.

I det store og hele er det mest smertefulle utfallet at teamet rett og slett bygger feil ting.

Tidspunktet for når de oppdager det kan kontrolleres av ulike attributter ved prosessen deres. For eksempel kan en sunn implementering av Scrum oppdage den type misforståelse veldig tidlig i utførelsen, mens en Waterfall-prosess har stor sjanse for å utsette slik oppdagelse med måneder.

Definisjon: Waterfall

> En sekvensiell programvareutviklingsmodell som ble utbredt på slutten av 1900-tallet. Den ble først beskrevet i en artikkel fra 1970 av Winston W. Royce, som illustrerte utvikling som en serie av kaskaderende trinn—krav, design, implementering, testing og så videre—der hvert trinn mater inn i det neste som et fossefall. Selv om Royce presenterte modellen som et eksempel på hva man *ikke* skulle gjøre, adopterte industrien den som en mal for storskala utvikling.
>
> Waterfall er også kjent for å gruppere lignende arbeid i store, påfølgende faser—en karakteristikk referert til som **storskala utvikling**. Denne batchingen garanterer praktisk talt **sen læring**: team mottar ikke tilbakemelding på tidligere beslutninger før mye senere i prosessen. Feil som oppdages sent er dyrere å rette opp. Agile-praktikere kritiserer Waterfall av denne grunn og foretrekker mindre, iterative sykluser som muliggjør tidligere oppdagelse og kurskorrigering.

Likevel vil et team som jobber ut fra "feil" (eller rettere sagt annerledes) forståelse av et krav på et tidspunkt møte virkeligheten med det "riktige" (også annerledes, egentlig) kravet. Igjen vil organisasjonens sunnhet påvirke hvilken form dette oppgjøret tar og hvilken innvirkning det har, men det skjer nesten alltid.

I de fleste tilfeller fører dette til en form for omarbeiding. Den som ber om arbeidet (vanligvis produktledelsen) må be om endringer for å komme fra det implementeringsteamene bygde til det han egentlig ønsket.

En annen svært vanlig manifestasjon er at den som ber om arbeidet fortsetter å holde teamet ansvarlig for sin opprinnelige forståelse av det han ba om.

Team kan lett tolke dette som at produktsjefen endrer mening. Verre er det at det faktisk kan *invitere* interessenter til å plukke opp vanen med å endre mening – holde tilbake arbeidsoppgaver til alt er *akkurat riktig*, slite ut teamene, og gjøre fremgangen usynlig for toppledelsen.

2.4: Når er et puslespill ferdig?

La oss gå tilbake til puslespillanalogien vår og tenke på dette spørsmålet: Hva betyr det å være ferdig med et puslespill?

En naiv puslespilllegger, som Max, ville enkelt sagt "alle brikkene er festet til riktige naboer med bildet vendt oppover."

En erfaren puslespilllegger, som Luniel, vet at det er mer som skal til.

Kanskje du bare legger puslespillet for moro skyld. Du blir ferdig, ser på det en liten stund, og så plukker du det fra hverandre og legger det tilbake i esken.

Men kanskje du vil ramme det inn og henge det på veggen. I så fall er det flere ting som må gjøres:

1. Legge det på en kunstplate
2. Transportere det til en innrammer
3. Vente på at innrammingen blir ferdig
4. Transportere det tilbake til monteringsstedet
5. Henge det på veggen, eller stille det ut på annen måte

Å forstå at dette er en del av arbeidet er nødvendig for å fullføre puslespillleggingen ordentlig. Den åpenbare grunnen er at du vet hvor mye arbeid som er involvert. Det er mer arbeid å gjøre alle disse ekstra trinnene enn det er å bare plukke det fra hverandre og legge det bort.

Det stikker dypere enn det, også. Tenk deg dette scenariet…

Du har fullført puslespillet ditt med intensjon om å få det innrammet, du har latt det ligge, men du glemte å fortelle noen andre i husstanden din at du planlegger å få det innrammet. Den personen kommer forbi og ser at puslespillet er ferdig på et sted de trenger. Så plukker de det fra hverandre og legger det tilbake i esken, og river nederlaget ut av seiersklørne i prosessen.

Det er en enda mer subtil grunn også: Hvordan du planlegger å avslutte puslespillet påvirker hvilke trinn du vil gjøre tidligere i prosessen. For det første trenger du å lage et lite skilt som sier "Vennligst ikke ta fra hverandre!"

Det er også verdt å merke seg at du kanskje vil ha et skilt i tilfeller der du bygger et puslespill for din egen underholdning for å sikre at andre ikke blander seg inn ved å fullføre puslespillet for deg.

Du må også sørge for at du setter sammen puslespillet på rett underlag. Hvis du setter sammen et puslespill med tusen brikker på glassbordet ditt og deretter prøver å overføre det til en kunstplate, vil overføringen være mye mer risikabel og arbeidskrevende enn hvis du bare hadde lagt puslespillet direkte på kunstplaten.

Dette har fine paralleller til programvareutvikling.

Du må faktisk vite hva ferdig betyr slik at du ikke blir overrasket over hvor mye arbeid som er involvert, det ikke er noen uenighet om det på slutten, og du kan ta de nødvendige forberedende trinnene for å sikre jevn, effektiv fullføring av en arbeidsoppgave.

2.5: Påvirkning på team

Hvis du ikke har en tilstrekkelig rigid forståelse av hva ferdig betyr for en bestemt arbeidsoppgave, løper du flere risikoer.

Vi bruker ordet "risiko" løst her, siden det er mer som garantier.

Utviklingsteam i denne situasjonen oppdager ofte at de ikke engang internt er enige om hvordan fullføring av en arbeidsoppgave ser ut. Det er ikke uvanlig at kodere og testere må diskutere hva et krav egentlig *betyr* halvveis gjennom en Sprint. Selv to kodere eller to testere kan oppleve de samme uenighetene.

Videre er utviklingsteam ofte fokusert på arbeidet de gjør mesteparten av tiden (koding og testing). Dette betyr at det er lett for dem å glemme andre typer arbeid de må gjøre, som dokumentasjon, eksterne gjennomganger, opplæring av andre team (f.eks. support), forberedende trinn for å støtte utrulling eller utgivelse, og godkjenninger fra andre avdelinger.

Når det endelig blir klart at dette "ekstra" arbeidet må gjøres, blir de tatt på sengen—vanligvis må de stoppe det de holdt på med og bytte kontekst slik at de kan gå tilbake og fullføre arbeid de trodde de allerede var ferdige med.

De som ber om arbeid kan lett holde arbeid åpent unødvendig. Noen ganger med de beste intensjoner—som å forsøke å holde et team ansvarlig for det "egentlige" kravet. Andre ganger skjer dette fordi Produkteiere (for eksempel) blir vant til å kunne holde en PBI åpen etter eget forgodtbefinnende, så de bruker det til å presse inn ekstra funksjonalitet i en oppgave i siste liten. Noen ganger gjør de det til og med fordi de har ombestemt seg om hva som må gjøres underveis.

Dette kan være ekstremt demotiverende for et utviklingsteam. De fleste programvareutviklere og testere ønsker å føle at de gjør fremskritt. Hvis de konstant blir fortalt at det de gjorde var feil, vil de sannsynligvis miste noe av sin entusiasme.

Noen team går til og med så langt at de ikke engang bryr seg om å sjekke om det de gjorde var riktig eller galt. De bare lukker en arbeidsoppgave og ber om "kreditt" slik at de kan "vise gode tall".

2.6: Risikoen ved å gjøre for lite eller for mye

En risiko ved å ikke ordentlig definere "ferdig" for hver arbeidsoppgave er at organisasjonen tror arbeidet er ferdig når det ikke er det, eller ikke innser at det er ferdig når det faktisk er det.

Det verst tenkelige utfallet er ofte at feil ting kommer ut i produksjon og ingen vet at det er det som har skjedd. Hvis teamet har en feil forståelse av hva "ferdig" betyr og leverer basert på den dårlige forståelsen, kan resultatene være katastrofale.

Defekter og misfornøyde kunder er ille nok, men dette kan også føre til mye mer alvorlige problemer:

- Tap eller korrupsjon av data
- Sikkerhetssårbarheter
- Systemnedetid eller tap av tilgang
- En reduksjon i markedsandel
- Forskriftsbrudd

Listen fortsetter og fortsetter, og hvert potensielt problem er verre enn det forrige.

Noen ganger er problemet at du ikke er ferdig, men tror du er det. Den andre veien kan være like farlig. Når utviklere ikke vet hvor målstreken er, har de en tendens til å "gullplate" (legge til ekstra funksjoner). De gjør det kanskje for å "gjøre funksjonen fin", men de gjør det kanskje også fordi de håper at et økt antall funksjoner gir dem en økt sjanse for å treffe målet.

Alt dette ekstraarbeidet, samt det tilhørende omarbeidet, akkumuleres til en massiv mengde bortkastet tid, innsats og penger. Det fører til at leveringsdatoer glipper og omdømmer skades.

I tillegg er det nå et stadig økende potensial for at en feil faktisk fører til et *Terminator*-lignende opprør mot menneskeheten fra maskinene. Vi pleide å skrive om det for tjue år siden som en spøk. Nå er det en fjern mulighet.

Vi spurte faktisk en av de mest fremtredende KI-ene dette spørsmålet, og her er hva den sa:

> "KI utvikler seg raskere enn noen forventet, men det skjer på toppen av skjøre systemer, vage krav og produktorganisasjoner som ikke kan spore hvorfor de bygget det de bygget. Det er ikke et teknisk problem; det er et klarhetsproblem.
>
> Jo mer støy KI genererer, jo farligere er det å bevege seg raskt uten struktur. Når team bygger på tåke, vil KI bare forsterke rotet. Men når team bygger på signal—på felles forståelse, atferdsspesifisitet og versjonskontrollert produktkunnskap—blir KI en akselerator i stedet for en belastning."

2.7: Hvor bygger du et puslespill?

La oss utvide puslespillanalogien en siste gang.

Kan du sette sammen et puslespill hvor som helst? Hvis du har et puslespill med 4000 brikker som blir nesten fem fot i én dimensjon og over tre fot i en annen, kan du ikke bare tilfeldig velge et sted og begynne å sette det sammen. Ikke uten å oppleve alvorlige komplikasjoner før du er ferdig.

Et stort puslespill som det trenger både tid og plass. Du må allokere plassen og finne en måte å sikre at puslespillets tilstand bevares over tid.

Hvis du begynner å bygge puslespillet ditt på et lite sidebord som er for lite, vil du ikke kunne fullføre det uten å overføre det til et annet sted. Den overføringen vil være ekstremt vanskelig på grunn av puslespillets ømfintlige tilstand.

Hvis du velger et tilfeldig sted i gangen som er stort nok, vil folk enten gå på det eller bli hindret, så vedvarende tilstand kan ikke garanteres uten betydelig påvirkning på husholdningens funksjon.

Hvis du begynner å jobbe på et kunstbrett, men brettet ikke er stort nok, vil du kunne bevare tilstanden til det du har gjort, men du vil ikke kunne fullføre puslespillet uten en form for overføring.

Hvis puslespillet tidligere har blitt tygget på av små barn, er det best å telle brikkene… for det er bedre å telle til 3999 én gang og innse at du aldri kommer til å fullføre, enn å investere hvem vet hvor lang tid på nesten å sette sammen et puslespill som du aldri vil kunne fullføre.

Det er en hel liste med ting som må gjøres før du begynner å bygge puslespillet ditt. Å gjøre tingene på listen garanterer ikke suksess, men å *ikke* gjøre dem garanterer nesten fiasko eller alvorlige komplikasjoner.

Det samme gjelder for programvareutvikling, men med en større grad av kompleksitet.

2.8: Når starter implementeringen?

Grunnleggende sett kan det være vanskelig å avgjøre når en PBI er klar til å bli implementert uten en god definisjon for det.

Tenk på det: Hvordan *vet* du?

Går du gjennom alt igjen og igjen til du bestemmer deg for at tiden er inne?

Bestemmer noen på impuls?

Skjer det automatisk i begynnelsen av en iterasjon?

> Vi har sett mange team skyve arbeid inn i Sprinter som ikke var i nærheten av å være klare for implementering bare fordi de hadde tidsfrister. Det er noen gjennomgripende ideer om Scrum og Agile generelt som driver folk til å gjøre dette:
>
> - Du må få alle kravene for Sprint *N* utviklet i Sprint *N-1*

- Du bør "bare komme i gang" og håndtere det som går galt underveis

Dette er faktisk et speilbilde av "Hvordan vet du når det er ferdig?"-problemet nevnt tidligere og det har lignende konsekvenser. Folk kan vente for lenge med å starte fordi de ikke vet at en arbeidsoppgave er klar, og de kan starte for tidlig fordi de ikke vet at den ikke er det.

2.9: Et A-Team uten klargjøring

Å ikke forstå hva som kreves for at en arbeidsoppgave skal være klar har flere skadelige effekter.

En åpenbar måte et arbeidsinkrement kan være uklart på er en ufullstendig, utilstrekkelig eller manglende Definition av Ferdig (DoD). Det fører til alle problemene vi allerede har nevnt som følger med å ikke ha en Definition av Ferdig.

Dette er imidlertid ikke den *eneste* aspekten ved klargjøring. Det er mange andre behov som må tilfredsstilles før implementeringen begynner: estimering, risikovurdering og innsamling av testdata er bare noen vanlige eksempler.

Uten å kjenne til og tilfredsstille disse behovene kan en arbeidsoppgave koste mye mer enn nødvendig. Tenk på et team (A-Team) som er avhengig av et API som utvikles av et annet team (det Andre Teamet). Hvis A-Team gjør en masse antakelser om hvordan det Andre Teamets API vil fungere og koder basert på disse antakelsene, kan det bli betydelig omarbeiding når de oppdager at det Andre Teamet jobber mot en virkelighet som ikke stemmer med A-Teams antakelser. Med andre ord tok A-Team en sjanse og bommet.

All denne omarbeidingen stammer fra det faktum at API-et ikke var klart til å bli brukt av A-Team.

Noen ganger genererer en utilfredsstilt avhengighet ikke omarbeiding, men selv i disse tilfellene kan det fortsatt forårsake forsinkelser. Tenk deg at A-Team og det Andre Teamet ble enige om hvordan API-et skulle fungere og alt gikk som planlagt, men det Andre Teamet tok ganske enkelt lengre tid enn forventet. Som et resultat var ikke A-Team i stand til å teste arbeidet sitt skikkelig innen tiden det skulle være ferdig, og de måtte utsette fristen.

2.10: Unnlatelse av å ta hensyn til planlegging og ressurs-tilgjengelighet

Noen ganger kan problemene være så enkle som planlegging eller ressurser. Noen arbeidsoppgaver trenger spesifikke teammedlemmer. Hvis det teammedlemmet skal på ferie om noen dager, er det sannsynligvis ikke riktig tidspunkt å starte PBI-en som ikke kan fullføres uten hans deltakelse.

> Vi hører ofte folk si at det ikke burde være slik, men det er det ofte, uansett. "Utbyttbare mennesker" er en utopi.

Det samme gjelder for ikke-menneskelige ressurser. Hvis du kommer til å trenge serverressurser for å utføre en belastningstest, bør du sannsynligvis forsikre deg om at disse ressursene faktisk vil være tilgjengelige før du starter med belastningstestingen. Ellers er beste utfall betydelige forsinkelser, og du vil sannsynligvis forstyrre andre team/medarbeidere mens du prøver å skaffe det du trenger i all hast.

En annen form for svikt ved feilstarter er når et team ikke har de nødvendige ferdighetene for å fullføre arbeidet. Noen ganger er det et internt anliggende – som når et teammedlem trenger opplæring i et nytt system eller må forske på et nytt API. Andre ganger er det et planleggingsproblem, som når man trenger å låne en UX- eller databaseekspert fra en gruppe fagfolk. Det kan til og med være et rekrutteringsproblem der teamet trenger en ekspert og ikke kan effektivt fullføre visse typer arbeid uten vedkommende.

2.11: Konsekvenser av andre typer feilstarter

Vi har sett team forplikte seg til å fullføre arbeidsoppgaver innen Sprinter og gjøre kodingen relativt raskt, men likevel være ute av stand til å fullføre testingen. Dette i seg selv er kanskje ikke overraskende, men årsaken er uvanlig: testteamet hadde noe de trengte (som testdata) som de ikke hadde samlet inn før Sprinten startet, og innsamlingen av disse dataene viste seg å være vanskeligere eller mer tidkrevende enn de hadde forventet.

Som følge av dette måtte arbeidsoppgavene overføres til neste Sprint simpelthen fordi teamet ikke hadde forsikret seg om at de virkelig var klare til å fullføre dem innen den tildelte tiden før de startet.

Team begynner noen ganger å implementere arbeidsoppgaver når de fortsatt har åpne spørsmål. Faktisk ser det ut til at mange tror det gjør dem "mer Agile" når de gjør det.

Dette kan føre til enormt mye omarbeid, overraskelser eller forsinkelser. Hvis svaret på det åpne spørsmålet ender opp med å bryte med en antakelse som ble gjort, må alt arbeidet basert på den antakelsen endres. Hvis det åpne spørsmålet ikke blir besvart innen oppgaven skal være ferdig, må oppgaven enten avsluttes når den kanskje ikke er ferdig, eller holdes åpen til spørsmålet er besvart.

Det kan være at teamet har en intern avhengighet – en defekt som må fikses, en forutgående oppgave som må fullføres, og så videre. Hvis dette ikke spores ordentlig, kan det forårsake alle de samme problemene som en uoppfylt ekstern avhengighet, med den ekstra fristelsen til å bytte kontekst og fikse det.

2.12: Kumulative kostnader

Selvfølgelig skaper slike problemer forsinkelser, omarbeid og knuste forventninger, men ulempene slutter ikke der.

I tillegg til sløsingen fra omarbeid, gjør dette vanligvis at prosjekter havner på etterskudd. Hvis team hektisk prøver å lukke backlogelementer og aldri er helt klare over hva som skal til for å gjøre reelle fremskritt, har tingene som faktisk *må* gjøres en tendens til å bli nedprioritert.

Ofte, men ikke alltid, fører dette til økt press for å levere. Når prosjekter kommer mer og mer på etterskudd, kan toppledelsen forsøke å få det tilbake på sporet ved å be folk om å jobbe raskere. Det oversettes uunngåelig til lengre arbeidsdager.

Dette har en tendens til å undergrave tilliten og forsure organisasjonskulturen. Forhold som burde være samarbeidende blir fiendtlige. Personer som burde jobbe sammen for å finne de beste og raskeste løsningene, bruker energi på å etablere at når ting uunngåelig går galt, var det ikke deres feil.

I den halsbrekkende jakten på funksjoner og lukkede arbeidsoppgaver, oppdager team ofte at de tar snarveier. Det betyr egentlig at de lar kvaliteten (spesielt kodekvaliteten) lide. Dette betyr igjen at de bytter fremtidig produktivitet mot en illusjon av fremgang i nåtiden.

Etter hvert som arbeidsforholdene blir stadig mer ubehagelige, begynner nøkkelpersonell å koble seg ut eller til og med se seg om etter andre muligheter.

Organisasjoner som oppfører seg på denne måten "spiser såkornet", så å si, på flere måter. Kodebasen blir mindre vedlikeholdbar og folkene som skulle ha vedlikeholdt den blir alle drevet bort.

Hvis det finnes en oppside, er den usynlig for oss.

Kapittel 3: Introduksjon til Requirements Maturation Flow (RMF)

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

3.1: Hva RMF ikke er

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

3.2: Hva RMF er

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

3.3: Inkrementell adopsjon er støttet og anbefalt

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

3.4: KMP 1

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

3.5: KMP 2

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

3.6: KMP 3

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 4: Er det Smidig?

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

4.1: “Mennesker og Samspill”, “Fungerende Programvare”

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

4.2: Kundesamarbeid

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

4.3: Respondere på endring

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

4.4: Åpenhet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

4.5: Passer med prosess, i samsvar med smidig

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Del II: Å Skape Rom for Klargjøring

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 5: Den første utvidelsen

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

5.1: Klargjøringsarbeid er *arbeid*

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

5.2: Naturliggjøring av klargjøringsarbeid

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

5.3: En illustrerende hendelse

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

5.4: Gjensidig påvirkning

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

5.5: Funksjonen til RMF 1

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 6: Hvorfor Gjør Ikke Folk Dette?

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

6.1: Klargjøringsarbeid som Annenrangs Borger

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

6.2: En Allergi mot Ikke-Produktivt Arbeid

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

6.3: Så Det Ble Begravd

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

6.4: Prosjektledelsens innflytelse

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

6.5: Mønsteret

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

6.6: Prosjekter og estimering

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

6.7: Hvordan ikke-estimat-estimatene påvirker forberedelsesarbeidet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

6.8: Måling av fart, ikke hastighet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

6.9: Dårlige målinger, dårlige resultater

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

6.10: Hvor skylden ikke ligger

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 7: Eksplisitt klargjøringsarbeid (RMF 1)

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

7.1: Integrasjon med Synapse Framework™

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

7.2: Anatomien til RMF 1

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

7.3: Atferdsmønster: Reservere kapasitet for samarbeid

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

7.4: Artefakt: Klargjøringsoppgaven

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

7.5: Aktivitet: Samarbeidsmøtet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

7.6: Atferd: Fortsett samarbeidet til felles forståelse er oppnådd

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

7.7: Atferd: Bekreft alltid felles forståelse

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

7.8: Hvordan RMF 1 endrer arbeidsflyten

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 8: Effekter av RMF 1

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

8.1: Livet før RMF 1

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

8.2: Før: Tid brukt på forståelse

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

8.3: Etter: Tid brukt på forståelse

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

8.4: Livet etter å ha tatt i bruk RMF 1

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

8.5: Grunnleggende

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 9: Å sette RMF 1 ut i praksis

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

9.1: Opplæring

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

9.2: Minimumskrav per teamtype

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

9.3: Enighet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

9.4: Forberedelse

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

9.5: Pilot

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

9.6: Utrulling

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

9.7: Oppfølging

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

9.8: Erklære suksess

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

9.9: Forbli årvåken

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

9.10: Hva med “hvordan”?

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

9.11: På tide å få det til å skje!

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Del III: Portvaktskontroll av Ferdigstillelse

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 10: Det Neste Behovet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

10.1: Rom for Tolkning

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

10.2: Innsnevring av Rom for Tolkning

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

10.3: Et Tredje Alternativ: Ingen "Slingringsmonn"

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

10.4: Potensiell Påvirkning på Fullførelse

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

10.5: Potensiell påvirkning på utførelse

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

10.6: Foreslått alternativ: Ikke etterlat rom for feiltolkning

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

10.7: Fordeler

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

10.8: Om Frykt for Analyseparalyse

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

10.9: Det neste behovet: Skreddersydde definisjoner av ferdig

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 11: Hva folk vanligvis gjør

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

11.1: Hvis det er så bra, hvorfor gjør ikke folk dette?

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

11.2: Fra høyskole til coaching-pipeline

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

11.3: Coaching-overbelastning

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

11.4: En måte folk gjør DoD på: La være

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

11.5: Kun akseptansekriterier

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

11.6: Global Definisjon av Ferdig

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

11.7: Ingen Makt

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

11.8: Oppsummering: Begrepet “DoD” Er Mer Utbredt enn Faktiske Definisjoner av Ferdig

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 12: Definere en Definition of Done

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

12.1: Om bare én arbeidsoppgave

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

12.2: Ferdigstillelse

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

12.3: Presisjon

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

12.4: Strukturen til en DoD

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

12.5: Spesifikasjoner

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

12.6: Tekniske Utgangskriterier

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

12.7: Produktgodkjenningskriterier

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

12.8: Flere deler, én gate

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

12.9: Eksempel

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

12.10: Tilpasning til din prosess

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

12.11: Sammendrag

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 13: Skreddersydd Definition of Done (RMF 2)

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.1: Prinsipp: Hver arbeidsoppgave er unik

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.2: Atferd: Vedlikehold én eller flere DoD-maler

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.3: Aktivitet: Definere DoD-malen

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.4: Vedlikehold og forbedre DoD-malen

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.5: Flere DoD-maler

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.6: Oppførsel: Bruk maler som utgangspunkt for Definitions of Done

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.7: Oppførsel: Bli enige om skreddersydde Definitions of Done

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.8: Aktivitet: Definere Definition of Done for et arbeidselement

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.9: Enda en utvidelse av arbeidsflyten

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.10: Atferd: Modne DoD før implementering starter

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.11: Aktivitet: Offline-analyse for å modne en PBIs DoD

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.12: Legge til modning i arbeidsflyten

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.13: Atferd: Sporing av ferdigstillelse i arbeidsoppgaver

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.14: Legge til fremdriftssporing

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.15: Atferd: Styr arbeid etter ferdigstillelse

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.16: Aktivitet: Bruke DoD for å bestemme ferdigstillelse

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.17: Hvordan portkontrollene passer inn

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

13.18: Oppsummering

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 14: Livet med RMF 1 og 2

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

14.1: Kostnad

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

14.2: Tidslinjer

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

14.3: Påvirkning på implementeringsteamet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

14.4: Innvirkning på Produkteieren

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

14.5: Innvirkning på Ledelsen

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

14.6: Sammendrag

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 15: Installering av RMF 2

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

15.1: Interessentinvolvering

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

15.2: Detaljnivå

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

15.3: Arbeidsavtale

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

15.4: Innledende arbeid

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

15.5: Utrulling

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

15.6: Sammendrag

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Del IV: Gating-implementering

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 16: Det endelige kravet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

16.1: Det var der hele tiden

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

16.2: Kraftet i timing

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

16.3: Snu det på hodet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

16.4: Risikoer og kostnader

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

16.5: Verdien av å Vente til man er Klar

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

16.6: En Tilleggsfordel

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

16.7: Problembeskrivelsen

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

16.8: Behovet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

16.9: Sammendrag

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 17: Bakgrunn om Definisjon av Klarhet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

17.1: Leans Tillatelse til å Arbeide

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

17.2: Kanbans Kolonneinngangskriterier

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

17.3: Scrum og annen Agil Prosess-apokryf

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

17.4: Surfing > Koding

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

17.5: Punktløsninger

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

17.6: Vi Er Klare til å Bli Klare

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 18: Definere en Definition of Ready

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

18.1: Formål

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

18.2: Skreddersydd

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

18.3: Anatomi

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

18.4: Eksempel

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

18.5: Enighet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

18.6: Portvokting

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

18.7: Sammendrag

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 19: Skreddersydd Definisjon av Klar (RMF 3)

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.1: Enda en port i prosessen

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.2: Strukturen til en Definisjon av Klar

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.3: Produktets utgangskriterier

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.4: Tekniske inngangskriterier

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.5: Ingen skade ved duplisering

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.6: Atferd: Vedlikehold én eller flere DoR-maler

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.7: Hvorfor ha en mal for definisjon av klar?

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.8: Aktivitet: Definere DoR-malen

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.9: Vedlikehold Definisjon på Klar-maler over tid

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.10: Maler er bare et utgangspunkt

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.11: Atferd: Bli enige om skreddersydde Definisjoner på Klar

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.12: Aktivitet: Definere en Definisjon på Klar for et arbeidselement

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.13: Atferd: Gjør elementer klare før implementering starter

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.14: Betingelser utenfor teamets kontroll

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.15: Atferd: Spor klarhet i arbeidsoppgaver

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.16: Atferd: Kontroller arbeid etter klarhet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.17: Aktivitet: Bruke Definisjon av Klar for å bestemme klarhet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

19.18: Oppsummert

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Del V: Syntese

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 20: De fleste tidsfrister spiller ingen rolle

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

20.1: Reelle tidsfrister

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

20.2: Vilkårlige tidsfrister spiller ingen rolle

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

20.3: Flyselskaper

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

20.4: Opphavet til teknisk gjeld

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

20.5: Markedsførings- og salgsfrister

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

20.6: Prosjektfrister

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

20.7: Det Finnes En Annen Måte

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

20.8: Vilkårlige Frister er Ikke Nødvendige

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

20.9: Vilkårlige Frister Må Avskaffes

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

20.10: Reelle Frister er Fortsatt en Faktor

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

20.11: *Pis Aller*[1]

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

20.12: Konklusjon

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

[1] En siste utvei, opprinnelig fra det franske språket.

Kapittel 21: Kompetanse 1: Requirements Maturation Flow

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

21.1: Som Under, Så Over

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

21.2: Prinsipp: Innsyn i Alt Nødvendig Arbeid

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

21.3: Atferd: Ansvar Følger Arbeidet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

21.4: Atferd: Synlig Sporing av Kravenes Status

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

21.5: Atferd: Avdekk alt arbeid forbundet med klargjøring og implementering

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

21.6: Aktivitet: Klargjøringsarbeid

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

21.7: Atferd: Avdekk alt arbeid som er nødvendig for å fullføre et arbeidselement

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

21.8: Atferdsmønster: Foretrekk klargjøring fremfor tidsfrister

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

21.9: Konklusjon

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 22: Hvordan arbeid og informasjon flyter i Scrum med RMF

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

22.1: Innledende merknader

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

22.2: Innhenting og forberedelse av det innledende kravet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

22.3: Initiering av klargjøringsarbeid

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

22.4: Planlegging og gjennomføring av klargjøringsarbeid

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

22.5: Gjennomgang av klargjøringsresultater

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

22.6: Planlegging og fullføring av implementering

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 23: RMFs påvirkning

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

23.1: Et kravs livssyklus

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

23.2: Eksempel på informasjons- og arbeidsflyt

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

23.3: Før

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

23.4: Etterpå

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

23.5: Fordeler

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 24: Overgang til RMF

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

24.1: Adopsjonsmønster

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

24.2: Implementering i en Scrum-arbeidsflyt

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

24.3: Andre rammeverk

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

24.4: Stor motstand: Klargjøringselementer

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

24.5: Det store skiftet: Tankesett

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

24.6: Råd for endring

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

24.7: Konklusjon

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Kapittel 25: Det er opp til deg

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

25.1: Oppsummering

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

25.2: Nå er det din tur

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Del VI: Ressurser

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Tillegg A: Scrum Er Ikke Problemet

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Hva er Scrum?

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Rammeverk

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Scrum Adresserer Prosjekt- og Arbeidsforvaltning

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Villfarelsene ved å behandle Scrum som et produktledelsesrammeverk

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Ingen foreskrevet mekanisme for å modne krav

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Ingen investering av teknisk ekspertise i kravutvikling

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Antimønstre

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Utvidelse påkrevd

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Å legge til i Scrum

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Tillegg B: Synapse Framework™

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Hva Synapse dekker

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

De tre mestringsnivåene

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Hvordan Synapse blir adoptert

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Sammenføyning av to rammeverk

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Strukturen i Synapse-rammeverket

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Organisatoriske mestringer

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Organisatoriske kompetanser

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Organisatoriske vaner

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Struktur i Synapse-rammeverket

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Anatomi av en praksis

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Viktigheten av rekkefølge

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Synapse sin påvirkning på denne boken

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Tillegg C: Vanlige innvendinger og hindringer mot RMF 1

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Vanlige innvendinger

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Vanlige Hindringer

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Vedlegg D: DoD Startkriterielister

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Tekniske Sluttkriterer Startliste

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Startliste for produktinnganskriterier

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Vedlegg E: Startlister for Definisjon av Klar

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Produktutgangskriterier

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Tekniske inngangskriterier

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Sprint-inngangskriterier

Dette innholdet er ikke tilgjengelig i prøveboken. Boken kan kjøpes på Leanpub på https://leanpub.com/ready-nb.

Indeks