Levla
Levla upp din verksamhet!

Agil Komplexitetshantering

Agil komplexitetshantering

En titt i backspegeln

I mitt arbete med olika produktportföljer sedan mitten av 90-talet har komplexitet varit ett begrepp som både blivit mer diffust, men också mer konkret.

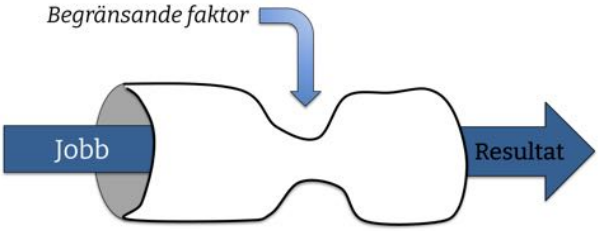
Med en färsk magistersexamen i datavetenskap var komplexitet intressant och spännande. Med tillräckligt mycket tid kändes det som att allt gick att lösa, och i någon mening är det nog sant för de flesta ingenjörsproblem - men tiden som krävdes fanns sällan där! Som programmerare blev jag irriterad på att det aldrig verkade finnas tillräcklig tid att lösa problemen *ordentligt,* och som skolan lärt oss, på egen hand.

Som projektledare blev jag varse en annan dimension av komplexitet. I första hand blev jag nu utsatt för hur svårt det kan vara för utvecklare att förklara komplexitetsnivån i ett problem så att det går att ta förnuftiga beslut i projektet - men nu inträdde också komplexiteten i samverkande team, som skapar och underhåller samverkande delar. Nu var det istället budgeten som aldrig riktigt verkade räcka. Säkert skulle vi kunna knäcka problemen om vi bara hade fem vassa utvecklare till i projektet? Tio?

När jag senare började arbeta med produktledning, blev jag så införstådd med den tredje dimensionen av komplexitet - att bygga rätt produkter, som löser rätt problem, i rätt tid och för rätt människor. Här pratar vi om *time to market, effektkartläggning, lean startup* och andra begrepp som vi kommer att spara till en annan handbok. Här ska vi främst diskutera *teknisk komplexitet och risk, och hur vi gör den synlig och agerar på den.*

En komplex portfölj

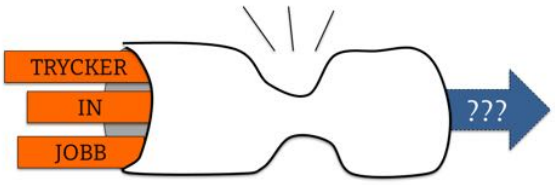
Man skulle kunna likna en organisation vid ett rör som transformerar idéer och behov till konkret nytta (t.ex. i form av produkter och produktfunktioner). Kapaciteten i "röret" utgörs av människor, kompetenser, och andra resurser, och har förstås sina begränsningar.



I en fabrik är begränsningarna ofta tydliga och enkla att spåra, medan i verksamheter som bedriver kunskapsbaserad utveckling, till exempel produktutveckling, tenderar begränsningarna att röra sig och bero på de behov som är högst efterfrågade för stunden.

Att ignorera dessa *föränderliga flaskhalsar* leder gärna till en upplevelse av en stokastisk, oförutsägbar portfölj, där resultat verkar komma i olika takt, och stressnivån kan vara omotiverat hög. Att inte känna till sin *kapacitet,* det vill säga hur mycket arbete man klarar av parallellt, och om man riskerar att överlasta sina "flaskhalsar" är ett vanligt

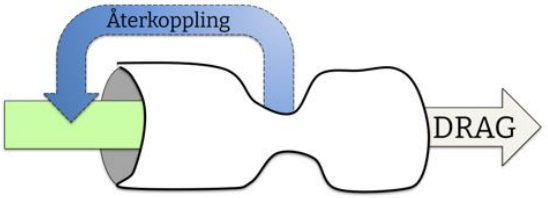
problem i produktutvecklande portföljer. Detta leder oss till en alltför vanlig bild av hur dessa portföljer ofta styrs, utan vetskap om varför resultaten ter sig oförutsägbara.



Jag brukar referera till denna form av styrning som *kapacitetsignorant,* eller för att välja en term från Lean-världen: Ett *tryckande* (“push”) system - där önskemål och projekt trycks in i röret, utan hänsyn till tillgänglig kapacitet.

Vad skulle då alternativet vara? Inom Lean brukar man lyfta fram ett *dragande* (“pull”) system eller arbetssätt som både effektivare och mer uthålligt.

En *dragande* verksamhet låter helt enkelt begränsningarna, eller flaskhalsen, styra hur mycket arbete som tas in vid varje given tidpunkt, och undviker på så sätt att överlasta portföljens kapacitet, vilket möjliggör att låta arbetet fortgå med optimal genomströmning. Jag har med åren noterat att agila arbetssätt kan tillföra den saknade pusselbiten genom att låta teamen själva planera och styra inflödet av arbete från portföljen samt sekvensera arbetet på ett vettigt sätt, ofta genom regelbundna s.k. *big room planning*-sessioner där samtliga inblandade team planerar kommande arbete samtidigt.



Ett effektivt sätt att genomföra denna typ av samplanering, och följa upp med regelbunden portföljstyrning och pulsmöten finns beskrivet i handboken om Levlamodellen.

Vad menar vi med portfölj?

Vid det här laget bör vi definiera vad begreppet *portfölj* innebär i den här handboken. Då det finns många typer av portföljer i olika industrier (inklusive sådana man en gång bar omkring papper i) gör jag det enkelt för mig, och definierar en portfölj som *en samling produkter eller tjänster som utvecklas med inblandning av en och samma mängd personal och resurser.* Så fort du har ett eller flera team (och en eller flera avdelningar) som behöver arbeta med fler än en sak, så har du en situation typisk för en portfölj, som det beskrivs här.

Men man ska väl inte jobba med flera saker?

I teorin, nej. I praktiken behöver man ofta hantera utveckling och förvaltning/support på flera produkter, och då är det svårt att undvika det hela. I många fall skulle en sådan strategi leda till konstant överkapacitet utom i några få fall, vilket skulle vara ett mycket kostsamt sätt att driva verksamheten på!

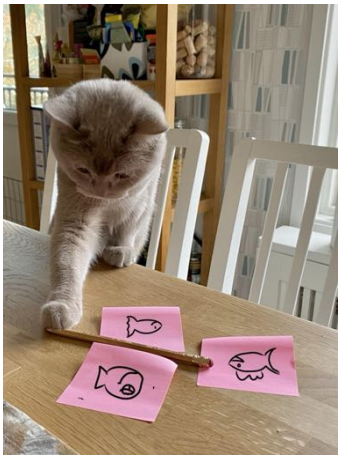
(Dessutom har vi ju många tekniker för att undvika eller hantera flaskhalssituationer och skifta fokus från beläggning till genomströmning inom både det agila och lean produktutveckling)

Agil portföljhantering

Så vad särskiljer då agil portföljhantering från traditionell? Jag skulle hävda att den viktigaste skillnaden är att *kapacitet och komplexitet* introduceras som viktiga beslutsunderlag, och att vi arbetar med KPI-er som *genomströmning (throughput), WIP (parallellt arbete)* och *cykeltid, mfl.* Här ska vi främst beröra hur vi tacklar komplexitet, så lämnar vi de övriga områdena till andra handböcker i serien.

Man kan betrakta den långsiktiga, eller *strategiska* horisonten i portföljen som ett *lågkostnadsställe* för identifierade behov, idéer och problem, då få personer är ännu inblandade i det, och den mer taktisk-operativa horisonten som *högkostnadsställe* då saker vid det laget är på väg in i, eller redan planerat i sprintar, och därmed hanteras av många fler personer. Med det tankesättet faller det naturligt att ha någon form av process för att släppa in arbetspaket från det ena till det andra, och där kommer kapacitetsstyrningen in i bilden.

Vikten av prioritering



Det kan tyckas tämligen självklart att man inte vill starta flera projekt med hög komplexitet samtidigt, och undvika situationer där allt pågående arbete närmaste tiden har hög komplexitet och osäkert utfall, och de flesta portföljer jag studerat genom åren har arbetat med riskanalys och riskmitigering i någon form. Problemet uppstår när man antingen inte gör kopplingen mellan komplexitet och risk tydlig nog, eller undgår att upptäcka ökande komplexitet under arbetets gång - alltför vanligt när det börjar bli bråttom inför deadlines och annat.

Faktum är att ett vanligt skäl till att verksamheter underlåter att agera på komplexitet beror på en oförmåga, eller kanske snarare ovilja, att *prioritera.* Att prioritera betyder ibland att säga nej, eller senarelägga saker, vilket kan innebära jobbiga samtal med kunder, vitesförlägganden, eller andra besvärliga aktiviteter man helst skulle vilja slippa ur portföljens hänseende.

Ändå är det just oförmågan till objektiv prioritering som kan försätta portföljen i en situation där man väljer att bortse från komplexitet, och istället hoppas att det löser sig. Otaliga är gångerna jag hört ledningsgruppen säga att “det brukar ordna sig om man bara ‘ingenjörar’ problemet tillräckligt”. Men, som jag brukar upprepa om och om igen:

Hopp är inte en strategi

Levla upp dina tidsestimat

I det här avsnittet ska vi gå igenom ett mycket kraftfullt sätt att adressera komplexitet - genom att levla upp våra tids- eller storleksestimat!

Ett problem jag ser med tidsuppskattningar är att de tillåts vara statiska, när de i själva verket förändras ju mer vi lär oss om en arbetsuppgift eller problemställning. Ett sätt att komma runt det är att arbeta med *regelbunden omestimering* av det arbete man håller på med, och framtida arbetsuppgifter som påverkas av det som sker för stunden.

Ett annat problem är att vi bakar in för mycket information i ett enda värde - ofta innehåller tidsestimaten inte bara en uppskattning av storleken på arbetet, utan även komplexitet, risk, resursfrågor (olösta) samt en implicit idé om *hur* arbetet borde gå tillväga. Om du sitter som produkt- eller portföljägare och ofta ser stora estimat, eller estimat med "∞"-tecknet, är detta ofta ett varningstecken om att man ser stora risker eller svårigheter med arbetet, och därmed säkrar upp tid för att lösa dem.

Då är det betydligt bättre att *bryta ut* den dolda informationen, och istället för ett enda storleksestimat arbeta med en *valuta*, bestående av flera ingående delar!

Refinement, eller förädling

För att adressera den första problematiken och bygga en kultur av att arbeta med *prognoser* och *löpande omestimering* brukar jag lära team att planera in (och tidsuppskatta) *refinement-sessioner* ("förfining" eller "förädling" om du vill hålla det svenskt) så fort de stöter på problem som verkar komplexa och/eller svårlösta.

Värdet av en refinement-session är en ökad förståelse för hur ett problem ska angripas, komplett med en enkel strategi, samt en riskanalys och riskhantering.

Genom att märka upp refinement-sessioner med en separat symbol, färg, eller om man arbetar i verktyg som t.ex. Jira, en separat tagg ,kan man enkelt få en uppfattning om komplexiteten i ett projekt eller en backlog genom att se fördelningen mellan refinement och genomförande.

En nyckelskillnad mellan traditionella och agila projekt är att agila projekt anser att det finns ett värde i att inte *försöka lösa alla problem genom stora förstudier innan projektet startas, då det i vissa fall kan vara direkt kostnadsineffektivt.*

Exempel på vad en refinement-session kan innehålla sträcker sig från proof-of-concept och designsprintar till marknadsundersökningar, effektkartläggning, tekniska granskningar, osv.

Valutan för Refinement

Jag föreslår en valuta innehållande tre komponenter: *Storlek, Komplexitet,* och *Kritisk resurs (Scarcity).*

Storlek: Detta är vårt normala tidsestimat, uttryckt antingen i absolut tid (dagar, veckor, månader, ...) eller relativ storlek (story points, ...)

Komplexitet: Ett mått på hur komplext *utförande team* bedömer uppgiften att vara, enligt någon modell. *Vi ska titta på två modeller längre ned.*

Kritisk resurs: En indikation om arbetsuppgiften kommer att involvera en *kritisk resurs* och till vilken grad. Det kan röra sig om ett team, eller en enskild medarbetare med unik kompetens eller erfarenhet, en lokal eller labb som är svårbokat, verktyg med mycket få licenser, osv.

Jag har själv arbetat med denna typ av estimatvaluta sedan början av 2010-talet, och det har visat sig ge en mycket större mängd information till portföljen jämfört med endast storleksuppskattningar - till en förhållandevis låg kostnad.

Hos en kund mätte vi hur mycket mer tid det gick åt att lägga till komplexitet och kritisk resurs, och det visade sig röra sig om ca 5-10% längre tid per session jämfört med att bara göra tidsuppskattningar - en mycket liten investering i förhållande till mängden beslutsunderlag!

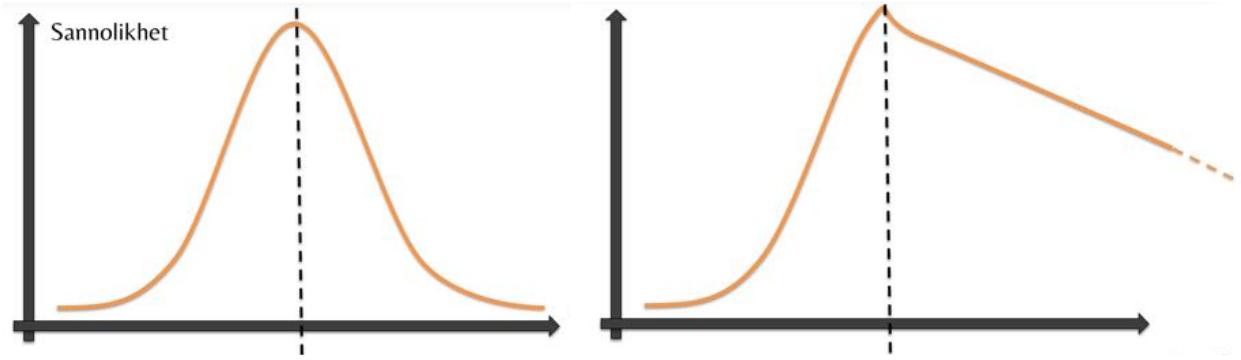
I praktiken innebär det alltså att vi i varje planeringstillfälle gör en skattning på storlek, komplexitet och kritisk resursanvändning, schemalägger refinement-sessioner för att lösa ut komplexitet, använder t.ex. big room planning för att undvika flaskhalsscenarios med kritiska resurser - och ser till att planera om regelbundet.

Asymmetriska tidsestimat

Jag har märkt att kvaliteten på tidsuppskattningar ökar när man bryter upp prognosen i en flerdelad valuta som beskrivits, helt enkelt för att teamen inte längre känner ett behov av att öka estimat eller på olika sätt signalera risk och komplexitet genom estimatet. Att regelbundet omestimera pågående arbete samt arbete som påverkas av nya insikter bidrar också till att skapa en kultur där man är mer benägen att lämna ifrån sig tidsuppskattningar även på icke nedbrutna aktiviteter, helt enkelt för att man vet att det finns möjlighet att justera dem framöver, i något som bildar ett slags självreglerande system.

Ett enkelt men kraftfullt tips i samband med storleksestimering är att ta hjälp av team eller experter för att klassificera krav som *asymmetriska.* Om vi tänker oss att tidsuppskattningar på krav följer någon form av normalfördelningskurva, där den faktiska tidsåtgången kan skilja mot estimatet med fallande sannolikhet åt ökande/minskande håll i tid, så finns det krav där det finns en risk att tiden kan öka dramatiskt under vissa omständigheter. Jag brukar kalla denna typ av krav *asymmetriska*

just för att normalfördelningskurvan inte längre är symmetrisk. Eller, på svenska: Går det åt pipan så går det *rejält åt pipan*.



Tidsestimat och sannolikhetsfördelning - till vänster en normalfördelningskurva där ursprungliga tidsuppskattningen visas av den streckade linjen, men där den faktiska tidsåtgången illustreras av kurvan i orange - det kan ta längre eller kortare tid, med fallande sannolikhet. Till höger karakteristiken hos ett s.k. asymmetriskt krav, där tidsåtgången kan bli dramatiskt mycket längre än förväntat, om oturen är framme.

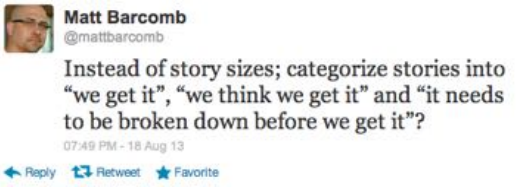
I min erfarenhet kan ofta erfarna team relativt omgående göra en klassning av en backlog av krav där de identifierar vilka de bedömer som asymmetriska, något som genast ger möjlighet att följa upp med en komplexitetsanalys för att förstå problematiken bättre.

Komplexitetsanalys

Så hur gör vi en komplexitetsanalys av kommande arbete? Behöver vi inte bryta ned allt för att förstå innehållet bättre? Jag skulle svara "ja och nej" på den frågan. För att veta *vad* vi behöver lägga energi på att bryta ned och angripa med olika typer av problemlösningsstrategi behöver vi göra en första bedömning redan *innan* kraven brutits ned för långt! Beroende på problemets karaktär kan en stor utredningsinsats i förväg vara svår, eller rent av bortkastad. Men vi kommer till det!

Barcombing

Faktum är att man ofta inte behöver krångla till det för sig. En teknik jag brukar referera till som "barcombing" kommer från en tweet skriven av Matt Barcomb, där han helt enkelt beskriver hur han låter sina team dela in kommande arbete i tre högar:



1. Det här vet vi hur vi ska göra
2. Det här tror vi oss veta hur vi ska göra
3. Det här? Vi har ingen aning!

Påfallande ofta räcker den här analysen för att ge en uppfattning om laddningen av komplexitet i en eller flera kommande sprintar.

Men det finns ännu en teknik som är kraftfull för planering av större insatser.

Cynefin

Uttalas "kynévin" och är ett walesiskt ord som betyder ungefär "plats för mina många tillhörigheter". Jag låter Dave Snowden förklara den med sina egna ord:

> *"Ramverket Cynefin utvecklades för att hjälpa ledare att förstå utmaningar och att fatta beslut inom sitt sammanhang. Genom att särskilja olika domäner (delsystemen där vi verkar) utgår ramverket från att våra handlingar måste matcha den verklighet vi befinner oss i genom en process av att skapa mening ur sammanhang.*
>
> *Detta hjälper ledare att inse vad som verkligen är komplext och vad som inte är det och agera därefter så att man inte slösar energi på att övertänka agerandet, men heller aldrig försöka få komplexa problem att passa in i standardlösningar."*

Cynefin delar in händelser, eller problem, i fyra kategorier. Vi kommer att utelämna den fjärde kategorin, "Kaos", då den används för icke förutsägbara händelser (med potentiellt katastrofala följder) och av naturliga skäl är den inte möjlig att använda sig av i förväg!

De övriga kategorierna är däremot användbara för oss. De är som följer:

Enkel (Simple)

Simpla problem där orsak-verkan-kedjan är uppenbar. Ofta finns det en entydig lösning, "best practice", som kräver minimal expertkunskap, där befintliga rutiner och checklistor kan följas.

Sammansatt (Complicated)

"Known unknowns", som Donald Rumsfeld kallade dem, är problem som ofta har flera lösningar - det finns ingen "best practice" men flera bra lösningsvägar. Specialister är ofta till stor hjälp då analysen av orsak-verkan-kedjan kräver expertkunskap. Traditionellt bryter vi ned de här problemen i mindre delar som får en egen komplexitet.

Komplexa (Complex)

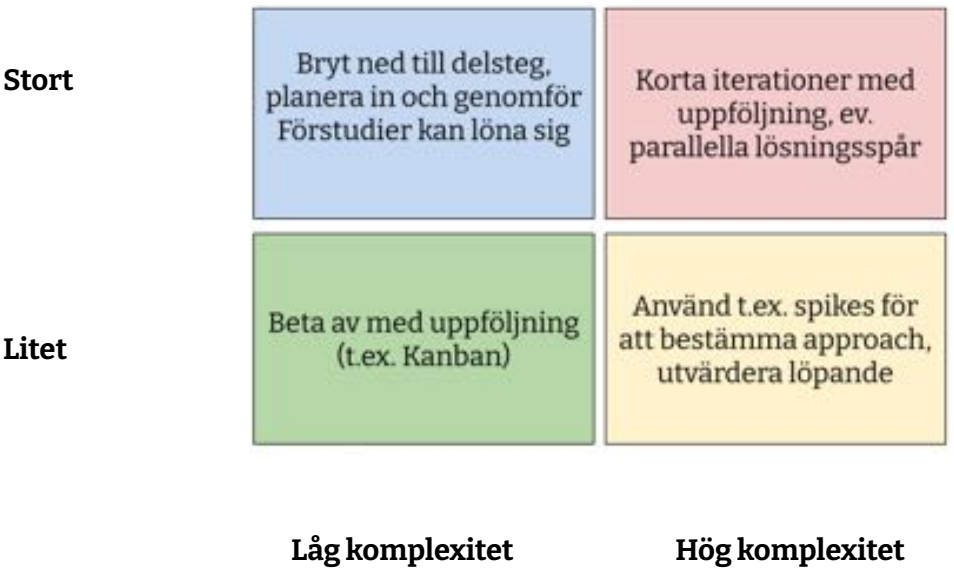
Dessa rackare till problem, där orsak-verkan ofta är uppenbar först i efterhand ville jag inte kännas vid i början av min karriär. Detta är problem där team ofta vägrar göra tidsuppskattningar eller säger att "det kommer att ta tid att ens förstå problemet". Experimentella åtgärder krävs ofta, och den slutgiltiga lösningen är inte tydlig förrän problemet är löst, medan lösningsmodeller och ny praxis arbetas fram under tiden.

Problem av komplex karaktär kan vara olösbara, men resulterar ofta i stor payoff, varför många innovativa eller disruptiva idéer bygger på att lösa just komplexa problem.

Praktisk användning

Hur använder man då dessa tekniker i praktiken? Genom att regelbundet analysera det som ligger prioriterat i portföljen (i form av roadmaps, produktbackloggar, eller liknande) med hjälp av experter kan vi få en uppfattning om riskerna och den laddning av komplexitet som ligger framför oss - men också lägga en strategi för hur vi ska angripa problemen, och i vilken ordning.

- Gör regelbunden omplanering, och estimera kommande arbete med den tidigare föreslagna valutan.
- Använd komplexitetsanalys för att välja angreppssätt
    - Komplexa krav bör ha en enkel fallback-lösning som kan användas om inte problemet visar sig lösbart inom rimliga ramar. Sätt upp i förväg ett slutdatum där problemlösningen avbryts och fallback-lösningen används istället.
    - Projekt som saknar en uppenbar fallback bör betraktas som högriskinsatser så att adekvata resurser avsätts för att hantera konsekvenserna, eller att man helt enkelt kan överge arbetet om det visar sig orimligt att lösa inom uppsatta ramar.
- Använd resultatet av Kritisk Resurs-analysen för att sekvensera arbetet så att flaskhalssituationer undviks och fokus på flöde (genomströmning) vidmakthålls.



Ovanstående matris kan ge handledning i hur man kan strukturera arbetet för att undvika att addera komplexitet i onödan. Vi ser att små uppgifter med låg komplexitet kan planeras genom förstudier eller till och med bara “betas av” med uppföljning medan uppgifter som är större och med högre komplexitet lönar sig att arbeta i högre grad med refinement (“spikes”) och korta iterationer, eventuellt med parallella lösningsspår.

Checklista för attribut som ökar komplexiteten

Här är en enkel lista på saker som, om de ingår i kommande arbete, riskerar att öka komplexiteten om de inte hanteras. Använd den i samband med bedömningar, för att se om ytterligare åtgärder behöver göras för att förenkla uppgifterna.

- ☐ Interaktionstyp - kräver lösningen avancerat stöd till användaren, eller medges att användaren har manuell kontroll?
- ☐ Antal vägar genom systemet / utfall - Ju fler, desto högre komplexitet
- ☐ Om man har ett komplext klarkriterium som involveras till hög grad tenderar det att öka komplexiteten
- ☐ Externa regulatoriska krav tenderar att tillfoga alla projekt ökad komplexitet, som behöver hanteras. En bra taktik kan vara att dela upp projektet / backloggen i delar som är utsatta för regulatoriska krav, respektive delar som *inte* är det, för att hantera komplexiteten
- ☐ Teknisk skuld, eller otestade delar av befintlig produkt, erbjuder alltid ökad komplexitet om vi behöver ge oss in i de delarna

Löpande riskhantering

Vare sig man följer ISO 31000 eller andra ramverk för riskhantering, är min uppfattning att det viktigaste är att man gör en *regelbunden klassning av risker.* Ett enkelt sätt att göra det på är att kategorisera risker utifrån den påverkan de har på projektet eller uppgiften om de infaller, respektive sannolikheten att de infaller. När det är gjort - ofta en övning som går relativt snabbt om den sker regelbundet - tar man fram riskhantering för *risker som har hög påverkan och hög sannolikhet.* Övriga risker övervakas, men vi lägger inte tid och energi på att ta fram mitigering och hantering av samtliga risker då vi kommer att omklassificera och eventuellt planera in nya mitigeringar löpande, gärna månadsvis.

Framtiden

Vad erbjuder framtiden? I år (2023) ser vi på allvar maskininlärningssystem, populärt benämnda "AI" få genomslag på bred front. Tveklöst erbjuder portföljstyrning extremt komplexa problem och beslut som i teorin skulle kunna utföras med större objektivitet än människor, av ett system som tränats på mängder av simulerade förlopp i portföljer.

I praktiken är detta dock ännu några år bort. I själva utvecklingsarbetet kommer vi ofta att ställas inför problem som de närmaste åren kommer att kräva expertis i att skapa en miljö för maskininlärning att kunna assistera oss, och där kanske det är mer lämpligt att konstatera att "AI ersätter inte chefer. Men chefer som inte använder AI kommer att bli ersatta".

Om författaren

Mikael Lundgren har arbetat med produktutveckling sedan mitten av 90-talet, och kom tidigt i kontakt med agila arbetssätt och lean produktutveckling. Idag arbetar han som sakkunnig expert, förändringsledare, utbildare och mentor till verksamheter och ledare som vill få bättre avkastning på sin produkt- och tjänsteutveckling. Han går även in som interimschef ibland, och implementerar effektiva processer för beslutsfattande och genomförande.

Mikael är Certified Scrum Trainer, certifierad SAFe programkonsult, och har en magistersexamen i datavetenskap, samt utbildningar i organisationsteori/pedagogik, och AI med Deep Learning i bagaget.

Han är pragmatisk och lösningsorienterad. Sedan 2011 driver han Levla AB som arbetat med många namnkunniga företag i Sverige i att skräddarsy effektiva beslutsprocesser för deras unika förutsättningar och regelbundet ger utbildningar inom området.

På fritiden pysslar Mikael med musikproduktion, och tycker om att läsa, skriva, fotografera och fluffa med katten Cosmos.

Kontakta författaren genom information@levla.se eller via www.levla.se !