Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for ingeniørvitenskap
Institutt for maskinteknikk og produksjon

Jørgen Slettahjell

Smidig gjennomføring av IT-prosjekter i forsvarsindustrien

Masteroppgave i organisasjon og ledelse, spesialisering i prosjektledelse og samhandling
Veileder: Nils Olsson

Januar 2022

NTNU
Kunnskap for en bedre verden

Jørgen Slettahjell

Smidig gjennomføring av IT-prosjekter i forsvarsindustrien

Masteroppgave i organisasjon og ledelse, spesialisering i prosjektledelse og samhandling
Veileder: Nils Olsson
Januar 2022

Norges teknisk-naturvitenskapelige universitet
Fakultet for ingeniørvitenskap
Institutt for maskinteknikk og produksjon

NTNU
Kunnskap for en bedre verden

Sammendrag

Programvareutviklingsprosjekter i forsvarssektoren er preget av forsinkelser og kostnadsoverskridelser. Dette begrenser forsvarssektorens muligheter til å benytte teknologi for å øke den operative evnen, være fremtidsrettet og nyskapende, koordinert og delegert – og til å utnytte nye plattformer.

Denne oppgaven ser på hvordan smidig gjennomføring av IT-prosjekter kan bidra til å møte noen av utfordringene som forsvarsindustrien står ovenfor. Dette gjøres ved å belyse forskjellene på tradisjonelle og smidige gjennomføringsmetoder, vurdere om smidig gjennomføring av IT-prosjekter er hensiktsmessig og hvordan man eventuelt kan gå frem for å bruke smidige metoder i denne bransjen.

Det brukes en kvalitativ metode med en fenomenologisk tilnærming for å finne svar på problemstillingen. Det gjennomføres dybdeintervjuer av prosjektledere og prosjektdeltakere i to ulike prosjekter hos en organisasjon som leverer programvare til forsvarssektoren. Deretter blir funnene drøftet opp mot relevant teori.

Arbeidet med oppgaven viser at forutsetningene for å ta i bruk smidige gjennomføringsmetoder ikke er til stede. På et overordnet nivå så skyldes dette at kontrakts- og anskaffelsesforholdene er for rigide. Det er ikke rom for arbeidsmåter som setter samarbeid og tillit foran kontraktsforhandlinger. På et lokalt nivå, med utgangspunkt i oppgavens empiri, så kan det virke som at casebedriften mangler de kulturelle forutsetningene som er nødvendig for å lykkes med smidig gjennomføring av IT-prosjekter.

Samtidig så vil en erkjennelse av at det er menneskene, og ikke prosessene, som er de primære driverne for prosjektenes suksess kunne dreie organisasjonene i en smidig retning. Personer og samspill er viktigere enn prosesser og verktøy – og med økt fokus på læring, medvirkning og selvstyring så vil det på sikt kunne bryte ned rigide organisasjonsstrukturer og føre til mer plastiske organisasjoner.

Abstract

Software development projects in the defence sector are riddled with delays and cost overruns. This limits the ability for the sector to take advantage of technology to increase the operational capability, to be future-oriented and innovative, coordinated and delegated – and to utilize new platforms.

This thesis looks at how agile execution of IT projects may be of aid when facing some of the challenges in the defence industry. This is done by highlighting the differences between traditional and agile software development methods, assessing whether agile execution of IT projects is appropriate, and how one can go forward with agile methods in this industry.

A qualitative method with a phenomenological approach is used to find answers to the research questions. In-depth interviews of project managers and project participants are conducted in two different projects at an organization that delivers software to the defence sector. The findings are then discussed against relevant theory.

The thesis shows that the prerequisites for applying agile software development methods are not present. At an overall level, this is due to the fact that the contract and procurement conditions are too rigid. There is not enough space to enable working conditions that value customer collaboration over contract negotiation. At a local level, based on the empirical data of the thesis, it may seem that the company lacks the cultural prerequisites necessary to succeed with agile execution of IT projects.

At the same time, recognizing that it is the people and not the processes that is the main driver for project success could make the organizations more agile. Individuals and interactions are more important than processes and tools – and with more focus on learning, participation, and self-management – we could see less rigid structures and more plastic organizations.

Forord

Denne oppgaven markerer slutten på en lang og lærerik reise gjennom det erfaringsbaserte masterstudiet innen organisasjon og ledelse ved NTNU Videre.

Studiet har vært utfordrende, inspirerende og lærerikt – samtidig som det har bydd på mye frustrasjon, stress og følelsen av utilstrekkelighet. Dyktige forelesere til tross, det er de mellommenneskelige relasjonene som har ført til størst vekst og utvikling. Møter med helt unike mennesker som har delt raust av sine erfaringer. Takk.

En takk er også på sin plass til informantene som til tross for travle arbeidshverdager tok seg tid til å bidra med de gode samtalene som ble lagt til grunn for oppgaven.

Takk til casebedriften som velvillig stilte med informanter, prosessdokumenter og gode innspill – og som ga meg tilliten til å forvalte dette på en ordentlig måte.

Takk til Nils Olsson for ekteføIt interesse, gode innspill og veiledning.

Stor takk til familien som har utvist stor tålmodighet med mine stadig tilbakevendende «jeg må jobbe med oppgaven»-utspill.

Til slutt en spesielt stor takk til Maria som har stått stødig i stormen og tatt seg av hus, hjem og barn. Vi gjør vel ikke dette igjen med det første.

Jørgen Slettahjell

Trondheim, januar 2022

Innholdsfortegnelse

Figurliste

Forkortelser

FD	Forsvarsdepartementet
FFI	Forsvarets forskningsinstitutt
FLO	Forsvarets logistikkorganisasjon
FMA	Forsvarsmateriell
HR	Menneskelige ressurser (Human Resources)
IKT	Informasjons- og kommunikasjonsteknologi
IT	Informasjonsteknologi
KD	Kunnskapsdepartementet
NASA	Den amerikanske romfartsorganisasjonen (National Aeronautics and Space Administration)
NATO	Forsvarsorganisasjon for land i Europa og Nord-Amerika (North Atlantic Treaty Organization)
NSD	Norsk senter for forskningsdata
NSM	Nasjonal sikkerhetsmyndighet
NTNU	Norges teknisk-naturvitenskapelige universitet
PC	Personlig datamaskin (Personal Computer)
PMI	Sertifiseringsorgan innen prosjektledelse (Project Management Institute)
PRINCE2	Sertifisert prosjektledelsesmodell (Projects in a controlled environment)
PRINSIX	Forsvarssektorens prosjektmodell
TSD	Tjenester for sensitive data
UiO	Universitetet i Oslo

1 Innledning

1.1 Bakgrunn

I 1939 startet den andre verdenskrig. Tysklands fremgang de første årene var fremragende. De tyske soldatene gjorde det bedre enn de allierte i omtrent alt de foretok seg. *Auftragstaktik* (Nelsen II, 1987) var en strategi som handlet om å desentralisere beslutningsmyndigheten. Avdelingene fikk tydelige mål, men de måtte selv finne ut av hvordan målene skulle realiseres. Det var kultur for å omfavne ansvar og besluttsomhet, og takhøyden var stor for å la soldater være uenige og kritiske til beslutninger som kom fra høyere hold. Dette gjorde at soldatene kjapt kunne skifte strategi og tilpasse seg uforutsette hendelser på slagmarken.

> In general, one does well to order no more than is absolutely necessary and to avoid planning beyond the situation one can foresee. These change very rapidly in war. Seldom will orders that anticipate far in advance and in detail succeed completely to execution.
>
> Helmut Karl Bernhard von Moltke, Instructions for Large Unit Commanders (1869)

En prosjektleder i et nyoppstartet teknologiselskap vil nok nikke anerkjennende til den prøyssiske feltmarskalkens ord. Den teknologiske utviklingen går i et rivende tempo, hurtige og hyppige endringer i samfunnet krever organisasjoner som er mer plastiske enn tidligere, og vi ser en dreining mot flatere strukturer og arbeid i team (Colbjørnsen, 2002).

Mange vestlige makter har studert den prøyssiske ledelsesfilosofien som dannet grunnlaget for den tyske strategien, og mange har forsøkt å kopiere den. Det norske Forsvaret er intet unntak. Med *oppdragsbasert ledelse* som sin filosofi så er forsvarssjefens grunnsyn at den med best situasjonsforståelse kan handle selvstendig i tråd med den overordnedes intensjon (Forsvaret, 2020).

Den 9. januar 2019 godkjente forsvarsministeren en ny IKT-strategi for forsvarssektoren (Forsvarsdepartementet, 2019). Hensikten med strategien er å gjøre forsvarssektoren i bedre stand til å *benytte teknologi for å øke den operative evnen*, å bli *fremtidsrettet og nyskapende*, *koordinert og delegert* med *evne til å utnytte nye plattformer*.

Den nye IKT-strategien er et svar på at organisasjonen og prosjektene ikke henger med på den rivende teknologiske utviklingen og de raske endringene i samfunnet. Forsvarssektoren er preget av prosjekter med lang gjennomføringstid (gjennomsnitt i sektoren er 8 år), at det ofte

er forsinkelser (50% av prosjektene i sektoren er mer en ett år forsinket), at det er vanskelig å gjennomføre prosjekter med en gitt kostnad og med riktig bruk av ressurser – og at det kan ta lang tid fra et behov oppstår og til det er dekket (behovet endrer seg underveis i prosjektet) (Forsvarsdepartementet, 2019).

IKT-strategien tar grundig for seg utfordringene med dagens situasjon og skisserer mål og tiltak for å nå dem. Disse berører alt fra hvordan organisasjonen må bygges opp til hvordan kulturen må tilpasses innovasjon og andre måter å jobbe på.

Det er ikke bare forsvarssektoren (FD, Forsvaret, FMA, FFI, NSM og Forsvarsbygg) som blir berørt av den nye strategien. Det er et utall leverandører til sektoren, både store og små, som må tilpasse seg en ny hverdag.

1.2 Problemstilling

Som arbeidstaker hos en av leverandørene til forsvarssektoren så synes jeg at det er naturlig å se på utfordringene i bransjen fra eget ståsted. Vår organisasjon gjennomfører for det meste prosjekter på den tradisjonelle måten, men vi har også noen prosjekter som bruker smidige metoder.

I denne oppgaven så ser jeg på hvordan **smidig gjennomføring av IT-prosjekter** kan bidra til å møte noen av utfordringene som forsvarsindustrien står ovenfor. For å gjøre dette så ønsker jeg å svare på følgende:

- Hva er forskjellene på smidig og tradisjonell gjennomføring av IT-prosjekter?
- Er smidig gjennomføring av IT-prosjekter hensiktsmessig i forsvarsindustrien?
- Hva kan vi gjøre for å ta i bruk smidige gjennomføringsmetoder i forsvarsindustrien?

Hva er smidig?

Smidighet innen prosjektarbeid handler i grunn om evnen til å håndtere endringer som skjer underveis i gjennomføringen. Smidig, eller kanskje mest det engelske ordet Agile, har i de senere årene vært et mye brukt moteord i IT-bransjen. Dermed har det også ført til at mange prosjekter har kalt seg smidig uten å egentlig være det.

Det smidige manifestet (Beck et al., 2011) har hatt stor betydning for utbredelsen av *smidig programvareutvikling* de siste tjue årene. Smidige programvareutviklingsmetoder kjennetegnes ved at krav og løsninger blir utviklet underveis i prosjektet i samarbeid med kunder og brukere, hvor arbeidet blir gjennomført av kryssfunksjonelle og selvorganiserende team.

Den rake motsetningen til å være smidig kan sies å være rigid. Tradisjonelle modeller som vannfallsmodellen, v-modellen og varianter av disse beskrives ofte som rigide.

1.3 Struktur

Denne oppgaven følger et tradisjonelt oppsett for monografibaserte masteroppgaver – med innledning, teori, metode, resultat, diskusjon og konklusjon.

Teorigrunnlaget forklarer hvordan prosjekter organiseres på et overordnet nivå og hvordan programvareutvikling gjennomføres i prosjekter. Det tar for seg ulike prosjektmodeller og ser på hvorfor teamarbeid, brukerinvolvering og læring er viktig for å lykkes med smidig programvareutvikling. Deretter hvordan forsvarssektorens prosjektmodell påvirker gjennomføring av IT-prosjekter.

Det empiriske underlaget er hovedsakelig generert ved hjelp av dybdeintervjuer av prosjektledere og prosjektdeltakere i en organisasjon som leverer programvare til forsvarssektoren. Empirien blir drøftet opp mot relevant teori for å gi svar på problemstillingen.

Oppgaven drar tidvis paralleller mellom smidig programvareutvikling og den prøyssiske ledelsesfilosofien *Auftragstaktik*. Selv om det er vesentlige forskjeller på å lage programvare og å kjempe i krig så finnes det noen interessante likheter. Dette gir oppgaven en ekstra dimensjon som kan bidra til å øke lesverdigheten.

2 Teori

Dette kapittelet presenterer teori som er relevant for å kunne drøfte problemstillingen.

Det forklarer hvordan prosjekter organiseres på et overordnet nivå ved bruk av ulike prosjekt-modeller. Dernest mer spesifikt hvordan programvareutvikling gjennomføres i prosjekter, med et historisk blikk på hvor de ulike metodene kommer fra. Dette er viktig for å kunne forstå hvorfor organisasjonene jobber som de gjør.

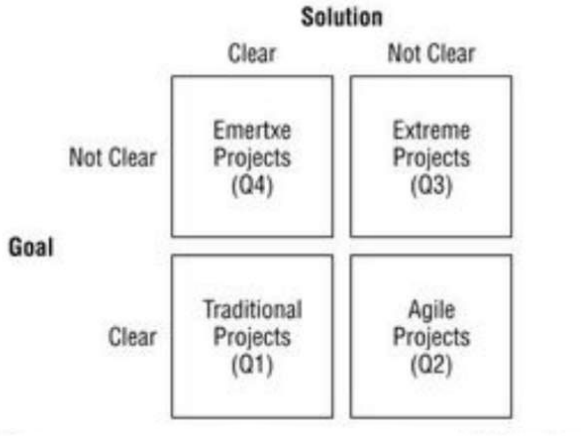
De følgende delkapitlene om teamarbeid, brukerinvolvering og læring underbygger viktige og sentrale aspekter ved effektiv programvareutvikling.

Til slutt så beskrives IT-prosjekter i kontekst av Forsvaret. Dette viser «den andre siden», og gir et innblikk i hvordan blant annet prosjektmodellen PRINSIX påvirker gjennomføringen.

2.1 Prosjektmodeller

Alle former for prosjektarbeid følger en mer eller mindre definert mal, en prosjektmodell. Hensikten med prosjektmodellen er å gi en struktur til prosjektarbeidet, slik at det blir forutsigbart og håndterbart (Rolstadås et al., 2014). Prosjektfaget er ikke entydig i hva en prosjektmodell skal inneholde eller hvor omgripende modellen skal være. Rolstadås et al. (2014) gir imidlertid eksempler på ulike prosjektmodeller hvor de beskriver prosjektfaser, prosjektroller, arbeidsprosesser, aktiviteter, leveranser og beslutningspunkter.

Robert Wysocki (2019) argumenterer for at det er prosjektets egenskaper som bestemmer hvilken prosjektmodell som bør velges for et prosjekt. Med egenskaper så mener han hvorvidt prosjektet har et klart mål og en klar løsning.



Figur 1 – Prosjektlandskapets fire kvadranter (Wysocki, 2019)

Figuren viser at det bør velges en tradisjonell prosjektmodell dersom både målet og løsningen er klar, en smidig prosjektmodell dersom målet er klart og løsningen er uklar, en ekstrem prosjektmodell dersom både målet og løsningen er uklar – og en mertske («ekstrem» bak-frem) prosjektmodell dersom målet er uklart og løsningen er klar.

Rolstadås et al. (2014) skriver at det ikke er noe i veien for å velge en tradisjonell prosjekt-modell for store prosjekter hvor det er hensiktsmessig å ha klare faser med klare leveranser, og samtidig bruke smidig tilnærming innenfor en fase, f.eks. for å løse en konkret utrednings-oppgave.

Det er verdt å merke seg at verken Rolstadås et al. eller Wysocki nevner eksterne faktorer (som organisasjonsstruktur, kontrakts- og anskaffelsesforhold eller kultur) som viktig for valg av prosjektmodell, men ser utelukkende på prosjektets egenskaper.

Boehm og Turner (2003) ser på valg av prosjektmodell i et risikoperspektiv, hvor sannsynligheten for å oppnå «et metodisk suksessfullt prosjekt» øker dersom man skreddersyr prosjektmodellen etter de fem faktorene personell, kritiskhet, størrelse, kultur og dynamiskhet. Hver av faktorene graderes et sted mellom ytterpunktene tradisjonell og smidig.

«Et metodisk suksessfullt prosjekt» er ifølge Cockburn (2002), som Boehm og Turner (2003) har basert sitt arbeid på, et prosjekt som blir levert og hvor leveransen (produktet, tjenesten etc.) blir tatt i bruk, hvor prosjektledelsen ikke har fått sparken og hvor prosjektmedlemmene ville valgt å jobbe på samme måte neste gang.

Her er en kort forklaring på de fem faktorene som Bohem og Turner (2003) legger til grunn for valg av prosjektmodell:

Personell er hvorvidt prosjektdeltakerne evner å forstå metodene som brukes i prosjektet. Tradisjonelle prosjekter krever mindre forståelse enn smidige prosjekter.

Kritiskhet er hvor store konsekvenser en feil kan få. Tradisjonelle prosjekter håndterer kritiske systemer bedre enn smidige prosjekter.

Størrelse er antall deltakere på prosjektet. Mange deltakere taler for tradisjonelle prosjekter og få deltakere for smidige prosjekter.

Kultur er hvorvidt prosjektdeltakerne trives med orden eller kaos. Tradisjonelle prosjekter lykkes bedre i en kultur med orden og smidige prosjekter lykkes bedre i en kultur med kaos.

Dynamiskhet er sannsynligheten for at det vil skje endringer underveis i prosjektet. Tradisjonelle prosjekter krever at det ikke skjer endringer, mens smidige prosjekter håndterer både få og mange endringer.

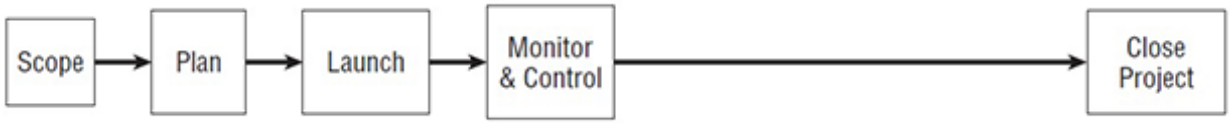
På et overordnet nivå så beskrives gjerne prosjektmodellen som en prosess med ulike faser, også kalt prosjektets livssyklus. Wysocki (2019) deler livssyklusen inn i fem faser; **oppstart**, **planlegging**, **utføring**, **overvåking/kontroll** og **avslutting**. Hver fase inneholder ulike aktiviteter, og til sammen så utgjør disse alle gjøremålene i prosjektet.

Disse fasene går ofte igjen, men de heter ikke alltid det samme og antallet kan variere. Poenget er at aktivitetene i prosjektet er nedfelt og systematisert, slik at organisasjonen sikrer at alt blir gjort og i riktig rekkefølge.

Wysocki grupperer ulike prosjektmodeller inn i lineære, inkrementelle, iterative, adaptive og ekstreme prosjektmodeller – og viser med det hvordan fasene i et prosjekt kan organiseres på forskjellige måter.

2.1.1 Tradisjonelle prosjektmodeller

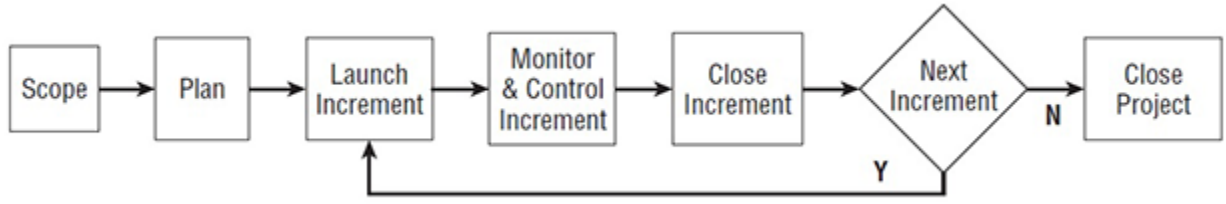
Lineær prosjektmodell



Figur 2 – Lineær prosjektmodell (Wysocki, 2019)

I en lineær prosjektmodell så følger alle fasene etter hverandre i en rett linje fra start til slutt, uten noen steg tilbake til foregående faser (Wysocki, 2019). Det betyr at det ikke er rom for endring av planene underveis i prosjektet, og skulle det mot formodning allikevel oppstå overraskelser så vil de mest sannsynlig føre til store forsinkelser og kostnadsoverskridelser. Ofte vil man i større prosjekter forsøke å ta høyde for dette ved å legge inn reserveposter for å dekke uspesifiserte og uforutsette kostnader (Rolstadås et al., 2014).

Inkrementell prosjektmodell

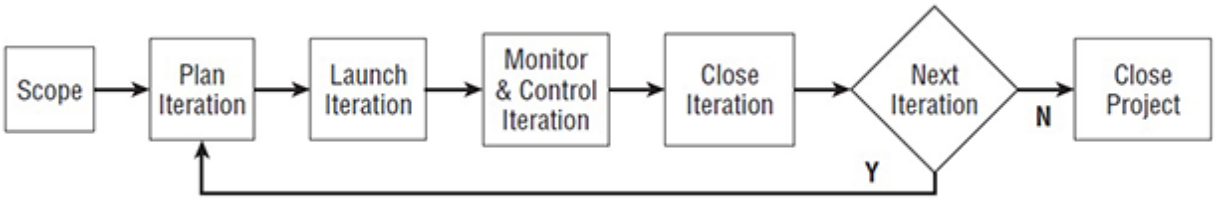


Figur 3 – Inkrementell prosjektmodell (Wysocki, 2019)

En inkrementell prosjektmodell er lik en lineær prosjektmodell med at planer og løsninger for hele prosjektet blir gjort før gjennomføringen starter, men åpner videre opp for endringer og tilleggsarbeid (Wysocki, 2019). Ofte gjøres dette etter at opprinnelig løsning har blitt levert, hvor kunden enten løser ut opsjoner eller ønsker endringer på arbeidet som har blitt gjort.

2.1.2 Smidige og ekstreme prosjektmodeller

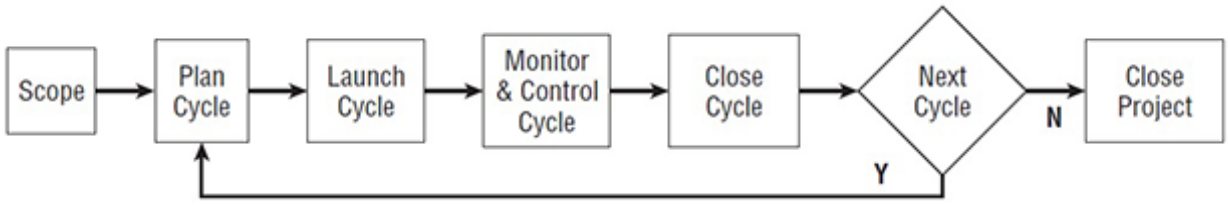
Iterativ prosjektmodell



Figur 4 – Iterativ prosjektmodell (Wysocki, 2019)

I en iterativ prosjektmodell så er målet med prosjekt klart, men ikke løsningen. Hensikten er å levere deler av løsningen for så å få tilbakemelding om endringer og videre arbeid. Dette fortsetter helt til løsningen er komplett (Wysocki, 2019). En slik prosjektmodell krever mye involvering av kunden, da prosjektet ikke kan fortsette med neste iterasjon før kunden har gitt sin tilbakemelding.

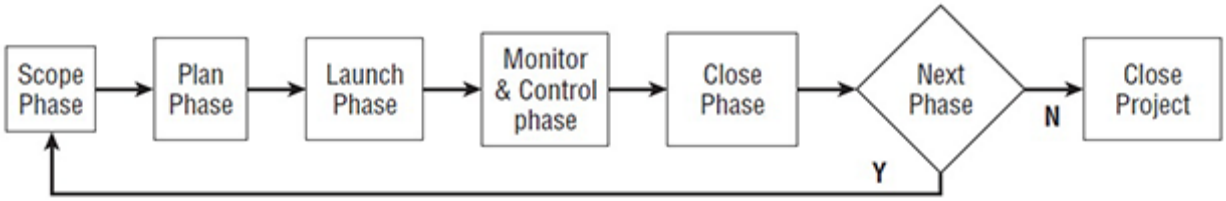
Adaptiv prosjektmodell



Figur 5 – Adaptiv prosjektmodell (Wysocki, 2019)

En adaptiv prosjektmodell har et likt mønster som en iterativ prosjektmodell, men inneholder større usikkerhet om mål og løsning (Wysocki, 2019). Modellen brukes primært for prosjekter innen programvareutvikling, og ofte er det kun grove trekk ved funksjonaliteten som er kjent ved oppstart. Denne modellen krever enda mer kundeinvolvering enn en iterativ modell, da løsningen i stor grad blir definert og utviklet underveis i prosjektet.

Ekstrem prosjektmodell



Figur 6 – Ekstrem prosjektmodell (Wysocki, 2019)

Den ekstreme prosjektmodellen brukes vanligvis i forsknings- og utviklingsprosjekter hvor verken målet eller løsningen er kjent. Den krever kontinuerlig involvering av kunden og den er ofte forbundet med høy risiko med mye prøving og feiling (Wysocki, 2019). Noen ganger er løsningen kjent, men ikke målet, da havner den i den «mertske» kategorien – ofte brukt dersom man har utviklet en ny teknologi, men enda ikke helt vet hva den skal brukes til.

2.2 Programvareutviklingsmetoder

Der hvor prosjektets gjennomføringsmodell strukturerer arbeidet fra prosjektlederens perspektiv så vil prosjektets utviklingsmetodikk strukturere arbeidet for programvareutviklerne i prosjektet.

Det er fire fundamentale aktiviteter involvert ved utvikling av en programvare (Sommerville, 2011):

Programvarespesifikasjon, hvor kunden og utviklerne definerer hva som skal lages.

Programvareutvikling, hvor utviklerne designer og utvikler produktet.

Programvarevalidering, hvor utviklerne (og eventuelt testere) sjekker at produktet oppfyller alle kravene.

Programvareevolusjon, hvor produktet endres basert på nye krav eller endringer i markedet.

Merk at begrepet «programvareutvikling» her brukes i både vid og snever forstand, henholdsvis etter «software engineering» og «software development». Generelt i denne oppgaven så brukes det i vid forstand, som inkluderer alle de fire fundamentale aktivitetene.

Programvareutviklingsmetoden er en prosess som går parallelt med prosjektmodellen. Den er hverken over- eller underordnet prosjektmodellen, men heller komplementerende og med et mer produktrettet perspektiv. Når prosjektet er avsluttet så vil produktet ofte bli utviklet videre gjennom andre prosjekter eller gjennom vedlikehold. Programvareutviklingsmetoden er derfor ofte varig og godt forankret i organisasjonen.

2.2.1 Tradisjonelle utviklingsmetoder

Tradisjonelt så har prosjekter blitt gjennomført som en lineær prosess, hvor ferdigstillelsen av én fase har vært en forutsetning for å starte den neste. I industrier som driver med vare-produksjon eller konstruksjon av for eksempel veier eller boliger så er det også ganske åpenbart at man må planlegge før man begynner å bygge, da det kan bli veldig kostbart å gjøre endringer senere i prosjektet. Dette tenkte man også om den gryende IT-industrien på sekstitallet, hvor programvareutviklingen begynte å bli en akseptert gren av ingeniørvitenskapen.

Programvareutviklingen ble imidlertid ikke tatt inn i varmen fordi den var kjent for å gå vitenskapelig til verks. Det var hovedsakelig maskinvare man utviklet på den tiden, hvor programvareutvikling var mer som en bigeskjeft å regne. Johnson (2010) beskriver en hverdag hvor små selvstyrte team hadde ansvar for all utviklingen – som inkluderte arbeid med krav, dokumentasjon, design, programmering, testing og demo med brukere. Utviklerne hadde daglige statusmøter hvor de flyttet rundt på gule lapper og fortalte om utfordringer. Det var imidlertid ingen metode eller prosess som dikterte hvordan de skulle jobbe.

Det økende potensialet som programvaren fikk som erstatter av det manuelle arbeidet førte til høyre krav til utviklingen. Programmene skulle ha mer og mer funksjonalitet, og arbeidet ble mer og mer komplekst. Utviklerne slet med å holde tritt, og det utviklet seg en aksept for at programvareutvikling innebar store budsjettsprekker og programmer med mange feil. Da disse programmene så rullet ut til sykehus, romferger og andre kritiske systemer så gikk det ut over liv og helse. Denne krisen blir kalt «programvarekrisen» (software crisis), og ble først omtalt på en programvarekonferanse i regi av NATO i 1968 (Naur & Randell, 1969).

Et klassisk eksempel på denne krisen er utviklingen av IBM sitt operativsystem OS/360. Frederick P. Brooks (1975) beskriver dette prosjektet i boken *The Mythical Man-Month*. Det var et virkelig stort prosjekt, med omtrent 70 utviklere, et budsjett på 150 millioner dollar og med en tidshorisont på to år. De fant ganske raskt ut at de nok ville bli forsinket, så de økte

antall utviklere fra 70 til 150. Men etter hvert som de ansatte flere utviklere så ble både kvaliteten på arbeidet og produktiviteten redusert, så i frykt for å bli enda mer forsinket så ansatte de enda flere utviklere. På det meste så var de omtrent 1000 utviklere på prosjektet, det ble over et år forsinket og IBM gikk på en smell på en halv milliard dollar.

Brooks har senere fått navnet sitt på en «lov», Brooks Lov, som sier at det «å legge arbeidskraft til et forsinket programvareutviklingsprosjekt gjør det enda mer forsinket». Brooks (1975) argumenterer for at man ved å ansette flere utviklere vil oppleve økt koordinering og kommunikasjon i prosjektet – og at de som allerede er produktive vil måtte bruke tiden sin på opplæring av de nye.

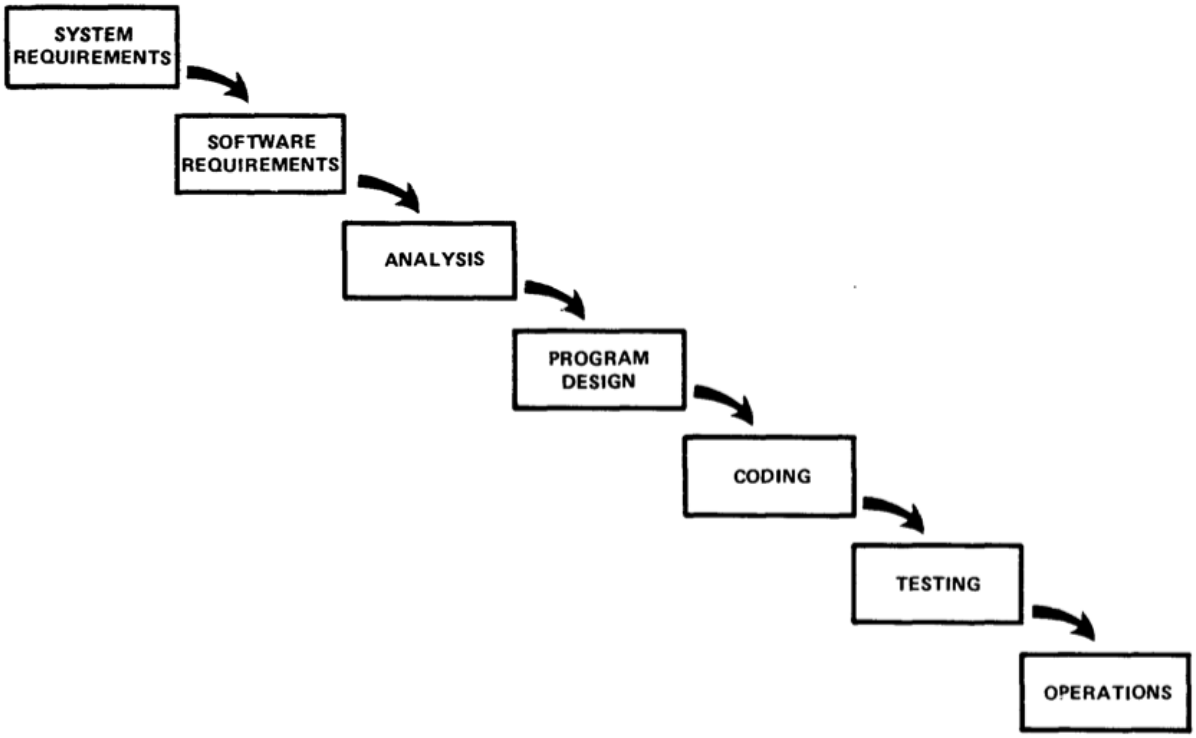
På slutten av sekstitallet så begynte det vitenskapelige selskapet Association for Computing Machinery (Bertrand, 2013) å omtale programvareutviklingen som «software engineering» nettopp for å tvinge bransjen til å behandle programvareutvikling som et ingeniørfag. Dette skulle få programvareutviklingen inn i mer ordnede former, med metoder og prosesser som tidligere var forbundet med maskinvareutvikling. Det er uklart hvem som var først ute med denne benevnelse, for også NATO (Naur & Randell, 1969) og programvareutviklingspioneeren Margaret Hamilton (som jobbet med Apollo 11 hos NASA) (Cameron, 2018) var tidlig ute.

Den mest kjente tradisjonelle programvareutviklingsmetoden kalles for **vannfallsmetoden**, fordi dens faser ofte illustreres som et vannfall hvor vannet renner nedover. Den amerikanske programvareutviklingspioneeren Winston W. Royce er kjent for å ha oppfunnet vannfalls-metoden i publikasjonen *Managing the Development of Large Software Systems* (Royce, 1970), sannsynligvis takket være en referanse i en artikkel av Bell og Thayer (1976). Han brukte ikke akkurat denne terminologien, men han beskrev en lineær modell for utvikling av programvare som inneholdt tydelige faser og beslutningspunkter. Denne ble raskt fanget opp av prosjektledere, som på grunn av dens strukturerte tilnærming (og av mangel på alternativer) innførte modellen i sine prosjekter.

På åttitallet så ble vannfallsmetoden ytterligere forsterket som standard for IT-prosjekter, da det amerikanske forsvarsdepartementet i 1985 krevde at alle som leverte programvare til dem måtte bruke metoden (DOD-STD-2167, 1985). NATO og andre myndigheter fulgte på. Det viste seg raskt at metoden ga elendige resultater, og i 1988 oppfordret derfor det amerikanske forsvarsdepartementet leverandørene om å heller bruke iterative og inkrementelle

metoder (DOD-STD-2167A, 1988). Vannfallsmetoden var da allerede blitt en de facto standard for IT-prosjekter.

Den originale metoden til Royce (1970):



Figur 7 – Vannfallsmetoden (Royce, 1970)

De fire fundamentale aktivitetene som er involvert ved utvikling av programvare (Sommerville, 2011) har i denne metoden blitt fordelt på syv faser. Programvarespesifikasjon dekkes av System Requirements, Software Requirements og Analysis. Programvareutvikling dekkes av Program Design og Coding. Programvarevalidering dekkes av Testing. Programvareevolusjon dekkes av Operations.

Etter å ha lest de første halvannen sidene av publikasjonen til Royce (1970), som inneholder beskrivelsen og illustrasjonen av metoden, så endrer ordlyden seg raskt. På andre side så skriver han:

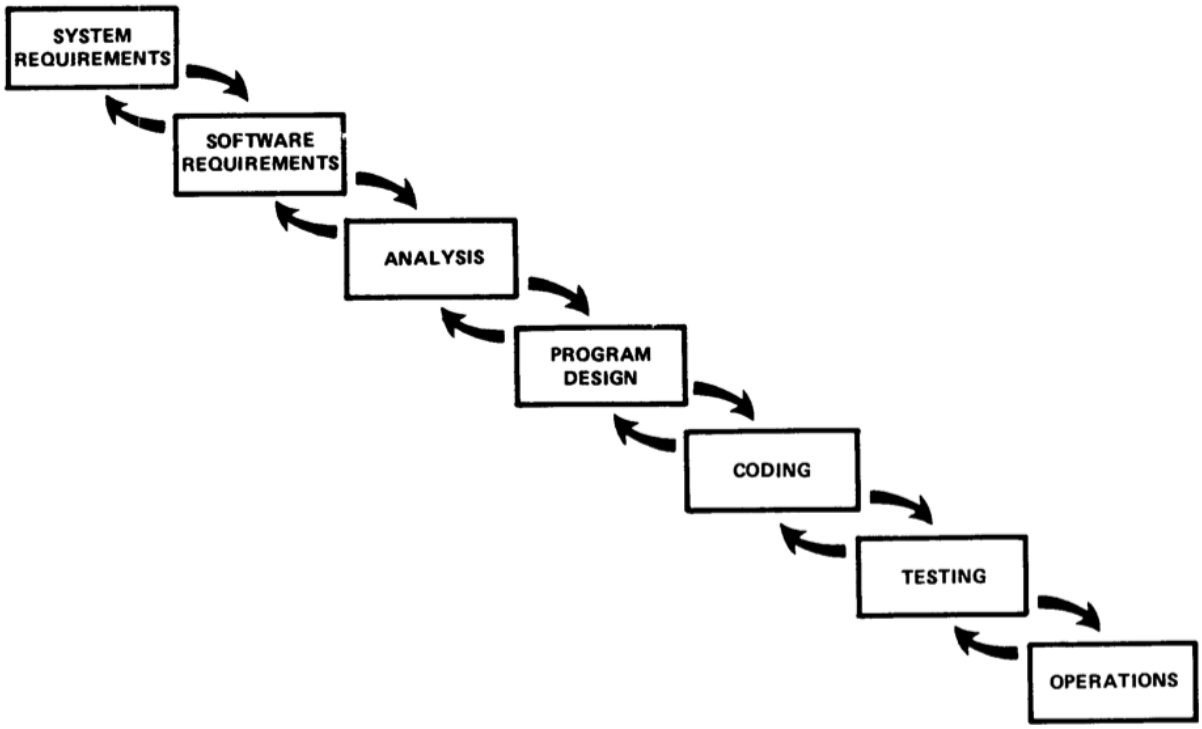
> I believe in this concept, but the implementation described above is risky and invites failure.
>
> Winston W. Royce, Managing the Development of Large Software Systems (1970), s.329

Videre så beskriver han mange problemer med metoden, blant annet med at testingen foregår sent i utviklingsløpet. Feil som blir avdekket i testingen vil sannsynligvis få store konsekvenser:

> The required design changes are likely to be so disruptive that the software requirements upon which the design is based and which provides the rationale for everything are violated. Either the requirements must be modified, or a substantial change in the design is required. In effect the development process has returned to the origin and one can expect up to a 100-percent overrun in schedule and/or costs.
>
> Winston W. Royce, Managing the Development of Large Software Systems (1970), s.329

I stedet for å anbefale denne metoden så foreslår han en rekke endringer som kan bøte på noen av utfordringene som blir identifisert. Dette inkluderer utstrakt bruk av dokumentasjon, prototyping, testing underveis og ikke minst kontinuerlig involvering av kunden. Noen av disse forslagene kjenner vi igjen i smidige utviklingsmetoder.

En av tilpasningene som Royce anbefaler til modellen kjenner vi igjen som **den iterative vannfallsmetoden**:

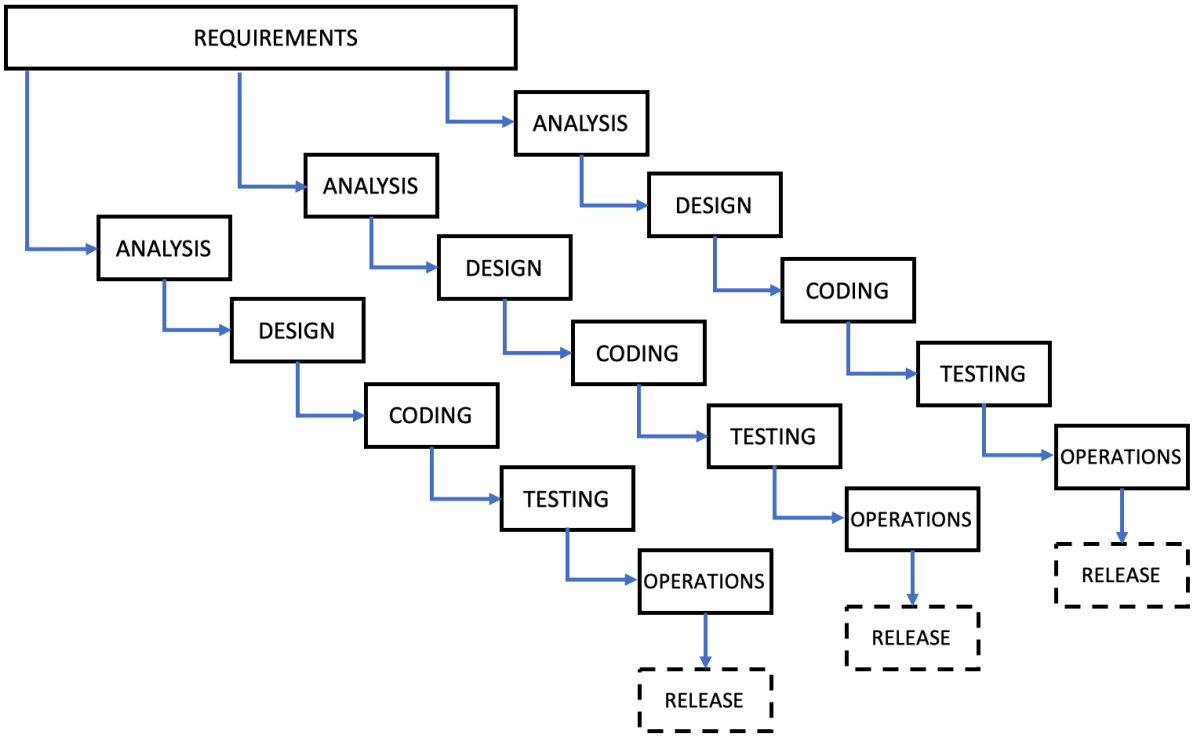


Figur 8 – Den iterative vannfallsmetoden (Royce, 1970)

Royce forklarer at man med denne metoden kan gå tilbake å gjøre arbeid i tidligere faser dersom det oppstår uforutsette hendelser. Ideelt sett så trenger man bare å gå ett steg tilbake for å gjøre endringer, men i praksis så må man ofte hoppe mange steg tilbake.

Den iterative vannfallsmetoden tilpasser egentlig bare metoden til den praktiske gjennomføringen, da det sjelden er mulig å drive med programvareutvikling uten å måtte gå tilbake for å gjøre noen endringer (Alleman, 2002; Pressman, 2015).

I stedet for å gjøre alt arbeidet i én sekvens, slik vannfallsmetoden predikerer, så kan man dele opp programvareutviklingen i mindre deler. Dette illustreres med den **inkrementelle metoden**:



Figur 9 – Den inkrementelle metoden (bearbeidet fra Pressman, 2015)

Denne metoden implementerer vannfallsmetoden i flere inkrementer (Pressman, 2015). Den kjennetegnes ved at systemet blir brutt ned i mindre deler (eller delprosjekter). Disse kan så gjennomføres parallelt og/eller sekvensielt. Resultater fra de ulike inkrementene kan enten leveres hver for seg eller samles opp til en større leveranse. Ofte så leveres et inkrement med grunnleggende funksjonalitet først. Dette gjør at brukere kan komme med tilbakemelding tidlig i prosjektet. Deretter legges det til mer og mer funksjonalitet.

Det finnes flere eksempler på tradisjonelle programvareutviklingsmetoder. **Spiralmetoden** gjennomfører fire sekvensielle faser om og om igjen i en spiral, hvor hver runde leverer en forbedret versjon av programvaren (Boehm, 1988; Pressman, 2015). **V-metoden** er en variant av vannfallsmetoden hvor fasene Requirements, System Design, Software Design og Coding illustreres som et vannfall nedover, mens fasene Software Verification, System Verification

og System Validation illustreres som en stige som går oppover (Pressman, 2015). Dette viser til hvordan de tre siste fasene verifiserer og validerer de tre første fasene i metoden.

Alle de tradisjonelle programvareutviklingsmetodene har til felles at de er prediktive og derfor godt egnet for planlegging og oppfølging i et prosjekt.

2.2.2 Smidige utviklingsmetoder

I februar 2001 så møttes sytten programvareutviklere på et fjellhotell i Utah (Highsmith, 2001). Hensikten var å danne en felles front mot den tradisjonelle måten å gjennomføre IT-prosjekter på. Det fantes mange mer eller mindre veldefinerte metoder i bruk, men det manglet en felles definisjon og enighet om hva som var viktig for å lykkes med programvareutvikling.

Dette var anerkjente og erfarne folk som representerte ulike retninger innen programvare-utvikling – som blant annet Extreme Programming, Scrum, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development og Pragmatic Programming (Highsmith, 2001).

Disse sytten programvareutviklerne kom fram til et manifest, **det smidige manifestet** (Beck et al., 2011), som de alle signerte og som dannet grunnlaget for det vi i dag kaller *smidig programvareutvikling*.

Manifestet består av et sett med sentrale verdier:

- Personer og samspill fremfor prosesser og verktøy
- Programvare som virker fremfor omfattende dokumentasjon
- Samarbeid med kunden fremfor kontraktsforhandlinger
- Å reagere på endringer fremfor å følge en plan

For å underbygge disse verdiene så har den smidige alliansen definert tolv prinsipper:

1. Vår høyeste prioritet er å tilfredsstille kunden gjennom tidlige og kontinuerlige leveranser av programvare som har verdi.
2. Ønsk endringer i krav velkommen, selv sent i utviklingen. Smidige prosesser bruker endringer til å skape konkurransefortrinn for kunden.
3. Lever fungerende programvare hyppig, med et par ukers til et par måneders mellomrom. Jo oftere, desto bedre.
4. Forretningssiden og utviklerne må arbeide sammen daglig gjennom hele prosjektet.
5. Bygg prosjektet rundt motiverte personer. Gi dem miljøet og støtten de trenger, og stol på at de får jobben gjort.
6. Den mest effektive måten å formidle informasjon inn til og innad i et utviklingsteam, er å snakke ansikt til ansikt.
7. Fungerende programvare er det primære målet på fremdrift.
8. Smidige metoder fremmer bærekraftig programvareutvikling. Sponsorene, utviklerne og brukerne bør kunne opprettholde et jevnt tempo hele tiden.
9. Kontinuerlig fokus på fremragende teknisk kvalitet og godt design fremmer smidighet.
10. Enkelhet – kunsten å maksimere mengden arbeid som ikke blir gjort – er essensielt.
11. De beste arkitekturer, krav og design vokser frem fra selvstyrte team.
12. Med jevne mellomrom reflekterer teamet over hvordan det kan bli mer effektivt og så justerer det adferden sin deretter.

Smidig programvareutvikling handler altså mer om verdier og prinsipper enn om prosesser og faser. Dette er mindre håndfast og vanskeligere å definere, men flere har forsøkt:

Boehm og Turner (2003) definerer en smidig metode som en «lett» prosess bestående av korte og iterative faser med hyppig involvering av brukere for å lage, prioritere og verifisere krav. Den legger vekt på uformell kunnskap i teamet fremfor dokumentasjon.

Cockburn (2002), en av grunnleggerne av den smidige alliansen, legger størst vekt på at smidige metoder er tidlig ute med å levere forretningsverdi (business value). For å få til dette så kreves det god kommunikasjon i teamet og tett interaksjon med brukere.

Rolstadås et al. (2014) omtaler smidige metoder som en reaksjon på de tradisjonelle metodene, hvor man bruker iterative fasemodeller med korte planleggingshorisonter og klare slutt-leveranser.

Den mest kjente og brukte smidige programvareutviklingsmetoden kalles **Scrum**. Den omtales mer som et rammeverk enn som en metode. Dette er fordi Scrum ikke sier noe om hvordan design og planer skal dokumenteres og godkjennes eller hvordan noe skal produseres. I stedet så er det et poeng i Scrum å eliminere kontrollpunkter og andre kilder til støy som ofte er forbundet med metodebegrepet (Verheyen, 2013). I denne oppgaven så omtaler jeg imidlertid Scrum som en programvareutviklingsmetode fordi den er et alternativ til vannfallsmetoden som programvareutviklingsmetode.

Terminologien ble først brukt i en artikkel av Hirotaka Takeuchi og Ikujiro Nonaka (1986). Med stadig hurtigere og hyppigere endringer i det voksende printer- og computermarkedet (med bedrifter som Fuji-Xerox, Canon, Honda, NEC, Epson, Brother, 3M, Xerox og HP) så de et stort behov for å gå bort fra den tradisjonelle lineære produktutviklingen og over til det de kalte en rugby-basert produktutvikling. Ordet «scrum» spiller på interaksjonen og samarbeidet mellom spillere i rugbysporten, hvor ballen passeres mellom spillere med ulike egenskaper fra start til slutt. De mente at utvikling basert på et slikt teamarbeid ville levere produkter kjappere og på en mer fleksibel måte, at det ville oppmuntre til prøving og feiling og at det ville stimulere til læring og nye måter å tenkte på gjennom hele organisasjonen. På sikt ville det bryte ned rigide organisasjonsstrukturer og føre til mer plastiske organisasjoner.

Det var flere som utviklet og påvirket prinsippene til Scrum som programvareutviklingsmetode gjennom nittitallet, men det er arbeidet til Jeff Sutherland og Ken Schwaber som er mest kjent. I 1993 introduserte Sutherland Scrum i selskapet Easel Corporation (Sutherland, 2004), inspirert av Degrace og Stahl (1990) sine fire grunner til at vannfallsmetoden stadig feiler for programvareutvikling:

- Kravene er aldri helt forstått før prosjektet starter
- Brukerne vet ikke hva de vil ha før de har sett første versjon av programvaren
- Kravene endres ofte underveis når programvaren blir laget
- Nye verktøy og ny teknologi gjør implementasjonsstrategien uforutsigbar

I tillegg så hadde Degrace og Stahl undersøkt «alt-samtidig»-metoden, som kort sagt gitt ut på å gjøre kravarbeid, analyse, design, koding og testing samtidig. I praksis så betydde det at én superprogrammerer gjorde alt arbeidet selv. Dette fungerte utrolig bra, men var helt umulig å skalere til større prosjekter. Det fungerte også bra med to programmerere dersom de brukte

teknikker som nå er kjent fra Extreme Programming (Beck, 2000), som blant annet parprogrammering hvor to utviklere programmerer sammen på samme PC.

Sutherland var også inspirert av den japanske produktutviklingen og han nevner spesielt Hirotaka og Takuchi (1986) sitt bidrag til arbeid i team. Den japanske produktutviklingen var også inspirasjonskilden til den kjente metoden **Lean** (Womack & Jones, 1997), som i stor grad var basert på arbeidet hos bilprodusenten Toyota. Lean og smidig har mange likheter, men der hvor smidige programvareutviklingsmetoder håndterer uforutsette omgivelser og nybrottsarbeid så trives Lean i mer forutsigbare omgivelser og med repetitive arbeidsoppgaver.

Kombinasjonen av å bygge selvstendige team, bruke verktøy som underbygde «alt-samtidig»-metoden og «best practices» fra bedrifter som Honda, Canon og Fujitsu ble til første iterasjon av det vi i dag kjenner som Scrum.

Det hele startet med at teamet brukte en halv dag på å planlegge hva som skulle lages i løpet av de neste seks månedene. Disse ble så brutt ned i seks deler som skulle gjennomføres i løpet av seks 30 dager lange iterasjoner. Disse iterasjonene ble kalt «sprints». Den samlede oversikten over oppgaver ble kalt «product backlog». Til den første sprinten så ble jobben brutt ned i arbeidsoppgaver som ikke var større enn at de kunne gjennomføres på mindre enn en dag.

Daglige møter lot alle i teamet se og dele status på prosjektet. Denne statusen hadde en mye større verdi enn den som typisk kom ut av et gantt-skjema. Dette førte til en mye mer effektiv fordeling av ressurser på teamet, da alle fikk innblikk i hva alle de andre jobbet med. Dersom noen slet med noe så kunne andre bistå. Små endringer i implementasjonen til den ene utvikleren kunne føre til avkortet arbeid hos en annen.

Kontinuerlig testing, integrering, omskriving av kode og bygging ble viktige aktiviteter for å holde de effektive hjulene i gang. Dette er elementer som er kjent fra Extreme Programming (Beck, 2000).

Hver fredag demonstrerte de produktet til likesinnede eksperter i bransjen, som kom med både positive og negative tilbakemeldinger på det de hadde gjort.

På slutten av hver måned, av hver sprint, så demonstrerte de produktet til sjefen og lot han prøve det ut. Han kom med tilbakemeldinger og en liste med funksjoner som han ønsket å få med. Før neste sprint så ble listen over funksjonalitet prioritert, og oppgaver plukket ut og brutt ned.

Sjefen så forbedringer for hver sprint og mente at produktet var klart for å leveres etter det femte inkrementet.

Samtidig så var Ken Schwaber i ferd med å lage tilsvarende beskrivelser, og publiserte artikkelen *SCRUM Development Process* (Schwaber, 1997), hvor han rammet metodikken inn ved hjelp av konsepter fra industriell prosesskontroll (teoretisk versus empirisk prosess-kontroll) og kompleksitetsteori for å forklare hvorfor systemutvikling alltid medfører uforutsette endringer.

Både Schwaber, Sutherland og Beck ble viktige og sentrale i utviklingen av det smidige manifestet (Beck et al., 2011). Samtidig så gikk Schwaber og Sutherland sammen og lagde *The Scrum Guide* (Schwaber & Sutherland, 2010). Denne har blitt videreutviklet og oppdatert gjennom årene, men prinsippene er de samme. *The Scrum Guide* (Schwaber & Sutherland, 2020) blir ansett som den offisielle definisjonen på metodikken.

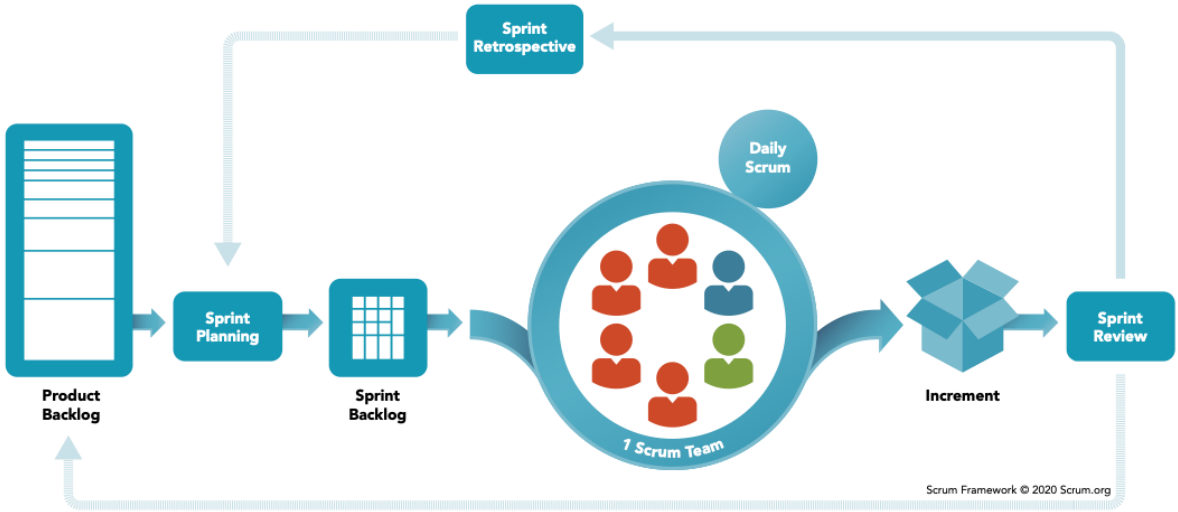
Beck (2000) utviklet Extreme Programming, som på mange måter er en egen smidig programvareutviklingsmetode. Den baserer seg også på korte iterasjoner, men har ikke like mange regler og roller som Scrum. Metoden fokuserer på verktøy som parprogrammering, test-drevet-utvikling, kodeomskriving, kontinuerlig bygging og integrering. Dette er verktøy som også er viktige i Scrum.

Selv om Scrum har utviklet seg siden 1993 så er prinsippene de samme. Som en prosess så består den av Sprint Planning, Sprint, Daily Scrum, Sprint Review og Sprint Retrospective. Disse fasene representerer planlegging, gjennomføring og avslutting av en iterasjon. Sprint Retrospective sørger for en tilbakemeldingssløyfe som muliggjør forbedringer underveis.

Rollene i Scrum består av en Product Owner, Developers og en Scrum Master. Førstnevnte representerer interessentene, ofte produkteieren eller kunden, og har som ansvar å prioritere oppgavene som skal gjøres. Dette dokumenteres i en Product Backlog. Det er utviklerne (Developers) som bestemmer hvor mange oppgaver fra Product Backlog som skal bringes over til Sprint Backlog og detaljeres. Teamet er kryssfunksjonelt og selvdrevet. Scrum Master er en rolle som på dette nivået erstatter prosjektlederen – men der hvor prosjektlederen tradisjonelt sett er den som organiserer arbeidet, så er Scrum Master den som har størst kjennskap til og organiserer prosessen. Han har for øvrig ingen myndighet over hvem som gjør hva, når eller hvordan.

> Like sheep in an open field, individuals in a project tend to stray. The ScrumMaster´s job is to keep the flock together. In fact, I often compare a ScrumMaster to a sheepdog, responsible for keeping the flock together and the wolves away.
>
> Ken Schwaber, Agile project management with Scrum (2004), s.23

Selv om *The Scrum Guide* ikke inneholder en visuell modell så har det blitt laget mange basert på beskrivelsene i den. Her er en modell basert på siste versjon (gjengitt med tillatelse) (Scrum.org, 2020):



Figur 10 – Scrum (Scrum.org, 2020)

Sutherland og Schwaber har senere kommersialisert Scrum-metodikken ved å selge sertifiseringer til blant annet Professional Scrum Master, Professional Scrum Product Owner og Professional Scrum Developer (Scrum.org, 2021).

I 2010 så kom det amerikanske forsvarsdepartementet nok en gang med oppfordringer til hvordan bransjen skulle gjennomføre programvareutvikling. De innebar tidlig og kontinuerlig involvering av brukere, hyppige leveranser og tidlige suksessive prototyper for å støtte en evolusjonerende fremgangsmåte (Bellomo, 2011). Dette er elementer som vi forbinder med smidige utviklingsmetoder, men samtidig så kan det tolkes som en videreføring av den iterative fremgangsmåten som ble anbefalt tilbake i 1988.

2.3 Teamarbeid

Den prøyssiske feltmarskalken Moltke skrev mye om hvor viktig usikkerhet er for utfallet av kamphandlinger (Hughes, 2009). Han mente at selv den mest gjennomtenkte planen ikke kan ta høyde for alle tilfeldigheter, og at mennesker av natur gjør uforutsette ting. For å møte dette så må man trene på kritisk tenkning og samarbeid. Ifølge Moltke så er hierarki et hinder for effektivt teamarbeid. Dette tok tyskerne med seg inn i soldatopplæringen, hvor de fokuserte mye på selvstendighet og bygging av tillit mellom soldater og ledere. På slagmarken så førte denne tilliten til et todelt ansvar – hvor lederne hadde ansvar for å forstå det store bildet og utlede effektive strategier og soldatene hadde ansvar for å reagere kjapt og effektivt på endringer på slagmarken.

Effektivt teamarbeid er sentralt i smidig programvareutvikling (Lindsjørn et al., 2016). Ifølge Highsmith og Cockburn (2001) så er det ikke den spesifikke programvareutviklingsmetoden som brukes som er viktigst, men selve erkjennelsen av at det er menneskene som er de primære driverne for prosjektets suksess. Dette er i kontrast til den tradisjonelle læren fra Pinto (2013), som setter prosjektlederen i sentrum. Pinto trekker frem den erfarne prosjektlederen Mike Brown (fra Rolls-Royce PLC) som et forbilde for prosjektledere. Brown mener at man løser problemene i et prosjekt før det starter og at det er «management», ikke «leadership», som er viktigst for prosjektets suksess. Det er prosjektlederen som skal planlegge, organisere, styre og kontrollere.

2.3.1 Team vs gruppe

På begynnelsen av 60-tallet så var nordmannen Einar Thorsrud sentral i et samarbeidsforsøk mellom LO og NAF (datidens NHO), hvor målet var å stoppe det sviktende engasjementet og den økende fremmedgjøringen blant arbeidstakerne i industrien (Herbst, 1977; Thorsrud & Emery, 1970). Dette samarbeidet vakte internasjonal oppsikt. Arbeidstakerne skulle få økt medvirkning, endret ledelse, mer demokrati, selvstyring og opplæring i grupper. Thorsrud definerte blant annet et sett med psykologiske jobbkrav som lå langt forut for sin tid og som etter hvert ble innarbeidet i arbeidsmiljøloven (Skogstad, 2011). Dette arbeidet ble rammen for det som han kalte *delvis selvstyrte grupper*, som på 80-tallet ble omdøpt til *team*.

Det er ikke klart om Thorsrud hadde et bevisst forhold til forskjellen på *team* og *gruppe* – eller om ordet *delvis* i sin tid ble brukt for å tekke etablerte ledere (som muligens ville sett på *selvstyrte grupper* alene som noe truende).

Ifølge Thompson (2008) så er den viktigste forskjellen på teamarbeid og gruppearbeid at medlemmene av et team har ett eller flere felles mål som de er gjensidig avhengig av hverandre for å oppnå. Ingen av medlemmene er i stand til å realisere målene alene, men må samarbeide og koordinere arbeidet seg imellom. Dersom disse egenskapene ikke er til stede så arbeider de heller i en gruppe.

Katzenback og Smith (1993) definerer et team som et mindre antall mennesker som har komplementære ferdigheter, som er forpliktet til en felles hensikt og som holder hverandre ansvarlige for prestasjonsmål og tilnærmingsmåter.

Etter hvert som man dykker ned i teamforskningen så blir det klart at begrepet også har en normativ forståelse (Bang, 2010). Å kunne kalle seg et team er i så måte noe man må gjøre seg fortjent til, man må samarbeide godt og være en sammenspleiset gjeng. Hjertø (2000) legger til at deltakernes trivsel og utvikling/læring er viktig i teamarbeid og Hackman (1987) legger vekt på at medlemmene ser på seg selv (og blir sett på) som en samlet sosial enhet i et større system.

2.3.2 Teamutvikling

Både den deskriptive og den normative forståelsen av teambegrepet forutsetter at det må skje en form for utvikling. En gruppe med mennesker som aldri har møtt hverandre før blir ikke til et team over natten. Det tar tid å lære å samarbeide, etablere normer og kjøreregler, skape felles mål, utvikle trivsel og få en følelse av samhørighet.

Tidlig på 60-tallet så fikk psykologiprofessor Bruce Tuckman i oppdrag av den amerikanske marinen å finne ut av hvordan mannskaper på små marinefartøy kunne bli til velfungerende team. Som underlag til jobben så fikk han utdelt femti artikler som handlet om hvordan blant annet terapigrupper utviklet seg over tid. Han oppdaget at gruppene gikk igjennom de samme fire fasene (Tuckman, 1965): Forming, Storming, Norming og Performing. Denne modellen ble en stor suksess i teamforskningen. Tuckman (1984) mente at det var de fengende navnene på fasene som var årsaken til at modellen ble så suksessfull.

Tuckman fikk imidlertid kritikk for at forskningen var dårlig forankret (Tuckman, 1984), og etter å ha revurdert modellen så kom han og stipendiaten Mary Ann Jensen frem til en modell som hadde en ekstra fase (Tuckman & Jensen, 1977). De slet med å finne et godt navn som rimet, men landet til slutt på Adjourning.

Forming er fasen hvor medlemmene blir kjente med hverandre og arbeidsoppgavene. De bygger relasjoner og forsøker å finne ut av hva som forventes. Dette er det første som skjer når gruppen blir dannet, men det kan også skje når det kommer nye medlemmer inn.

Storming er fasen hvor det oppstår uenigheter om hvordan oppgavene skal løses og forvirring rundt hvem som skal gjøre hva. Medlemmene har blitt varme i trøya og tør å utfordre. Én eller flere tar gjerne lederrollen og etablerer kjøreregler for resten av gruppa.

Norming er fasen hvor normene og kjørereglene har blitt etablerte. Medlemmene har fått tillit til hverandre, de samarbeider godt og de kan fokusere på å løse arbeidsoppgavene sine.

Performing er fasen hvor effektiviteten er størst. Rammene og rollene er satt og medlemmene har fått felles forståelse for målene og hvordan de skal nå dem. De er selvstyrte og krever lite ledelse.

Adjourning er fasen som markerer avslutningen på samarbeidet. Her deler de det som har vært bra og det som har vært mindre bra.

Selv om modellen ofte blir referert til i teamforskingen så blir den også ofte kritisert. Sjøvold (2006) skriver at mange grupper fungerer bra uten å ha brukt mye tid i de ulike fasene. Videre at modellen har liten overføringsverdi fordi den er basert på studier av terapigrupper. Dette sammenfaller med det Tuckman selv skriver om modellen (Tuckman, 1965). Han skriver også at observasjonene er mer kvalitative enn kvantitative og at modellen derfor er mest egnet for andre å bygge videre på. Selv etter å ha revurdert modellen i 1977, hvor han blant annet underbygger modellen med det empiriske arbeidet til Runkel et al. (1971), så skriver han at det empiriske grunnlaget er tynt (Tuckman & Jensen, 1977).

Andre forskere mener at modellen er for enkel, ikke tar hensyn til at et team kan gå gjennom fasene mange ganger og at ulike team kan oppleve dem i ulike rekkefølger (Miller, 2003; Rickards & Moger, 2000).

2.3.3 Effektivitet

Arbeid blir organisert i team fordi man antar at det er mer effektivt enn om arbeidet skulle blitt gjort av enkeltindivider. Det er i så måte naturlig å vurdere effektivitet ved å sammenlikne de oppnådde resultatene opp mot målet eller forventningene til disse. Hjertø (2000) legger til at effektivitet også er bestemt av hvorvidt teamet har oppnådd jobbtilfredshet og om det har opplevd læring, og viser til arbeidet til den amerikanske psykologiprofessoren J. Richard Hackman. Hackman er kjent for å ha studert alt fra store orkester til små flybesetninger, og er nok den som har hatt størst betydning for forskningen innen teameffektivitet.

Hackman (2002) beskriver tre kriterier som må være på plass for at et team skal kunne regnes som effektivt:

Evnen til å tilfredsstille forventninger

> The productive output of the team (that is, its product, service, or decision) meets or exceeds the standards of quantity, quality, and timeliness of the team´s clients – the people who receive, review, or use the output.
>
> J. R. Hackman, Leading Teams: Setting the Stage for Great Performances (2002), s.23

Hva som skal bli ansett som *riktig* mål på effektivitet er ikke rett frem i dagens kunnskaps-organisasjoner. Selv for team som gjør arbeid som ser ut til å ha tydelige og objektive måle-kriterier så viser de sjelden det fulle bildet. For eksempel så har man forsøkt å måle effektiviteten til programvareutviklere ved å telle hvor mange kodelinjer de produserer per time eller hvor raskt de trykker på tastaturet. For en kunde så kan dette kanskje virke som fornuftige målekriterier, men for en programvareutvikler så viser det fullstendig mangel på forståelse for hva en programvareutvikler driver med.

Hackman (2002) er imidlertid tydelig på at det ikke er teamet sine standarder eller vurderinger som ligger til grunn for å vurdere hvorvidt det er effektivt eller ikke – og heller ikke teamets leder sine standarder og vurderinger. Lederen sin jobb er å hjelpe teamet med å identifisere disse og så hjelpe teamet med å møte dem.

Teamet kan ofte ha helt andre meninger om hva som er godt arbeid enn kunden. For eksempel så kan et symfoniorkester mene at en god konsert innebærer musikk som er komplisert og utfordrende å spille – mens publikum ønsker å høre melodiske og kjente verk. Dersom orkesteret fortsetter å spille det de ønsker å spille så kan det føre til at publikum uteblir og at

de går konkurs. Dersom de i motsatt fall hører på publikum og kun spiller poplåter så kan det føre til at orkesteret mistrives og at kvaliteten visner.

Kunden kan ha alt for høye forventninger, det er ikke sikkert de vet hva de selv ønsker, det de ønsker kan være tvilsomt og til og med ulovlig – og teamet kan ha flere kunder samtidig med motstridende behov og ønsker.

Noen team er så heldige at alle de involverte er enige om hvordan arbeidet skal vurderes. Andre team må enten utdanne kunden om jobben de gjør, bruke politiske pressmidler, forsøke å erstatte kunden med en mer velvillig kunde – eller gi seg hen og gjøre det som kunden ønsker. Hva et team skal gjøre er alltid avhengig på den aktuelle situasjonen.

Gode team jobber med å tilfredsstille kundens forventninger. Virkelig gode team jobber aktivt med å justere kundens forventninger for så å overgå dem.

Evnen til å utvikle og forbedre

> The social processes the team uses in carrying out the work enhance members´ capability to work together interdependently in the future.
>
> J. R. Hackman, Leading Teams: Setting the Stage for Great Performances (2002), s.27

Det andre kriteriet som Hackman legger til grunn for å ha et effektivt team handler om felles mål og forpliktelser, kommunikasjon, evnen til å lære av hverandre og oppdage feil tidlig. Dette i motsetning til gjensidig antagonisme, uenigheter og gjentakende feil.

Teamet blir dyktige til å detektere og rette feil, de gjør hyppige vurderinger av hvordan det går og de utnytter muligheter som dukker opp underveis. Et effektivt team vil ha en vekst som gjør dem i stand til å gjøre en enda bedre jobb neste gang.

Evnen til å gi gode opplevelser

> The group experience, on balance, contributes positively to the learning and personal well-being of individual team members.
>
> J. R. Hackman, Leading Teams: Setting the Stage for Great Performances (2002), s.28

Det tredje kriteriet handler om å sitte igjen med læring og en følelse av velvære etter at arbeidet er ferdig. De enkelte medlemmene av teamet må føle at de har fått noe ut av arbeidet og at de har hatt en god opplevelse. Uten dette så kan ikke et team bli ansett som effektivt. Dersom

arbeidet hindrer medlemmene i å gjøre det dem ønsker, hvis det ikke fører til personlig utvikling eller hvis det fører til at de blir frustrerte og skuffet – så er kostnadene ved arbeidet for høyt.

Selv om disse tre kriteriene må være på plass for å si at et team er effektivt så betyr det ikke at alle er like viktige til enhver tid. Kriteriene vil vektes ulikt avhengig av arbeidsoppgavene og omgivelsene for øvrig. Noen ganger så vil omstendighetene for eksempel kreve at man ofrer både læring og fornøyelser for å dra i land et prosjekt. Dette kan kreve ubekvem arbeidstid og at man må gjøre arbeid som man verken har interesse eller glede av. Over tid så bør imidlertid gleden og fordelene ved arbeidet være større enn ulempene.

2.4 Samarbeid og brukerinvolvering

Det er bred konsensus om at samarbeid og gode relasjoner mellom kunder og leverandører er viktig for å oppnå suksess i programvareutviklingsprosjekter (Beath & Orlikowski, 1994). Det smidige manifestet trekker også frem kundetilfredsstillelse som det viktigste prinsippet for smidig programvareutvikling:

> Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
>
> Agile Alliance, The Agile Manifesto (Beck et al., 2011)

Litteraturen er ikke konsekvent i bruken av terminologien på dette området og bruker «kunder» og «brukere» litt om hverandre. Det virker imidlertid å være bredest oppfatning om at «kunder» representerer de som har kjøpt eller bestilt et produkt, mens «brukere» er de som til syvende og sist skal bruke det. Dette kan være de samme menneskene, men trenger ikke å være det. Samtidig så kan brukere være de som er avhengig av det som kommer ut av produktet, være de som dytter ting inn i produktet, uttrykker behovene for produktet, godkjenner kjøpet av produktet – og vil på mange måter kunne påvirke i hvilken grad kunden er fornøyd eller misfornøyd med produktet. Sånn sett så er «kunder» et videre begrep enn «brukere».

Tradisjonelt så er programvarespesifikasjon den viktigste aktiviteten ved utvikling av en programvare:

> The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the

> detailed technical requirements … No other part of the work so cripples the resulting system if done wrong. No other part is as difficult to rectify later.
>
> F. P. Brooks, No Silver Bullet: Essence and Accidents of Software Engineering (1987)

Å konseptualisere og spesifisere hva som skal lages krever ikke bare en dyp forståelse av problemet som skal løses, men også en forståelse av hvordan programvaren kan hjelpe til med å løse det. Kunden forsøker å beskrive hva som skal lages, kravene utarbeides og sendes videre til utviklere som så skal lage programvaren. Dette er imidlertid langt fra en perfekt form for kommunikasjon, og behov og hensikt kan ofte forsvinne på veien. Etter at funksjonaliteten har blitt implementert og testet mot kravene så er det ikke sikkert at den løser problemet slik kunden hadde sett for seg.

Smidige metoder baserer arbeidet i større grad på brukerhistorier (user stories) enn på forhåndsdefinerte krav (Maiden & Jones, 2010). I tillegg så er det forventet at smidige team jobber tett med kunden for å justere disse gjennom hele prosjektet. Det er imidlertid stadig utfordringer med at kunden ikke har gode nok kunnskaper om kompleksiteten til systemene som utvikles. Disse utfordringene blir ekstra fremtredende når kunden i tillegg ikke setter av tid og ressurser på å følge opp samarbeidet.

Et annet aspekt er tilbakemelding fra brukere. I motsetning til spesifisering av krav så må tilbakemeldinger nødvendigvis komme underveis eller i hvert fall etter at prosjektet har startet. I tillegg til formelle og uformelle klager på produktet så skal tilbakemeldinger også fange opp skjulte behov og gode ideer (Fundin & Bergman, 2003).

Tilbakemelding fra brukere blir ansett som enklere ved bruk av smidige metoder, hvor dette skjer kontinuerlig gjennom hele prosjektet. Tilbakemeldingene er imidlertid lite verdt dersom de ikke når frem til programvareutviklerne eller dersom de ikke bidrar til å forbedre produktet. Brukerne må ofte motiveres til å holde seg involverte og obligatorisk deltakelse fører sjelden frem.

2.5 Læring

Etter den prøyssiske feltmarskalken Moltkes bortgang så begynte utviklingen å gå mer i retning av den konservative og ordrebaserte *Befehlstaktik*, en strategi som på mange måter slo bena under den humanistiske utdanningstradisjonen som var forbundet med *Auftragstaktik*. Denne

dreiningen mot det autoritære skyldtes ifølge Mattsson (2003) et økt press fra konservative krefter og en større satsing på teknologi og naturvitenskap. Dette bidro nok til at tyskernes inntog i Frankrike i begynnelsen av den første verdenskrig ble til tre lange og kalde år i skyttergravene, hvor både tyskerne og de allierte bet seg fast ved hjelp av rigide taktikker, sentralisert styring og forbud mot tilbaketrekning. For tyskerne så ble dette en bekreftelse på at strategien var mislykket – og fokuset ble nok en gang plassert på læring og utvikling av individet.

> The principal thing now is to increase the responsibilities of the individual man, particularly his independence of action, and thereby to increase the efficiency of the entire army.
>
> Hans von Seeckt, Observations of the Chief of the Army Command (1925)

Læring ble igjen sentralt i *Auftragstaktik*. Tyskerne oppfattet krigens natur som kompleks og uforutsigbar, og det var derfor helt avgjørende å utvikle soldatene til å tenke og handle selvstendig (Nelsen II, 1987).

Fra et systemisk og organisatorisk perspektiv så er læring en endring som skjer som følge av en prosess – hvor individet, teamet eller organisasjonen bevisst skaper kunnskap (Bateson, 1972; Qvortrup, 2004). Læring skjer altså ikke av seg selv, men er en villet handling.

Qvortrup (2004) trekker frem tre viktige former for kompetanse som fremmer læring i systemer:

Refleksjonskompetanse er evnen til å innta en annens holdning til egne meninger og handlinger og reflektere over disse.

Relasjonskompetanse er evnen til å se den andres synspunkter og være i stand til å se en sak fra flere sider.

Meningskompetanse er evnen til å identifisere en meningshorisont for fellesskapet, for eksempel ved å finne mening i politikk og økonomi, og til å stille spørsmål ved felles verdier.

Har man disse kompetansene så vil man til dels ha fremmet læring på et individuelt plan, til dels ha fremmet læring på et kommunikativt plan og til dels ha fremmet læring på et sosialt plan.

Bateson (1972) mener at læring kommer innenfra og avhenger av våre antakelser om hvordan verden rundt oss fungerer – hvor vi i samhandling med omgivelsene lager en subjektiv konstruksjon av virkeligheten som gir oss forståelse og mening. Denne antakelsen kan utfordres av hendelser, refleksjon eller dialog med andre mennesker, som dermed kan føre til at den endres.

Bateson og Qvortrup er ikke veldig tydelige i sine beskrivelser av hva som fremmer læring, men Qvortrup (2004) mener at det må finnes rom for å reflektere over ulike perspektiver. Selv om læring kan oppstå alene i form av indre dynamikk, for eksempel når man jobber med krevende oppgaver, så oppstår den som regel etter en eller annen form for ekstern stimuli. Hvordan mennesker samhandler har derfor stor betydning for læring (Qvortrup, 2004).

Organisasjonen må være bevisst på hvorfor den ønsker å organisere i arbeidet i team. Dette må sees i sammenheng med oppgavene som skal gjøres. Det er ikke alle oppgaver som er hensiktsmessig å gjøre i team, for eksempel oppgaver som krever høy individuell kompetanse eller er av repeterende art (Oldham & Hackman, 2010). Samtidig så kan det allikevel være hensiktsmessig å gjøre slike oppgaver i team dersom det gjør at de ansatte trives bedre (Assmann, 2008). Motivet kan også være å vise omverden at organisasjonen er i stand til å tilpasse seg dagens moderne arbeidsformer.

Det sosiale aspektet i teamarbeidet har fått økt fokus i moderne organisasjoner (Oldham & Hackman, 2010). Samarbeid med andre om felles oppgaver kan føre til at man identifiserer seg med og er i stand til å se andres perspektiver – noe som dermed kan føre til læring. I kryssfunksjonelle team så er det ikke bare de faglige spesialiseringene som kan gi andre perspektiver, men også de personlige. En designer har kanskje en annen bakgrunn og en annen personlighet enn en programmerer. Deltakere kan komme fra ulike befolkningsområder, ha forskjellige interesser og politiske syn. Slike sosiale aspekter er viktige for å kunne se ting på andre måter og å utvikle læring (Oldham & Hackman, 2010).

Selv om arbeidet er organisert i kryssfunksjonelle team og organisasjonen har et bevisst forhold til læring så garanterer ikke dette at det oppstår læring. Ifølge Qvortrup (2004) så er det også en forutsetning at arbeidet skjer i en læringskultur, det vil si at det finnes en variasjonsramme for læringsprosesser som på den ene siden synliggjør hva som er mulig og ikke mulig (for eksempel gjennom tradisjon, historie, felles vedtekter og regler) og på den andre siden gjør

dette i form av rammestyring. Det er ikke nok at den enkelte har en målrettet kommunikasjon i så måte, det krever et kontinuerlig samvirke av mange kommunikasjoner.

For å utvikle en sterk læringskultur så er det en forutsetning at forhold mellom ledere og de ansatte, likeså mellom de ansatte, er basert på tillit (Filstad, 2016). Kunnskap som begrep innebærer å kombinere informasjon med erfaring, mening, forståelse og refleksjon (Filstad, 2016) – og dermed vil deling av kunnskap også til en viss grad være et uttrykk for en emosjonell tilstand. Med dette så kan deling av kunnskap altså føre til at man havner i en sårbar posisjon, og innebære en risiko med konsekvenser for stillingen og posisjonen i organisasjonen.

Filstad (2016) mener at det er i de uformelle relasjonene at kunnskapsdelingen ofte skjer, hvor man involverer seg med og identifiserer seg med de man stoler på og har tillit til.

I denne sammenhengen så er det også relevant å trekke frem den prisvinnende boka til Gunnar Ekman (2004), *Fra prat til resultat*, som handler om hvordan man utøver lederskap gjennom disse uformelle møtene. Ifølge Ekman så er det småpraten rundt kaffemaskina som skaper tillit, og det er gjennom småpraten hvor den tause kunnskapen spres.

2.6 IT-prosjekter i Forsvaret

Siden den første krigeren så sitt snitt til å plukke opp en stein til sin fordel i nærkamp så har teknologien formet hvordan vi kjemper og forsvarer oss. Tubbs et al. (2002) beskriver hvordan spesielt utviklingen innen telekommunikasjon har formet krigføringen. Telekommunikasjon var også fundamentet for internettet – hvor distribuerte datamaskiner dannet et nettverk for deling av informasjon, først som et militært eksperiment, men som senere utviklet seg til å bli ryggraden i det moderne samfunnet.

Tubbs et al. (2002) beskriver videre hvordan både kommersielle og offentlige initiativtakere satte sikkerheten til side til fordel for kapabilitet, kostnader og tidsplaner når de utviklet nye internettløsninger. Sikkerhet brukte for mye systemressurser, som gjorde systemet tregt, og de satte begrensninger for hva brukerne av systemet kunne bruke det til. Sikkerhet var dyrt, og det tok uforholdsmessig lang tid å utvikle programvare uten feil. Et distribuert system har ingen «single-point-of-failure», så det er dermed vanskelig å stoppe eller å forstyrre. Paradoksalt nok så fører dette også til at det er vanskelig å stenge ute de med uredelige hensikter.

I Norge så har vi hatt store satsinger på digital infrastruktur de senere årene. Kommunal- og moderniseringsdepartementet har uttalte ambisjoner om at vi skal ha «verdens beste digitale tjenester og en offentlig sektor som legger til rette for innovasjon og nyskaping» (Kommunal- og moderniseringsdepartementet, 2018). Ifølge Statistisk sentralbyrå (Røgeberg, 2019) så er vi allerede blant verdens beste i bruk av digitale tjenester.

Et samfunn som i høy grad er avhengig av digital teknologi for å fungere vil selvsagt også være veldig sårbart for digitale trusler. Både Politiets Sikkerhetstjeneste og Forsvarets Etterretningstjeneste mener at «trusler i det digitale rom» er blant de største truslene som Norge står ovenfor (Etterretningstjenesten, 2021; PST, 2021), hvor de særlig trekker frem digitale etterretnings- og påvirkningsoperasjoner fra fremmede stater. Dette setter dertil høye krav til de vi har satt til å beskytte oss.

Ansvaret for å beskytte oss mot digitale trusler er delt mellom Nasjonal sikkerhetsmyndighet (NSM) og Forsvaret. Førstnevnte har ansvar for den sivile infrastrukturen og sistnevnte for den militære cyberberedskapen. Med en økt dreining fra konvensjonell til hybrid krigføring så blir skillet mellom militære og sivile mål visket ut. Dette setter høye krav til samarbeid mellom aktørene. Nasjonal strategi for digital sikkerhet understreker viktigheten av internasjonal og sivil-militært samarbeid.

Ifølge generalmajor Inge Kampenes, som er sjef for det norske Cyberforsvaret, så er vi dårlig rustet mot utenlandske cyberangrep (Eide & Nørstebø, 2017). Dette bekreftet også tidligere forsvarssjef Haakon Bruun-Hanssen med sine fagmilitære råd til regjeringen i 2019, hvor han poengterte viktigheten av å styrke Forsvarets digitale modenhet for ikke å miste avgjørende kampevne (Forsvaret, 2019).

I 2014 så innbød Forsvarsdepartementet til en anbudskonkurranse om bistand til modernisering og effektivisering av forsvarssektoren (McKinsey & Company, 2015). Hensikten var å identifisere, kvantifisere og beskrive potensialet for modernisering og effektivisering av utvalgte forvaltningsområder og funksjoner. I 2015 så leverte McKinsey en rapport som tok for seg all ikke-operativ virksomhet, blant annet Forsvarets IT-investeringer (McKinsey & Company, 2015). Rapporten er knusende og beskriver IT-investeringene som preget av rot, inkompetanse, sløsing og ansvarspulverisering. Der hvor gjennomsnittlig gjennomføring av et IT-prosjekt i offentlig og privat sektor er på to og et halvt år så er gjennomsnittet for IT-prosjekter i Forsvaret på over åtte år. Videre omtales IT-investeringene som pengesluk, både

fordi de tar lang tid og fordi de er kostbare i drift. Det fører til at «utstyr og programvare både er utdatert og for dyrt når det leveres» og at «operativ evne svekkes da nødvendig materiell og tjenester leveres lenge etter at behovet oppstår».

Rapporten peker på fem faktorer som forklaring på de dårlige resultatene:

- Sektoren har en uegnet organisasjonsstruktur.
- Det er store variasjoner i prosjektgjennomføring og viktige kompetansegap til beste praksis.
- Investeringsprosessen er for omfattende og unødvendig komplisert.
- En for høy andel av kostnadene går gjennom prosjektfinansiering.

I tillegg til reorganisering så trekker rapporten frem endringer i prosjektgjennomføringen som viktige tiltak, blant annet at det er nødvendig med opplæring i hvordan prosjektmodellen kan brukes på en mer smidig måte.

I etterkant av McKinsey-rapporten så har deler av Forsvaret vært gjennom en reorganisasjon. I januar 2016 så ble FMA opprettet, hvor deler av FLO ble underlagt. Hensikten var blant annet å oppnå kvalitetsforbedring og effektivisering av materiellinvesteringer og materiellforvaltning (Forsvarsdepartementet, 2015).

Den 9. januar 2019 godkjente forsvarsministeren en ny IKT-strategi for forsvarssektoren (Forsvarsdepartementet, 2019). Hensikten med strategien er å gjøre forsvarssektoren i bedre stand til å *benytte teknologi for å øke den operative evnen*, å bli *fremtidsrettet og nyskapende*, *koordinert og delegert* med *evne til å utnytte nye plattformer*.

IKT-strategien går i dybden på flere av poengene fra McKinsey-rapporten, blant annet hvorfor det er så lang ledetid på prosjektene. Her trekkes utarbeidelse av beslutningsdokumentasjon fram som svært tidkrevende og at det er et stort antall beslutningspunkter og steg som gjør prosessen lite fleksibel. Andre poenger er at planleggingen ofte er for optimistisk og ikke tar inn over seg kompleksiteten som ofte følger skreddersydde løsninger – og at prosjekt-gjennomføringen er lite effektiv.

Investeringsprosjektene til Forsvaret planlegges årlig for de neste 10-15 årene (Forsvarsdepartementet, 2019). Et ledd i denne planleggingen er å utarbeide perspektiv-planer med kort (4 år), mellomlangt (8 år) og langt (20 år) perspektiv. Ifølge IKT-strategien så planlegges det alt for ambisiøst fremover i tid og at en mulig årsak er at det mangler

kompetanse og forståelse for hvordan IKT påvirker måloppnåelse i de øvrige investeringsområdene.

Det er forsvarssektorens prosjektmodell PRINSIX som benyttes for gjennomføring av investeringsprosjekter (FMA, 2020). Denne beskriver faser, beslutningspunkter, roller og ansvar – og følger definerte faser på ulike nivåer fra prosjektidé til fremskaffelsesløsning. Forsvarsdepartementet (2019) skriver at metoden i utgangspunktet er fleksibel, men at manglende kompetanse er en viktig grunn til at den ofte blir for komplisert og langtekkelig. Andre grunner som nevnes er at det er for stort fokus på å beskrive den tekniske løsningen i detalj fremfor hvilke behov den skal løse, og at antall beslutningspunkter gjør det krevende å få til smidige gjennomføringer.

Figur 11 – Forsvarssektorens prosjektmodell, PRINSIX (FMA, 2020)

Veilederen til FMA (2020) er sparsom med informasjon om de ulike fasene. Gjennomføringsfasen er kanskje spesielt tynt beskrevet. Det er lagt størst vekt på tidligfase og det er blant annet spesifisert eksplisitt at detaljerte kravspesifikasjoner må være på plass før oppdrag kan lyses ut til potensielle leverandører.

PRINSIX utdanningsprogram er et helhetlig og målrettet utdannings- og sertifiseringsløp innen prosjektledelse som er beregnet på ansatte i forsvarssektoren (FMA, 2021). Utdanningsprogrammet benytter ifølge nettsiden internasjonalt anerkjent prosjektmetodikk (PMI, PRINCE2) og forsvarsspesifikk prosjektpraksis spesielt basert på PRINSIX rammeverk. Læringsmålene som er listet for de ulike kursene viser at deltakerne får grundig opplæring i praktisk prosjektledelse. De avslører også at prosessene har tunge tradisjonelle aner, blant annet så blir V-modellen (en variant av vannfallsmetoden) nevnt flere ganger. Her er det ingen læringsmål som sier noe om tilrettelegging for smidig gjennomføring.

3 Metode og forskningsdesign

I dette kapittelet så vil jeg redegjøre for og begrunne hvorfor jeg har valgt å bruke en **kvalitativ metode** med en **fenomenologisk tilnærming** for å finne svar på problemstillingen.

Johannessen et al. (2010) mener at det er viktig å ha gjort seg opp en mening om hva eller hvem man skal studere og hvordan studien skal gjennomføres før man gyver løs på den. I denne oppgaven så har det vært ganske klart hva den skal handle om, men jeg må innrømme at jeg har famlet en del i min søken etter å konkretisere hvordan den skal løses.

3.1 Rammebetingelser

Før jeg kan velge en problemstilling og et forskningsdesign så må jeg kartlegge ramme-betingelsene for oppgaven.

En essensiell knapphetsfaktor for denne oppgaven er tiden. Det er tolv måneder til rådighet, og oppgaven skal skrives ved siden av full jobb og et hektisk familieliv. Jeg må derfor velge en fremgangsmåte som lar meg jobbe fleksibelt. Det vil for eksempel være vanskelig å gjøre observasjoner i en ekstern organisasjon over lengre tid, siden min egen arbeidsgiver er avhengig av at jeg er på jobb på dagtid.

En annen knapphetsfaktor er egen kompetanse. Det er viktig at jeg har tilstrekkelig med kunnskaper om problemområdet som skal belyses. Samtidig så er dette en anledning for å ta inn nye kunnskaper. Jeg kan velge å se problemstillingen i lys av mitt eget ståsted eller velge andre perspektiver. Dette er en balansekunst.

Hva oppgaven skal brukes til er også en viktig faktor for valg av problemstilling, metode og forskningsdesign. Gitt knapphets- og kompetansefaktorene nevnt over så tenker jeg at det er mest hensiktsmessig å bruke oppgaven for å tilegne meg læring. Læring om temaet, om prosessen og forhåpentligvis noe om meg selv. Dernest så tenker jeg at oppgaven kan gi organisasjonen kunnskaper om temaet og innsikt i forholdene på arbeidsplassen. Til slutt så håper jeg at oppgaven kan bli brukt av andre som utgangspunkt for videre forskning. Med utgangspunkt i at det er min egen læring som har høyeste prioritet så mener jeg at det blir mest riktig å la oppgaven lede meg i den retningen som jeg synes blir mest interessant.

3.2 Komparativ casestudie

Yin (2009) beskriver en casestudie som en empirisk undersøkelse av et avgrenset fenomen i dets naturlige omgivelser. Et fenomen kan være en bedrift, et program, en aktivitet, en hendelse, handlinger, samhandlinger – egentlig hva som helst. Formålet er å gå i dybden på det som studeres.

En casestudie kan inneholde undersøkelser av flere fenomener (Dul & Hak, 2007), men jo flere fenomener man tar for seg i en gitt tidsperiode jo mer overfladisk må nødvendigvis undersøkelsene bli.

I denne oppgaven så tar jeg for meg to programvareutviklingsprosjekter som gjennomføres hos en organisasjon i forsvarsindustrien. Det ene prosjektet bruker en tradisjonell prosjektmodell og det andre bruker en smidig prosjektmodell. Jeg går i dybden på disse prosjektene hver for seg, for så å forsøke å skape forståelse ved å drøfte og sammenlikne dem.

Yin (2009) innleder boken sin med å forklare hvordan en casestudie er både en lineær og en iterativ prosess. Lineær i den forstand at den går gjennom steg med planlegging, design, forberedelser, datainnsamling, analyse og deling – og iterativ i den forstand at man ved hvert steg må tilbake å revurdere tidligere avgjørelser. Hvert kapittel står sterkt alene, men er samtidig avhengig av hverandre. Dette forsøker jeg å leve opp til i gjennomføringen av denne oppgaven.

3.3 Kvalitativ metode

I en casestudie så kan man bruke både kvalitative og kvantitative metoder for å undersøke et fenomen. Kvalitative metoder legger mest vekt på den subjektive og følelsesmessige forståelsen av fenomenet, mens kvantitative metoder bruker målbare tall og indikatorer (Tjora, 2012).

Kvantitative data er hensiktsmessig å bruke for generalisering. Dette fungerer best når man undersøker mange varianter av fenomenet, hvor det er mulig å finne felles målbare variabler. Verdiene for disse kan så analyseres ved hjelp av statistiske metoder.

Kvalitative data er hensiktsmessig å bruke for å få dybdekunnskap og for å oppnå en forståelse av fenomenet. Dette er ikke egnet for å generalisere, men kan brukes for å formulere hypoteser, utvikle begreper og teorier – og for å foreta teoretiske generaliseringer.

Med de rammebetingelsene som er kartlagt, og med en casestudie som utgangspunkt, så mener jeg at det er mest hensiktsmessig å bruke kvalitative metoder for å samle inn data i denne oppgaven. Dette gjør jeg ved å utføre dybdeintervjuer av prosjektledere og prosjektdeltakere i to ulike prosjekter. Denne fremgangsmåten tror jeg også er mer interessant og gir mer læring enn å samle statistikk, selv om statistikk over bruk av prosjektmodeller på tvers av mange organisasjoner i forsvarsindustrien muligens ville vært mer nyttig å bruke for å svare på problemstillingen.

Det er flere elementer som spiller inn i valget av informanter. Mange forskere mener at antall informanter har betydning; Polkinghorne (1989) anbefaler mellom 5 og 25, Adler and Adler (2012) foretrekker mellom 12 og 60, Brannen (2012) mener at alt fra 1 til 260 er passe, mens Baker and Edwards (2012) er veldig konkret og mener at 15 må være best. Glaser and Strauss (1967) mener at man bør stoppe når informantene begynner å gjenta hverandre (data saturation, diminishing returns). Yin (2009) mener imidlertid at det ikke er så viktig hvor mange man intervjuer, men at det viktigste er å få ulike perspektiver på fenomenet.

Jeg tenker at det er viktig å få synspunkter fra ulike roller i prosjektene som jeg skal undersøke. Det er imidlertid en utfordring at prosjektene gjennomføres på to vidt forskjellige fysiske lokasjoner. Reiserestriksjoner (på grunn av den pågående pandemien) gjør at det ikke er mulig å gjennomføre fysiske intervjuer med alle deltakerne.

I denne oppgaven så prioriterer jeg å få gjennomført dybdeintervju med begge prosjektlederne, da de har størst innsikt i prosessene som brukes. Dernest så forsøker jeg å få gjennomført dybdeintervju med to prosjektdeltakere i hvert prosjekt for å få innsikt i hvordan de forstår og opplever arbeidshverdagen.

For å få et bredere og en mer holistisk forståelse for prosjektene som jeg undersøker så henter jeg også data fra prosessbeskrivelsene som prosjektene bruker.

3.4 Fenomenologisk tilnærming

Fenomenologi er ofte et begrep som forbindes med den tyske filosofen Edmund Husserl, som sent på attenhundretallet hadde som mål å skape en metode som lot fenomenene fremtre «rene», slik de egentlig er, uavhengig av subjektive fordømmelser og sosiale forutsetninger (Kvalsund, 2014). Et fenomenologisk vitenskapssyn tar utgangspunkt i den subjektive

opplevelsen og forsøker å forstå den dypere meningen som ligger bak (Thagaard, 2009). Albert Einstein (1936) sa at vitenskap ikke er noe annet enn en raffinering av de hverdagslige tankene. Det fenomenologiske vitenskapssynet til Einstein var helt grunnleggende for utarbeidelsen av mange av hans teorier, ikke minst i arbeidet med den generelle relativitets-teorien (Weyl, 1956). På mange måter så kan vi si at fenomenologi er en grunnstein for all vitenskap, da den vitenskapelige metode alltid starter med å se og oppdage.

I denne oppgaven så forsøker jeg å være trofast mot informantenes opplevelser av hvordan det er å arbeide i de to prosjektene. Det er klare strukturelle rammer rundt deres arbeidshverdag, med henholdsvis bruk av tradisjonelle og smidige prosjektmodeller, men jeg er mest interessert i hvordan de opplever sine egne virkeligheter innenfor disse rammene. For å få til dette så bruker jeg minst mulig dirigering i samtalene, og bruker aktiv lytting for å la samtalene gå dit det føles naturlig. Det er viktig at vi holder oss innenfor temaet, så en del spørsmål er allikevel klargjort på forhand.

3.5 Hermeneutisk perspektiv

Siden jeg selv har jobbet i programvareutviklingsprosjekter i mange år, og har inngående kjennskap og erfaring med problemstillingen i denne oppgaven, så er det ikke til å unngå at jeg tar med meg en forforståelse, personlige erfaringer og ideer inn i oppgaven. Dette kan virke selvmotsigende, siden fenomenologien nettopp skal legge egen forforståelse til side i møte med informanten – men samtidig så tror jeg at jeg ved å være bevisst dette utgangspunktet kan få en mer helhetlig forståelse av informantens sannhet.

Om det i det hele tatt er mulig å sette en forforståelse til side, eller om den faktisk er helt nødvendig for å forstå fenomenet, er forskere noe uenig om (Bengtsson, 2006). Jeg tenker at det viktigste som forsker er å være bevisst og tydelig på egen subjektivitet i så måte. Bengtsson (2006) mener at man i en hermeneutisk fortolkning må la forforståelsen bli konfrontert av fenomenet, og så eventuelt gå tilbake å endre den. Det er da det vil oppstå ny kunnskap.

3.6 Semistrukturert intervju

Et semistrukturert intervju tar utgangspunkt i en intervjuguide – men lar tema, spørsmål og rekkefølge variere fra intervju til intervju (Johannessen et al., 2010). Dette gir nok fleksibilitet

til at informanten kan fortelle fritt samtidig som jeg kan styre samtalen innenfor fastlagte rammer.

Arbeidet med teorien til oppgaven avdekker at det finnes mange oppfatninger og meninger om de ulike modellene og metodene som oppgaven tar for seg, blant annet hva det vil si å jobbe smidig. Her lar jeg informantene selv fortelle hva de legger i disse begrepene og hvordan de opplever arbeidet i kontekst av deres arbeidshverdag.

Én av utfordringene med denne fremgangsmåten er å stille åpne nok spørsmål til at informantene åpner seg om sine egne opplevelser samtidig som samtalene skal holde seg innenfor oppgavens tema. En annen utfordring er at kvaliteten på dataene som jeg samler inn fra informantene vil være varierende. Noen av informantene kjenner jeg veldig godt, mens andre har jeg aldri møtt. Noen av intervjuene gjennomføres digitalt, andre ansikt til ansikt.

Intervjuguiden tar utgangspunkt i emnene som er omtalt i teorikapittelet om hvordan det er å arbeide i tradisjonelle og smidige prosjekter. Hensikten er å bygge oppgaven rundt informantenes personlige erfaringer, så det er viktig å få de til å dele sine opplevelser. Intervjuene blir delt opp etter anbefaling av Tjora (2012) – med oppvarmingsspørsmål, refleksjonsspørsmål og avrundingsspørsmål. Disse fungerer som rammer for intervjuene. Videre så brukes aktiv lytting som oppmerksomhetsferdighet (Kvalsund, 2018). Aktiv lytting innebærer blant annet parafrasering, paraspråk, refleksjon av kroppsspråk og følelsesorienterte responser. Det er viktig å være klar over at man med oppmerksomhetsferdigheter styrer hva som skal være i fokus, og at aktiv lytting derfor påvirker hva som blir formidlet av informanten (Kvalsund, 2018).

3.7 Gjennomføring av intervju

Intervjuene gjennomføres i bedriftens lokaler med medlemmene i prosjektet som bruker tradisjonell prosjektmodell og over videosamband med medlemmene i prosjekt som bruker smidig prosjektmodell. Siden jeg benytter lydopptaker så er det viktig at dette skjer i en ubeskyttet sone, både fordi bedriftens sikkerhetsinstrukser bestemmer det og for at de jeg snakker med skal være bevisste på at de snakker med meg i en annen kontekst enn til vanlig.

I denne oppgaven så blir alle opplysninger om de jeg intervjuer anonymisert, men siden personopplysninger blir behandlet underveis i arbeidet så er det nødvendig å få godkjennelse

fra Norsk senter for forskningsdata (NSD) før intervjuene finner sted. Lydopptak regnes som personopplysninger. NSD er et statlig organ som er underlagt Kunnskapsdepartementet (KD) - hvis formål er å arkivere, tilrettelegge og formidle data til blant annet forskningsmiljøer (NSD, 2021). NSD har en egen seksjon som behandler søknader som gjelder personvern fra forskere og studenter.

Lydopptakene blir automatisk lagret i en tjeneste for lagring av sensitive forskningsdata, og de blir slettet når oppgaven ferdigstilles. Denne tjenesten, Tjenester for Sensitive Data (TSD), blir levert av Universitetet i Oslo (UiO). NTNU har inngått en databehandleravtale med UiO for behandling av personopplysninger.

I forkant av hvert intervju så får den som skal intervjues tilsendt et samtykkeskjema per e-post. Samtykkeskjemaet informerer om studien, hvordan data behandles og anonymiseres. Dette leses og godkjennes før intervjuet begynner. Godkjenningen gis ved å besvare e-posten.

3.8 Transkribering

Transkribering innebærer å overføre intervjuene fra tale til tekst (Kvale et al., 2009). Tjora (2012) mener at man må være systematisk og veldig nøye med å få med alle detaljer. Helst så bør man ha med både pauser, gjentakelser, kroppsspråk og stemninger. Dette kan imidlertid være et svært omfattende arbeid, i hvert fall hvis man har flere timelange intervjuer. NTNU tilbyr et digitalt verktøy, NVivo, for å automatisere denne jobben. Det finnes også profesjonelle tilbydere av transkriberingstjenester som er egnet til formålet. Tjora (2012) anbefaler at man bruker koder og kodegrupperinger for å systematisere og for å få oversikt over datamaterialet.

Jeg skjønner hva som er bakgrunnen for Tjora sine anbefalinger. Boka hans er basert på den såkalte *stegvis-deduktive induktive metoden* (Tjora, 2012), som veldig kort fortalt er en lineær metode for bruk i kvalitativ forskning hvor man starter med å frembringe empiri, for så å jobbe seg mot konsepter/modeller og til slutt teori. Underveis så har man tilbakeblikk for å validere tidligere steg. Dette minner om den iterative vannfallsmetoden som brukes i programvareutviklingsprosjekter (se kapittel 2.2.1).

Denne oppgaven følger ikke metoden til Tjora. Den bruker en mer smidig metode for gjennomføring, hvor teori og intervjuguide utvikles parallelt og i flere iterasjoner. Her er det behov for å legge et visst lag med teori i grunn før intervjuene kan gjennomføres.

Med en mer deduktiv tilnærming så kommer gjennomføring av intervjuer ganske sent i prosessen, og det er allerede blitt dannet en oppfattelse av hva som er viktig og hva som er mindre viktig å få ut av dem. Derfor velger jeg å skrive referat til hvert intervju i stedet for avskrift. Da skriver jeg svarene på spørsmål og tema med mine egne ord basert på samtalen som har funnet sted, og jeg plukker ut det som jeg synes er mest relevant innenfor temaene som er definert i intervjuguiden. Dersom dette forandrer seg underveis, eller hvis det oppstår behov for å plassere et utsagn i sin rette kontekst, så kan jeg alltids høre intervjuet på nytt.

Å bruke digitale verktøy eller profesjonelle transkriberingstjenester er problematisk av hensyn til personvernet. Intervjuene skal ikke lagres på privat datamaskin, og de skal i hvert fall ikke sendes til tredjeparter som NTNU ikke har databehandlingsavtaler med.

Et referat er ikke veldig godt egnet for dokumentasjon og det er sterkt farget av referentens tolkninger av det som blir sagt. Dette gjelder også til en viss grad for ordrette transkriberinger. Det er umulig å skrive ned intonasjoner, pauser, kroppsspråk og stemmebruk uten å gjøre egne tolkninger. I tillegg så mister man alltid det sosiale samspillet som skjer når to mennesker har en samtale (Kvale et al., 2009). I denne oppgaven så er jeg bevisst min egen påvirkning, som starter allerede når jeg formulerer problemstillingen. Det er jeg som velger hvilke spørsmål som jeg skal stille og hvordan de skal stilles. Jeg velger hvilke teorier er som er relevante og hvem jeg skal høre på. Ikke minst så er det jeg som drøfter meg frem til et svar på problemstillingen. Det er altså ikke gjennom transkriberingen at objektiviteten til en kvalitativ studie sikres.

3.9 Forskningskvalitet

Begrepene reliabilitet og validitet brukes ofte når man skal *måle kvaliteten* til forskningen som utføres (Tjora, 2012). Reliabilitet brukes for å si noe om hvor pålitelige dataene er og hvor nøyaktig innsamlingen av den er. Validitet brukes for å angi i hvilken grad dataene er gyldige og relevante for problemstillingen. I utgangspunktet så er begrepene forbundet med kvantitative studier, men de brukes også i kvalitative studier (Yin, 2009).

I denne oppgaven så bruker jeg begrepene pålitelighet og gyldighet i stedet for reliabilitet og validitet. Dette er fordi jeg mener at reliabilitet og validitet ikke er dekkende. For en kvalitativ studie så er det viktigere å vurdere den totale troverdigheten enn å utelukkende vurdere datamaterialet.

3.9.1 Pålitelighet

Pålitelighet handler om konsistens og nøyaktighet, og siden kvalitative studier ofte er vanskelige å gjenskape så er det desto viktigere å beskrive fremgangsmåten som er brukt. Thagaard (2009) mener at man må redegjøre for hvordan dataene er samlet inn og bearbeidet.

Gjennomsiktighet og grundig redegjørelse av gjennomføringen gjør at studien fremstår som troverdig. Oppgaven beskriver hvorfor jeg velger å gjøre en casestudie med bruk av semistrukturerte intervjuer, hvordan en fenomenologisk tilnærming med et hermeneutisk perspektiv kan være hensiktsmessig for akkurat denne oppgaven og hvordan intervjudataene blir transkribert og lagret. Troverdigheten svekkes imidlertid noe ved at jeg ikke går nærmere inn på egne erfaringer eller utdyper hvilke forhold jeg har til informantene. Dette er en vurdering som er gjort opp mot hensynet til casebedriftens og informantenes anonymitet.

3.9.2 Gyldighet

Gyldighet handler om hvorvidt fremgangsmåten og funnene reflekterer formålet med studien og om de representerer virkeligheten (Johannessen et al., 2010). For at svarene som jeg gir på forskningsspørsmålene skal være troverdige så må jeg for det første argumentere for at fremgangsmåten som jeg har valgt er hensiktsmessig for problemstillingen, og for det andre så må jeg være åpen om hvordan jeg tolker og drøfter resultatene underveis. Yin (2009) mener at det er viktig å etablere en beviskjede, samt å sørge for at man ikke utelukker viktige alternativer når man drøfter mulige kausale sammenhenger.

Flere av informantene i denne oppgaven gjengir noe av den samme informasjonen. Dette er med på å styrke gyldigheten. Samtidig så er hensikten med intervjuene å få ulike perspektiver og forståelser for temaet, ikke at informantene nødvendigvis skal validere hverandres informasjon. Spørreundersøkelser med mange informanter vil nok være mer hensiktsmessig å bruke dersom gyldigheten er det viktigste, men da på bekostning av den dypere forståelsen.

Å sammenlikne datagrunnlaget fra denne oppgaven med datagrunnlag fra andre tilsvarende oppgaver vil også kunne styrke gyldigheten. Med tanke på at denne oppgaven belyser temaet gjennom et ganske smalt vindu så tror jeg imidlertid at gyldigheten bare blir styrket marginalt. En litteraturstudie vil nok få større utbytte av å sammenlikne data fra flere oppgaver.

Alle informantene har fått tilsendt utkast til oppgaven slik at de kan bekrefte eller avkrefte fremstillingen som er gjort. Tilbakemeldingene har imidlertid vært beskjedne. Dette skyldes nok oppgavens omfang, og i retrospekt så ser jeg at det kan være mer hensiktsmessig å sende nøkkelinformasjon for gjennomlesing enn en hel oppgave.

Det er også relevant å vurdere hvorvidt funnene i oppgaven er gyldige i andre sammenhenger. For kvantitative studier så kalles dette ekstern validitet eller generalisering. Yin (2009) mener at kvalitative casestudier er lite egnet for generalisering, siden utvalget er så marginalt. Thagaard (2009) mener at gjennomsiktighet og detaljerte beskrivelser av fremgangsmåten vil kunne gi slike oppgaver gyldighet også i andre sammenhenger. I denne oppgaven så er jeg bevisst på at temaet belyses et lite utvalg og at det er usikkert om andre forskere vil komme frem til de samme konklusjonene gitt de samme premissene. Samtidig så vil et altfor kritisk blikk på generalisering kunne hindre utvikling av ny kunnskap, og så lenge fremgangsmåten er tilstrekkelig redegjort for så mener jeg at funnene kan ha en viss overføringsverdi.

3.10 Etiske prinsipper

Denne oppgaven følger de fire etiske grunnprinsippene til Diener og Crandall (1978):

Ikke gjør skade er et prinsipp som handler om å redusere risikoen for at deltakere eller andre med tilknytning til forskningen blir skadelidende. En informant bør ikke oppleve intervjusituasjonen som ubehagelig eller stressende, føle seg tvunget til å svare på spørsmål eller sette seg selv og bedriften i et dårlig lys. I denne oppgaven så reduserer jeg skadepotensialet ved å anonymisere bedriften og informantene, gi informantene god informasjon på forhand, la informantene komme med tilbakemeldinger og gjøre det mulig for informantene å trekke seg. I tillegg så er jeg svært forsiktig med håndteringen av lydopptakene og hvilken informasjon som blir henter ut fra disse. Ved å få oppgaven innmeldt og godkjent av NSD så sikrer jeg at personopplysningene blir behandlet i tråd med lovverket.

Informert samtykke er et prinsipp som handler om at potensielle deltakere må få god nok informasjon til at de kan gjøre veloverveide valg. Det er viktig at informasjonen er ærlig og redelig, og at det ikke senere viser seg at forskeren i tillegg har drevet med skjult observasjon. I denne oppgaven så får informantene god informasjon før de velger å delta og i forkant av intervjuene. Oppgaven benytter ikke observasjon.

Krenkelse av privatlivet er et prinsipp som handler om i hvilken grad det er akseptabelt å gå inn i deltakernes private sfære. Selv om deltakerne gir tillatelse til å la seg intervjue så gir ikke dette anledning til å spørre om hva som helst. Det er imidlertid ikke alltid opplagt for forskeren hva som kan oppleves som for privat for den enkelte, så det er viktig å informere informanten om at man ikke må svare på alle spørsmålene og at man har mulighet til å trekke seg fra studien. Skjult observasjon kan også bli ansett som krenkelse av privatlivet. Temaet for denne oppgaven utforsker ikke informantenes privatliv, og selv om den går inn på hvordan informantene opplever sine arbeidshverdager så er det lite sannsynlig at spørsmålene vil oppleves som krenkende. Dersom informantene allikevel ønsker å avbryte, så er jeg forberedt på det.

Bedrageri er et prinsipp som handler om hvorvidt forskningen blir fremstilt som noe annet enn den egentlig er. Dette gjelder både ovenfor deltakerne i studien og ovenfor omgivelsene. I denne oppgaven så streber jeg etter å være oppriktig ovenfor de involverte, spørsmålene er åpne og tydelige, informasjonen blir i minst mulig grad plukket ut av sin opprinnelige kontekst og fremgangsmåten er redegjort for så godt det lar seg gjøre.

4 Resultater

I denne oppgaven så tar jeg for meg to programvareutviklingsprosjekter hos en organisasjon som leverer kritisk infrastruktur og programvare til forsvarssektoren. Det ene prosjektet bruker en tradisjonell gjennomføringsmodell (heretter kalt det tradisjonelle prosjektet) og det andre bruker en smidig gjennomføringsmodell (heretter kalt det smidige prosjektet). Begge prosjektene har Forsvarsmateriell (FMA) som kunde.

Funnene som legges frem i dette kapittelet er utledet av intervjuer og prosessbeskrivelser. Underveis i gjennomføringen så har det blitt klart at medlemmene i det smidige prosjektet ikke er tilgjengelig for intervju. Dette henger sammen med at prosjektet er avsluttet og at de fleste medlemmer ikke lenger jobber i organisasjonen. Derfor er det kun gjennomført dybdeintervju av prosjektlederen fra dette prosjektet. Fra det tradisjonelle prosjektet så er det gjennomført dybdeintervju av prosjektlederen og to prosjektmedlemmer.

Det tradisjonelle prosjektet er et vedlikeholdsprosjekt med en varighet på ett år. Prosjektlederen har lang erfaring som prosjektleder, men er ny i denne organisasjonen. Prosjektmedlemmene og organisasjonen rundt prosjektet er for øvrig veletablerte og har jobbet sammen og på samme måte i mange år.

Det smidige prosjektet er et utviklingsprosjekt med en varighet på omtrent fem år. Prosjektlederen for dette prosjektet har også lang erfaring innen prosjektlederfaget, både i denne og i andre organisasjoner. Siden prosjektet tar frem et nytt produkt så er prosjektet satt sammen og organisert for anledningen.

Organisasjonen er organisert som en sterk matriseorganisasjon, hvor alle aktiviteter skjer gjennom prosjekter (tilbudsprosjekter, utviklingsprosjekter, vedlikeholdsprosjekter, service- og supportprosjekter, leveranseprosjekter) og hvor prosjektene henter ressurser fra ulike ressurs-grupper. Det er ingen driftsorganisasjon for produktene – men basisorganisasjonen tar seg av innkjøp, kvalitetssikring, arkivering, IT-støtte, økonomi, HR, sikkerhet og linjeadministrasjon.

Figurene som illustrerer prosjektmodellene og programvareutviklingsmetodene i dette kapittelet er basert på intervjuer og prosessbeskrivelser, og de er derfor farget av egne tolkninger og forståelser.

4.1 Prosjektmodeller og programvareutviklingsmetoder

Begge prosjektlederne er bevisste på forskjellen mellom prosjektmodell og programvare-utviklingsmetode, og begge er tydelige på at førstnevnte i stor grad blir bestemt av kontrakts-formen. I denne bransjen så er det fastpriskontrakter som gjelder, også for disse prosjektene. Dette innebærer at omfang, pris og leveransemilepæler er fastsatt når kontrakten blir inngått.

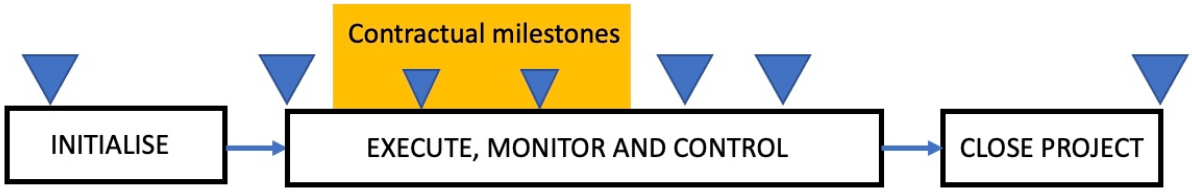
Prosjektmedlemmene har et bevisst forhold til programvareutviklingsmetoden, men ikke til prosjektmodellen. I det tradisjonelle prosjektet så har programvareutviklingsmetoden vært fast og urørt i mange år, mens den i det smidige prosjektet har blitt utviklet og tilpasset underveis i prosjektet.

Prosessrammeverket som brukes av organisasjonen definerer ikke en detaljert prosjektmodell, men det er tydelig på at prosjektet må ta hensyn til flere prosesser med avhengigheter til hverandre. Tilbud, løsning, programvareutvikling, maskinvareutvikling, innkjøp, leveranse, tjenester og produkt – alle har sine egne livssykluser og prosesser. Disse påvirker hverandre i større eller mindre grad avhengig av om prosjektet for eksempel lager programvare og maskinvare, bare programvare eller om det leverer et produkt som allerede har blitt laget. I denne oppgaven så ser jeg på prosjekter som kun driver med programvareutvikling.

Figur 12 – Casebedriftens prosjektmodell (bearbeidet fra Anonym casebedrift, 2021)

Prosjektmodellen likner på en lineær modell, men det kan være verdt å merke seg at hele prosjektet gjennomføres i én fase. Det er ikke definert en egen planleggingsfase, noe som åpner for at prosjektet kan gjennomføres på flere måter.

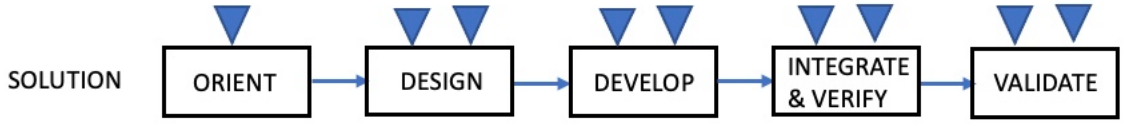
Modellen har en del faste milepæler, og noen milepæler som er basert på kontrakten.



Figur 13 – Casebedriftens prosjektmodell med milepæler (bearbeidet fra Anonym casebedrift, 2021)

Milepælene består ofte av ulike sjekklister som blir gjennomgått for å sikre at alt er tenkt på før prosjektet starter, i gjennomføringen av prosjektet og ved avslutting av prosjektet. Figuren viser at kontrakten legger føringer for gjennomføringen med milepæler som prosjektet må forholde seg til. Disse er det gjerne også knyttet betalinger til.

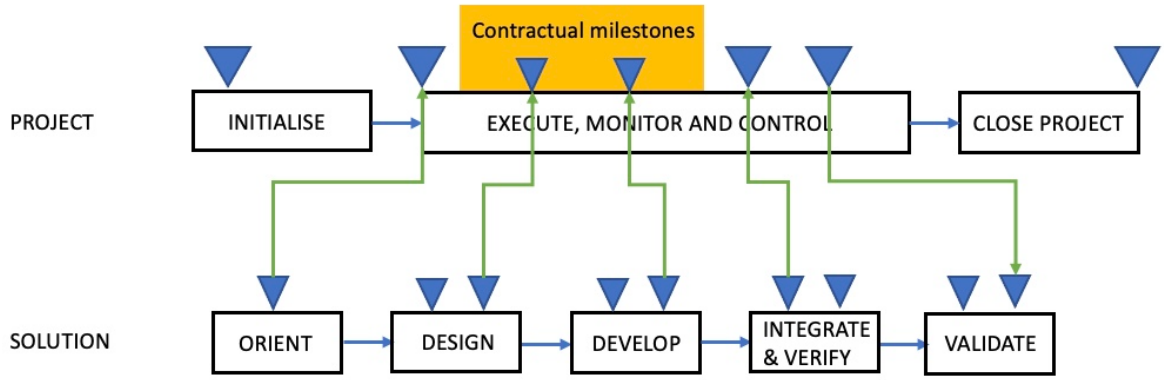
Som nevnt så må prosjektet forholde seg til flere prosesser – deriblant prosessen for *løsningen*, som rammer inn alt som prosjektet skal produsere og levere. Dette kan inkludere programvare, dokumentasjon, opplæring, maskinvare og annet utstyr.



Figur 14 – Casebedriftens tradisjonelle løsningsmodell (bearbeidet fra Anonym casebedrift, 2021)

På løsningsnivå så definerer prosessrammeverket flere modeller – inkludert vannfallsmodellen, v-modellen, inkrementell modell og iterativ modell. Figuren over viser vannfallsmodellen. Denne modellen definerer også et sett med milepæler som verifiserer at alt som skal gjøres i én fase blir gjort før den neste starter.

Modellen inneholder også en oppfølgingsfase som går parallelt med de andre fasene. Den er ikke tegnet inn i figuren. Oppfølging skjer gjennom hele prosjektet.



Figur 15 – Casebedriftens tradisjonelle løsningsmodell med milepæler (bearbeidet fra Anonym casebedrift, 2021)

Flere av milepælene på løsningsnivå er innspill til milepæler på prosjektnivå.

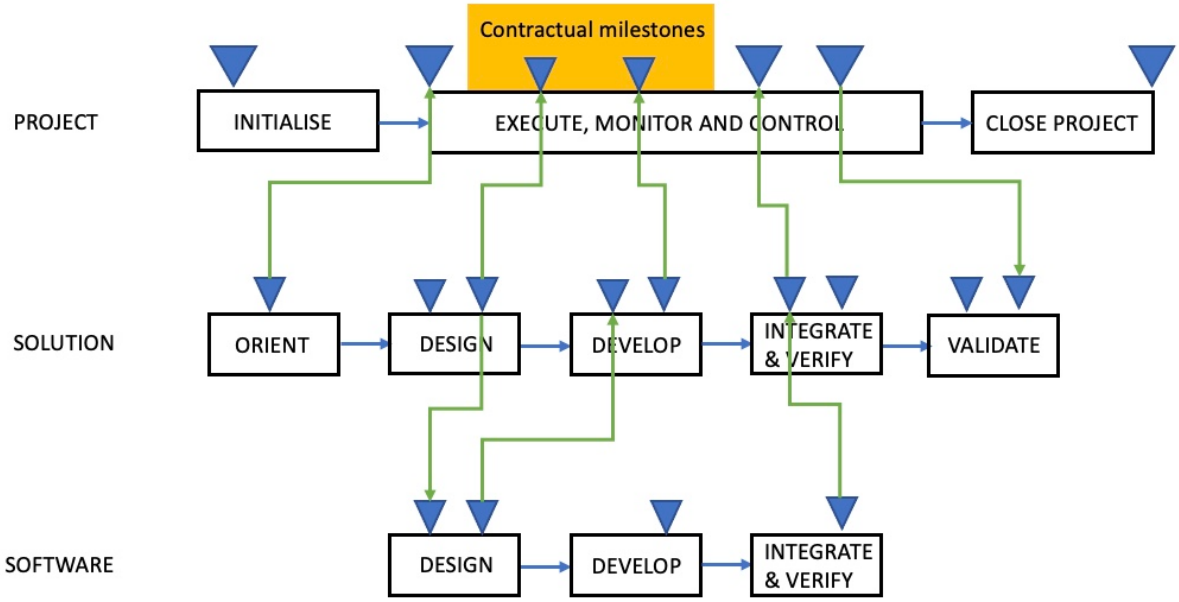
Prosessrammeverket gir veiledning til hvilke modeller som bør brukes på løsningsnivå. Kriterier for kravmodenhet, kompleksitet, produktutvikling, tidspress, tekniske risikoer, behov for tilbakemelding fra brukere, bruk av ny teknologi og grad av arbeidsbelastning lager en

matrise med anbefalt, sterkt anbefalt eller ikke anbefalt for hver modell. Vannfallsmetoden (og v-metoden) er «ikke anbefalt» for alle kriterier i matrisen bortsett fra kriteriet for produkt-utvikling, hvor det er notert at metoden er anbefalt dersom det kun skal utføres mindre vedlikeholdsarbeid på produktet. Iterativ gjennomføring er anbefalt for de fleste kriteriene i matrisen.

Programvareutviklingsmetodene som beskrives av prosessrammeverket kalles for smidig og ikke-smidig.

Det tradisjonelle prosjektet

Det tradisjonelle prosjektet bruker vannfallsmodell på løsningsnivå og ikke-smidig metode på programvarenivå.



Figur 16 – Casebedriftens tradisjonelle programvareutviklingsmetode (bearbeidet fra Anonym casebedrift, 2021)

Den ikke-smidige programvareutviklingsmetoden går hånd-i-hånd med vannfallsmetoden på løsningsnivå. Figuren viser at avsluttende milepæl i designfasen på løsningsnivå er innspill til starten på designfasen på programvarenivå. Likeledes så er avsluttende milepæl i designfasen på programvarenivå innspill til starten på utviklingsfasen på løsningsnivå.

I designfasen så brytes overordnede krav og design ned i arbeidspakker og tilordnes en arbeidspakkeleder. Videre i utviklingsfasen så går hver arbeidspakke gjennom sin egen prosess med kravutredning, detaljert design, estimering og godkjenning før arbeidet starter. Avhengig

av hvor mye arbeid som skal gjøres i en arbeidspakke så blir den tildelt ressurser i form av programvareutviklere, utviklingstimer og tidsfrister. Arbeidspakkelederen har ofte stor systemforståelse og er i stand til å ta avgjørelser om løsningen. I tilfeller hvor arbeidspakkelederen ikke kan finne en god løsning innenfor de kravene som er stilt i kontrakten så må dette flagges videre til produktleder og prosjektleder. Videre så kan det bli en byråkratisk behandling (og forhandling) om endringer til kontrakten for å gjøre tilpasninger for den nye løsningen.

Hver arbeidspakke gjennomfører sine egne integrasjonstester før de avsluttes, men det er i integrasjons- og verifikasjonsfasen at hele systemet verifiseres. Her blir det gjennomført en ganske stor og omfattende systemtest. Feil som oppdages i denne fasen kan føre til kostbare feilrettinger. Dersom mange eller alvorlige feil blir rettet i denne fasen så kan det bli behov for en ny runde med systemtesting.

Både prosjektlederen og prosjektdeltakerne gir inntrykk av at det i praksis er liten bevissthet i prosjektet om de ulike milepælene på løsnings- og programvarenivå. Sjekklistene blir sjeldent fulgt opp, da disse oppfattes som i overkant kompliserte. Noen av sjekklistene på programvarenivå danner grunnlaget for å svare ut sjekklistene på løsningsnivå, og siden prosjektet kun leverer programvare så blir disse ofte slått sammen eller blir sett på som irrelevante. Noen milepæler blir erstattet av etablerte arbeidsrutiner. Disse inngår i avdelingens *tailoring*, som uttrykkes i form av et godt forankret prosessdokument med prosesser for dokumentoppdatering, versjonshåndtering, kodeinnsjekk, arbeidspakker, feilrapporter og hvem som har ansvar for de ulike delene av systemet.

Prosjektene i organisasjonen blir stadig gjennomgått av eksterne evaluatorer for å sikre at det foreligger god nok kontroll og sporing av arbeidet som blir gjort, og disse gjennomgangene blir som regel godkjent uten nevneverdige mangler.

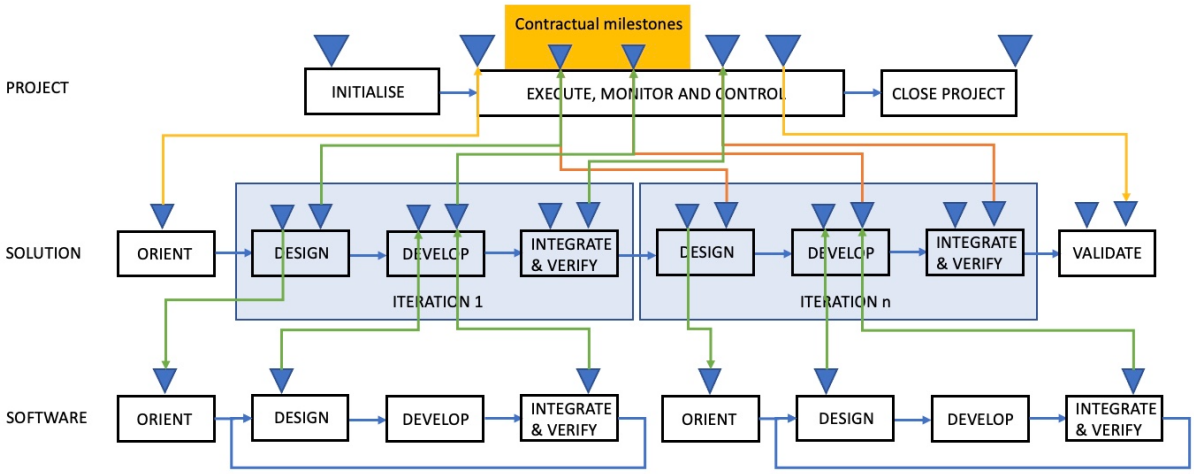
Prosjektlederen og de intervjuede prosjektdeltakerne i det tradisjonelle prosjektet mener at arbeidsprosessene er dypt forankret i organisasjonen, både strukturelt og kulturelt, og at det derfor er svært vanskelig for nyansatte å komme med innspill til forbedringer. Dette blir forsterket av at organisasjonen har stort fokus på månedlige fiscalrapporteringer og at den i stor grad virker å være styrt av enkeltindividers ønsker og behov.

At svært mye kompetanse ligger på få hender blir også trukket frem som den største risikoen for prosjektet. Dette gjelder også på kundesiden, og prosjektlederen mener at det er vanskelig

å få tak på de uformelle prosessene som foregår mellom partene. Selv om disse nærmer seg pensjonsalder så virker det ikke være en prioritet å lære opp arvtakere.

Prosjektlederen opplever å i aller høyeste grad bli målt på økonomi. Samtidig så sier de intervjuede prosjektdeltakerne at det er kvaliteten som i praksis er den viktigste styrings-parameteren. Ifølge en av prosjektdeltakerne så blir arbeid ofte hengende i ulike godkjennings-prosesser i ukesvis fordi enkelte av de som er sentrale med godkjenning av kodeinnsjekker og dokumentoppdateringer er altfor pirkete. Prosjektdeltakeren forteller at forslag for å gjøre disse mer effektive har blitt avfeid som «cowboy-tendenser», og at det virker som om de som er sentrale i disse prosessene kjører et parallelt løp ved siden av prosjektet og ikke tar inn over seg at prosjektet kan være presset på tid.

Det smidige prosjektet



Figur 17 – Casebedriftens smidige programvareutviklingsmetode (bearbeidet fra Anonym casebedrift, 2021)

Modellen for det smidige prosjektet ser i første øyekast komplekst ut. Her brukes det en iterativ modell på løsningsnivå, hvor det planlegges og gjennomføres flere sekvensielle leveranser underveis i prosjektet. Hver av disse iterasjonene bruker en smidig programvare-utviklingsmetode.

Det er ikke alle milepælene på løsningsnivå som er relevante for hver iterasjon. Prosjektlederen forteller at noen av sjekkpunktene bare blir gjennomgått ved første og siste iterasjon, og noen er relevante kun dersom det blir forhandlet inn mer funksjonalitet i kontrakten.

Selv om prosjektet forholder seg til en fastpriskontrakt, hvor krav og overordnet design er bestemt på forhand, så har den spesifiserte leveransemilepæler underveis. Dette er formelle

kontrollpunkter hvor kunden får sjekket ut at prosjektet går som planlagt og at produktet som utvikles lever opp til forventningene. Samtidig så åpner det for en iterativ gjennomføring. Prosjektlederen mener at det er forbundet høy risiko ved å planlegge langvarige prosjekter i detalj på forhånd – og at det i realiteten også er en umulig oppgave. For det første så vil det alltid dukke opp uforutsette ting underveis som fører til endringer. Og for det andre så må man ofte først utvikle deler av systemet for å få nok kunnskaper til å detaljere andre deler av det. Med flere leveranser underveis så kan prosjektet ta for seg én leveranse om gangen.

Kontrakten har en fast del samt et sett med opsjoner. Dette kommer av at Forsvaret i sine anskaffelser har en kort og en lang horisont. Den faste delen dekker eksisterende behov, mens opsjonene dekker mulige fremtidige behov. Prosjektlederen mener at dette gir en viss fleksibilitet, men fortsatt innenfor gitte spesifikasjoner og kostnadsrammer.

Prosjektet bruker Scrum som programvareutviklingsmetode. Iterasjonene på dette nivået gjennomføres langt hyppigere enn de på løsningsnivå, noe som betyr at det ikke er en direkte sammenheng mellom leveransene på programvarenivå og leveransene på løsningsnivå. Det er kun leveransene på løsningsnivå som leveres til kunden.

Prosjektlederen mener at den største utfordringen for dette prosjektet er bruken av ny teknologi og at produktet som utvikles er nytt og veldig komplekst. Produkter kan heller ikke settes i operativ drift før de er ferdig utviklet og godkjent for bruk, og det kan være utfordrende å simulere komplekse operative miljø.

De ulike interessentene i prosjektet prioriterer ulike suksesskriterier. Prosjektlederen sier at det viktigste kriteriet for Forsvaret i dette prosjektet er tiden, da produktet som leveres skal erstatte et produkt som holder på å bli utfaset. Samtidig så er kontrakten tydelig på at prosjektet skal imøtekomme alle kravene. Prosjektlederen er mest opptatt av at formålet med prosjektet skal bli oppfylt og at brukerne skal bli fornøyde. Da er det viktig med arenaer hvor man får mulighet til å justere forventningene underveis, for eksempel ved å holde demonstrasjoner og la brukerne teste produktet. Dette bygger tillit og kan være med på å gi større aksept dersom prosjektet mot formodning skulle bli forsinket. For organisasjonen så er resultatet viktig, og det forventes at prosjektlederen når resultatmålene samtidig som forsinkelser og endring i omfang blir håndtert.

4.2 Teamarbeid

Begge prosjektene gjennomføres med samlokaliserte prosjektdeltakere, hvor alle fagdisipliner sitter i det samme lokalet. Begge prosjektene er også i utgangspunktet tradisjonelt organisert – med prosjektleder, systemarkitekter, programvareutviklere, testere og støttefunksjoner. Begge prosjektlederne er tydelige på at samlokalisering er viktig. Det fører til en betydelig effektivisering av kommunikasjonen og muliggjør hyppige avklaringer. Samtidig så er det lite tilrettelegging for hjemmekontor for programvareutviklere i organisasjonen.

Det tradisjonelle prosjektet

Prosjektlederen sier at det tradisjonelle prosjektet består av 12-14 utviklere som er delt opp i flere små team. Disse jobber på ulike deler av det samme produktet. Det er kravene i kontrakten og arbeidet definert i de påfølgende arbeidspakkene som bestemmer teamsammensetningene. For eksempel så vil en arbeidspakke som består av arbeid med databasen kreve utviklere med kompetanse på databasen. Likeledes så vil en arbeidspakke med arbeid på brukergrensesnittet kreve utviklere med kompetanse på brukergrensesnittet. Arbeidspakkene har gjerne hovedvekt av oppgaver rettet mot ett kompetanseområde, med noe arbeid på ett eller flere andre kompetanseområder.

En av prosjektdeltakerne forteller at den som er arbeidspakkeansvarlig ofte har systemansvar for den berørte delen av systemet, driver med oppfølging av fremdriften, sørger for sysselsetting av teammedlemmene, tar de tekniske avgjørelsene ved løsningen og sørger for at arbeidet til slutt blir godkjent gjennom integrasjonstesting.

Både prosjektlederen og de intervjuede prosjektdeltakerne i det tradisjonelle prosjektet trekker litt på svaret når jeg spør om teamene er effektive.

Prosjektlederen mener at enkelte personer kan være effektive, og at enkelte team kan være effektive dersom arbeidspakkelederen driver tett oppfølging av arbeidet. Ineffektiviteten begrunnes med at utviklerne kan bli sittende og vente på andre i teamet eller på å få arbeidsoppgaver av teamlederen. Samtidig så kan teamlederen bli passiv og vente på at arbeid skal bli gjort av de i teamet eller vente på arbeid fra andre team som man er avhengig av.

Den ene prosjektdeltakeren opplever at den ene arbeidspakkelederen er autoritær og detaljstyrer arbeidet. Samtidig så er den samme arbeidspakkelederen veldig pirkete når kodeinnsjekker skal godkjennes. Dette oppleves som ineffektivt og kostnadsdrivende.

Den andre prosjektdeltakeren mener at teamarbeidet kan være effektivt av og til, når mellomstore team (5 til 8 teammedlemmer) jobber med samme arbeidspakke over en lengre periode. Da kan det utvikle seg et felles ansvar og eierskap til arbeidet. Dette skjer, men ikke veldig ofte, og det har ikke skjedd i dette prosjektet. Som regel så er det mindre team hvor en ansvarlig teamleder gjør alt designarbeidet før han deler ut oppgaver til utviklerne. Da har ikke disse noe forhold til helheten og om arbeidspakken bruker for mange timer eller blir forsinket.

Det smidige prosjektet

Prosjektlederen forteller at det smidige prosjektet består av 11 utviklere, de fleste innleide konsulenter. Disse er satt sammen i flere utviklingsteam. Det var i utgangspunktet teamleder og ekstern systemarkitekt som tok de fleste avgjørelsene for et team. Dette gjorde at teamet fikk et veldig smalt perspektiv og kun så arbeidet innenfor hver iterasjon. Prosjektet valgte derfor å plassere systemarkitekten inne i teamet slik at det overordnede perspektivet ble mer synlig. Siden prosjektet bruker Scrum som utviklingsmetode så ble denne tildelt rollen som Product Owner. Når arbeidet i en iterasjon, eller Sprint, ble ferdigstilt så ble det levert videre til testerne. Disse hadde ansvaret med å skrive og gjennomføre integrasjonstestene, og de fikk således erfaring inn mot systemtesten som skulle foregå mot slutten av prosjektet. Etter hvert så ble det tydelig at disse også måtte inn i teamene. Testerne har et brukerperspektiv som utviklerne mangler, og de kan stille kritiske spørsmål underveis i gjennomføringen. Dessuten får testerne tidlig kompetanse på systemet som skal testes.

Teamene har etter hvert blitt mer kryssfunksjonelle og selvdrevne. Product Owner jobber metodisk med Product Backlog og forbereder Sprint Planning. Teamdeltakerne bidrar med å skissere forslag til løsninger som kvalitetssikres gjennom Product Owner og produktsjefen. Dersom det ikke er mulig å finne gode løsninger innenfor rammene som kravene stiller, eller at det blir funnet bedre løsninger som krever at kravene blir endret, så blir det en byråkratisk behandling (og forhandling) om endringer. Dette synes prosjektlederen er veldig ryddig.

Prosjektlederen er tydelig på at prosjektarbeidet ikke har vært særlig effektivt. Det tok lang tid å utvikle programvareutviklingsmetoden til noe som fungerte bra. I tillegg så var det nødvendig å bygge kompetanse på ny teknologi og skape forståelse for et nytt produkt. Det ble ansatt

mange konsulenter til prosjektet. Disse hadde egne erfaringer med smidige metoder som utfordret måten prosjektet jobbet på. Prosjektarbeidet ble effektivt når metode og teknologi hadde satt seg, men det tok lang tid å komme dit. Mot slutten så ble prosjektet nedskalert og kun de faste ansatte ble igjen. Disse jobber nå på andre prosjekter. Prosjektlederen synes at det er synd at så mye av kompetansen som ble bygget opp i løpet av disse årene er borte.

4.3 Samarbeid og brukerinvolvering

Det tradisjonelle prosjektet

Før kontrakten ble signert og prosjektet satt i gang så møttes enkelte representanter fra bedriften og kunden for å diskutere hva som skulle være med. Dette er et vedlikeholdsprosjekt som tar inn forbedringer til et eksisterende produkt som er i drift, så flere av innspillene er basert på tilbakemeldinger fra brukere.

Prosjektlederen forteller at prosjektet har kvartalsvise fremdriftsmøter, hvor representantene fra organisasjon møter kunden for å diskutere eventuelle endringer, gjøre avrop på opsjoner, melde inn operative behov og avtale assistanse ved øvelser og on-site support. I tillegg så blir status for fremdriften lagt frem.

Ingen av de intervjuede prosjektdeltakerne i det tradisjonelle prosjektet har noen kontakt med kunden eller brukerne. All forståelse av det som skal gjøres går gjennom de faste representantene. Siden ingen andre blir involvert i disse prosessene så er det også disse som gjør alle overordnede designvalg. Selv om disse er svært dyktige så oppfattes dette som en svakhet i arbeidsprosessen. Den ene prosjektdeltakeren forteller om designvalg som har fått dyrekjøpte konsekvenser, som etter mye arbeid har blitt gjort om på grunn av at viktig innsikt har kommet frem sent.

Det smidige prosjektet

Før det smidige prosjektet ble startet så ble det gjennomført et forprosjekt. Kunden beskrev all funksjonalitet og alle krav, men med vår organisasjon i en rådgiverrolle. På denne måten så ble begge parter omforent med hva som skulle lages før prosjektet startet og alle fikk en god forståelse av behovet som skulle dekkes.

Prosjektlederen hadde innledningsvis en lang diskusjon med kunden for å avklare hvordan brukerne best kunne involveres i prosessen. Brukerinvolvering krever at kunden forplikter seg med tid og ressurser, forteller prosjektlederen, og tilbakemeldingene må komme tidsnok til at kursen kan endres. I dette prosjektet så har de samlinger hver tredje måned, hvor arbeideidet blir vist frem til en fast kjernegruppe på tolv personer. Disse representerer ikke alle brukergruppene, men en god del av dem. En problemstilling som man må være bevisst på i så måte er at det kan oppstå konflikter mellom de ønskene som brukerne kommer med og det som er bestemt gjennomført av kunden. Her er det viktig å være tydelig på at tilbakemeldinger er verdifulle, men at ønsker ikke nødvendigvis blir tatt inn i produktet. Prosjektlederen forteller at prosjektet har en viss handlefrihet innenfor rammene av kontrakten, men at det tross alt er en fastpriskontrakt.

Med Scrum som gjennomføringsmetodikk så er det Product Owner sitt ansvar å ivareta de ulike interessentenes behov. De øvrige prosjektdeltakerne har derfor ingen samhandling med kunder eller brukere.

4.4 Læring

Hva som ligger i ordet læringskultur varierer mellom de intervjuede, men det virker å være enighet på tvers av prosjektene om at en læringskultur må innebære læring ut over den man opplever i prosjektgjennomføringen – det må komme fra visjonene til organisasjonen og reflekteres i porteføljestyring, linjeledelse og oppfølging av medarbeidere. I så måte så er det enighet om at bedriften ikke har en etablert læringskultur.

I begge prosjektene så er det imidlertid stor takhøyde for å gjøre feil og for å tørre å si at man har gjort feil. Prosjektene består av høyt utdannede, intelligente og profesjonelle folk som stort sett behandler hverandre som likeverdige.

Begge prosjektene benytter også ny teknologi, noe som blir trukket frem som positivt for læring og utvikling.

Det tradisjonelle prosjektet

Det tradisjonelle prosjektet har retrospective som en obligatorisk oppgave i alle arbeidspakker. Dette innebærer at de som har vært deltakere på en arbeidspakke setter seg sammen og diskuterer seg frem til hva som har vært bra, hva som har vært mindre bra og mulige forslag til

forbedringer. Det er imidlertid ingen som tar dette videre, og både prosjektlederen og de intervjuede prosjektdeltakerne mener at det bare er noe man må gjennom. I beste fall så blir noe av det tatt med videre i underbevisstheten til neste prosjekt eller arbeidspakke.

Noen av teamene inkluderer seremonier som kan fremme læring, som daglige stand-up-møter og kanban-tavler. Førstnevnte er korte møter hvor alle på teamet forteller de andre hva de jobber med og om de har noen utfordringer. Sistnevnte er en tavle som viser hvilke oppgaver de ulike teammedlemmene jobber med. Begge disse seremoniene er kjente fra smidig programvare-utvikling og er med på å skape felles forståelse og ansvar i teamet. Prosjektlederen ønsker at alle teamene bruker stand-up-møter og kanban-tavler, men sier at det ikke er tradisjon for dette.

Både prosjektlederen og de intervjuede prosjektdeltakerne trekker frem medarbeidersamtalen som en arena som burde lagt til rette for læring. Samtalen handler imidlertid oftere om leveransemål enn om personlig utvikling, og det er ingen karrierestige eller utviklingsløp å følge. En av de intervjuede prosjektdeltakerne opplever at kursing kun blir sett på som en utgiftspost og at det ikke er noe ønske fra bedriften om å utvikle medarbeiderne. Dette inntrykket forsterkes ved at det kun er enkelte sentrale prosjektdeltakere som deltar på bransjespesifikke konferanser.

Det smidige prosjektet

Prosjektlederen i det smidige prosjektet mener at læringen skjer i prosjektet ved at man jobber tett sammen om felles oppgaver. Her er det også etablerte seremonier som fremmer læring, som retrospective, stand-up-møter og kanban-tavle – det er også tatt i bruk parprogrammering. Parprogrammering innebærer at to utviklere sitter sammen og samarbeider om å arbeide på én datamaskin, hvor de bytter på å bruke tastaturet. Prosjektlederen mener imidlertid at dette på sikt ikke er effektivt, men at det er grei måte for noen å lære raskt på.

Prosjektlederen er tydelig på at læringskulturen må reguleres på et overordnet nivå, utenfor prosjektet. Det er viktig at medarbeidere er i utvikling, drar på kurs og konferanser – og at de får arbeide med det de synes er givende.

5. Drøfting

Dette kapittelet belyser temaet *smidig gjennomføring av IT-prosjekter i forsvarsindustrien* med bruk av funnene fra casebedriften og oppgavens teorigrunnlag. Hensikten med drøftingen er å komme frem til et synspunkt gjennom kombinasjon og syntese som svarer på oppgavens problemstilling:

- Hva er forskjellene på smidig og tradisjonell gjennomføring av IT-prosjekter?
- Er smidig gjennomføring av IT-prosjekter hensiktsmessig i forsvarsindustrien?
- Hva kan vi gjøre for å ta i bruk smidige gjennomføringsmetoder i forsvarsindustrien?

Kapittelet belyser dette gjennom en tematisk fremstilling som til slutt kulminerer i en oppsummering.

5.1 Prosjektmodeller

Teorikapittelet skiller på hvordan prosjektarbeidet struktureres og hvordan programvare-utviklingen gjennomføres. Dette er i tråd med erfaringene fra casebedriften, hvor ulike aspekter ved prosjektarbeidet følger ulike prosesser. I casebedriften så er imidlertid prosjektmodellen svært generell. Samtidig så har bedriften definert en løsningsmodell som likner på hvordan prosjektmodellen er definert i teorikapittelet. Av intervjuene så kan det virke som om prosjektmodell og løsningsmodell i praksis er det samme. Dette henger nok sammen med at prosjektene kun lager programvare og dermed ikke har noe forhold til maskinvare og andre elementer som kan inngå i en løsningsmodell.

Ifølge Wysocki (2019) så bør man velge en tradisjonell prosjektmodell dersom målet og løsningen er klar og en smidig prosjektmodell dersom målet er klart og løsningen er uklar. Begge prosjektene i casebedriften har klare mål til hva de skal oppnå, men det kan diskuteres i hvilken grad løsningen er klar. På et overordnet nivå så vil jeg si at begge prosjektene har løsningen klar. Alle oppgavene er estimerte, og det er tatt avgjørelser på hvordan de skal løses. Samtidig så må alle oppgaver brytes ned på et detaljert nivå som del av programvare-utviklingen. Her må det utformes systemkrav, arbeidspakker, oppgaver for programvare-utviklerne, estimater, dokumentasjon og tester.

De fem faktorene som Boehm og Turner (2003) legger til grunn for valg av prosjektmodell peker også i retning av tradisjonelle prosjektmodeller. Begge prosjektene i casebedriften leverer kritiske systemer hvor feil kan få alvorlige følger for nasjonens sikkerhet og føre til tap av liv. Det er relativt mange utviklere i prosjektene, det er krav til dokumentasjon og sporbarhet i arbeidet – og organisasjonen har en kultur som omfavner orden. Til tross for at prosjektene legger til rette for forutsigbart arbeid så dukker det alltid opp endringer underveis. Det siste punktet er imidlertid det eneste som taler for bruk av smidige prosjektmodeller.

Alle de intervjuede i casebedriften mener at det er kontrakten som bestemmer hvilken prosjektmodell som blir valgt. Ser vi på Wysocki (2019) sine kriterier i lys av dette så er det tydelig at løsningen er klar fordi prosjektmodellen dikterer det, og ikke motsatt. Dette sammenfaller også med forsvarssektorens prosjektmodell, PRINSIX (2020), som eksplisitt sier at kravspesifikasjoner må være på plass før oppdrag kan lyses ut.

I det tradisjonelle prosjektet så følger alle fasene etter hverandre som i en lineær prosjektmodell. Da er det i utgangspunktet ikke rom for å gjøre endringer underveis, og endringer som oppstår vil kunne føre til store forsinkelser og kostnadsoverskridelser. Dette stemmer godt med erfaringene fra casebedriften og fra bransjen for øvrig (McKinsey & Company, 2015).

Selv om det smidige prosjektet bruker den samme kontraktsformen som det tradisjonelle prosjektet så beskrives prosjektmodellen for det smidige prosjektet som en iterativ modell. Ifølge Wysocki (2019) så er dette en modell som brukes når man ikke har en kjent løsning og hvor hensikten er å få tilbakemeldinger underveis om endringer og det videre arbeidet. Det smidige prosjektet har som nevnt en kjent løsning, det er kravene i kontrakten som bestemmer hvilken funksjonalitet som skal være med og som er styrende for utviklingen av produktet. Jeg mener derfor at prosjektet i praksis har en lineær prosjektmodell med milepæler underveis som brukes til å justere kundens forventninger og til å bygge tillit til prosjektets gjennomførings-evne. Dette gjøres med demonstrasjoner og ved å gi brukere anledning til å komme med tilbakemeldinger. Jeg tror at prosjektmodellen beskrives som en iterativ modell fordi prosjektet definerer ferdigkriterier for deler av arbeidet i samsvar med de nevnte milepælene. Dette blir kanskje oppfattet som leveranser internt i prosjektet, men det gjennomføres ikke systemtesting eller idriftsetting av programvaren hos kunden. Dette gjøres ikke før i valideringsfasen, før prosjektet avsluttes.

Min konklusjon er at det er forsvarssektorens prosjektmodell som gjennom kontraktsformen bestemmer hvilken prosjektmodell som benyttes i casebedriften, og at begge prosjektene bruker en lineær prosjektmodell. Prosjektene har i så måte de samme forutsetningene for håndtering av endringer underveis. Prosjektlederne sier ikke noe om hvorvidt det har oppstått endringer i akkurat disse prosjektene som har ført til store kostnader eller forsinkelser, og begge er tydelige på at det er en viss grad av fleksibilitet innenfor de kravene og kostnadsrammene som er definert i kontrakten.

5.2 Programvareutviklingsmetoder

Avdelingen som gjennomfører det tradisjonelle prosjektet har over mange år utviklet en *tailoring*, som i praksis er en lokal tilpasning av vannfallsmodellen. Noen av sjekklistene som er beskrevet i bedriftens prosessrammeverk blir her erstattet av egne skjema eller arbeidsrutiner. En slik innarbeidet prosess gir forutsigbarhet og kontinuitet i en verden hvor prosjekter kommer og går. Samtidig så er den nok med på å hindre endring og utvikling, i og med at prosjektene på avdelingen ikke gis mulighet til å utvikle egne måter å jobbe på. Dette blir forsterket ved at tailoringen kombinerer måten prosjektet jobber på med hvordan produktet utvikles. Den inkluderer håndtering av versjoner, vedlikehold, feilrapportering og support på tvers av prosjekter – og den kombinerer prosjektroller og produktroller. Dermed så må eventuelle endringer i prosessen forankres i både prosjektaksen og i produktaksen.

Det smidige prosjektet har i så måte et bedre utgangspunkt til å forme en egnet programvare-utviklingsmetode. Casebedriftens prosessrammeverk er ikke veldig detaljert i beskrivelsen av hva som ligger i det å være *smidig*, tvert imot så kan det se ut som om det bare innebærer bruk av iterasjoner. Fordelen med dette er som nevnt at prosjektet kan forme sin egen metode – noe som også kan være en bakdel, i og med at hvert prosjekt i mangelen på en godt innarbeidet smidig metode må finne opp hjulet på nytt.

Tradisjonelle programvareutviklingsmetoder er lett å gjenkjenne og lett å definere. Dette gjør de også lett å følge. Smidige metoder er på den annen side mindre definerbare. Litteraturen sammenlikner tradisjonelle og smidige metoder primært ved hvordan aktivitetene deles opp og gjennomføres. Både Rolstadås et al. (2014) og Boehm og Turner (2003) legger vekt på iterative faser i deres definisjoner. Å gå fra lineære til iterative faser krever lite av en organisasjon, det er konkret og isolert. Dette perspektivet har nok i så måte gjort det lettere for

mange organisasjoner å gå fra tradisjonelle til smidige metoder. Samtidig så tror jeg at det har bidratt til at mange organisasjoner har gått glipp av potensialet til smidige metoder. Den første sentrale verdien fra det smidige manifestet (Beck et al., 2011) er at personer og samspill er viktigere enn prosesser og verktøy – da er det et tankekors at det er prosesser og verktøy som har fått størst plass i litteraturen.

Prosesser og verktøy har også fått mye fokus i Scrum, som er den mest utbredte smidige programvareutviklingsmetoden. Scrum er kjent for sine roller og regler. Iterasjonene har en bestemt lengde, alle møter og rutiner er nøye definert og planlagt – og alle roller har egne navn og definisjoner. Det er sågar en egen rolle som har ansvaret for at alle reglene blir fulgt.

Det smidige prosjektet har tilegnet seg Scrum og forsøkt å tilpasse de ulike rollene og reglene til egen organisasjon. Ifølge prosjektlederen så har dette vært en gradvis prosess som etter mye prøving og feiling har endt med en modell som sammenfaller med beskrivelsen av Scrum (Schwaber & Sutherland, 2020).

5.3 Teamarbeid

Teorikapittelet viser at teambegrepet kan beskrives både deskriptivt og normativt. I et team så er medlemmene gjensidig avhengig av hverandre og det legges til rette for samarbeid. Samtidig så jobber man ikke i et team før man *føler* og *oppfører* seg som et team – man har tillit til hverandre, spiller hverandre gode og tar felles ansvar.

Begge prosjektene i casebedriften er organisert i flere små team. Det er ikke gitt uttrykk for at det er noen bevissthet rundt hvorvidt det jobbes i team eller grupper.

Prosjektlederen til det smidige prosjektet forteller at de i begynnelsen hadde en tradisjonell organisering hvor avgjørelser ble tatt av teamleder og systemarkitekt utenfor teamet. Prosjektlederen var også vant til at det var han som måtte fordele roller og oppgaver. Med denne organiseringen så var det nok både samarbeid og gjensidig avhengigheter i teamet, men uten tillit og med liten grad av selvstyring så vil jeg si at de jobbet i en gruppe. Etter hvert så plasserte de systemarkitekten inne i teamet og prosjektlederen ga teamet tillit til selv å fordele oppgaver og ta ansvar for arbeidet. Jeg vet ikke i hvilken grad de følte seg som et team eller tok felles ansvar, men sannsynligheten for at de jobbet i et team ble adskillig større etter at det ble tilrettelagt for det.

I det tradisjonelle prosjektet så er det teamlederen som har systemansvaret, tar de tekniske avgjørelsene, fordeler oppgavene og følger opp på tid og kostnader. Selv om teammedlemmene kan være gjensidig avhengig av hverandre og samarbeide så blir det ikke lagt til rette for tillit og felles ansvar. Den ene prosjektdeltakeren forteller at det av og til har oppstått teamarbeid hvor teamet har hatt felles eierskap til arbeidet. Jeg mener at dette bare kan skje dersom teamlederen er bevisst på å dele på ansvaret og stole på at teammedlemmene styrer arbeidet selv. Rollen til teamlederen blir da å håndtere eksterne avhengigheter og være den de går til for å få tatt hurtige avklaringer om produktet.

Den ene prosjektdeltakeren forteller at han på det ene teamet opplever at teamlederen er autoritær, detaljstyrer arbeidet og er unødvendig pirkete når arbeidet skal gjennom godkjenningsprosessen. Dette tyder på at teamlederen har liten tillit til de øvrige prosjekt-deltakerne. Samtidig så sier prosjektlederen at arbeidet kan være effektivt dersom teamlederen driver med tett oppfølging. Dette tyder på prosjektlederen heller ikke har så stor tillit til prosjektdeltakerne. Inntrykket av manglende tillit forsterkes ved at ledelsen i liten grad legger til rette for hjemmekontor, da bruk av hjemmekontor fordrer at de ansatte i større grad styrer sine egne arbeidshverdager. Det kan imidlertid være andre årsaker til dette. Det er blant annet strenge krav til oppbevaring av graderte dokumenter og kodebaser.

Teamarbeidet i det smidige prosjektet var lite effektivt i starten, men ble effektivt mot slutten av prosjektet. Dette kan sees i sammenheng med teamutviklingsfasene til Tuckman (1965; 1977) som viser at det tar tid å bli kjent, bygge tillit til hverandre og få felles forståelse for målene. Samtidig så tror jeg ikke at det ble tilrettelagt for teamarbeid i denne fasen av prosjektet. Prosjektlederen forteller at det tok lang tid før de fikk utviklingsmetoden til å fungere godt. Siden teamutviklingsfasene nettopp handler om utviklingen av de egenskapene som gjør et team til et team så mener jeg at de ikke er relevante for gruppearbeid – og dermed heller ikke for det smidige prosjektet. Prosjektlederen forteller også at det var nødvendig å bygge kompetanse på ny teknologi og skape forståelse for et nytt produkt, noe som sannsynligvis også bidro til at teamarbeidet ble oppfattet som lite effektivt.

Hackman (2002) forklarer at det er evnen til å tilfredsstille forventninger som er det viktigste kriteriet for å avgjøre om et team blir ansett som effektivt. Hos det tradisjonelle prosjektet så er det igjen teamlederen som sitter på dette ansvaret. En teamleder som ferdigstiller arbeidet ved å bruke færre timer enn det som er estimert vil selvsagt få mer anerkjennelse enn en teamleder som ferdigstiller arbeidet ved å bruke flere timer enn det som er estimert – til tross for at dette

i seg selv ikke sier noe som helst om hvorvidt det er teamet som er effektivt eller estimatet som er for høyt. For å bli ansett som effektive så må teamlederen derfor sørge for å stadig justere forventningene som omgivelsene har til arbeidet som gjøres.

5.4 Samarbeid og brukerinvolvering

For å lykkes med smidig programvareutvikling så må både kunden og brukerne være aktivt involvert i prosessen. Dette er viktig for å kunne forstå problemet som skal løses og for å avdekke skjulte behov og gode ideer underveis.

Både det tradisjonelle og det smidige prosjektet i casebedriften gjennomfører forprosjekter med kunden før kontraktene blir signert og prosjektene satt i gang. Partene blir der enige om hva som skal lages og hvilke behov som skal dekkes. Denne prosessen inkluderer enkelte representanter fra casebedriften og enkelte representanter fra kunden. Siden prosessen ikke inkluderer brukere eller programvareutviklere så er det nok trygt å si at det er stor avstand mellom disse gruppene når prosjektene starter.

Underveis i prosjektet så håndteres tilbakemeldinger noe ulikt i prosjektene. Det tradisjonelle prosjektet har kvartalsvise fremdriftsmøter med kunden, men ingen arenaer som lar brukerne komme med tilbakemeldinger. Det tradisjonelle prosjektet har møter hver tredje måned hvor det viser frem funksjonalitet til en gruppe med brukerrepresentanter. Dette gjør det mulig å få tilbakemeldinger underveis. Teamene er imidlertid ikke involvert i disse møtene, så deres forståelser vil være prisgitt hvordan tilbakemeldingene blir tatt med videre inn i prosjektet.

Forsvarssektoren består av svært mange brukere, og ved store anskaffelser så er det nok en utfordring for sektoren å klare avdekke de ulike behovene som finnes. Videre så vil aktiv involvering fra brukere kunne kreve kapasitet som ikke finnes. Mange jobber på skift, og det kan være vanskelig å fristille brukere slik at de skal kunne delta.

I casebedriften så er mye kompetanse konsentrert på noen få mennesker. Dette er trukket frem som den største risikoen for det tradisjonelle prosjektet, hvor noen få er sentrale i omtrent alt som foregår. Det smidige prosjektet har etter hvert gitt teamene frie tøyler til å organisere arbeidet – men siden systemarkitektene i prosjektet også har gått inn i rollen som Product Owner så er det disse som gjør all kommunikasjon med omgivelsene, sitter på løsnings-forslagene og tar de viktige avgjørelsene.

5.5 Læring

Selv om læring er et sentralt element i alle kunnskapsorganisasjoner så er det kanskje særlig viktig i organisasjoner som driver med smidig programvareutvikling. Fremdriften styres av hyppige iterasjoner og tett kommunikasjon, og prosjektet må reagere på endringer i krav og omstendigheter. Dette fordrer ikke bare at teamet lærer fra iterasjon til iterasjon, men også at prosjektet lærer i et samspill med brukerne.

Ifølge Qvortrup (2004) så må læring skje som en villet handling. Begge prosjektene i casebedriften tar i bruk seremonier som i og for seg tilrettelegger for læring.

I det tradisjonelle prosjektet så tar noen av teamene i bruk daglige statusmøter, bruker kanban-tavle og har restrospective etter endt arbeid. Sistnevnte er imidlertid mer en tvungen formalitet. Siden resultatene fra disse bare blir liggende i en database så vil ikke dette føre til læring i organisasjonen, det kan i beste fall føre til refleksjon og læring på et individuelt nivå. De øvrige seremoniene er mer eller mindre tilfeldig brukt og ikke etablert i prosessen. Teamledere som tar de i bruk kan enten gjøre det bevisst for å fremme læring eller fordi de har erfart at det gjør oppfølgingen enklere. Jeg holder en knapp på det siste. Uansett så virker det ikke å være bevisst brukt for å oppnå læring av prosjektet.

Det smidige prosjektet har etablert disse seremoniene som del av gjennomføringsprosessen. Siden retrospective skjer mellom hver iterasjon så er det mer sannsynlig at det fører til læring og endringer underveis i prosjektet. Prosjektlederen forteller at endringer i organiseringen, med å plassere systemarkitekter og testere i teamene, har ført til mer effektivitet. Dette er endringer som kan ha kommet av erfaring og læring gjennom blant annet retrospective. Det er imidlertid ikke kjent om disse seremoniene brukes bevisst for å fremme læring eller om de brukes fordi de er del av Scrum-prosessen.

I det smidige prosjektet så holdes det også jevnlige demonstrasjoner av arbeidet for et utvalg brukere, med tilrettelegging for tilbakemeldinger. Dette fremmer først og fremst læring dersom det er teamene som demonstrerer og har interaksjon med brukerne. Min oppfatning er at dette er mer en arena som produkteier og prosjektleder bruker for å justere forventinger og bygge tillit. Tilbakemeldingene er uansett viktige for teamet, men jeg er usikker på om de fører til endringer annet enn justering på funksjonalitet som allerede er definert og bestemt.

Oldham og Hackman (2010) mener at det ikke bare er faglig kryssfunksjonalitet som er viktig for å fremme læring i teamarbeid, men også personlige. Det er vanskelig å gjøre noen konkrete vurderinger av personlighetene til prosjektdeltakerne i casebedriften, men erfaringen tilsier at de er en relativt homogen gruppe. Strenge krav til sikkerhetsklarering utelukker folk fra andre land og kulturer – og menn er fortsatt i overtall på teknologistudiene. Men selv blant norske mannlige dataingeniører så kan det være ulike personligheter. Hvordan de ulike teamene er satt sammen virker imidlertid ikke å være påvirket av disse i det hele tatt, men være fullt og helt basert på faglig kompetanse.

Filstad (2016) trekker frem tillit som en forutsetning for å kunne utvikle en læringskultur. Begge prosjektene i casebedriften har generelt sett stor takhøyde for å gjøre feil og for å si ifra om at man har gjort feil. Det virker derfor som at det er stor tillit mellom prosjektdeltakerne. Samtidig så opplever både prosjektdeltakere og prosjektledere at ledelsen er mer opptatt av økonomiske resultater enn av å utvikle medarbeiderne. Dette kommer frem av beskrivelsene som er gitt om hvordan prosjektene blir fulgt opp, hva som er fokus på medarbeidersamtaler og hvordan kurs og konferanser blir nedprioritert. De ansatte har nok derfor liten tillit til ledelsen. Samtidig så kan det også være at ledelsen har liten tillit til de ansatte. Tradisjonell organisering og gjennomføring gjør at ledelsen kun ser de kvantitative resultatene for et prosjekt. Prosjektet får ikke vist hva det skaper underveis og det er vanskelig å se hva prosjektet har å si for bedriften på lengre sikt. Kanskje er det viktigere for prosjektet å oppnå dets formål enn å bli ferdig til riktig tid og med færrest mulig kostnader. Dersom det stadig er sprekk på tid og kostnader så kan dette også redusere tilliten til de ansatte.

5.6 Oppsummering

Den prøyssiske *Auftragstaktik* danner grunnlaget for ledelsesfilosofiene som brukes av de fleste vestlige militærmakter i dag, som den amerikanske «mission control» og den norske «oppdragsbasert ledelse». Selv om teorigrunnlaget for disse filosofiene er grundig studert blant militærinstitusjonene så tar de fleste forfattere lett på det historiske og kulturelle bakteppe til den prøyssiske filosofien. Forsvarssjefen forklarer oppdragsbasert ledelse med at de underordnede skal bruke sin situasjonsforståelse til å handle i tråd med den overordnedes intensjon (Forsvaret, 2020). Som en prosess så er dette lett å gjennomføre. Den øverste leder uttrykker sin intensjon ved å formulere en hensikt og ønsket resultat til sine underordnede.

Disse uttrykker sine intensjoner, selvsagt innenfor lederens intensjon, til sine underordnede. Slik fortsetter det nedover hierarkiet helt til de når soldatene som skal sette det ut i praksis.

Den militære organisasjonen vil imidlertid aldri lykkes med å oppnå den fulle effekten av *Auftragstaktik* med mindre organisasjonen har en kultur som setter verdien av den enkelte soldat først, hvor gjensidig tillit mellom leder og de undergitte er vesentlig. Selv om de tyske offiserene ikke hadde tilstrekkelig med informasjon eller var trygge på ordrene som ble gitt så var de pålagt å handle raskt. Det var bedre å gjøre en adekvat handling med én gang enn en bedre handling for sent. Uerfarne offiserer gjorde derfor mange skjebnesvangre feil, så trening og opplæring var høyt prioritert. Muth (2011) forteller om store forskjeller på opplæringen til de amerikanske og de tyske offiserene under den andre verdenskrig. Den amerikanske opplæringen var tøff og strukturert. Det fantes ikke rom for fleksibilitet eller personlige tilpasninger og alle skulle gjennom den samme opplæringen. På den andre siden så var både fleksibilitet og innovasjon sentralt hos tyskerne. Dedikert trening, felles mål og verdier, tillit og reelle ønsker om å ta ansvar var avgjørende for at flat struktur og selvstyrte avdelinger skulle fungere.

På samme måte som en militær organisasjon ikke vil oppnå den fulle effekten av å adoptere prosessene fra den tyske strategien så vil det ikke være hensiktsmessig å implementere smidige prosesser i en organisasjon uten å gjøre noe med kulturen.

Jeg tror at det smidige prosjektet i casebedriften i så måte er typisk for tradisjonelle organisasjoner som begynner med smidig programvareutvikling. De tar til seg de smidige prosessene, tilordner de ulike rollene og håndhever reglene – men opplever kanskje at resultatene uteblir eller at de ikke oppnår de forbedringene som ble «lovet».

Casebedriften består av flere avdelinger og prosjekter enn de jeg har sett på, men basert på prosjektene som jeg har undersøkt så er mitt inntrykk at den ikke tilrettelegger for smidig programvareutvikling. Prosjektmodellen er lineær og åpner ikke for reelt samarbeid med kunder og brukere, bedriften har ingen læringskultur, det er liten grad av gjensidig tillit mellom ledelsen og de ansatte – og verken prosjektene eller teamene har reell beslutningsmyndighet eller eierskap til produktene som utvikles. Sistnevnte av ulike grunner. Det smidige prosjektet ble oppløst etter at produktet ble ferdigstilt, og prosjektdeltakerne har derfor gått videre til andre prosjekter og produkter. Prosjektdeltakerne til det tradisjonelle prosjektet jobber kontinuerlig

med det samme produktet over mange prosjekter – men der ligger eierskap og beslutnings-myndighet på noen få sentrale individer.

Forsvarssektoren består av Forsvarsdepartementet (FD), Forsvaret, Forsvarsmateriell (FMA), Forsvarets forskningsinstitutt (FFI), Nasjonal sikkerhetsmyndighet (NSM) og Forsvarsbygg. Forsvaret består igjen av fjorten driftsenheter som er spredt over hele landet. Anskaffelser av IT-systemer kan berøre svært mange brukere og ulike interessenter. Den nye IKT-strategien søker å finne mer fleksible måter å gjøre disse anskaffelsene på, men det betyr ikke at sektoren vil forkaste PRINSIX til fordel for Scrum med det aller første, og kanskje også med rette. Smidige metoder blomstrer i omgivelser som tillater hyppig utrulling og tett samarbeid med brukere. Dette er det vanskelig å se for seg er mulig i denne bransjen, og selv om det er brukernes hverdager som ofte skal forbedres med de nye IT-systemene så er det ikke sikkert at det er brukerne som er de viktigste interessentene. Sikkerhet er ofte en vesentlig faktor. Og i så måte så er både sikkerhetsmyndigheter og kontrakts- og innkjøpsorganisasjoner viktige interessenter.

Det finnes mange argumenter for at det ikke er hensiktsmessig å drive med smidig programvareutvikling i denne bransjen. Det er høye krav til etterrettelighet og sporbarhet i arbeidet som gjøres. Produkter kan ikke settes i drift før de har vært gjennom omfattende sikkerhetsgodkjenninger. Det er lang avstand mellom brukere og programvareutviklere. Ikke minst så legger ikke kontraktene til rette for hyppige leveranser og tilbakemeldinger underveis i gjennomføringen. Selv om de smidige programvareutviklingsmetodene kolliderer med disse hindringene så er det viktig å huske at smidig programvareutvikling handler om verdier:

Personer og samspill fremfor prosesser og verktøy – Prosesser og verktøy er viktig – men det er viktigere å utvikle menneskene ved å gi dem ansvar, tillit, læring, trivsel og mestring. Målet er å skape selvorganiserende og autonome team med kompetente og motiverte mennesker med eierskap til produktet. Så kan teamet lage sin egen lille prosess. En prosess som stadig utvikler seg basert på hva teamet mener er nødvendig for at de skal kunne arbeide best mulig sammen og levere resultater.

Programvare som virker fremfor omfattende dokumentasjon – Overordnede skisser i kombinasjon med programkode og demonstrasjon av produktet har ofte større verdi enn detaljert og omfattende dokumentasjon. Det betyr ikke at krav til etterrettelighet og sporbarhet skal settes til side, eller at dokumentasjonen av sikkerhetsmekanismene skal kastes. Det handler

mer om at teamet skifter fokus bort fra at programvaren skal testes og leveres på slutten av prosjektet til at programvaren i stedet testes og kjøres hver dag. Dette kan føre til endringer (for eksempel automatisering av tester) som over tid gjør at den enorme og dyre systemtestperioden på slutten av prosjektet reduseres og kanskje til og med forsvinner helt.

Samarbeid med kunden fremfor kontraktsforhandlinger – Dette er en verdi som det kan være vanskelig å leve etter – da både kunden og leverandøren ønsker lav risiko og forutsigbar økonomi. Dette handler om å erkjenne at man tidlig i prosjektet ikke er i stand til å forutsi forløpet eller resultatet av en detaljert gjennomføringsplan. Da kan man tilby en fastpris som er romslig nok til å tillate smidighet innenfor rammene av relativt generelt beskrevne krav. Eller så kan man avtale en begrenset mengde med funksjonalitet om gangen. Dette gjør det mulig å levere verdi oftere, noe som utvikler gjensidig tillit. Underveis så vil det oppstå læring som gjør det lettere å legge planer for det videre arbeidet. Regler for offentlige anskaffelser og anskaffelsesprosesser i praksis gjør det vanskelig å prioritere samarbeid foran kontrakts-forhandlinger. Én tilnærming kan da være å gjøre arbeidet med kontrakten til en felles reise ved å sende inn utkast og tidlige versjoner for gjennomgang og diskusjoner. Jobbe litt tettere sammen. Transparens fremfor å forsøke å lure inn detaljer for å tjene ekstra. På den måten gjøre intensjonene kjente for hverandre og få felles eierskap.

Å reagere på endringer fremfor å følge en plan – Dette handler om å erkjenne at endringer underveis i prosjektet er uunngåelig. Det betyr ikke at man skal kaste planene og ty til kaos, man må vite hvilken vei man skal gå. Det er imidlertid viktig å justere kursen underveis, gjerne i små inkrementelle steg. Planlegging er en essensiell og integrert del av Scrum. Teamet legger en plan for hva som skal gjøres før det begynner på en iterasjon, og hver morgen samles de for å finjustere på den. I tillegg så blir det også laget en overordnet plan som strekker over flere iterasjoner, et såkalt «veikart». Planleggingen av et prosjekt med en lang tidshorisont vil nødvendigvis være basert på mange antakelser. Da er det desto viktigere å gjøre hyppige vurderinger og justeringer underveis.

Denne oppgaven bygger på dybdeintervjuer av noen få mennesker, som sammen med teorigrunnlaget kan bidra til å gi en økt forståelse for temaet. Det er viktig å poengtere at den reflekterer *min* forståelse. Det er jeg som har valgt informanter, stilt spørsmålene, lyttet og tolket – det er jeg som har funnet teorier som jeg synes er relevante – og det er jeg som har drøftet meg frem til svar på problemstillingen. Det betyr ikke at oppgaven ikke kan ha verdi for

andre. Den kan være til hjelp for andre som skal undersøke det samme fenomenet og den kan være et godt utgangspunkt ved design av andre typer (for eksempel kvantitative) studier.

En svakhet med oppgaven er at de to prosjektene fra casebedriften ikke er veldig sammenliknbare. Det ene prosjektet er et vedlikeholdsprosjekt med ansatte som har jobbet sammen i mange år. Det andre prosjektet utvikler et nytt produkt og består i stor grad av konsulenter som aldri har jobbet med hverandre før. Det er derfor vanskelig å vurdere i hvilken grad de ulike utviklingsmetodene spiller inn på effektivitet, teamarbeid og læring.

En annen svakhet er at det ikke blir gjennomført intervjuer av prosjektdeltakere fra det smidige prosjektet. Det er bare prosjektlederen sine erfaringer og synspunkter som gjenspeiles i oppgaven. Dette har noe å si for vurderingene som gjøres. Det kunne også vært interessant å fått innspill fra mennesker som jobber i forsvarssektoren, da primært i en organisasjon som driver med anskaffelser av IT-systemer og som kjenner til prosessene.

Et utgangspunkt for oppgaven, som også gjelder for programvareutviklingsmiljøet for øvrig, er at smidige metoder er ansett som bedre enn tradisjonelle metoder. Dette vil nok farge hvordan informantene opplever og forteller om sine arbeidshverdager. Da kan det være lettere å prate positivt om det som forbindes med smidige metoder og negativt om det som forbindes med tradisjonelle metoder. Generelt så kan det også være lettere å prate om det som oppleves som frustrerende enn det som fungerer bra – og dette kan igjen avhenge av forholdet og tilliten til den som gjennomfører intervjuet.

6 Konklusjon

Denne oppgaven ser på hvordan smidig gjennomføring av IT-prosjekter kan bidra til å møte noen av utfordringene som forsvarsindustrien står ovenfor. Temaet blir belyst ved at det gjennomføres dybdeintervjuer av prosjektledere og prosjektdeltakere i to ulike prosjekter hos en organisasjon som leverer programvare til forsvarssektoren. Deretter blir funnene drøftet opp mot relevant teori. Dette gir ikke anledning til å dra generelle konklusjoner, men det kan være med på å gi økt innsikt og forståelse for temaet.

For å svare på problemstillingen så tar oppgaven utgangspunkt i følgende forskningsspørsmål:

Hva er forskjellene på smidig og tradisjonell gjennomføring av IT-prosjekter?

Dette spørsmålet legger grunnlaget for oppgaven og blir grundig belyst både teoretisk og gjennom dybdeintervjuer. På et overordnet nivå så har tradisjonelle IT-prosjekter en lineær tilnærming til gjennomføringen hvor prosjektet ikke går videre til neste fase før den foregående fasen er ferdig, verifisert og godkjent. Smidige prosjekter har en iterativ tilnærming med fokus på å levere små og fungerende deler av produktet på hyppig basis. Oppgaven trekker frem at smidig programvareutvikling handler mer om mennesker og samhandling enn om prosesser og verktøy – så teamarbeid, læring, medvirkning og selvstyring blir tillagt stor vekt.

Er smidig gjennomføring av IT-prosjekter hensiktsmessig i forsvarsindustrien?

For å svare på dette så drøfter oppgaven hvorvidt prosjektene i casebedriften og forsvars-sektorens prosjektmodell legger til rette for smidig gjennomføring av IT-prosjekter.

Den konkluderer med at prosjektmodellen ikke legger til rette for smidig gjennomføring, og at dette kanskje også er naturlig gitt forsvarssektorens størrelse med mange brukere og ulike interessenter. Der hvor smidig gjennomføring fordrer tett samarbeid med brukere så er det kanskje andre interessenter som er viktigere i denne sektoren, hvor sikkerhetsmyndigheter og kontrakts- og innkjøpsorganisasjoner trekkes frem som eksempler.

Oppgaven konkluderer også med at prosjektene i casebedriften i liten grad legger til rette for smidig gjennomføring – og trekker frem mangel på læringskultur, liten gjensidig tillit mellom ledelsen og de ansatte og at teamene ikke har noen reell beslutningsmyndighet som årsaker.

Med forsvarssektorens tradisjonelle prosjektmodell og en kultur i bedriften som ikke legger til rette for smidig gjennomføring så virker det ikke å være hensiktsmessig å ta i bruk smidige metoder.

Hva kan vi gjøre for å ta i bruk smidige gjennomføringsmetoder i forsvarsindustrien?

Hensikten med smidig er å forme organisasjonene slik at de blir i stand til å reagere raskt på endringer og fange opp mulighetene i markedet. Erkjennelsen av at det er menneskene, og ikke prosessene, som er de primære driverne for suksess vil kunne dreie organisasjonene i en smidig retning. Personer og samspill fremfor prosesser og verktøy. Programvare som virker fremfor omfattende dokumentasjon. Samarbeid med kunden fremfor kontraktsforhandlinger. Å reagere på endringer fremfor å følge en plan.

Videre arbeid

Temaet *smidig gjennomføring av IT-prosjekter i forsvarsindustrien* kan utforskes videre på mange måter. Et naturlig neste steg kan være å gjøre en litteraturstudie for å finne ut om andre har liknende erfaringer. Dersom det ikke er gjort mange studier innen programvareutvikling i forsvarsindustrien så kan det være aktuelt å inkludere studier for sammenliknbare industrier. Dette kan være programvare til medisinsk utstyr, sikkerhetssystemer i energisektoren eller til autonome kjøretøy. Det kan også være interessant å se problemstillingen fra forsvarssektorens perspektiv og eventuelt vurdere om tiltakene som er skissert i IKT-strategien har påvirket eller vil kunne påvirke prosjektmodellen som sektoren bruker.

For casebedriften så kan det være naturlig å se videre på hvordan teamene kan benytte automatisering og verktøy for kontinuerlig testing og integrering for å bygge ned siloer og komme nærmere de smidige verdiene. Her er DevOps og kanskje spesielt DevSecOps aktuelle metoder å se på. Dette er metoder som på mange måter kan være med på å bygge opp under smidig programvareutvikling, men som har fokus på den tekniske gjennomføringen.

7 Litteraturliste

Adler, P., & Adler, P. (2012). Expert voices. In S.E. Baker & R. Edwards. *How many qualitative interviews is enough* (National Centre for Research Methods Review Discussion Paper), 8-11. http://eprints.ncrm.ac.uk/2273

Alleman, G. (2002). Is there an underlying theory of software project management? (A critique of the transformational and normative views of project management). (Version 5.0, 29 June 2015).

Anonym casebedrift. (2021). *Prosessrammeverk* [Upublisert].

Assmann, R. (2008). *Teamorganisering: Veien til mer fleksible organisasjoner*. Fagbokforlaget.

Baker, S. E., & Edwards, R. (2012). Introduction. In S.E. Baker & R. Edwards. *How many qualitative interviews is enough* (National Centre for Research Methods Review Discussion Paper), 8-11. http://eprints.ncrm.ac.uk/2273

Bang, H. (2010). Effektivitet i ledergrupper: En studie av sammenhengen mellom gruppeprosesser og teameffektivitet i ledermøter.

Bateson, G. (1972). *Steps to an ecology of mind*. Ballantine Books.

Beath, C. M., & Orlikowski, W. J. (1994). The contradictory structure of systems development methodologies: Deconstructing the IS-user relationship in information engineering. *Information systems research*, *5*(4), 350-377.

Beck, K. (2000). *Extreme programming explained: Embrace change*. addison-wesley professional.

Beck, K., Cockburn, A., & Fowler, M. (2011). *The Agile Manifesto*. Agile Alliance. http://agilemanifesto.org/

Bell, T. E., & Thayer, T. A. (1976). *Software requirements: Are they really a problem?* Proceedings of the 2nd international conference on Software engineering, San Francisco, California, USA.

Bellomo, S. (2011). *A closer look at 804: A summary of considerations for DoD program managers*.

Bengtsson, J. (2006). *Å forske i sykdoms-og pleieerfaringer: Livsverdensfenomenologiske bidrag*. Høyskoleforlaget.

Bertrand, M. (2013). *The origin of «software engineering»*. Retrieved 30.06.2021 from https://bertrandmeyer.com/2013/04/04/the-origin-of-software-engineering/

Boehm, B. (1988). A spiral model of software development and enhancement. *IEEE Computer*(21), 61-72. http://www-scf.usc.edu/~csci201/lectures/Lecture11/boehm1988.pdf

Boehm, B., & Turner, R. (2003). *Balancing agility and discipline: A guide for the perplexed*. Addison-Wesley Professional.

Brannen, J. (2012). Expert voices. In S.E. Baker & R. Edwards. *How many qualitative interviews is enough* (National Centre for Research Methods Review Discussion Paper), 8-11. http://eprints.ncrm.ac.uk/2273

Brooks, F. P. (1975). *The Mythical Man-Month*. Addison-Wesley Publishing Company.

Brooks, F. P. (1987). *No silver bullet: Essence and accidents of software engineering*.

Cameron, L. (2018). What to know about the scientist who invented the term «software engineering». *IEEE Software Magazine*. https://www.computer.org/piblications/tech-news/events/what-to-know-about-the-scientist-who-invented-the-term-software-engineering

Cockburn, A. (2002). *Agile software development*. Addison-Wesley.

Cockburn, A., & Highsmith, J. (2001). Agile software development: The people factor. *Computer*, *34*(11), 131-133.

Colbjørnsen, T. (2002). Byråkratiets endelikt. *Ledelse av fleksible organisasjoner og selvbevisste medarbeidere. I A. Skogstad og S. Einarsen (red.) Ledelse på godt og vondt. Effektivitet og trivsel*, 377-399.

DeGrace, P., & Stahl, L. H. (1990). *Wicked problems, righteous solutions*. Yourdon Press.

Diener, E., & Crandall, R. (1978). *Ethics in social and behavioral research*. U Chicago Press.

DOD-STD-2167. (1985). *Defense system software development, military standard*.

DOD-STD-2167A. (1988). *Defense system software development, military standard*.

Dul, J., & Hak, T. (2007). *Case study methodology in business research*. Routledge.

Eide, O. K., & Nørstebø, C. (2017). Vår evne til å stå imot et cyberangrep er marginal. *Forsvarets forum*. https://forsvaretsforum.no/vår-evne-til-å-stå-i-mot-et-cyberangrep-er-marginal

Einstein, A. (1936). Physics and reality. *Franklin Institute Journal*(221), 313-347.

Ekman, G. (2004). *Fra prat til resultat: Om lederskap i hverdagen*. Abstrakt.

Etterretningstjenesten. (2021). Fokus. https://www.forsvaret.no/aktuelt-og-presse/publikasjoner/fokus

Filstad, C. (2016). *Organisasjonslæring: Fra kunnskap til kompetanse*. Bokforlaget.

FMA. (2020). *PRINSIX*. Retrieved 30.09.2021 from https://www.fma.no/prinsix

FMA. (2021). *PRINSIX utdanningsprogram*. https://forsvaret.metierportal.no

Forsvaret. (2019). *Forsvarssjefens fagmilitære råd 2019: Rapport fra cyberutredningen*. Forsvarsstaben.

Forsvaret. (2020). *Forsvarets grunnsyn på ledelse*. FOBID Retrieved from https://www.forsvaret.no/soldater-og-ansatte/regelverk/Forsvarets-grunnsyn-ledelse.pdf

Forsvarsdepartementet. (2015). *Ny etat for materiell i forsvarssektoren*. Retrieved from https://www.regjeringen.no/no/aktuelt/ny-etat-for-materiell-i-forsvarssektoren/id2407364/

Forsvarsdepartementet. (2019). *IKT-strategi for forsvarssektoren*. Retrieved from https://www.regjeringen.no/no/dokumenter/ikt-strategi-for-forsvarssektoren/id2685492/

Fundin, A. P., & Bergman, B. L. (2003). Exploring the customer feedback process. *Measuring Business Excellence*.

Glaser, B. G., & Strauss, A. L. (1967). *The discovery of grounded theory*. Aldine.

Hackman, J. R. (1987). The design of work teams. *Handbook of organizational behavior*, 315-342.

Hackman, J. R. (2002). *Leading teams: Setting the stage for great performances*. Harvard Business Press.

Herbst, P. G. (1977). *Alternativ til hierarkisk organisasjon*. Tanum.

Highsmith, J. (2001). *History: The Agile Manifesto*. https://agilemanifesto.org/history.html

Hjertø, K. B. (2000). *Tid for effektive team: Veien fra klisjé til realiteter*. https://www.magma.no/tid-for-effektive-team-veien-fra-klisj-til-realiteter

Hughes, D. (2009). *Moltke on the art of war: Selected writings*. Presidio Press.

Johannessen, A., Tufte, P. A., & Christoffersen, L. (2010). *Introduksjon til samfunnsvitenskapelig metode* (Vol. 4). Abstrakt Oslo.

Johnson, J. (2010). Is agile old school? *Software Magazine*. https://www.umsl.edu/~sauterv/analysis/AgileOldSchool.pdf

Katzenbach, J. R., & Smith, K. S. (1993). The discipline of teams. *Best of Harvard Business Review 2000*, 163-171.

Kommunal- og moderniseringsdepartementet. (2018). *Tidenes største satsing på digitalisering* https://www.regjeringen.no/no/aktuelt/tidenes-storste-satsing-pa-digitalisering/id2614074/

Kvale, S., Brinkmann, S., Anderssen, T. M., & Rygge, J. (2009). *Det kvalitative forskningsintervju*. Gyldendal Akademisk.

Kvalsund, R. (2014). *Coaching: Metode, prosess, relasjon*. Synery Publishing.

Kvalsund, R. (2018). *Oppmerksomhet og påvirkning i hjelperelasjoner*. Fagbokforlaget.

Lindsjørn, Y., Sjøberg, D. I. K., Dingsøyr, T., Bergersen, G. R., & Dybå, T. (2016). Teamwork quality and project success in software development: A survey of agile development teams. *Journal of Systems and Software*, *122*, 274-286.

Maiden, N., & Jones, S. (2010). Agile requirements can we have our cake and eat it too? *IEEE software*, *27*(3), 87-88.

Mattsson, P. A. (2003). Upplysning, romantik och Auftragstaktik. In A. Cedergren & P. A. Mattsson (Eds.), *Uppdragstaktik*. Försvarshögskolan.

McKinsey & Company. (2015). *Modernisering og effektivisering av stabs-, støtte og forvaltningsfunksjoner i forsvarssektoren*. McKinsey & Company. https://www.regjeringen.no/globalassets/departementene/fd/dokumenter/rapporter-og-regelverk/150317modernisering-og-effektivisering-av-forsvarssektoren.pdf

Miller, D. L. (2003). The stages of group development: A retrospective study of dynamic team processes. *Canadian Journal of Administrative Sciences/Revue Canadienne des Sciences de l'Administration*, *20*(2), 121-134.

Moltke, H. K. B. (1869). Instructions for large unit commanders. In D. Hughes (Ed.), *Moltke on the art of war: Selected writings*. Presidio Press.

Muth, J. (2011). *Command culture: Officer education in the US army and the German armed forces, 1901-1940, and the consequences for World War II*. University of North Texas Press.

Naur, P., & Randell, B. (1969). Software engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th-11th October 1968.

Nelsen II, J. T. (1987). "Auftragstaktik": A Case for decentralized battle. *The US Army War College Quarterly: Parameters*, *17*(1), 22.

NSD. (2021). *Norsk senter for forskningsdata*. https://www.nsd.no

Oldham, G. R., & Hackman, J. R. (2010). Not what it was and not what it will be: The future of job design research. *Journal of organizational behavior*, *31*(2‐3), 463-479.

Pinto, J. K. (2013). *Project management achieving competitive advantage* (3 ed.). Pearson Educational Limited.

Polkinghorne, D. E. (1989). Phenomenological research methods. In *Existential-phenomenological perspectives in psychology* (pp. 41-60). Springer.

Pressman, R. S. (2015). *Software engineering: A practitioner's approach* (8 ed.). McGraw-Hill Education.

PST. (2021). Nasjonal trusselvurdering. https://www.pst.no/alle-artikler/trusselvurderinger/nasjonal-trusselvurdering-2021/#Bruk%20av%20nasjonal%20trusselvurdering

Qvortrup, L. (2004). *Det vidende samfund: Mysteriet og viden, læring og dannelse*. Unge Pædagoger.

Rickards, T., & Moger, S. (2000). Creative leadership processes in project team development: An alternative to Tuckman's stage model. *British journal of Management*, *11*(4), 273-283.

Rolstadås, A., Olsson, N., Johansen, A., & Langlo, J. A. (2014). *Praktisk prosjektledelse: Fra idé til gevinst*. Fagbokforlaget.

Royce, W. W. (1970). Managing the development of sarge software systems. *Proceedings, IEEE WESCON*.

Runkel, P. J., Lawrence, M., Oldfield, S., Rider, M., & Clark, C. (1971). Stages of group development: An empirical test of Tuckman's hypothesis. *The Journal of Applied Behavioral Science*, *7*(2), 180-193.

Røgeberg, O. (2019). Norge i europatoppen i bruk av offentlige nettjenester. https://www.regjeringen.no/no/aktuelt/tidenes-storste-satsing-pa-digitalisering/id2614074/

Schwaber, K. (1997). Scrum development process. In *Business object design and implementation* (pp. 117-134). Springer.

Schwaber, K. (2004). *Agile project management with Scrum*. Microsoft press.

Schwaber, K., & Sutherland, J. (2010). The Scrum Guide.

Schwaber, K., & Sutherland, J. (2020). The Scrum Guide.

Scrum.org. (2020). *The Scrum Framework Poster*. https://www.scrum.org/resources/scrum-framework-poster

Scrum.org. (2021). Professional Scrum Certifications. https://www.scrum.org/professional-scrum-certifications

Sjøvold, E. (2006). *Teamet: Utvikling, effektivitet og endring i grupper*. Universitetsforlaget.

Skogstad, A. (2011). Psykososiale faktorer i arbeidet. I A. Skogstad & S. Einarsen (Red.). *Det Gode arbeidsmiljø: krav og utfordringer*, *2*, 16-41.

Sommerville, I. (2011). Software engineering. *ISBN-10*, *137035152*(9), 18.

Sutherland, J. (2004). Agile development: Lessons learned from the first scrum. *Cutter Agile Project Management Advisory Service: Executive Update*, *5*(20), 1-4.

Takeuchi, H., & Nonaka, I. (1986). The new new product development game. *Harvard business review*, *64*(1), 137-146.

Thagaard, T. (2009). *Systematikk og innlevelse: En innføring i kvalitativ metode* (3 ed.). Fagbokforlaget.

Thompson, L. L. (2008). *Making the team: A guide for managers* (3 ed.). Pearson Prentice Hall.

Thorsrud, E., & Emery, F. (1970). *Mot en ny bedriftsorganisasjon*. Tanum.

Tjora, A. (2012). *Kvalitative forskningsmetoder i praksis* (Vol. 2). Gyldendal Akademisk.

Tubbs, D., Luzwick, P. G., & Sharp Sr, W. G. (2002). Technology and law: The evolution of digital warfare. *International Law Studies*, *76*(1), 25.

Tuckman, B. W. (1965). Developmental sequence in small groups. *Psychological Bulletin*.

Tuckman, B. W. (1984). Citation classic: Developmental sequence in small groups. *Current Concerns*.

Tuckman, B. W., & Jensen, M. A. (1977). Stages of small group development revisited. *Group and Organizational Studies*, *2*, 419-427.

Verheyen, G. (2013). Scrum: Framework, not methodology. https://guntherverheyen.com/2013/03/21/scrum-framework-not-methodology/

Weyl, H. (1956). *Erkenntnis und besinnung*. F. Rouge.

Womack, J. P., & Jones, D. T. (1997). Lean thinking: Banish waste and create wealth in your corporation. *Journal of the Operational Research Society*, *48*(11), 1148-1148.

Wysocki, R. K. (2019). *Effective project management: Traditional, agile, extreme, hybrid*. John Wiley & Sons.

Yin, R. K. (2009). Case study research: Design and methods. *Applied social research methods series*, *5*.

8 Vedlegg

8.1 Intervjuguide

A. Introduksjon

Redegjørelse for studien og hvordan dataene vil bli behandlet.

a. Takk for at du ønsker å delta på dette intervjuet.

b. Formålet med studien er å se på hvordan vi opplever å jobbe med programvareutvikling i forsvarsindustrien, med fokus på smidige metoder.

c. Intervjuet vil vare mellom 30 og 60 minutter.

d. Alt du sier vil bli behandlet anonymt. Ingen navn vil bli brukt. Det vil komme frem av oppgaven om du er prosjektleder eller deltaker i et prosjekt som bruker enten en tradisjonell eller en smidig metode, at prosjektene gjennomføres på to ulike lokasjoner og at bedriften leverer kritisk infrastruktur / programvare til forsvarssektoren.

e. Du svarer bare på de spørsmålene som du ønsker.

B. Bakgrunn

a. Kan du fortelle litt om din tidligere arbeidserfaring?

b. Hvor lenge har du jobbet i denne bedriften?

c. Kan du fortelle litt om hva jobben din innebærer?

d. Dersom du er en prosjektleder, hvordan har du lært prosjektlederfaget? Erfaring, utdanning, kurs?

e. Har du erfaring med forskjellige prosjektmodeller eller programvareutviklingsmetoder?

C. Prosjektmodell og programvareutviklingsmetode

Dette temaet handler om hvordan prosjektarbeidet blir delt opp og gjennomført.

I denne oppgaven så skiller jeg på en prosjektmodell og en programvare-utviklingsmetode, men dette skillet er ikke nødvendigvis så synlig i praksis.
Det viktigste er hvordan du opplever gjennomføringen av prosjektet.

a. Opplever du at prosjektet bruker både en prosjektmodell og en programvare-utviklingsmetode, eller går dette på det samme?

b. Hvordan vil du beskrive prosjektmodellen som brukes i prosjektet? Er den lineær, inkrementell, iterativ, adaptiv ...?

c. Hvordan vil du beskrive programvareutviklingsmetoden som brukes i prosjektet? Er det en tradisjonell vannfallsmetode (spiralmetode, v-metode) eller en smidig metode?

d. Hvem har bestemt hvilken prosjektmodell og hvilken programvare-utviklingsmetode som skal brukes? Hvorfor?

e. Vil du si at målet med prosjektet har vært tydelig fra starten?

f. Vil du si at løsningen (krav, design) har vært klart fra starten, eller har den utviklet seg underveis i gjennomføringen?

g. Har kravene endret seg underveis i prosjektet?

h. Har planen endret seg underveis i prosjektet?

i. Opplever du at det oppstår gjentakende problemer i prosjektet? I hvilke faser oppstår dem?

j. Hva er de største risikoene i prosjektet?

k. Hvordan vurderer du prosjektets suksess? Hvordan tror du andre vurderer den?

l. Er det noen av de typiske styringsparameterne tid, kostnad og omfang/kvalitet som blir oftere prioritert enn andre? Hvorfor tror du det er sånn?

m. Hva tror du er de største utfordringene (om noen) med å bruke smidige metoder i et prosjekt som leverer programvare til forsvarssektoren? Spørsmålet stilles uavhengig av om prosjektet bruker tradisjonelle eller smidige metoder.

D. Teamarbeid

   a. Kan du fortelle om hvor mange dere er og hvilke roller dere har? Er dere samlokaliserte?

   b. Jobber dere på andre prosjekter samtidig? Hvordan er organiseringen?

   c. Vil du si at dere jobber i team eller i en gruppe? Hvordan vil du beskrive forskjellen på team og gruppe?

   d. Hva kjennetegner et effektivt team? Er dere effektive? Hele tiden?

   e. Vil du si at teamet er selvstendig? Hvem tar avgjørelser? (teamdeltakerne, teamlederen, prosjektlederen, produktlederen)

   f. Hvordan koordineres arbeidsoppgavene?

   g. Har teamet felles mål eller hensikt?

   h. Hva skjer med teamet etter at prosjektet er over?

   i. Hva gjør deg motivert til å jobbe i prosjektet? Kan være både ekstern og intern motivasjon..

E. Samarbeid og brukerinvolvering

   a. Hvordan vil du beskrive samarbeidet med kunden?

   b. Hvordan er brukerne representert i arbeidet?

   c. Legger kunden til rette for samarbeid med brukerne?

   d. Dersom teamet oppdager at noe må løses annerledes enn planlagt, hvordan håndteres det? Kontraktsforhandlinger, arbeidsmøte med brukere ...

F. Læring

   a. Vil du si at organisasjonen har en læringskultur? På hvilken måte, eller hvorfor ikke?

b. I hvilken grad blir det gitt tilbakemeldinger underveis? På hvilket nivå? System, organisasjon, team eller individ? (kan være tilbakemeldinger i form av hva som har vært bra, hva som har vært mindre bra og hva som kan gjøres bedre neste gang)

c. Opplever du at det læring er prioritert i prosjektet? Er det definerte seremonier som fremmer læring? (retrospective, stand-up, tilbakemeldingskultur)

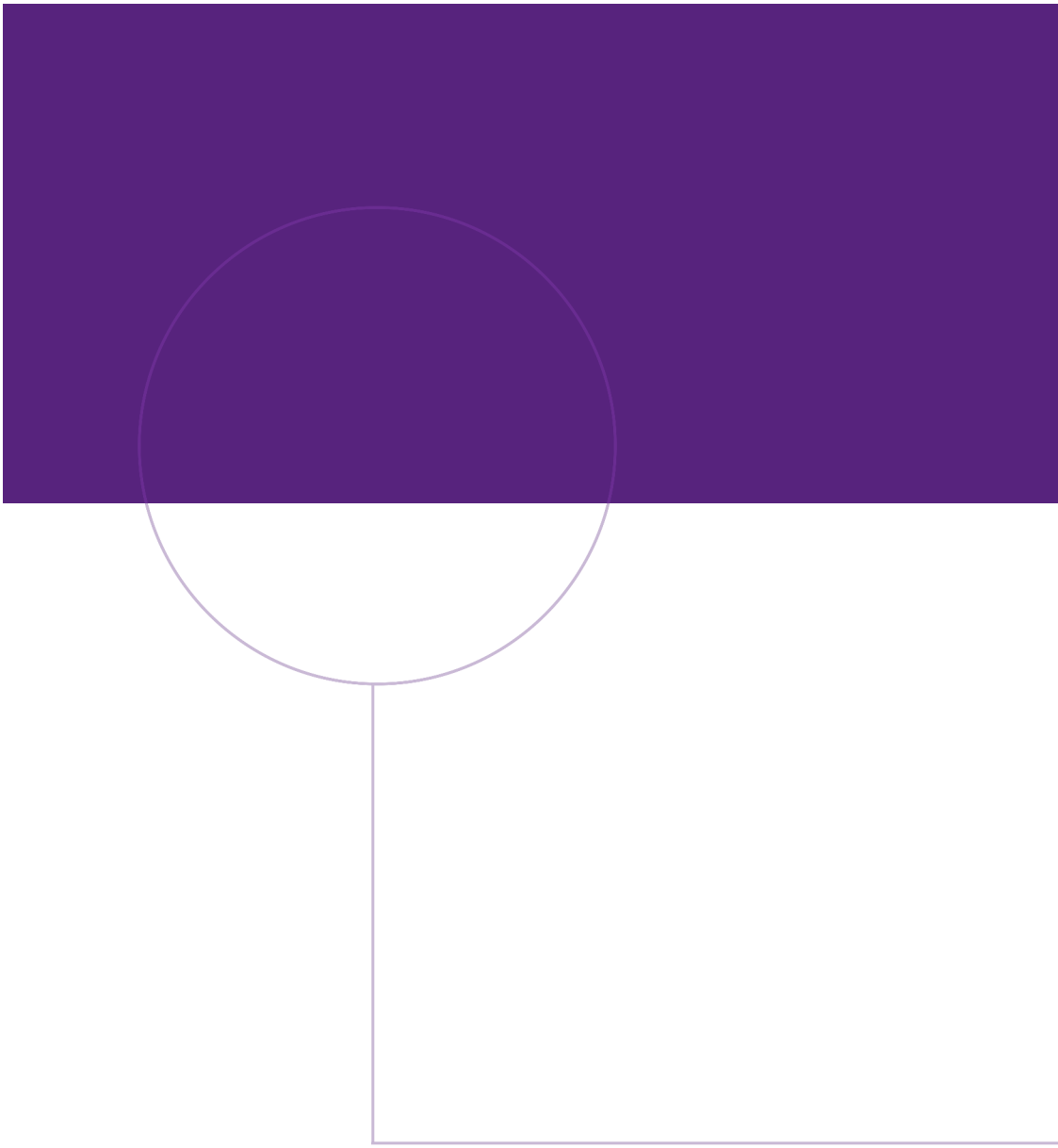
d. Vil du si at du allerede har kunnskaper om de fleste oppgavene som du tar eller får tildelt, eller hender det at oppgavene krever at du må tilegne deg helt ny kunnskap?

e. Føler du at det er trygt å dele meninger og informasjon med de andre i teamet? Også hvis du har gjort en feil?

G. Avslutning

a. Er det noe du ønsker å legge til som du tror kan være relevant for temaet?

b. Har du noen spørsmål om lagring av data eller hva den skal brukes til? Eller om andre aspekter ved oppgaven?



NTNU

Kunnskap for en bedre verden