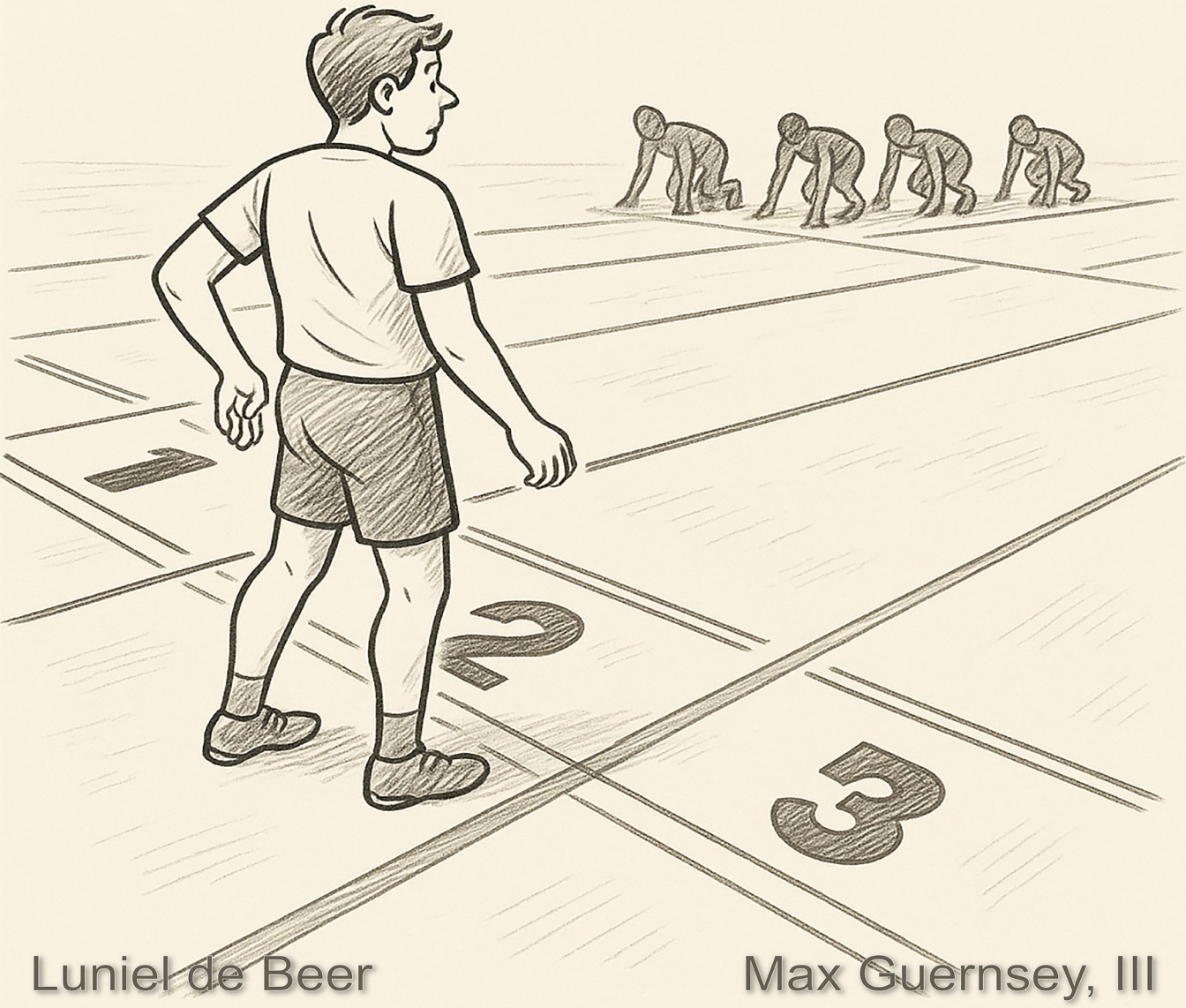
READY

WHY MOST SOFTWARE PROJECTS FAIL AND HOW TO FIX IT



Luniel de Beer

Max Guernsey, III

Svensk Utgåva

Twittra om denna bok!

Hjälp gärna Luniel de Beer och Max Guernsey, III genom att sprida ordet om denna bok på Twitter!

Den föreslagna tweeten för denna bok är:

Jag har precis köpt Ready — en bok för mjukvaruledare och team som vill eliminera omarbetning, överföring och bristande samordning i mjukvaruleveranser. #CodeReady

Den föreslagna hashtaggen för denna bok är #CodeReady.

Ta reda på vad andra säger om boken genom att klicka på denna länk för att söka efter denna hashtag på Twitter:

#CodeReady

Redo (Svensk Utgåva)

Varför de flesta mjukvaruprojekt misslyckas och hur man åtgärdar det

Luniel de Beer och Max Guernsey, III

Denna bok finns tillgänglig på https://leanpub.com/ready-sv

Denna version publicerades den 2025-10-22

Detta är en Leanpub-bok. Leanpub ger författare och förläggare möjlighet att använda Lean Publishing-processen. Lean Publishing är processen att publicera en bok under utveckling med hjälp av lättviktiga verktyg och många iterationer för att få läsarfeedback, justera tills du har rätt bok och bygga dragkraft när du väl har det.

Till minne av Johann van Aardt, som såg min passion, introducerade mig till riktig programmering och hjälpte mig hitta vägen till ett nytt hem. Och till mina föräldrar, vars orubbliga stöd—från min första kund till mitt första hem i USA—gjorde allt detta möjligt.

—Luniel

Till min familj, med vilka solen går upp och ner.

—Max

Innehåll

Om denna bok

Ready är en bok för alla som är involverade i mjukvaruutveckling och som är trötta på **underleverans**, **kroniskt omarbete** och **oklara krav**.

Du kanske redan har försökt investera i teamets färdigheter, förbättrat implementeringen av ditt processramverk eller renoverat koden och fortfarande behöver mer förbättring.

Detta beror på att den huvudsakliga begränsningen för de flesta mjukvaruutvecklingsteam inte är teamets färdigheter, utan kravmognaden. **Även mogna team med rätt kompetens kämpar fortfarande** när de arbetar med omogna krav.

Ready introducerar RMF (Requirements Maturation Flow, på svenska: kravmognadsflöde), ett praktiskt och djupt strukturerat tillvägagångssätt för att få produkt och utveckling i linje utan att ersätta din befintliga process.

Oavsett om du använder Scrum, Kanban eller något eget, hjälper RMF dig att **stabilisera omfattningen**, **eliminera överföring** och **leverera det som verkligen betyder något**.

Om dina team känner sig fast vid kanten av "nästan klart", kommer denna bok att visa dig hur du kan **bryta cykeln** och låsa upp ditt/dina team **för gott**.

Vem boken är till för

Denna bok är bokstavligen till för alla som är involverade i mjukvaruutveckling. Alla från utvecklare till produktchefer och från enskilda medarbetare till chefer.

Denna bok är för dig om du är involverad i mjukvaruutveckling och har märkt att ett team du arbetar med eller i har ett eller flera av följande problem:

- Arbete förs ofta över från en iteration till nästa
- Implementeringsteam känner att de försöker träffa rörliga mål
- Arbete hålls öppet för länge
- Arbete markeras som avslutat men saker är inte riktigt klara
- Utfört arbete stämmer inte överens med förväntningarna
- Arbetet genererar regelbundet ett stort antal defekter
- Stora mängder arbete måste göras om regelbundet

Om något av dessa problem känns bekant kan *Ready* hjälpa.

Hur du använder denna bok

Denna bok är utformad för att vara praktisk. Det är inte en teoretisk avhandling eller en strategipresentation - det är en handbok för att installera RMF (Requirements Maturation Flow) baserad på verkligt klientarbete och fälttestad under verkligt leveranstryck.

Kapitlen är skrivna i sekvens, men RMF själv är modulär. Den består av tre grundläggande metoder:

- RMF 1: Samarbeta för gemensam förståelse
- RMF 2: Styra färdigställande av arbete med skräddarsydda definitioner av färdigt
- RMF 3: Styra implementering med skräddarsydda definitioner av redo

Varje del, eller "Vana" som vi kallar dem, står på egna ben, men de bygger på varandra. Boken är utformad för att hjälpa dig att ta dig an dem en i taget, i ordning. Den strukturen speglar hur vi rekommenderar att team antar RMF i praktiken - där varje Vana läggs till först efter att den föregående fungerar.

Detta undviker att övervälma team och ger varje förändring bästa möjliga chans att fastna. Du kommer att lära dig mer om hur du gör det med början i Kapitel 9.

Om du söker hjälp - vare sig det gäller råd, coachning eller någon att prata med ditt ledningsteam - är du välkommen att kontakta oss direkt.

Och om du letar efter formellt stöd för att implementera RMF erbjuder Producore en hel serie program utformade för att vägleda införandet steg för steg. Du kan lära dig mer på https://ready-book.link/rmf.

Om författarna

Luniel de Beer är skaparen av Requirements Maturation Flow (RMF), ett praktiskt system för att åtgärda gapen mellan produktintention och utvecklingsgenomförande. Han har över 15 års erfarenhet av att leda agila transformationer, överbrygga produkt och utveckling, och hjälpa team att leverera med klarhet och självförtroende.

Luniel är också upphovsman till Producores Capability Management-system, ett spårbart och skalbart sätt att modellera produktegenskaper. Han utvecklade PKB-Driven Development (PKBDD), ett versionshanterat system för att hantera bestående produktkrav. Dessa verktyg utgör en del av ett större leveransramverk utvecklat på Producore.

Max Guernsey, III är mjukvaruarkitekt, utbildare och medgrundare av Producore, ett konsultföretag dedikerat till att åtgärda leveransmisslyckanden genom strukturell och teknisk noggrannhet. Med över två decenniers erfarenhet inom objektorienterad design, refaktorisering, testdriven utveckling och designmönster har han både levererat verksamhetskritiska system och coachat utvecklingsteam i stor skala. Hans arbete kombinerar djupa tekniska metoder med beteende- och processomvandling för att hjälpa organisationer att uppnå hållbar leveransexcellens.

Max bidrog väsentligt till PKBDD och ledde utvecklingen av Producores tillvägagångssätt för Beteendedriven utveckling (BDD) genom sin djupa expertis inom beteendespecifikation.

Tillsammans integrerar deras arbete tydlighet, spårbarhet och grindstyrning till ett sammanhängande system för programvaruleverans som kan skalas från teampraktik till organisatorisk förmåga.

Förord

Not till tekniska ledare

Om du är en senior ledare i en teknisk organisation har du förmodligen inte brist på ansträngning, disciplin eller smarta människor. Ändå händer det att projekt stannar upp. Mål glider iväg. Förväntningar uppfylls inte. Inte för att dina team är lata - utan för att något grundläggande är trasigt i hur arbete definieras, formas och levereras.

Den här boken är ingen ledarskapguide. Det är ett verktyg för strukturell diagnos. Den avslöjar vad som faktiskt händer inom dina team - varför "nästan klart" fortsätter att bli "inte klart", och varför lokala framsteg så sällan översätts till strategiska resultat.

Du kanske inte ser *dig själv* i dessa sidor. Men om dina team inte kan leverera det du behöver kommer du att se *dem*. Och när du gör det kommer du äntligen ha språket - och systemet - för att åtgärda det.

Från Luniel

Först och främst hade denna bok inte varit möjlig utan Max, vars förmåga att se genom dimman och agnarna och destillera en idé till dess kärna är helt bortom min förmåga.

Hur hamnade vi här?

När jag ser tillbaka tror jag det beror på att jag alltid har velat förstå hur saker verkligen fungerar. Oavsett om det handlade om religion, näringslära eller mjukvaruutveckling, stötte jag ständigt på samma problem: ytliga svar som inte höll under press. Så jag fortsatte gräva - frågade inte bara vad vi gör, utan varför, och vad som saknas när det inte fungerar.

En av de tidigaste sprickorna i systemet visade sig i en roll där jag bar tre hattar: Scrum Master, Product Owner och Development Manager (!!) för ett team som levererade datatjänster på ett välkänt techföretag. Vi gjorde det som Scrum föreskrev - korta Sprintar, stories i en backlog, planering på en halv dag - men varje gång vi påbörjade en ny Sprint stötte vi på friktion. Teamet förstod inte problemet fullt ut, vi var tvungna att revidera kraven mitt i Sprinten, förutsägbara beroenden dök upp och försenade oss, och viktiga steg missades.

Så jag började göra något annorlunda. Jag samlade teamet och intressenterna i ett rum för varje story, gick igenom problemet i detalj, brainstormade lösningen tillsammans, och först därefter skrev jag storyn. Sprint-planeringen krympte till en timme, och vår leveransframgång sköt i höjden.

Sakta började jag inse att framgång inte kommer från att arbeta hårdare inom Sprinten. Det kommer från strukturen du sätter på plats *innan* den börjar.

Senare, efter att ha hört Jeff Sutherland prata om "Definitions of Ready", visste jag att det fanns något värdefullt där - men det var inte tillräckligt. Min erfarenhet med krav, UX, UI, forskning och senare med BDD visade mig att olika arbetsuppgifter kräver olika typer av beredskap. Vissa behöver beteendespecifikationer. Vissa behöver systemåtkomst. Vissa behöver en fullständig kapabilitetsanalys.

Och alla behöver en delad förståelse som faktiskt är bekräftad - inte antagen.

När jag arbetade med fler team såg jag samma mönster överallt: missade steg, ouppfyllda beroenden, team som gjorde sitt bästa men ständigt kämpade med att åtgärda problem som borde ha förhindrats. Även fantastiska team kämpade - inte för att de var svaga, utan för att de saknade en struktur som gjorde beredskapen explicit.

Resultatet av allt detta lärande, iteration och frustration är ett strukturerat system för att hantera beredskap.

Det är vad den här boken handlar om.

Jag hoppas den ger dig klarhet i var de verkliga problemen ligger och hur man åtgärdar dem. Jag hoppas den ger dig språket att försvara metoder som kan verka "extra" men som faktiskt är väsentliga. Och framför allt hoppas jag att den hjälper team att leverera med mindre stress, färre överraskningar och mycket bättre resultat.

Om vi får detta rätt kommer vi att spara branschen miljarder kronor.

Men ännu viktigare, vi kommer att ge människor deras sinnesfrid tillbaka.

Från Max

Jag har arbetat med detta problem från olika vinklar i årtionden, men mina framsteg hämmades tills jag träffade Luniel.

Detta beror på att innan jag kände honom närmade jag mig problemet, i grunden, som ett tekniskt sådant. Jag fokuserade på att hjälpa team att anta saker som testdriven utveckling

(TDD), refaktorering, avancerad mjukvarudesign och, senare, acceptanstestdriven utveckling (ATDD) eller beteendedriven utveckling (BDD).

I de flesta av dessa fall behandlades problemet som tas upp i denna bok som en implementationsdetalj för att etablera dessa tekniska metoder.

Detta betyder inte att jag inte längre värdesätter de tekniska metoderna. Jag tycker fortfarande att de är djupt viktiga, men de adresserar inte direkt problemet med beredskap i mjukvaruutveckling. Istället *synliggör* de det problemet och sedan sätter folk en plåster på sin process för att hantera det "precis tillräckligt" för att stödja de tekniska metoder de försöker implementera.

Jag vill också ta upp frågan om vem denna bok kan hjälpa. Det korta svaret är "förmodligen nästan alla inom mjukvaruutveckling", men det verkliga svaret innehåller nyanser som hjälper till att kartlägga det till olika miljöer utan att ändra den grundläggande innebörden.

Det finns team som behöver lösningen som presenteras i den här boken. Du kommer att möta en förenklad version av ett sådant i Kapitel 1.

Det finns också team som inte absolut **behöver** ett system som det vi föreslår, men som ändå skulle kunna dra nytta av det.

Det bästa teamet jag någonsin har arbetat med—utan tvekan en full standardavvikelse över det **näst bästa teamet**, om inte två—fanns i utkanten av Central Oregon. De var så högpresterande att de kunde övervinna avsaknaden av ett sådant system genom ren och skär kompetens. Min chef vid den tiden, Tom Barreras, sa en gång något i stil med "Jag har märkt att våra användarberättelser går bättre när vi ägnar lite tid åt att prata om testerna i förväg."

Detta var återigen något som jag då såg genom linsen av testutveckling och tekniskt utförande, men nu vet jag att det var ytterligare en indikation på att *beredskap* var en faktor som påverkade teamet… det specifika teamet var bara så kompetent och snabbt att reagera att de kunde lyckas genom att hantera hinder när de uppstod istället för att förebygga dem från första början.

Även om du är den typ av person som inte absolut *behöver* oroa dig för beredskap eftersom du kan övervinna det, eller om du arbetar med ett team av samma kaliber, kan du fortfarande dra nytta av innehållet i den här boken.

Del I: Något Saknas

*När det inte hjälper att göra samma saker bättre, leta efter **vad som inte görs**.*

Kapitel 1: Det dolda problemet

Detta är en sann[1] historia om en bank. Vi kallar den helt enkelt för "Banken". Det är en typ av federerad kreditinstitution som utgör en del av USA:s nationella finansiella infrastruktur.

Vi (Luniel och Max) togs in på Banken eftersom de hade svårigheter med att leverera ett mjukvaruprojekt. Det var en av de mest dysfunktionella miljöer vi någonsin sett, och det är därför vi valde detta som den inledande fallstudien: om meningsfull förändring var möjlig på Banken, är det möjligt var som helst.

En snabb anteckning om projekt

När vi använder termen "projekt" i denna bok menar vi det i projektledningssammanhang. Även om det finns olika uppfattningar om vad ordet betyder, använder vi definitionen från Project Management Institute:

> "Ett projekt är ett **tillfälligt** åtagande som genomförs för att skapa en unik produkt, tjänst eller resultat."

Detta innebär att ett projekt har en definierad början och ett slut. När ett projekt avslutas arkiveras projektkunskap och artefakter, teammedlemmar frigörs och kontrakt slutförs.

I denna bok handlar ett projekt i grunden om genomförande. De flesta projekt, enligt PMI:s definition, börjar med genomförbarhet eller design. Vision och strategi har redan fastställts när ett projekt påbörjas.

Ett projekt föds ur den visionen och strategin, och dess framgång eller misslyckande baseras på huruvida de förutspådda målen uppnås—*inte* på om dessa mål var de rätta.

Du kanske använder ordet "projekt" på ett annat sätt, och det är okej. Var bara medveten om att när vi använder det syftar vi på definitionen och sammanhanget ovan.

Inget av detta är menat att antyda att vi godkänner användningen av projektledning för mjukvaruutveckling. Tvärtom. Men vi erkänner att det *används* trots allt. Vi tar itu med

[1] Vi har ändrat identifierande detaljer för att skydda integriteten hos de personer och institutioner som det hände.

det problemet senare, i Kapitel 6.

1.1: Den klassiska omskrivningen

Banken höll på att skriva om sin låneåterbetalningsportal av flera skäl.

Det gamla systemet, en helt C#/.NET-baserad lösning, var felbenäget. Förutom att försämra kundnöjdheten genererade det också ett oändligt flöde av mycket dyra supportärenden där någon manuellt behövde manipulera databasen för att korrigera ett fel som systemet hade skapat.

Det gamla systemet var också förfallet ur ett underhållsperspektiv. Det var nästan omöjligt för ingenjörerna att göra meningsfulla ändringar och, även när de kunde, var det ett extremt riskfyllt företag.

Omskrivningen var tänkt att ändra på det.

Det nya systemet skulle fortfarande ha en C#/.NET-backend, men den skulle vara helt täckt av tester. Frontenden skulle implementeras i OutSystems, en populär låg- eller ingen-kod-lösning som låter en organisation definiera en applikation på ett ställe och få en webbapp, en Android-app och en iOS-app automatiskt genererade när de bestämmer sig för att publicera sina ändringar.

Förhoppningen var att det nya systemet skulle vara felfritt, både förbättra kundnöjdheten och avsevärt minska supportkostnaderna.

De hoppades också att omskrivningen skulle frigöra utvecklarna—med kombinationen av ett mer disciplinerat tillvägagångssätt för backend och låg-kod-lösningen för frontend som kraftigt skulle minska kostnaderna och riskerna för nya funktioner.

En trevlig sidoeffekt av att flytta till OutSystems var att de skulle få en ren, modern mobilapp på båda de stora plattformarna.

Det var drömmen när de hade börjat tre år före början av denna historia. Verkligheten var att teamen hittills inte hade levererat **någonting**.

1.2: Perspektiv på problemet

När vi pratade med företagsledningen hörde vi en mycket naturlig frustration över att de hade gjort så stora investeringar utan någon som helst strategisk förflyttning.

De hade provat allt, från deras perspektiv. De hade ändrat personal, ökat personal, ändrat budget, ökat press och tagit in en rad konsulter (där det starkt antyddes att vi var sist i raden). Ingenting verkade göra det bättre—åtminstone inte på ett sätt som de kunde mäta, eftersom allt de såg var att "nålen" stod på noll ett kvartal och sedan fortfarande stod på noll nästa kvartal.

De ville inte ha mer "osynliga framsteg". De ville ha *resultat*.

När vi pratade med ledningen inom Produktorganisationen fick vi en något annorlunda (men fortfarande liknande) historia eftersom de arbetade mer direkt med Utveckling.

Det var inte så att teamen inte gjorde någonting, inte enligt dem i alla fall. Det var att teamen aldrig gjorde det som efterfrågades. Det var praktiskt taget garanterat: oavsett hur enkel förfrågan var och hur tydligt den formulerades, fick man något helt annat när det var dags att utvärdera vad teamen hade skapat.

Det hade gått så långt att det blivit ett stående skämt i stil med "Vi måste lista ut hur vi ska be om det vi inte vill ha, så har vi åtminstone en chans att få det vi faktiskt vill ha."

De tekniska cheferna såg saken helt annorlunda.

För dem var problemet att Product inte levererade genomförbara krav och att Product inte levererade *tillräckligt många* krav. Om Product bara kunde "hänga med i svängarna" skulle teamen kunna leverera det de ville ha i tid och inom budget.

De hade gjort betydande investeringar i att modernisera hur kod skrevs och levererades och, enligt deras uppfattning, levererade inte Product tydliga krav.

När vi pratade med andra konsulter (som hade rekommenderat oss till organisationen), fokuserade de med rätta på dysfunktionen de såg: Alla verkade mycket fokuserade på att skylla på någon annan. Anledningen till att de tog in oss från början var att de var oroade över bemanningsstrategin och ville ha en utvärdering av enskilda medarbetare, men de såg fingerpekandet och översittarbeteendet på chefsnivå som huvudkällan till problemen.

1.3: Vår undersökning

Vårt ursprungliga uppdrag var att utvärdera teamen och försöka hjälpa dem att förbättra sina färdigheter vid behov, så vi började titta närmare på personerna som arbetade i frontlinjen.

Det fanns definitivt utrymme för förbättring.

De enskilda medarbetarna på Product-sidan hade egentligen inte de färdigheter som krävdes. I verkligheten var de mestadels projektledare som hade kastats in i rollen som Product Owner (PO) eller Product Manager.

Som ett resultat skrev hälften av dem "svävande" krav och accepterade sedan (bokstavligen) vad som helst som teamen gjorde under den iterationen utan någon kritisk analys. Den andra hälften skrev samma typ av krav och hävdade sedan att teamen "borde ha vetat" saker som de aldrig hade pratat om och höll arbetsuppgifterna öppna mer eller mindre på obestämd tid.

Banken använde för att hantera och spåra sin backlog av arbete. Även om det vi tar upp i den här boken mestadels är oberoende av Scrum använder vi Scrum-terminologi genomgående eftersom majoriteten - eller åtminstone en pluralitet - av teamen använder Scrum.

Definition: Scrum

Scrum är ett lättviktigt ramverk som hjälper människor, team och organisationer att generera värde genom adaptiva lösningar för komplexa problem. I korthet:

1. En Product Owner prioriterar arbetet för ett komplext problem i en Product Backlog
2. Scrum-teamet omvandlar ett urval av arbetet till ett värdeinkrement under en Sprint
3. Scrum-teamet och dess intressenter inspekterar resultaten och justerar inför nästa Sprint
4. Upprepa

Om du är obekant med Scrum och dess terminologi rekommenderar vi att du läser 2020-versionen av Scrum Guide. Det är en snabb, upplysande läsning.

På samma sätt fann vi att de tekniska teamen låg långt under genomsnittet när det gällde programmeringsfärdigheter (-2σ, i bästa fall) och dessutom var mycket motståndskraftiga mot förändring. Som en naturlig följd var kodkvaliteten usel.

Ändå var detta enligt teamen inte anledningen till att de inte levererade. För dem var det vaga krav och förändringar under pågående Sprint från Product som förstörde projektet.

…och ingen pratade ens om det större problemet, det som var så absurt att det verkar påhittat tills man har upplevt det.

De tekniska teamen hade en vana att inte förstå ett krav, bygga något slumpmässigt, och sedan kräva erkännande för att ha "avslutat en arbetsuppgift".

Vi menar inte ett litet missförstånd. Vi menar en total frånkoppling: Vi bad dem att inaktivera tillämpningen av medel till kapitalbelopp under vissa omständigheter, och de inaktiverade istället möjligheten att lägga till en sekundär bekräftelse-e-postadress.

Sedan sa de att det var vad vi hade bett om.

1.4: Djupare grävande

Båda kompetensgapen kunde åtgärdas, men vi var skeptiska till att de var de verkliga hindren.

Det var något annat som inte stämde, så vi grävde djupare. Vi började med denna fråga: Varför tog det så lång tid att skriva krav och varför gav det så dåliga resultat?

En anledning var att kunskapen som krävdes för att skriva ett meningsfullt krav var en bristvara. En liten del av den fanns hos teknik- och Product-teamen.

En del av det fanns inbäddat i koden i det äldre systemet. En del var helt borta. Det mesta fanns dock lagrat som tyst kunskap hos ämnesexperter utspridda över bankens olika enheter. Detta innebar att att skapa ett krav som faktiskt främjade ett strategiskt mål var en extremt arbets- och tidskrävande aktivitet.

I kontrast till detta fanns en omättlig aptit på funktionalitet från ett funktionssvällt ledningsteam. Direktivet var "håll utvecklarna sysselsatta—fyll dem med krav". Fokus låg på kvantiteten av krav för att hålla teamen upptagna—ett perspektiv som var helt motsatt den omsorg och tid som behövdes för att definiera krav som skulle "göra skillnad".

1.5: Att Göra Saker Bättre Gjorde Det Inte Bättre

Detta är alla problem som går att åtgärda, men att åtgärda dem hjälpte inte.

Tidigare förbättringar av mjukvaruutvecklingstekniker hade inte hjälpt teamen att leverera, men Max gjorde en insats för att hjälpa teamen att förbättras ytterligare.

Han introducerade revolutionerande koncept från programmeringsdoktriner från mitten av 1900-talet som "kopiera och klistra inte in den koden 27[2] gånger, lägg den i en funktion och anropa den istället". Bara detta förslag förbättrade dramatiskt kvaliteten på ny kod och lät dem börja förbättra kvaliteten.

Det och andra grundläggande kodningsråd hjälpte dem att skriva bättre kod som de lättare kunde underhålla i framtiden.

…men det hjälpte inte projektet framåt.

På produktsidan kunde Luniel introducera BDD och säkerställa att produktägare grundligt granskade krav innan de överlämnades till teamen.

Han fick teamen att samarbeta kring dem och använda dem för att utvärdera om en produktbacklogpost (PBI) verkligen var klar.

Definition: Produktbacklogpost (PBI)

> En produktbacklogpost (PBI) är en separat arbetsenhet i produktbackloggen som representerar en potentiell förändring, tillägg eller förbättring av produkten. PBI:er kan ta många former—funktion, buggfix, teknisk förbättring, forskningsuppgift, etc.—och definieras av sitt bidrag till produktvärdet.
>
> Många team hänvisar till PBI:er som "stories" eller "användarberättelser", men den korrekta Scrum-termen är "produktbacklogpost" eller "PBI". När en PBI har åtagits i en sprint blir den också en del av sprintbackloggen. För enkelhetens och neutralitetens skull använder vi "produktbacklogpost" eller "PBI" för att hänvisa till alla arbetsuppgifter som Scrum-teamet hanterar—oavsett om du kallar det produktbacklogpost, sprintbacklogpost (SBI), användarberättelse, story, arbetsuppgift eller backlogpost.

Det skapade *tydlighet*, men det skapade inte *flöde*.

Med hjälp av partnerkonsulterna som hade tagit in oss kunde vi (tillfälligt) lätta på det absolut knäckande trycket som lades på teamen och kravförfattarna.

Det kan ha hjälpt till att skapa lite förtroende, men det gav inga påtagliga resultat.

[2] Detta är inte bara inte en överdrift, det är inte ens det värsta fallet. I ett fall fanns det nästan hundra exakta kopior av samma algoritm.

Vi började till och med bygga en kunskapsbas som hjälpte människor att spåra verksamhetskunskapen de behövde för att skriva krav och identifierade områden där det fanns luckor i kunskapen.

Det påskyndade kravskrivandet, men det fick inte ut produkten genom dörren.

Efter månader av interaktioner hade vi hjälpt ledningen att *se* var de befann sig, men de var ingenstans nära sina mål. Och de kom inte närmare.

De var på väg att återgå till sin gamla strategi att "överbelasta" teamen för att säkerställa att de alltid var upptagna.

1.6: En Elimineringsprocess

Det hade varit enkelt att bara ge upp och säga "det här är hopplöst". Det fanns många ursäkter man kunde falla tillbaka på:

- Utvecklingsteamet hade låg kompetens (det hade de)
- Kravförfattarna hade fel kompetens (det hade de)
- Ledningen krossade teamen med orealistiska förväntningar (det gjorde de)
- Organisationen saknade kritisk verksamhetskunskap som krävdes för att fungera (det gjorde den)
- Cheferna verkar inte lita på varandra (det gjorde de inte)

Allt detta var sant. Ändå hade förbättringar gjorts inom alla dessa områden och ingen av dem verkade göra det grundläggande problemet bättre. Ingen av dem hjälpte utvecklingsteamen, produkt, ledning eller chefsgruppen att komma närmare sina mål.

…och där har vi ledtråden till lösningen: Alla problem som listats tidigare var problem som folk redan kunde se.

Om de variabler du kan se inte gör någon skillnad måste det finnas en variabel du **inte ser** som gör det.

Det verkliga problemet var det som ingen ens visste fanns.

1.7: Jakten på den verkliga gärningsmannen

I sökandet efter den verkliga gärningsmannen - det som faktiskt hindrade banken från att uppnå sina mål - behövde vi börja någonstans.

En rimlig utgångspunkt var att titta på vad de misslyckade PBI:erna (de flesta av dem) hade gemensamt.

Vi började med att eliminera de saker som vi kunde se inte var gemensamma eftersom de varierade kraftigt:

- Vilken del av systemet: vissa PBI:er påverkade endast backend, andra endast frontend, ytterligare andra påverkade båda delarna
- Vilket team som utförde arbetet - det verkade inte spela någon roll vem som gjorde arbetet, risken för misslyckande var fortfarande hög
- Vilken PO som författade arbetet - samma sak som med teamen

Sedan började vi titta på de saker som var gemensamma. Listan var varken lång eller kort:

- Utvecklingsteamen
- Product owners
- Ledningen
- Kulturen
- Utvecklingsmiljön
- Utvecklingsmetodiken
- Kravspecificeringstekniken
- Domänen (finans)
- Beroendetjänsterna

Många av dessa kunde också avfärdas direkt. Teamen, product owners, ledningen, kulturen och utvecklingsmiljön hade alla nyligen förbättrats utan någon verklig påverkan på meningsfulla resultat. Vi hade personligen hjälpt till att förbättra utvecklings- och kravspecificeringsmetodiken och bekräftat att dessa förbättringar hade etablerat sig, men det hjälpte fortfarande inte.

Man kan knappast skylla på domänen. Finans är en av de äldsta typerna av beräkningar som utförts i dokumenterad historia. Den är extremt mogen. Dessutom utvecklade andra banker programvara, vilket på ett tydligt sätt motbevisade den (uppenbart långsökta) hypotesen att banker helt enkelt inte kan göra det.

Beroendetjänsterna kunde inte heller klandras, eftersom de hade lika stora problem med förändringar som det initiativ vi tittade på…

…men det fick oss att tänka: Tänk om vi började analysera orsakerna till misslyckandena?

1.8: Dissekering av misslyckandets frön

En PBI misslyckades med att föra produkten framåt eftersom teamet, som de hade en tendens att göra, gjorde något helt slumpmässigt och nästan helt orelaterat till förfrågan. Det är uppenbarligen ett tecken på att de inte förstod arbetsuppgiften. Så förståelse var en stor kandidat, även om vi på sätt och vis hade arbetat med det när vi hjälpte dem att införa BDD.

En annan PBI misslyckades eftersom de fick beräkningarna fel. Det är ytterligare ett bevis på att förståelse kunde vara kärnproblemet.

En tredje arbetsuppgift som vi analyserade blev inte riktigt ordentligt kontrollerad av Product Ownern - hon godkände den när teamet sa att det var dags att stänga den. Det satte vår hypotes på prov, men man kunde fortfarande argumentera för att hon inte förstod hur arbetsuppgiften passade in i en övergripande plan.

Kanske. På sätt och vis. Om vi kisade riktigt hårt när vi tittade på det på det sättet.

Sedan stötte vi på en PBI som inte alls passade in i mönstret. Teamet verkade förstå - även om det inte fanns något sätt att verifiera om de faktiskt gjorde det. Men det spelade ingen roll: de fick aldrig en chans att lyckas eller misslyckas på egen hand eftersom de stötte på ett beroende som behövde uppdateras och var tvungna att skjuta upp sitt arbete med flera Sprintar.

Även om de *inte* förstod vad de skulle göra, hade de aldrig en chans med den backloguppgiften, därför var förståelse **inte** problemet i det fallet.

Ett enstaka avvikande fall är förstås inte ett motbevis för en specifik grundorsak, men det väckte vår nyfikenhet. Vi började leta efter fler motbevis.

Och vi hittade dem. Det fanns arbetsuppgifter som:

- Misslyckades eftersom teamet visste att de inte förstod, men ingen kunde hitta en ämnesexpert för att lösa problemet
- Ändrades till en sämre användarupplevelse som en tillfällig lösning på hur uppströmstjänsterna fungerade
- Var tvungna att skjutas upp eftersom uppströmsberoenden inte var klara

- Inte kunde slutföras eftersom testarna inte kunde samla in testdata i tid
- Stängdes men behövde göras om eftersom själva förfrågan var felaktig
- Misslyckades eftersom teamet inte insåg hur komplex den befintliga koden redan var
- Helt enkelt inte var estimerade
- Var kraftigt underestimerade[3]
- Ändrades mitt i Sprinten eftersom PO:n äntligen fick den domänkunskap de behövde
- Verkade ändras efter Sprinten (från teamets perspektiv) eftersom PO:n och teamet aldrig var överens om vad det betydde

Listan fortsätter, men det räcker så för den här berättelsen.

Man skulle kunna argumentera för att var och en av dessa kan kopplas till "förståelse" på något sätt—och visst var bristande förståelse *inblandad* i många av dem—men det betyder inte att bristande förståelse var orsaken… särskilt eftersom vi hade arbetat en del med gemensam förståelse och det hade inte riktigt hjälpt.

Då slog det oss. Det saknades en mer grundläggande del. I de fall där bristande förståelse var inblandad var det bara den proximala orsaken.

Den distala orsaken var mycket bredare.

1.9: En skärningspunkt

Det enda som alla PBI:er vi analyserade för fellägen hade gemensamt var detta: De hade alla **påbörjats för tidigt**.

När ett arbetsmoment misslyckas för att teamet visste att de inte förstod problemet och inte kunde hitta en expert som kunde hjälpa dem förstå, betyder det att teamet påbörjade en PBI medan de visste att de inte förstod problemet.

När ett backlogmoment måste ändras till en sämre upplevelse på grund av hur en uppströmsservice fungerar, betyder det att arbetsmomentet påbörjades utan att man egentligen förstod uppströmsservicens påverkan.

Uppskjutning på grund av att ett uppströmsberoende inte var klart vid slutet betyder att det inte fanns någon garanti för dess beredskap vid början.

…och så var det med alla andra fall: Testarna hade inte data redo eller visste inte hur de skulle få tag på den innan en PBI påbörjades, kraven var inte ordentligt granskade

[3]Vi menar inte bara att de fick det fel. Det såg ut som att teamen kanske bara satte in värdet "3" i alla uppskattningsfält för en rad PBI:er.

innan de överlämnades till teamet, koden var inte undersökt innan arbetet påbörjades, uppskattningen var otillräcklig eller gjordes inte alls, domänkunskapen saknades, och förstås, den gemensamma förståelsen var inte verifierad.

Problemet visade sig vara att implementeringen påbörjades på arbetsmoment innan dessa arbetsmoment var *redo*.

Enligt vår erfarenhet misslyckas de flesta arbetsmoment med att leverera eftersom de **inte var redo** när implementeringen påbörjades.

Så vi gav oss ut för att hjälpa Banken med det.

Vid det här laget kanske du tänker att det borde räcka med att bara säga att PBI:er ska vara redo. Men det visar sig att det inte är så lätt att implementera. "Köp lågt, sälj högt" är en lika enkel idé.

Det saknas en del som behövs för att omsätta de goda råden i praktiken.

1.10: Den stora vändpunkten

Vi hittade den saknade pusselbiten som låste upp dem, vilket i sin tur låste upp initiativet.

När vi avslutade det uppdraget var utvecklarna fortfarande långt under mediannivån vad gäller kompetens. PO:erna hade fortfarande inte rätt färdigheter. Kulturen var fortfarande inte åtgärdad…

Ändå började produkten äntligen röra sig framåt och kom till slut ut genom dörren.

När du har läst klart den här boken kommer du att veta vad den saknade pusselbiten är och vad som krävs för att få den på plats. Och du kommer att kunna låsa upp organisationer som verkar hållas tillbaka av en osynlig vägg.

En snabb kommentar om omfattning

Den här boken handlar om ett mycket specifikt och genomgripande problem: bristen på struktur, tydlighet och mognad vid överlämningen mellan verksamhet och utveckling. Den utgår från att något har valts för implementering och fokuserar på att säkerställa att byggarbetet sker med gemensam förståelse, beredskap och spårbar färdigställning.

Teknikerna i den här boken talar inte om för dig vad du ska bygga, varför du ska bygga det eller hur du tar reda på om det är rätt sak att bygga. Om din organisation saknar verklig produkthantering eller meningsfulla återkopplingsloopar, försöker vi inte lösa det här. Vad vi istället erbjuder är ett sätt att göra dessa luckor mer synliga och att minska kostnaden för att upptäcka att du hade fel.

När det används i rätt sammanhang ger denna lösning flöde, trygghet och tydlighet. Men som alla system kan det missbrukas—särskilt när det används isolerat eller utan medvetenhet.

Kapitel 2: Kostnaden för Bristande Grunder

Det är värt att ägna lite tid åt att förstå *hur* allvarligt detta problem kan vara för vissa organisationer.

Vi har funnit att det finns tre huvudsakliga "kategorier" av problem som bristande beredskap skapar:

- Saknad eller ofullständig gemensam förståelse mellan och inom Produkt och Utveckling
- Bristande kontroll över vad det faktiska målet med en PBI är och när den verkligen är klar
- Avsaknad av kontrollpunkter för när ett arbetsmoment kan påbörja implementeringsfasen

Dessutom har vi märkt att det kan vara mycket utmanande att förändra dessa saker. Det är förståeligt: förändring är svårt.

Gamla vanor är svåra att bryta och nya vanor är svåra att etablera. I vår erfarenhet som konsulter har vi märkt att det är *extremt* lätt för människor att falla tillbaka i gamla vanor och jämförelsevis svårt för dem att etablera nya.

Så man måste ha en mekanism som upprätthåller de nya vanorna och motverkar de gamla.

För att hantera detta anser vi att det saknas ytterligare en komponent av ansvarsskyldighet och spårbarhet.

2.1: Pusslet utan bilden på lådan

Har du någonsin försökt lägga ett pussel utan bilden på lådan? Det går, men det är långsammare, mer frustrerande och fullt av felaktiga starter.

Du gör framsteg, river sedan isär det. Du ifrågasätter vad som passar var. Du tror att ni jobbar mot samma bild, tills du inser att ni inte gör det.

Så känns mjukvaruutveckling ofta.

Backloggen är full. Sprinten är igång. Alla arbetar hårt.

Men utan en gemensam bild av vad vi bygger blir samordning en fråga om tur, inte ett system.

Utan tydlighet känner även de bästa teamen frustration, utbrändhet och en känsla av att deras insats inte värdesätts.

2.2: Ett gammalt talesätt och en hård verklighet

Det finns en anledning till att så många metoder kring kravhantering, även vissa utvecklingsmetoder, fokuserar mycket på att skapa en gemensam förståelse mellan beställare och implementeringsteam. Inget skapar så mycket kaos som när utvecklare inte riktigt vet vad de ska bygga.

"Skräp in, skräp ut" är ett talesätt, inte en kliché.

Det engelska språket är fyllt av självmotsägelser…

- You can sanction someone's actions. Maybe that means you've given them permission in advance or maybe it means you're giving condemnation after the fact.
- You can lightly dust something but, if that something is a credenza, it means you're removing dust from it while, if in reference to a beignet, you're adding dust.
- If you hold up a team, you might be the reason that team is able to continue to function or you might be the reason they can't get anywhere.

Autoantonymer kan vara de mest slående exemplen, men de är bara en typ av tvetydighet. Vissa ord är inte bara sina egna motsatser, utan har många ytterligare potentiellt förvirrande alternativa betydelser.

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

Det är inte bara engelska heller. *Alla* naturliga språk som vi känner till har denna egenskap.

Och ändå är de de enda vi har att arbeta med när vi specificerar krav.

Som ett resultat, när ett utvecklingsteam inte har *bekräftat* att deras förståelse av ett krav är densamma som beställarens, förlitar sig teamet på tur. Det vill säga, det bästa möjliga utfallet är att de valde rätt tolkning och att beställaren inte ändrar sig längs vägen.

Det utfallet är långt ifrån garanterat.

2.3: Några av de vanliga resultaten

Utan bekräftad gemensam förståelse löper team flera risker.

I stort sett är det vanligaste och mest smärtsamma utfallet att teamet helt enkelt bygger fel sak.

När de upptäcker detta kan styras av olika attribut i deras process. Till exempel kan en välfungerande implementation av Scrum upptäcka den typen av missförstånd mycket tidigt i utförandet medan en Vattenfallsprocess riskerar att fördröja sådan upptäckt med månader.

Definition: Vattenfall

> En sekventiell mjukvaruutvecklingsmodell som blev utbredd under sent 1900-tal. Den beskrevs först i en artikel från 1970 av Winston W. Royce, som illustrerade utveckling som en serie av fallande steg—krav, design, implementering, testning, och så vidare—där varje steg matar in i nästa likt ett vattenfall. Även om Royce presenterade modellen som ett exempel på vad man *inte* skulle göra, antog branschen den som en blueprint för storskalig utveckling.
>
> Vattenfall är också känd för att gruppera liknande arbete i stora, på varandra följande faser—ett kännetecken som kallas **storskalig utveckling**. Denna gruppering garanterar praktiskt taget **sen återkoppling**: team får inte feedback på tidigare beslut förrän mycket senare i processen. Fel som upptäcks sent är dyrare att åtgärda. Agila utövare kritiserar Vattenfall av denna anledning och föredrar mindre, iterativa cykler som möjliggör tidigare upptäckt och kurskorrigering.

Förr eller senare kommer dock ett team som arbetar utifrån “fel” (eller snarare annorlunda) förståelse av ett krav att möta en uppgörelse med det “rätta” (också annorlunda, egentligen)

kravet. Återigen påverkar organisationens hälsotillstånd vilken form denna uppgörelse tar och vilken inverkan den har, men det händer nästan alltid.

I de flesta fall leder detta till någon form av omarbete. Beställaren (vanligtvis produktledningen) måste be om ändringar för att komma från det som implementationsteamen byggde till det han egentligen ville ha.

En annan mycket vanlig manifestation är att beställaren fortsätter att hålla teamet ansvarigt för hans ursprungliga förståelse av vad han bad om.

Team kan lätt tolka detta som att produktchefen ändrar sig. Ännu värre kan det faktiskt *uppmuntra* intressenter att ta upp vanan att ändra sig - genom att hålla tillbaka arbetsuppgifter tills allt är *precis rätt*, utmatta teamen och dölja framsteg för högre chefer.

2.4: När är ett pussel färdigt?

Om vi återgår till vår pusselläggningsanalogi, fundera över denna fråga: Vad betyder det att vara klar med ett pussel?

En naiv pusselläggare, som Max, skulle helt enkelt säga “alla bitar är fästa vid rätt grannar med bilden uppåt.”

En erfaren pusselläggare, som Luniel, vet dock att det finns mer till det.

Kanske lägger du bara pusslet för nöjes skull. Du kommer att bli klar, titta på det en stund och sedan ta isär det och lägga tillbaka det i lådan.

Å andra sidan kanske du vill rama in det och hänga upp det på väggen. I så fall finns det ytterligare saker som måste göras:

1. Lägga det på en konstskiva
2. Transportera det till en ramtillverkare
3. Vänta på att inramningen blir klar
4. Transportera tillbaka det till monteringsplatsen
5. Hänga upp det på väggen eller på annat sätt ställa ut det

Att förstå att detta är en del av arbetet är nödvändigt för att kunna slutföra en pusselläggning på rätt sätt. Den uppenbara anledningen är att du vet hur mycket arbete som krävs. Det är mer arbete att göra alla dessa extra steg än att bara ta isär det och lägga undan det.

Det går djupare än så dock. Föreställ dig detta scenario…

Du har slutfört ditt pussel med avsikt att rama in det, du har lämnat det på plats, men du glömde berätta för någon annan i ditt hushåll att du planerar att rama in det. Den personen kommer förbi och ser att pusslet är färdigt på en yta som han eller hon behöver. Så de tar isär det och lägger tillbaka det i lådan, och rycker därmed nederlag ur segerns käftar.

Det finns en ännu mer subtil anledning också: Hur du planerar att avsluta pusslet påverkar vilka steg du vill ta tidigare i processen. För det första behöver du göra en liten skylt där det står "Var vänlig ta inte isär detta!"

Det är också värt att notera att du kan vilja ha en skylt även i det fall där du bygger ett pussel för din egen underhållning för att säkerställa att andra personer inte stör genom att färdigställa pusslet åt dig.

Du måste också se till att du lägger pusslet på rätt yta. Om du lägger ihop ett pussel med tusen bitar på ditt soffbord med glasskiva och sedan försöker överföra det till en konstskiva, kommer överföringen att vara mycket mer riskfylld och arbetskrävande än om du bara hade lagt pusslet direkt på konstskivan.

Detta har tydliga paralleller med mjukvaruutveckling.

Du behöver faktiskt veta vad färdigt betyder så att du inte blir överraskad av hur mycket arbete som krävs, det finns ingen oenighet om det i slutet, och du kan ta de nödvändiga förberedande stegen för att säkerställa ett smidigt och effektivt slutförande av en arbetsuppgift.

2.5: Påverkan på team

Om du inte har en tillräckligt rigid förståelse för vad färdigt innebär för en särskild arbetsuppgift, utsätter du dig för ett antal risker.

Vi använder ordet "risk" löst här, eftersom de mer liknar garantier.

Utvecklingsteam i denna situation upptäcker ofta att de inte ens internt är överens om vad det innebär att slutföra en arbetsuppgift. Det är inte ovanligt att programmerare och testare upptäcker att de måste reda ut vad ett krav egentligen *betyder* halvvägs genom en Sprint. Även två programmerare eller två testare kan drabbas av samma meningsskiljaktigheter.

Dessutom är utvecklingsteam ofta fokuserade på det arbete de gör mest (kodning och testning). Detta innebär att det är lätt för dem att glömma andra typer av arbete de behöver göra, såsom dokumentation, externa granskningar, utbildning av andra team (t.ex. support), förberedande steg för att stödja driftsättning eller release, och godkännanden från andra avdelningar.

När det slutligen blir uppenbart att detta "extra" arbete måste göras, blir de tagna på sängen—vanligtvis måste de avbryta det de höll på med och byta kontext för att gå tillbaka och slutföra arbete som de trodde de redan hade avslutat.

De som begär arbete kan lätt hålla arbetet öppet i onödan. Ibland med de bästa avsikter—som att försöka hålla ett team ansvarigt för det "verkliga" kravet. Andra gånger händer detta eftersom Product Owners (till exempel) har vant sig vid att kunna hålla en PBI öppen efter eget tycke, så de använder det för att pressa in ytterligare funktionalitet i en uppgift i sista minuten. Ibland gör de det till och med eftersom de har ändrat sig om vad som behöver göras mitt under arbetets gång.

Detta kan vara extremt demotiverande för ett utvecklingsteam. De flesta mjukvaruutvecklare och testare vill känna att de gör framsteg. Om de ständigt får höra att det de gjorde var fel, kommer de sannolikt att tappa en del av sin energi.

Vissa team går till och med så långt att de inte ens bryr sig om att kontrollera om det de gjorde var rätt eller fel. De stänger bara arbetsuppgiften och ber om "kredit" så att de kan "visa bra siffror".

2.6: Risken med att göra för lite eller för mycket

En risk med att inte ordentligt definiera "klar" för varje arbetsuppgift är att organisationen tror att arbetet är klart när det inte är det, eller inte inser att det är klart när det faktiskt är det.

Det värsta möjliga utfallet är ofta att fel sak hamnar i produktion och ingen vet att det är vad som hänt. Om teamet har en felaktig förståelse av vad "klar" betyder och levererar baserat på den felaktiga förståelsen, kan resultaten bli katastrofala.

Defekter och missnöjda kunder är illa nog, men detta kan också leda till mycket allvarligare problem:

- Dataförlust eller korruption
- Säkerhetssårbarheter

- Systemavbrott eller förlorad åtkomst
- En minskning av marknadsandelar
- Regelöverträdelser

Listan fortsätter och varje potentiellt problem är värre än det föregående.

Ibland är problemet att man inte är klar men tror att man är det. Det motsatta kan vara lika farligt. När utvecklare inte vet var mållinjen är, tenderar de att "förgylla" (lägga till extra funktioner). De kanske gör det för att "göra funktionen trevlig", men de kanske också gör det eftersom de hoppas att ett ökat antal funktioner ger dem en ökad chans att träffa målet.

Allt detta extra arbete, liksom motsvarande omarbete, ackumuleras till en massiv mängd bortslösad tid, ansträngning och pengar. Det orsakar försenade leveransdatum och skadar rykte.

Dessutom finns det nu en ständigt ökande risk för att ett misstag faktiskt leder till ett *Terminator*-liknande uppror mot mänskligheten från maskinerna. Vi brukade skriva om det för tjugo år sedan som ett skämt. Nu är det en avlägsen möjlighet.

Vi ställde faktiskt denna fråga till en av de mest framstående AI-systemen, och här är vad det svarade:

> "AI expanderar snabbare än någon förväntade sig, men det sker ovanpå sköra system, vaga krav och produktorganisationer som inte kan spåra varför de byggde det de byggde. Det är inte ett tekniskt problem; det är ett tydlighetsproblem.
> Ju mer brus AI genererar, desto farligare är det att röra sig snabbt utan struktur. När team bygger på dimma förstärker AI bara röran. Men när team bygger på signal—på gemensam förståelse, beteendespecificitet och versionshanterad produktkunskap—blir AI en accelerator istället för en belastning."

2.7: Var bygger man ett pussel?

Låt oss utvidga pusselbyggaranalogin en sista gång.

Kan du lägga ett pussel var som helst? Om du har ett pussel med 4 000 bitar som blir nästan fem fot i en dimension och över tre fot i en annan, kan du inte bara slumpmässigt välja en plats och börja lägga. Inte utan att uppleva allvarliga komplikationer innan du är klar.

Ett stort pussel som detta behöver både tid och utrymme. Du måste allokera utrymmet och hitta ett sätt att säkerställa att pusslets tillstånd bevaras över tid.

Om du börjar bygga ditt pussel på ett litet sidobord som är för litet, kommer du inte att kunna slutföra det utan att flytta det till en annan plats. Den förflyttningen kommer att bli extremt svår på grund av pusslets ömtåliga tillstånd.

Om du väljer en slumpmässig plats i korridoren som är tillräckligt stor kommer folk antingen att gå på den eller bli hindrade, så varaktighet kan inte garanteras utan betydande påverkan på hushållets funktion.

Om du börjar arbeta på en konstskiva, men skivan inte är tillräckligt stor, kommer du att kunna bevara tillståndet för det du har gjort, men du kommer inte att kunna slutföra pusslet utan någon form av förflyttning.

Om pusslet tidigare har tuggats på av små barn är det bäst att räkna bitarna… för det är bättre att räkna till 3999 en gång och inse att du aldrig kommer att kunna slutföra det än att investera vem vet hur lång tid på att nästan färdigställa ett pussel som du aldrig kommer att kunna avsluta.

Det finns en hel lista med saker som måste göras innan du börjar bygga ditt pussel. Att göra sakerna på listan garanterar inte framgång, men att *inte* göra dem nästan garanterar misslyckande eller allvarliga komplikationer.

Detsamma gäller för mjukvaruutveckling, fast med en högre grad av komplexitet.

2.8: När börjar implementeringen?

I grunden kan det vara svårt att avgöra när ett PBI är redo att implementeras utan en bra definition för det.

Tänk efter: Hur *vet* du?

Går du igenom allt om och om igen tills du bestämmer att det är dags?

Bestämmer någon på en ingivelse?

Sker det automatiskt i början av en iteration?

> Vi har sett många team pusha in arbete i Sprintar som inte alls var redo att implementeras bara för att de hade deadlines. Det finns några genomgående idéer om Scrum och Agilt i allmänhet som driver folk att göra detta:
>
> - Du måste få alla dina krav för Sprint *N* utvecklade i Sprint *N-1*
> - Du bör "bara sätta igång" och hantera det som går sönder längs vägen

Detta är faktiskt en spegelbild av frågan "Hur vet du när det är klart?" som nämndes tidigare och det har liknande konsekvenser. Folk kanske väntar för länge med att börja eftersom de inte vet att en arbetsuppgift är redo och de kanske börjar för tidigt eftersom de inte vet att den inte är det.

2.9: Ett A-team utan beredskap

Att inte förstå vad som krävs för att en arbetsuppgift ska vara redo har flera skadliga effekter.

Ett uppenbart sätt som ett arbetsinkrement kan vara icke-redo på är en ofullständig, otillräcklig eller saknad Definition of Done (DoD). Det leder till alla de problem vi redan har nämnt som följer av att inte ha en Definition of Done.

Men det är inte den *enda* aspekten av beredskap. Det finns många andra behov som måste tillfredsställas innan implementeringen börjar: estimering, riskbedömning och insamling av testdata är bara några vanliga exempel.

Utan att känna till och tillfredsställa dessa behov kan en arbetsuppgift kosta mycket mer än vad den behöver kosta. Tänk på ett team (A-teamet) som är beroende av ett API som utvecklas av ett annat team (det Andra Teamet). Om A-teamet gör en massa antaganden om hur det Andra Teamets API kommer att fungera och kodar efter dessa antaganden, kan det bli omfattande omarbetning när de upptäcker att det Andra Teamet arbetar mot en verklighet som inte stämmer överens med A-teamets antaganden. Med andra ord tog A-teamet en chansning och missade.

All denna omarbetning härstammar från det faktum att API:et inte var redo att användas av A-teamet.

Ibland genererar ett otillfredsställt beroende inte omarbetning, men även i dessa fall kan det fortfarande orsaka förseningar. Föreställ dig om A-teamet och det Andra Teamet kom överens om hur API:et skulle fungera och allt gick enligt plan, men det Andra Teamet tog

helt enkelt längre tid än förväntat. Som resultat kunde A-teamet helt enkelt inte testa sitt arbete ordentligt när det skulle vara klart och de var tvungna att skjuta fram sin deadline.

2.10: Att misslyckas med att ta hänsyn till schemaläggning och resurstillgänglighet

Ibland kan problemen vara så enkla som schemaläggning eller resurshantering. Vissa arbetsuppgifter kräver specifika teammedlemmar. Om den teammedlemmen ska på semester om några dagar är det förmodligen inte rätt tidpunkt att påbörja det PBI som inte kan slutföras utan hans medverkan.

> Vi hör ofta folk säga att det inte borde vara så här, men det är det ofta, oavsett. "Utbytbar personal" är en from förhoppning.

Detsamma gäller icke-mänskliga resurser. Om du kommer att behöva serverresurser för att utföra ett belastningstest bör du förmodligen säkerställa att dessa resurser faktiskt kommer att vara tillgängliga innan du påbörjar arbetet med belastningstestningen. Annars är bästa scenariot betydande förseningar och du kommer förmodligen att störa andra team/medarbetare medan du försöker skyndsamt ordna det du behöver.

Ett annat sätt som falska starter kan misslyckas på är när ett team saknar de färdigheter som krävs för att slutföra arbetet. Ibland är det en intern fråga—som att en teammedlem behöver få utbildning i ett nytt system eller forska om ett nytt API. Andra gånger är det ett schemaläggningsproblem, som när du behöver låna en UX- eller databasexpert från en pool av kvalificerade medarbetare. Det kan till och med vara ett rekryteringsproblem där teamet behöver en expert och inte effektivt kan slutföra vissa typer av arbete utan denne.

2.11: Konsekvenser av andra typer av falska starter

Vi har sett team som åtar sig att slutföra arbetsuppgifter inom Sprintar och gör kodningen relativt snabbt men ändå inte kan slutföra testningen. Detta i sig kanske inte är överraskande, men anledningen är ovanlig: testteamet hade något de behövde (som testdata) som de inte

hade samlat in innan Sprinten började och insamlingen av dessa data visade sig vara svårare eller mer tidskrävande än de hade förväntat sig.

Som ett resultat behövde arbetsuppgifterna föras över till nästa Sprint helt enkelt eftersom teamet inte hade säkerställt att de verkligen var redo att slutföra dem inom den tilldelade tiden innan de började.

Team börjar ibland implementera arbetsuppgifter när de fortfarande har öppna frågor. Faktum är att många verkar tro att det gör dem "mer Agila" när de gör det.

Detta kan orsaka enorma mängder omarbete, överraskningar eller förseningar. Om svaret på den öppna frågan visar sig strida mot ett antagande som gjordes, måste allt arbete som baserades på det antagandet ses över. Om den öppna frågan inte får svar när uppgiften ska vara avslutad, måste uppgiften antingen stängas när den kanske inte är klar, eller hållas öppen tills frågan besvaras.

Det kan vara så att teamet har ett internt beroende—en defekt som behöver åtgärdas, en föregående uppgift som måste slutföras, och så vidare. Om detta inte spåras ordentligt kan det orsaka samma problem som ett ouppfyllt externt beroende med den extra frestelsen att byta kontext och åtgärda det.

2.12: Kumulativa kostnader

Naturligtvis skapar sådana problem förseningar, omarbete och grusade förväntningar, men nackdelarna slutar inte där.

Förutom slöseriet från omarbete gör detta vanligtvis att projekt hamnar efter. Om team frenetiskt försöker stänga backloggposter och aldrig riktigt är klara över vad som krävs för att göra verkliga framsteg, tenderar de saker som faktiskt *behöver* bli gjorda att hamna på efterkälken.

Ofta, men inte alltid, leder detta till ökat tryck att leverera. När projekt hamnar allt längre efter tidplanen, kan högre ledning försöka få det tillbaka på rätt spår genom att be människor att arbeta snabbare. Det översätts ofrånkomligt till längre arbetstider.

Detta tenderar i sin tur att erodera förtroendet och försura organisationens kultur. Relationer som borde vara samarbetsinriktade blir antagonistiska. Personer som borde arbeta tillsammans för att hitta de bästa, snabbaste lösningarna, lägger energi på att fastställa att när saker oundvikligen går fel, var det inte deras fel.

I den halsbrytande jakten på funktioner och avslutade arbetsuppgifter upptäcker team ofta att de tar genvägar. Det betyder egentligen att de låter kvaliteten (särskilt kodkvaliteten)

försämras. Det innebär i sin tur att de byter framtida produktivitet mot en illusion av framsteg i nuet.

När arbetsförhållandena blir alltmer otrevliga börjar nyckelpersoner tappa engagemanget eller till och med börja se sig om efter annat.

Organisationer som beter sig på detta sätt "äter upp sitt utsäde", så att säga, på mer än ett sätt. Kodbasen blir mindre underhållbar och de personer som skulle ha underhållit den drivs alla bort.

Om det finns någon fördel med detta är den osynlig för oss.

Kapitel 3: Introduktion till Kravmognadsflöde (RMF)

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

3.1: Vad RMF inte är

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

3.2: Vad RMF är

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

3.3: Stegvist införande stöds och rekommenderas

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

3.4: RMF 1

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

3.5: RMF 2

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

3.6: RMF 3

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 4: Är det Agilt?

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

4.1: “Individer och samspel”, “Fungerande programvara”

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

4.2: Kundsamarbete

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

4.3: Att reagera på förändring

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

4.4: Transparens

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

4.5: Passar med process, överensstämmer med agilt

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Del II: Att Skapa Utrymme för Beredskap

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 5: Den första utvidgningen

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

5.1: Förberedande arbete är *arbete*

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

5.2: Att göra förberedande arbete naturligt

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

5.3: En illustrativ incident

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

5.4: Ömsesidig Påverkan

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

5.5: RMF 1:s funktion

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 6: Varför Gör Inte Folk Detta?

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

6.1: Förberedande Arbete som Andra Klassens Medborgare

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

6.2: En Allergi mot Icke-Produktivt Arbete

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

6.3: Så Det Begravdes

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

6.4: Projektledningens inflytande

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

6.5: Mönstret

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

6.6: Projekt och uppskattningar

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

6.7: Hur icke-uppskattningarna påverkar förberedelsearbetet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

6.8: Mäter hastighet, inte velocity

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

6.9: Dåliga mätningar, dåliga resultat

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

6.10: Var skulden inte ligger

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 7: Explicit förberedande arbete (RMF 1)

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

7.1: Integration med Synapse Framework™

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

7.2: Anatomi av RMF 1

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

7.3: Beteende: Reservera Kapacitet för Samarbete

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

7.4: Artefakt: Beredskapsuppgiften

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

7.5: Aktivitet: Samarbetsmötet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

7.6: Beteende: Fortsätt samarbeta tills gemensam förståelse uppnås

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

7.7: Beteende: Bekräfta Alltid Gemensam Förståelse

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

7.8: Hur RMF 1 Förändrar Arbetsflödet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 8: Effekter av RMF 1

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

8.1: Livet före RMF 1

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

8.2: Före: Tid spenderad på förståelse

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

8.3: Efter: Tid som läggs på förståelse

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

8.4: Livet efter att ha antagit RMF 1

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

8.5: Grundläggande

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 9: Att omsätta RMF 1 i praktiken

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

9.1: Utbildning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

9.2: Minimikrav per teamtyp

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

9.3: Överenskommelse

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

9.4: Förberedelse

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

9.5: Pilot

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

9.6: Utrullning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

9.7: Uppföljning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

9.8: Att förklara framgång

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

9.9: Fortsatt vaksamhet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

9.10: Vad med “Hur”?

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

9.11: Dags att Förverkliga Det!

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Del III: Grindstyrning av arbetsfärdigställande

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 10: Nästa behov

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

10.1: Tolkningsutrymme

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

10.2: Att minska tolkningsutrymmet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

10.3: Ett tredje alternativ: Inget “spelrum”

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

10.4: Möjlig påverkan på slutförande

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

10.5: Potentiell påverkan på utförandet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

10.6: Föreslagen alternativ: Lämna inget utrymme för feltolkning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

10.7: Fördelar

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

10.8: Om Rädslan för Analysförlamning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

10.9: Nästa behov: Skräddarsydda definitioner av Färdig

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 11: Vad folk vanligtvis gör

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

11.1: Om det är så bra, varför gör inte folk det?

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

11.2: Högskolan till coaching-pipelinen

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

11.3: Coaching-överbelastning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

11.4: Ett sätt som folk gör DoD på: Gör det inte

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

11.5: Endast acceptanskriterier

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

11.6: Global Definition of Done

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

11.7: Ingen Kraft

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

11.8: Sammanfattning: Termen “DoD” Är Vanligare än Faktiska Definitioner av Done

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 12: Att definiera en Definition of Done

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

12.1: Om bara en arbetsuppgift

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

12.2: Färdigställande

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

12.3: Precision

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

12.4: Struktur för en Definition av Klar

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

12.5: Specifikationer

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

12.6: Tekniska utgångskriterier

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

12.7: Produktens ingångskriterier

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

12.8: Flera delar, en grind

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

12.9: Exempel

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

12.10: Anpassning till din process

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

12.11: Sammanfattning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 13: Skräddarsydd Definition of Done (RMF 2)

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.1: Princip: Varje arbetsuppgift är unik

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.2: Beteende: Underhåll en eller flera DoD-mallar

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.3: Aktivitet: Definiera DoD-mallen

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.4: Underhåll och förbättra DoD-mallen

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.5: Flera DoD-mallar

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.6: Beteende: Använd mallar som utgångspunkter för Definitions of Done

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.7: Beteende: Kom överens om skräddarsydda Definitions of Done

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.8: Aktivitet: Definiera en arbetsuppgifts Definition of Done

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.9: Ytterligare en utökning av arbetsflödet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.10: Beteende: Mogna DoD innan implementering påbörjas

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.11: Aktivitet: Offline-analys för att mogna ett PBI:s Definition av Klar

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.12: Att lägga till Mognad i Flödet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.13: Beteende: Spåra Färdigställande i Arbetsuppgifter

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.14: Lägga till framstegsspårning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.15: Beteende: Grinda arbete genom färdigställande

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.16: Aktivitet: Använda DoD för att fastställa färdigställande

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.17: Hur grindarna passar in

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

13.18: Sammanfattning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 14: Livet med RMF 1 & 2

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

14.1: Kostnad

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

14.2: Tidslinjer

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

14.3: Påverkan på implementationsteamet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

14.4: Påverkan på produktägaren

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

14.5: Påverkan på ledningen

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

14.6: Sammanfattning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 15: Installation av RMF 2

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

15.1: Intressentmedverkan

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

15.2: Detaljnivå

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

15.3: Arbetsöverenskommelse

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

15.4: Initialt arbete

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

15.5: Utrullning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

15.6: Sammanfattning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Del IV: Grindimplementering

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 16: Det slutgiltiga kravet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

16.1: Det fanns där hela tiden

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

16.2: Timningens kraft

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

16.3: Vända på det hela

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

16.4: Risker & Kostnader

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

16.5: Värdet av att Vänta Tills Man är Redo

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

16.6: En Extra Fördel

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

16.7: Problemformuleringen

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

16.8: Behovet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

16.9: Sammanfattning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 17: Bakgrund till Definition of Ready

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

17.1: Leans arbetsgodkännande

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

17.2: Kanbans kolumningångskriterier

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

17.3: Scrum och andra agila processapokryfer

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

17.4: Surfa > Koda

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

17.5: Punktlösningar

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

17.6: Vi är redo att bli redo

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 18: Definiera en Definition av redo

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

18.1: Syfte

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

18.2: Skräddarsydd

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

18.3: Anatomi

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

18.4: Exempel

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

18.5: Överenskommelse

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

18.6: Grindvakt

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

18.7: Sammanfattning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 19: Skräddarsydd Definition av redo (RMF 3)

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.1: Ytterligare en grind i processen

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.2: Strukturen för en Definition av redo

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.3: Produktavgångskriterier

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.4: Utvecklingens ingångskriterier

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.5: Ingen skada med duplicering

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.6: Beteende: Underhåll en eller flera DoR-mallar

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.7: Varför ha en Definition av redo-mall?

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.8: Aktivitet: Definiera DoR-mallen

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.9: Underhåll mallar för Definition av Redo över tid

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.10: Mallar är bara en utgångspunkt

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.11: Beteende: Enas om skräddarsydda Definitioner av Redo

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.12: Aktivitet: Definiera en arbetsuppgifts Definition av Redo

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.13: Beteende: Gör uppgifter färdiga innan implementering påbörjas

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.14: Villkor utanför teamets kontroll

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.15: Beteende: Spåra färdighet i arbetsobjekt

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.16: Beteende: Grinda arbete genom färdighet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.17: Aktivitet: Använda DoR för att bestämma färdighet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

19.18: Sammanfattningsvis

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Del V: Syntes

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 20: De flesta deadlines spelar ingen roll

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

20.1: Verkliga deadlines

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

20.2: Godtyckliga deadlines spelar ingen roll

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

20.3: Flygbolag

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

20.4: Ursprunget till teknisk skuld

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

20.5: Marknadsförings- och försäljningsdeadlines

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

20.6: Projektdeadlines

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

20.7: Det Finns Ett Annat Sätt

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

20.8: Godtyckliga Deadlines är Inte Nödvändiga

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

20.9: Godtyckliga Deadlines Måste Avskaffas

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

20.10: Verkliga Deadlines är Fortfarande en Faktor

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

20.11: *Pis Aller*[1]

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

20.12: Slutsats

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

[1] En sista utväg, ursprungligen från franska språket.

Kapitel 21: Kompetens 1: Kravmognadsflöde

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

21.1: Som nedan, så ovan

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

21.2: Princip: Transparens i allt nödvändigt arbete

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

21.3: Beteende: Ansvar följer med arbetet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

21.4: Beteende: Synligt Spåra Kravens Status

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

21.5: Beteende: Synliggör allt arbete kopplat till förberedelse och implementering

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

21.6: Aktivitet: Förberedelsearbete

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

21.7: Beteende: Synliggör allt arbete som krävs för att slutföra en arbetsuppgift

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

21.8: Beteende: Föredra Beredskap framför Deadlines

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

21.9: Slutsats

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 22: Hur arbete och information flödar i Scrum med RMF

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

22.1: Inledande anmärkningar

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

22.2: Fånga och förbereda det initiala kravet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

22.3: Initiera förberedelsearbete

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

22.4: Planera och utföra förberedelsearbete

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

22.5: Granska förberedelseresultat

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

22.6: Planering och färdigställande av implementering

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 23: RMF:s påverkan

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

23.1: Ett kravs livscykel

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

23.2: Exempel på informations- och arbetsflöde

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

23.3: Före

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

23.4: Efter

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

23.5: Fördelar

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 24: Övergång till RMF

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

24.1: Införandemönster

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

24.2: Installation i ett Scrum-arbetsflöde

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

24.3: Andra ramverk

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

24.4: Stort motstånd: Beredskapsarbetsposter

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

24.5: Den stora förändringen: Tankesättet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

24.6: Råd för förändring

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

24.7: Slutsats

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Kapitel 25: Det är upp till dig

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

25.1: Sammanfattning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

25.2: Nu är det din tur

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Del VI: Resurser

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Appendix A: Scrum är inte problemet

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Vad är Scrum?

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Ramverk

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Scrum hanterar projekt- och arbetshantering

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Dårskapen i att behandla Scrum som ett produkthanteringsramverk

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Ingen föreskriven mekanism för att mogna krav

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Ingen investering av teknisk expertis i kravutveckling

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Antimönster

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Utökning krävs

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Att bygga på Scrum

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Bilaga B: The Synapse Framework™

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Vad Synapse omfattar

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

De tre mästerskapen

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Hur Synapse införs

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Att sammanfoga två ramverk

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Synapse-ramverkets struktur

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Organisatoriska mästerskaper

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Organisatoriska kompetenser

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Organisatoriska vanor

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Synapse-ramverkets struktur

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Anatomi hos en praxis

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Vikten av ordning

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Synapse påverkan på denna bok

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Bilaga C: Vanliga invändningar och hinder mot RMF 1

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Vanliga invändningar

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Vanliga hinder

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Bilaga D: DoD Startlistor för Kriterier

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Startlista för Tekniska Exitkriterier

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Startlista för produktens ingångskriterier

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Appendix E: Startlistor för Definition av Redo-kriterier

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Produktutgångskriterier

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Ingångskriterier för utveckling

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Sprintens ingångskriterier

Detta innehåll är inte tillgängligt i provboken. Boken kan köpas på Leanpub på https://leanpub.com/ready-sv.

Index