Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for økonomi
Institutt for industriell økonomi og teknologiledelse

Magne Tøndel

OVERFORBRUK AV SCRUM?

ERFARINGER OM METODER OG PROSESSER INNEN PROGRAMVAREUTVIKLING I PRAKSIS

Masteroppgave i Organisasjon og ledelse
Veileder: Ola Edvin Vie
Medveileder: Jan Alexander Langlo
August 2023

NTNU
Kunnskap for en bedre verden

Magne Tøndel

OVERFORBRUK AV SCRUM?

ERFARINGER OM METODER OG PROSESSER INNEN PROGRAMVAREUTVIKLING I PRAKSIS

Masteroppgave i Organisasjon og ledelse
Veileder: Ola Edvin Vie
Medveileder: Jan Alexander Langlo
August 2023

Norges teknisk-naturvitenskapelige universitet
Fakultet for økonomi
Institutt for industriell økonomi og teknologiledelse

NTNU
Kunnskap for en bedre verden

Forord

Denne masteroppgaven er skrevet som en del av min videreutdanning i Organisasjon og ledelse ved Norges teknisk- naturvitenskapelige universitet i Trondheim. Oppgavens tema er innenfor prosjektledelse rettet inn imot IT-bransjen - og omhandler gjennomføring og styring av programvareutviklingsprosjekter ved bruk av ulike dokumenterte utviklingsmetoder og metodikker.

Fra tidligere studier er jeg utdannet sivilingeniør i Datateknikk – Komplekse datasystemer ved Norges teknisk- naturvitenskapelige universitet i Trondheim. Etter dette har jeg opparbeidet meg over 15 års erfaring med programvareutvikling i en rekke offentlige og private utviklingsprosjekter - både som systemutvikler, prosjektleder og personalansvarlig leder. I skrivende stund er jeg ansatt som leder for en utviklergruppe i firmaet Norconsult Digital AS. Problemstillingen til denne masteroppgaven er definert av meg selv og bygger på selverfarte utfordringer med gjennomføring og styring av mange forskjellige typer av systemutviklingsprosjekter - som til dels har vært veldig ulike både med tanke på innhold, målsetninger og ambisjonsnivå.

Jeg ønsker å rette en spesielt stor takk til min veileder Ola Edvin Vie ved Institutt for industriell økonomi og teknologiledelse. Uten hans gode hjelp og veiledning til strukturering av oppgaven og akademisk skriving ville denne masteroppgaven blitt noe helt annet. I denne forbindelse vil jeg også takke min medveileder Jan Alexander Langlo ved Institutt for maskinteknikk og produksjon for gode tips og støtte spesielt med tanke på gjennomføringen av forskningsprosjektet som tilhører denne masteroppgaven. En stor takk rettes selvfølgelig også til case-bedriften som har stilt opp med nysgjerrige og engasjerte systemutviklere både for intervjuer og samtaler – hvor utviklingslederen i denne bedriften må

nevnes spesielt. Jeg er takknemlig for å kunne skrive denne masteroppgaven på bakgrunn av en egendefinert problemstilling som jeg selv synes er interessant, men uten alle dere ville jeg ikke hatt denne muligheten.

Sist, men ikke minst vil jeg takke min kjære kone Elisabeth J. Tøndel og våre tre barn Fredrik, Lilly Karoline og Martin for deres forståelse og tålmodighet gjennom det siste året. Det å gjennomføre denne masteroppgaven i tillegg til full jobb og oppfølging av tre barn både på skolearbeid og fritidsaktiviteter har til tider vært utfordrende – men sammen så klarer vi alt!

Magne Tøndel - Orkanger, 12.August 2023

Sammendrag

I dagens systemutviklingsprosjekter så er det en utpreget bruk av allerede dokumenterte utviklingsmetodikker og tilpasninger av disse. Scrum er den mest kjente og mest brukte metodikken for gjennomføring av slike prosjekter i profesjonell sammenheng. Forskningsspørsmålet til denne masteroppgaven stiller spørsmål om denne metodikken passer til bruk i alle systemutviklingsprosjekter – og følgelig hva som kjennetegner et prosjekt hvor man ikke bør benytte utviklingsmetodikken Scrum:

Hva kjennetegner programvareutviklingsprosjekter hvor det ikke vil være hensiktsmessig å benytte "Scrum" som utviklingsmetode?

Funnene av denne kvalitative case-studien og forskningsprosjektet som denne studien omfatter viser at Scrum-metodikken nødvendigvis ikke passer til prosjekter som omfatter daglig oppfølging av kunder og brukere, samt forventninger til raske beslutninger og hyppige leveranser. Et eksempel på et slikt prosjekt vil være et forvaltning- og vedlikeholdsprosjekt hvor man videreutvikler eksisterende programvare som allerede er i markedet - og hvor det allerede eksisterer kunder og brukere med egne krav, ambisjoner og meninger. Et slikt prosjekt krever ofte små og hyppige leveranser for å minimere risiko. Små og hyppige leveranser gjøres typisk i disse prosjektene på bakgrunn av feilrettinger eller for hjelpe kunder og brukere videre med sitt eget arbeid så fort som mulig. Disse prosjektene egner seg derfor ikke for omfattende planlegging, prioritering og leveranser på fastsatte tidspunkt som Scrum-metodikken omfatter.

For å komme frem til funnene i denne case-studien er det utviklet et teoretisk rammeverk bestående av tre proposisjoner som er utviklet fra tilgjengelig litteratur på dette området, i tillegg til min egen kunnskap og erfaring om bruk

av forskjellige utviklingsmetoder i praksis. Disse proposisjonene videreutvikles og omformuleres i analysen av empirien til forskningsprosjektet, diskuteres med hensyn på teoretiske og praktiske implikasjoner og brukes tilslutt for å kunne besvare forskningsspørsmålet.

Selve forskningsspørsmålet er utledet for å undersøke bruk av dokumenterte metodikker innen programvareutviklingsprosjekter i praksis – og underbygges av forfatterens erfaringer med prosjektgjennomføringer i denne bransjen over en lang tidsperiode.

Abstract

In today's system development projects, there is a distinct use of already documented development methodologies and adaptations of these. Scrum is the best known and most used methodology for the implementation of such projects in a professional context. The research question for this master's thesis asks whether this methodology is suitable for use in all system development projects - and consequently what characterizes a project where the development methodology Scrum should not be used:

What characterizes software development projects where it would not be appropriate to use "Scrum" as development method?

The finding in this qualitative case-study and the research project that this study includes show that the Scrum methodology is not necessarily suitable for projects that include daily follow-up of customers and users, as well as expectations for quick decisions and frequent deliveries. An example of such a project would be a management and maintenance project where you further develop existing software that is already on the market - and where there are already customers and users with their own requirements, ambitions and opinions. Such a project often requires small and frequent deliveries to minimize risk. Small and frequent deliveries are typically done in these projects because of error corrections or to help customers and users continue with their own work as quickly as possible. These projects are therefore not suitable for extensive planning, prioritization and deliveries at fixed times, which the Scrum methodology includes.

In order to arrive at the findings of this case study, a theoretical framework has been developed consisting of three propositions which have been developed from available literature in this area, in addition to my own knowledge and expe-

rience of using different development methods in practice. These propositions are further developed and reformulated in the analysis of the empirical findings for the research project, discussed with regard to theoretical and practical implications and finally used to be able to answer the research question.

The research question itself is derived to investigate the use of documented methodologies within software development projects in practice - and is supported by the author's experiences with project contribution in this industry over a long period of time.

Innhold

Liste over figurer

Liste over tabeller

Kapittel 1

Introduksjon

Denne masteroppgaven tar utgangspunkt i systemutvikling i profesjonelle prosjekter og hvordan forskjellige utviklingsmetodikker kan tas i bruk for gjennomføring og styring av disse. Ifølge SSB så var det i utgangen av 2022 over 65 000 arbeidsforhold i Norge som var klassifisert som IT-tjenester – som blant annet gjennomfører nettopp systemutviklingsprosjekter. Prosjektene varierer selvfølgelig både i innhold og størrelse – helt fra små kundetilpasninger til nasjonale digitale satsninger, men er sammen viktige for å drive den teknologiske utviklingen videre.

Å utføre arbeid i prosjekter har etter hvert blitt en mer og mer vanlig måte å utføre arbeidsoppgaver på (Rolstadås m. fl., 2014). Prosjektledelse i en eller annen form har funnet sted i tusenvis av år, men det er i løpet av de siste 70 årene at organisasjoner begynte å innføre systematiske prosjektstyringsverktøy og teknikker for å løse komplekse prosjekter (Kwak, 2005). For 60 år tilbake så kunne organisasjoner velge om man ville ta i bruk prosjektledelse, men i dag er det ikke spørsmål om man skal innføre prosjektledelse eller ikke – men når? (Kerzner, 2017).

Suksess i et prosjekt defineres ofte som at prosjektet må fullføres innen planlagt tidsperiode, kostnad og omfang, og jerntrianglet brukes ofte i litteraturen for å sette bilde på denne konkrete sammenhengen mellom tid, kostnad og kvalitet (Atkinson, 1999). Samtidig så påpekes det at prosjekter i dag gjerne har andre

effekter som påvirker samfunnet utenfor prosjektet og at suksesskriterier som effektivitet, målsetninger, relevans, virkninger og levedyktighet bør vurderes for å måle faktisk suksess i et prosjekt (Samset, 2014).

Programvareutvikling som en del av prosjekter har sine røtter tilbake til tidlig på 1960 tallet. Før dette var større datamaskiner bare tilgjengelig for universiteter og forskningsmiljøer (Wirth, 2008). Fagfeltet innen utvikling av programvare har endret seg utrolig mye etter dette og denne utvikling pågår fortsatt i stadig økende takt. Denne utviklingen har ført til at mange metoder, rammeverk, prosesser og metodikker er blitt utviklet og tatt i bruk – for å prøve å følge et stadig økende krav i markedet på dette området.

Prosjekter som omfatter programvareutvikling er gjerne komplekse – noe som fører til at det ofte er vanskelig å oppnå suksess i slike prosjekter. Årsakene til dette er ofte at prosjektledere ikke alltid klarer å forstå kundes behov eller at omfanget av prosjektet ikke er godt nok definert. Programvareutviklingsprosjekter som ikke oppnår suksess har også gjerne designfeil eller andre svakheter fra begynnelsen av prosjektet – før en eneste linje med kode er implementert (Al-Ahmad m. fl., 2009). Kapittel 2 *Prosjektledelse* inneholder en nærmere beskrivelse av prosjektledelse, prosjekttyper og suksess i prosjekter.

For å sikre suksess i komplekse programvareutviklingsprosjekter måtte prosjektledelse og metodikker innføres. Fossefallsmodellen regnes for å være den eldste kjente utviklingsmetoden som ble tatt i bruk i forbindelse med programvareutvikling (Awad, 2005) – hvor en lineær prosess går ut på å planlegge, designe og dokumentere en løsning før implementering og testing kan forekomme. Etter hvert som prosjektene ble stadig mer og mer komplekse og kravene ble høyere og høyere ble også viktigheten av å kunne respondere på endringer hyppig mer og mer aktuelt - dette parallelt med en rivende teknologisk utvikling hvor behovet for økt samarbeid mellom kunde og utvikler stadig øker. Dette fører oss til de smidige utviklingsmetodene som brukes i slike utviklingsprosjekter i dag hvor *"Scrum"* er den mest kjente og utbredte metodikken. Kapittel 3 *Utviklingsmetoder* inneholder en oversikt over et utvalg av de mest kjente utviklingsmetodene – både tradisjonelle og nyere smidige metoder.

Forskningsspørsmålet jeg ønsker å finne ut i denne masteroppgaven er:

Hva kjennetegner programvareutviklingsprosjekter hvor det ikke vil være hensiktsmessig å benytte "Scrum" som utviklingsmetode?

For å prøve å finne svaret på dette forskningsspørsmålet fra et teoretisk perspektiv har jeg studert litteratur tilhørende fagfeltet prosjektledelse og utviklingsmetoder innen programvareutvikling - og for å avgrense omfanget for oppgaven har jeg definert følgende delspørsmål:

1. *Finnes det typer av programvareutviklingsprosjekter som favorittseirer en utviklingsmetode fremfor en annen?*

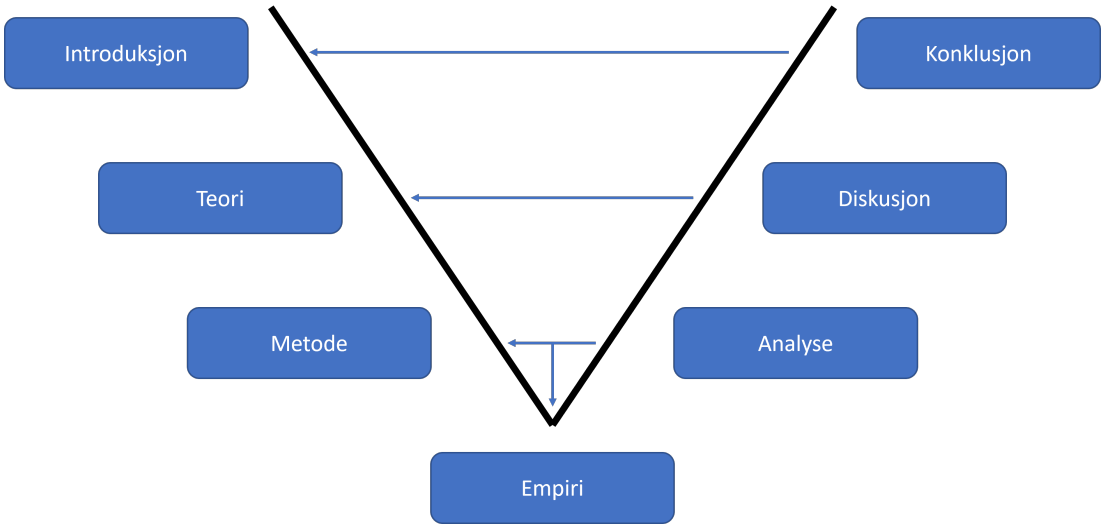
2. *Hvilke alternative utviklingsmetoder er mest utbredt i bransjen - og hva er egentlig kjennskapen og bruken av disse i bransjen i dag?*

3. *Hva kjennetegner et programvareutviklingsprosjekt der man bør velge "Scrum" som utviklingsmetode?*

For hver av disse delspørsmålene vil det utvikles en proposisjon som sammen utgjør det teoretiske rammeverket for denne masteroppgaven. Proposisjonene utvikles på bakgrunn av utvalgt litteratur, samt mine egne erfaringer på dette området. Det teoretiske rammeverket vil videre brukes aktivt igjennom hele denne masteroppgaven; det vil bli omformulert og tilpasset de empiriske data i løpet av analysen, diskutert med hensyn på teoretiske og praktiske implikasjoner, og til slutt brukt for å konkludere på det opprinnelige forskningsspørsmålet.

1.1 Rapportens struktur

Rapportens oppbygging og struktur er illustrert i figur 1.1.



Figur 1.1: Rapportens oppbygging og struktur

Denne rapporten er strukturert i 8 kapitler:

Kapittel 1 *Introduksjon* introduserer temaet for forskningsstudiet, problemstillingene og forskningsspørsmålet.

Kapittel 2 *Prosjektledelse* presenterer relevant litteratur omhandlende prosjektledelse. Dette kapitlet er inndelt i to deler - hvor den første delen omfatter prosjekttyper og den andre delen omfatter suksess i prosjekter.

Kapittel 3 *Utviklingsmetoder* presenterer de mest kjente og mest relevante utviklingsmetodene som er beskrevet - og som i større eller mindre grad brukes i systemutviklingsprosjekter i dag. Dette kapitlet er inndelt i to deler – hvor den første delen omfatter tradisjonelle utviklingsmetoder og den andre delen omfatter nyere smidige utviklingsmetoder. I slutten av dette kapittelet presenteres

det teoretiske rammeverket som legger grunnlaget for videre analyse, diskusjon og konklusjon i denne masteroppgaven.

Kapittel 4 *Metode* omfatter en presentasjon og forklaring av valgt metode for dette forskningsprosjektet, samt begrunnelse for valget av metode og strategi.

Kapittel 5 *Empiriske funn* inneholder en kort presentasjon av case-bedriften, samt resultatene eller de empiriske data som jeg har samlet inn i forbindelse med dette forskningsprosjektet.

Kapittel 6 *Analyse* inneholder en analyse av empirien som ble presentert i kapittel 5 og det teoretiske rammeverket som ble presentert på slutten av kapittel 3. I figur 1.1 er dette illustrert med en pil fra Analyse-delen til Empiri- og Metode-delen.

Kapittel 7 *Diskusjon* inneholder en oppsummering av hver enkelt proposisjon og en diskusjon av funnene sett i lys av litteraturen som er presentert i kapittel 2 og 3. I figur 1.1 er dette illustrert med en pil fra Diskusjons-delen til Teori-delen.

Kapittel 8 *Konklusjon* inneholder min konklusjon og en kort oversikt over hva mine resultater er. Forskningsspørsmålet som ble definert i kapittel 1 vil også bli besvart i dette kapitlet. I figur 1.1 er dette illustrert med en pil fra Konklusjons-delen til Introduksjons-delen.

Kapittel 2

Prosjektledelse

I dette kapitlet defineres prosjektledelse og det blir presenteret hvordan prosjektledelse treffer systemutviklingsprosjekter. Kapittelet inneholder også en oversikt over forskjellige prosjekttyper eller klassifiseringer av prosjekter, samt hvordan man kan måle suksess i prosjekter.

Å utføre arbeid i prosjekter har etter hvert blitt en mer og mer vanlig måte å løse konkrete arbeidsoppgaver på - både i næringsliv og forvaltning. Et prosjekt i denne forbindelse defineres som et tiltak som har karakter av en engangsforetelse med et gitt mål og avgrenset omfang, og som gjennomføres innenfor en gitt tids- og kostnadsramme (Rolstadås m. fl., 2014). Det finnes også mange andre definisjoner på et prosjekt og PMI (Project Management Institute) definerer et prosjekt som en midlertidig innsats gjennomført for å skape et unikt produkt, tjeneste eller resultat (Institute, 2001).

For å gjennomføre prosjekter utøves prosjektledelse, og PMI velger å benytte følgende definisjon på prosjektledelse (Institute, 2001, s. 443):

> *"Anvendelse av kunnskap, ferdigheter, verktøy og teknikker på prosjektaktiviteter for å imøtekomme prosjektets behov"*

Prosjektstyring innebærer planlegging av prosjektet samt oppfølging av fremdrift og kostnader slik at de definerte målene nås, mens prosjektledelse i tillegg omfatter spørsmål knyttet til organisering og tilrettelegging av arbeidet og de menneskelige ressursene (Rolstadås m. fl., 2014). Fagområdet prosjektledelse kan

også deles inn i to retninger – de som legger vekt på organisering og samarbeid, og de som vektlegger metoder og teknikker.

Innen programvareutviklingsprosjekter utøves det også prosjektledelse i de aller fleste prosjekter av en viss størrelse. Programvareutviklingsprosjekter er ofte komplekse og i stor grad avhengig av en del forutsetninger som grunnlag for en god prosjektledelse. Infrastruktur (maskinvare og programvare), kompetanse, ledelsesengasjement, overvåkning og inspeksjon trekkes frem som viktige faktorer i denne forbindelse (Lucky m. fl., 2014).

Programvareutviklingsprosjekter er som andre prosjekter unike og de kan også være veldig forskjellige med tanke på struktur, tidsfrister, budsjett, antall deltakere, omfang og mange andre egenskaper. Dette fører oss videre til klassifiseringer av prosjekter og forskjellige prosjekttyper.

2.1 Prosjekttyper

Klassifisering i typer og typologier er viktige verktøy når vi ønsker å sammenlikne likheter og ulikheter mellom forskjellige prosjekter (Morris m. fl., 2011). Klassifisering er kanskje en av de "mest sentrale og generiske av alle våre konseptuelle øvelser" og uten klassifisering så "ville det ikke eksistert noe avansert konseptualisering, resonnement, språk, dataanalyse eller for den saks skyld samfunnsvitenskapelig forskning" (Bailey, 1994, s. 1). Klassifisering eller typer av prosjekter er også essensielt og nødvendig når man utformer nye metoder og verktøy innen prosjektledelse som vil være effektive på forskjellige typer av prosjekter (Crawford m. fl., 2005).

I grove trekk kan vi si at det finnes tre prosjekttyper: konkrete prosjekter, ad-hoc prosjekter og åpne prosjekter, og at mange prosjekter havner et sted mellom disse typene og kombinerer trekkene (Briner m. fl., 2000). Ofte vil prosjekter starte åpne eller ad-hoc og bli mer og mer konkrete etter hvert som prosjektet gjennomføres. Klassifiseringen innenfor disse prosjekttypene styres av parameterne "Spesifisering av forventet resultat", "Graden av struktur og formalitet" og "Kunnskapsnivå" – hvor konkrete prosjekter scorer høyt på alle tre parameterne og åpne prosjekter scorer lavt (Briner m. fl., 2000).

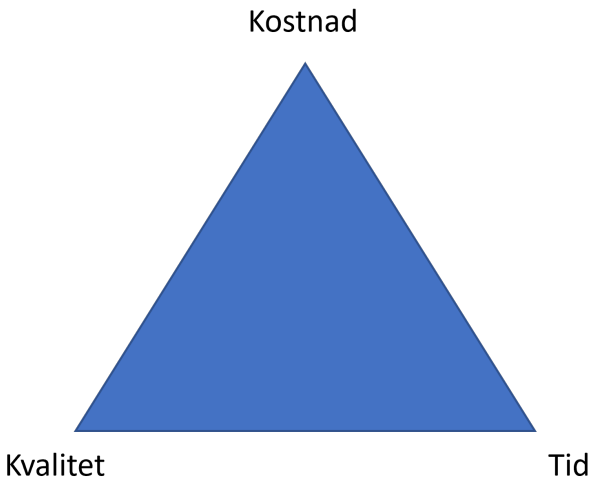
Innenfor programvareutviklingsprosjekter finnes det også forskjellige typer av prosjekter. Hansson velger å dele inn prosjektene i nyutviklingsprosjekter og vedlikeholdsprosjekter hvor vedlikeholdsprosjekter vil være knyttet til en eller flere allerede eksisterende applikasjoner, mens et nyutviklingsprosjekt kan føre frem til en ny applikasjon hvis prosjektet utvikles ferdig og settes i drift - og gjøres til gjenstand for et nytt vedlikeholdsprosjekt (Hansson, 2013). Det er også uttalt at vedlikehold utgjør over halvparten av innsatsen som investeres i programvaren i løpet av dens levetid (Rehman m. fl., 2018). Men det finnes også andre klassifiseringer av prosjekttyper innen programvareutviklingsprosjekter som investeringsprosjekter som defineres som en langsiktig tildeling av ressurser og midler for å oppnå konkrete samfunn-, effekt- og resultatmål (Brustad og Jarle, 2001) og forskningsprosjekter som OECD velger å definere slik (Sundnes, 2002, s. 1):

> *"Forskning og utviklingsarbeid (FoU) er kreativ virksomhet som utføres systematisk for å oppnå økt kunnskap - herunder kunnskap om mennesket, kultur og samfunn og omfatter også bruken av denne kunnskapen til å finne nye anvendelser"*

2.2 Suksess i prosjekter

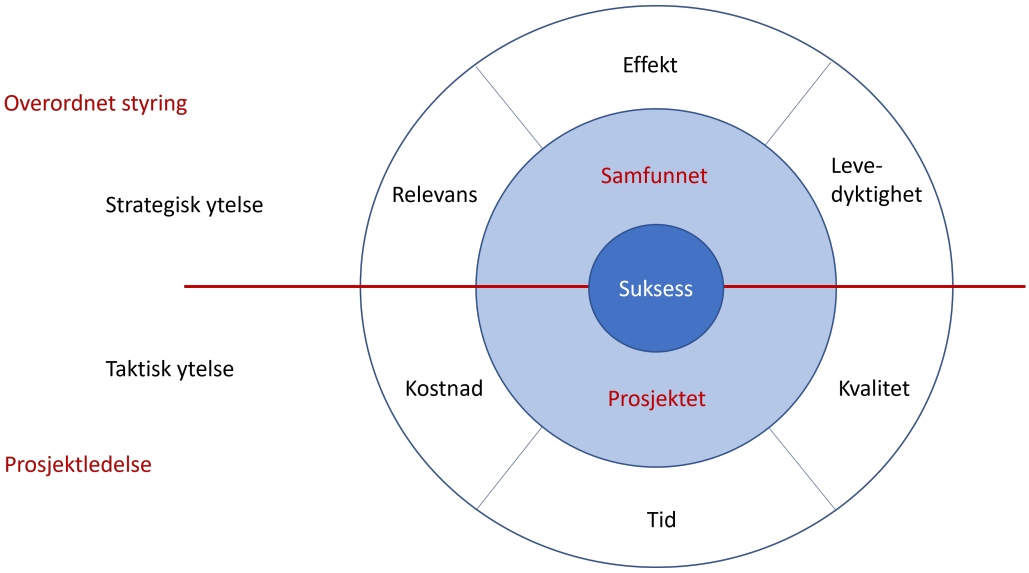
Alle prosjekter er mer eller mindre vellykket, og problemene som kan oppstå i prosjekter kan være mer eller mindre alvorlige og konsekvensene mer eller mindre langvarige (Samset, 2014). To av de mest vanlige problemene som kan oppstå i prosjekter er at tidspunkt for ferdigstillelse blir forsinket eller at prosjektet blir mer kostbart enn først antatt. Disse to problemene får ofte veldig stor oppmerksomhet, men er ofte de minst alvorlige problemene i et prosjekt – sett i forhold til de fremtidige inntektene eller samfunnsnytten som et vellykket prosjekt kan oppnå på sikt (Samset, 2014).

Jerntrianglet har tradisjonelt satt et bilde på hvordan suksess i et prosjekt kan måles hvor sammenhengen mellom tid, kost og kvalitet utgjør graden av suksess i et prosjekt – og brukes fortsatt i stor grad i litteratur omhandlende prosjektledelse (Atkinson, 1999). Figur 2.1 viser en typisk fremstilling av jerntrianglet.



Figur 2.1: Jerntrianglet

Ser man prosjekter i et større perspektiv, vil kravene for et vellykket prosjekt også måles i om prosjektet bidrar til avtalte målsetninger i samfunnet. Suksesskriterier som i denne sammenheng skal realiseres er da også effektivitet, målsetninger, relevans, virkninger og levedyktighet (Samset, 2014). Samset velger da å utvide vurderingen av prosjektet til å omfatte det han betegner som taktisk og strategisk ytelse, hvor taktisk ytelse er et uttrykk for om prosjektledelsen har lyktes i å gjennomføre prosjektet og strategisk ytelse gjelder spørsmålet om prosjektet er levedyktig og relevant gjennom hele levetiden. Figur 2.2 viser Samsets modell som gjenspeiler sammenhengen mellom strategisk ytelse og taktisk ytelse - hvor taktisk ytelse omfatter de tradisjonelle måleparametene for et prosjekt og hvor strategisk ytelse omfatter relevansen og effekten prosjektet har for samfunnet i tillegg til selve levedyktigheten til prosjektet.



Figur 2.2: Strategisk og taktisk ytelse i et prosjekt

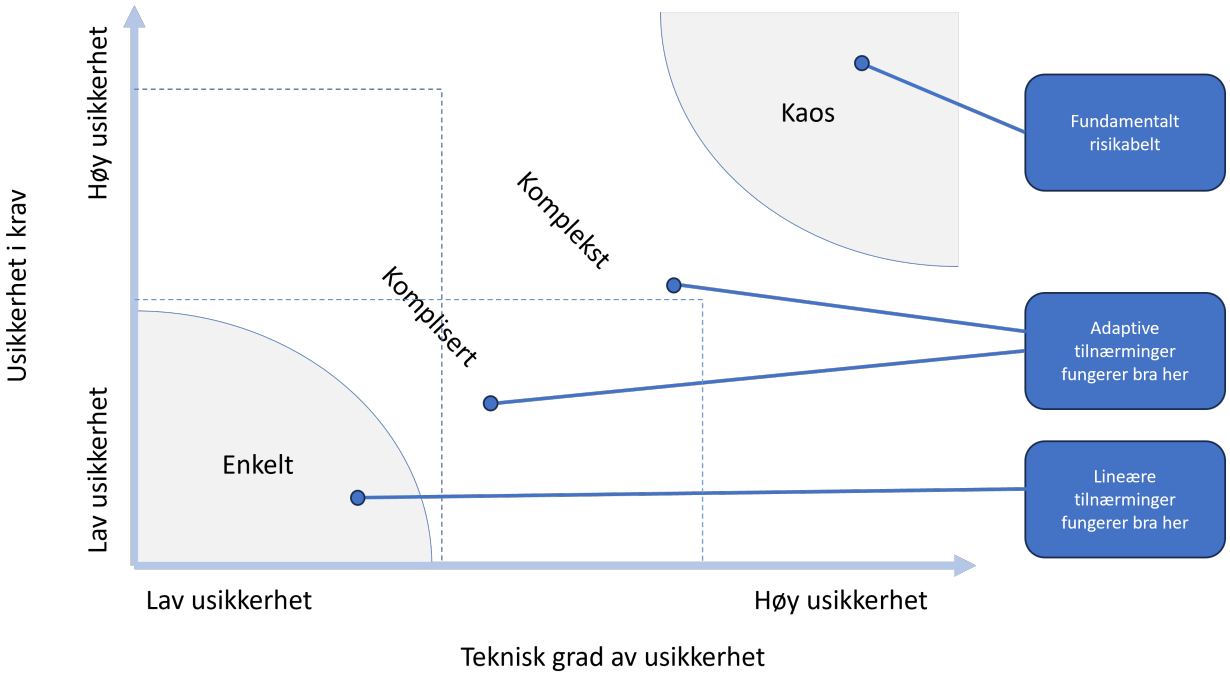
Graden av oppnådd nytte er et avgjørende element for å kunne si om et digitaliseringsinitiativ har vært vellykket eller ikke (Berg, 2021). I denne sammenhengen så blir nyttestyring aktuelt. Problemer og utfordringer med gjennomføring og innføring av større IT og digitaliseringsprosjekter de siste årene har virkelig satt nyttestyring i fokus, og i skrivende stund så kan innføringen av Helseplattformen i norsk helsesektor nevnes i denne forbindelse.

Digitaliseringsprosjekter er sjelden ferdig når prosjektet er avsluttet – det vil som oftest være et behov for fremtidig feilretting, tilpasninger og videreutvikling i slike leveranser. For å sikre dette behovet er det innført begrepet *kontinuerlig utvikling* – hvor ressursene fra den opprinnelige leveransen også får ansvaret for videre utvikling og forvaltning. Kontinuerlig utvikling er vært knyttet til kontinuerlige leveranser, der nytten realiseres løpende i utviklingen av tjenestene/produktene (Berg, 2021). Arbeidsformen i kontinuerlig utvikling skiller seg noe fra tradisjonell prosjektgjennomføring ved at arbeidet er organisert etter løpende leveranser uten noen prosjektplan – men behovet for nyttestyring vil stort sett være det samme. Helene Berg med flere mener derfor at kontinuerlig utvikling også åpner for kontinuerlig evaluering av oppnådde nytteeffekter un-

derveis i prosjektet og anbefaler derfor et sterkere fokus på kost-nytteanalyse og planlegging av nytterealisering i slike prosjekter.

2.3 Kompliserte eller komplekse prosjekter

Graden av kompleksheit vil også være avgjørende for hvordan et prosjekt bør styres og gjennomføres. Figur 2.3 viser et godt visuelt bilde på akkurat dette - hvor sammenhengen mellom usikkerhet i krav og graden av teknisk usikkerhet kan definere om et prosjekt kan klassifiseres som enkelt, komplisert, komplekst eller kan føre til kaos (Alliance, 2017).



Figur 2.3: Fra enkelt til kaos

Figuren til *Project Management Institute* viser at lineære tilnærminger ofte vil være tilstrekkelig for enklere prosjekter med lav grad av teknisk usikkerhet og

klarere krav. For mer kompliserte eller komplekse prosjekter så vil det være mer hensiktsmessig å innføre adaptive tilnærminger hvor man igjennom prosjektet planlegger og tilpasser arbeidsoppgavene i takt med omverden og endrer forutsetninger ved behov.

2.4 Oppsummering og utvikling av proposisjon

Med bakgrunn i litteraturen som er presentert i dette kapittelet og delspørsmål 1 av forskningsspørsmålet vil det avslutningsvis i dette kapittelet utvikles en proposisjon. Delspørsmål 1 ble i kapittel 1 *Introduksjon* definert slik:

Finnes det typer av programvareutviklingsprosjekter som favorittseirer en utviklingsmetode fremfor en annen?

Proposisjon 1: Forskjeller og ulikheter i systemutviklingsprosjekter.

Alle prosjekter er unike. For å belyse ulikheter og forskjeller i prosjekter og hvordan dette styrer hvordan prosjektledelse bør utøves ble det i dette kapittelet presentert et utdrag fra tilgjengelig litteratur omhandlende prosjekttyper, samt ulike måter å måle suksess i prosjekter. Systemutviklingsprosjekter som gjennomføres i både offentlig og privat sektor er gjerne både kompliserte og tidskrevende, og god og riktig nyttestyring er derfor veldig viktig og kommer nok ikke til å bli mindre aktuell i fremtiden. Delspørsmålet løfter problemstillingen om noen av de eksisterende utviklingsmetodene vil være mer hensiktsmessig å innføre i visse typer av systemutviklingsprosjekter for faktisk å kunne oppnå suksess. Med dette som bakgrunn og for å kunne underbygge svaret på delspørsmål 1 av forskningsspørsmålet har jeg formulert proposisjon 1 på følgende måte:

Egenskaper ved prosjektet eller forskjellige typer av systemutviklingsprosjekter påvirker arbeidsflyten i prosjektet og følgelig hvilken utviklingsmetodikk man burde velge for å kunne oppnå suksess.

I neste kapittel vil det presenteres et utvalg av tradisjonelle og mer smidige utviklingsmetoder som representerer mangfoldet og den historiske utviklingen av arbeidsmetodikker for systemutviklingsprosjekter.

Kapittel 3

Utviklingsmetoder

I dette kapitlet presenteres ulike utviklingsmetoder som er beskrevet i teorien på dette området, samt uttalte fordeler og ulemper med disse. Kapittelet er delt i to hvor et utvalg av tradisjonelle utviklingsmetoder blir presentert i del en, og noen av de viktigste av de nyere smidige utviklingsmetodene i del to. Tilslutt i del to vil jeg presentere noen proposisjoner som med grunnlag i teorien og min egen erfaring på dette fagområdet gjør antakelser med bakgrunn i forskningsspørsmålet og de to siste delspørsmålene som ble presentert i kapittel 1 *Introduksjon*. Avslutningsvis vil det presenteres et teknologisk rammeverk med de tre proposisjonene – ett for hvert av delspørsmålene i forskningsspørsmålet.

3.1 Tradisjonelle utviklingsmetoder

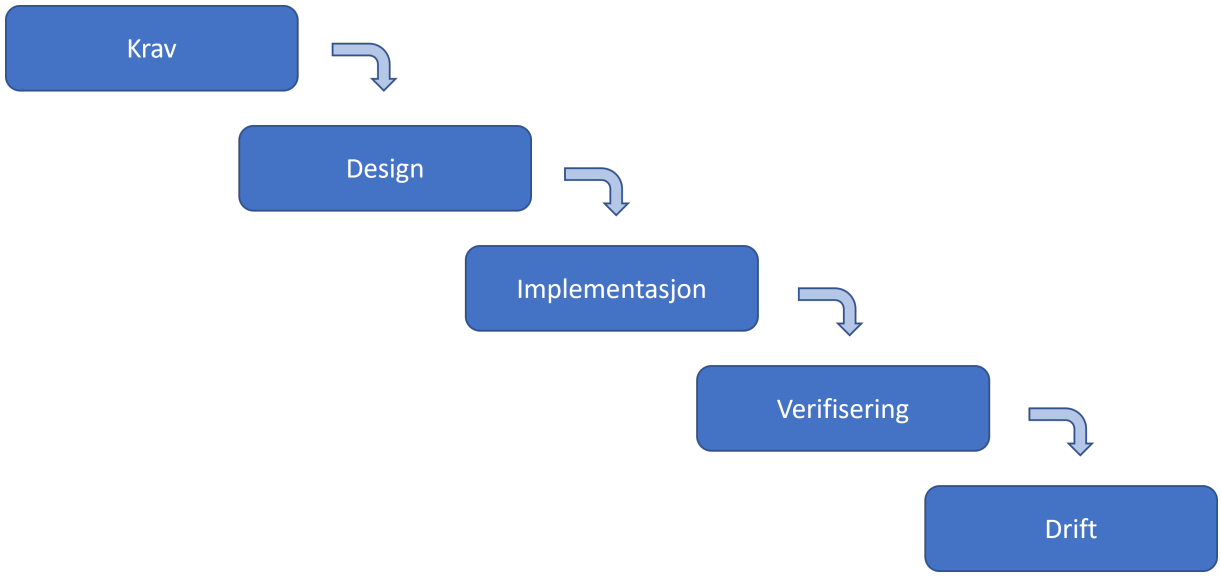
Hvis vi ser tilbake til 1960 tallet – når system- og programvareutvikling virkelig begynte å få oppmerksomhet så jobbet systemutviklerne etter en metode som senere har fått navnet *"The Code and Fix Model"*. Metoden gikk egentlig ut på å sette i gang med koding og rette feil etter hvert som de oppsto. Christophe Thibuat beskrev metoden slik *"... one year of slamming the code, one year of debugging..."* (Awad, 2005, s. 3). Mange profesjonelle systemutviklere i dag vil nok i en eller annen grad latterliggjøre denne modellen – men samtidig så er modellen helt klart aktuell den dag i dag. Utallige store og små prosjekter har startet akkurat bare med et problem eller et behov – hvor noen bare har begynt å løse problemer ved hjelp av tilgjengelig kodeverktøy, uten noen som

helst tanke på utfall eller resultat på forhånd.

Det er i dag flere kjente utviklingsmetoder til bruk i systemutviklingsprosjekter. En rivende utvikling og stadig økende krav og kompleksitet har gjort at flere mer eller mindre utviklingsmetoder har blitt utviklet. I dette kapittelet har jeg studert litteratur omhandlende de mer tradisjonelle utviklingsmetodene hvor de mest utbredte av disse er presentert med uttalte styrker og svakheter.

3.1.1 Fossefallsmodellen

Fossefallsmetoden regnes for å være den eldste kjente utviklingsmetoden innen programvareutvikling og brukes fortsatt i mange store prosjekter – både private og offentlige (Awad, 2005). Fossefallsmetodens bryter ned arbeidsoppgavene i et prosjekt til lineære og sekvensielle faser – hvor hver fase avhenger av resultatet fra den foregående fasen og korresponderer med en spesialisering av arbeidsoppgavene (for eksempel verifisering) (Petersen m. fl., 2009). Hovedpoenget med denne utviklingsmetoden er dermed at man sikter mot å fullføre fasens hovedoppgave før fasen avsluttes. Denne tilnærmingen til prosjektarbeid er typisk i flere fagområder i industrien. I forbindelse med programvareutvikling anses denne utviklingsmetoden å være blant de mindre iterative og fleksible metodikkene, ettersom prosessene stort sett flyter i en retning – som i en foss. Figur 3.1 viser de typiske sekvensielle prosjektfasene i fossefallsmodellen.

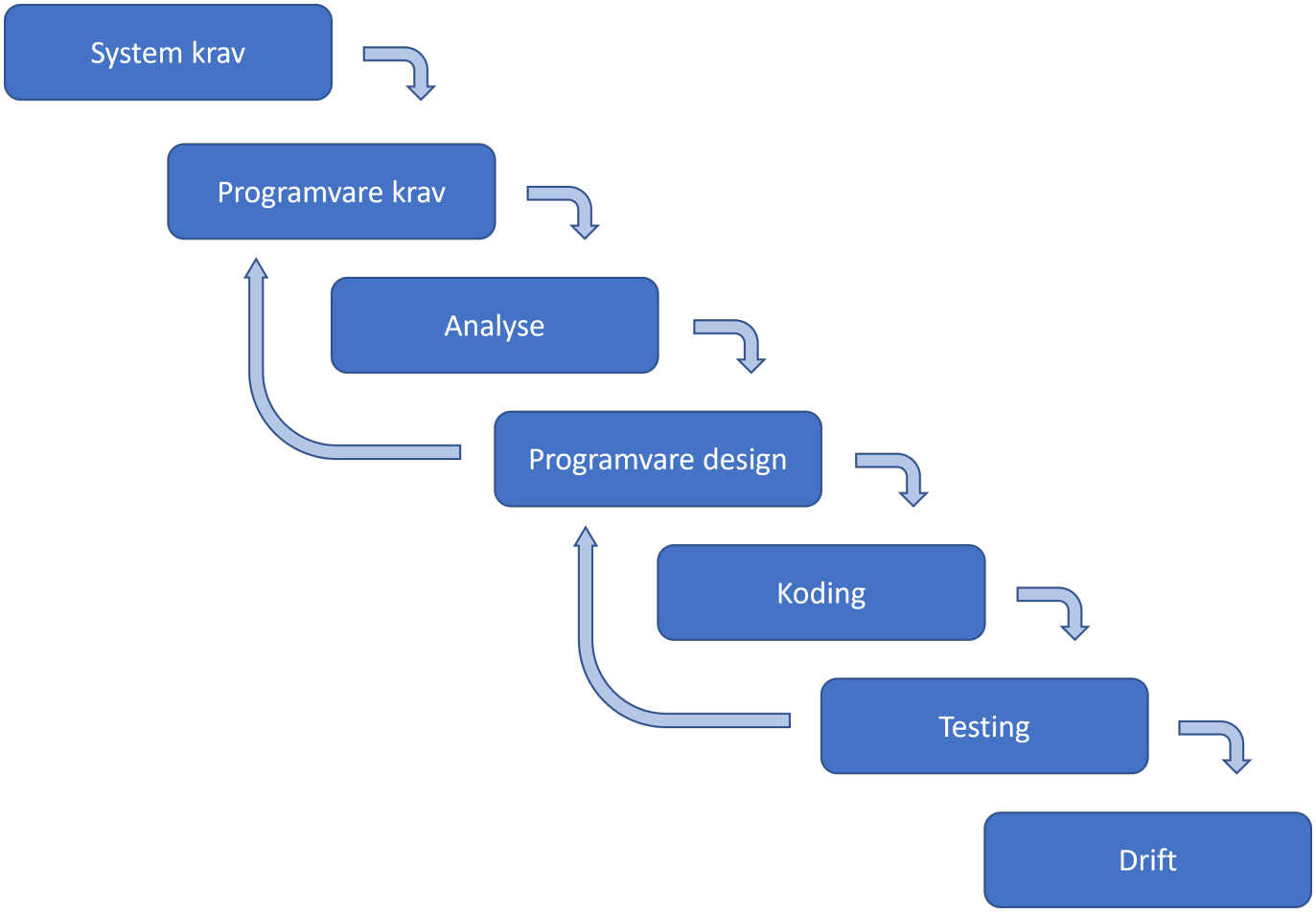


Figur 3.1: De sekvensielle fasene i fossefallsmodellen

De sekvensielle og ikke overlappende fasene i fossefallsmetoden sikrer at arbeidet i en fase er fullført før neste fase starter – eksempelvis så skal design-fasen være fullført før implementerings-fasen i prosjektet startes. Dette gjør at denne metoden vil egne seg godt i utviklingsprosjekter hvor kvalitetskontroll er viktig - dette på grunn av den intense bruken av planlegging og dokumentasjon (Munassar og Govardhan, 2010).

Winston W. Royce var den første som presenterte denne prosessen til bruk i forbindelse med programvareutvikling – selv om han ikke benyttet uttrykket "fossefall" i denne publikasjonen fra 1970 (Royce, 1970). Denne opprinnelige prosessen inneholdt fasene systemkrav, programvarekrav, analyse, design, koding, testing og drift. Royce mente også at denne prosessen hadde klare svakhetstrekk med tanke på at testing av programvaren bare forekom mot slutten av prosessen – noe som han beskrev som risikabelt og sårbart. For å eliminere denne risikoen introduserte han fem nye steg i denne iterative prosessen – hvor stegene omhandlet viktigheten med dokumentasjon i de forskjellige fasene i utviklingsprosessen. I tillegg til dette presiserte Royce at denne prosessen nødvendigvis ikke trenger å være fullstendig iterativ når den brukes i forbindel-

se med programvareutvikling – i praksis så kan det bli nødvendig å bevege seg motstrøms i prosessen, noe som vises i Royce endelige prosess som er vist i figur 3.2.



Figur 3.2: Royces endelige prosess

Fordelene med fossefallsmetoden er mange. For det første så er modellen enkel å forstå og ta i bruk (Awad, 2005), den er også veldig godt dokumentert og kan lett sammenliknes med andre liknende produksjonsprosesser innenfor andre

fagområder. Fossefallsmetoden inneholder klare prosjektfaser hvor resultatet i hver enkelt fase er relativt enkelt å forholde seg til for prosjektdeltakerne. Andre peker til at tid og ressurser som brukes på tidlige fase i programvaresyklusen kan redusere kostnaden i senere faser. For eksempel så er det totalt sett mye billigere å oppdage et problem i analyse-fasen i et prosjekt, enn at man oppdager det i test-fasen (McConnell, 1996). Foruten dette så er nok fossefallsmetoden mest kjent for sitt høye fokus på dokumentasjon i alle fasene i prosjektet. Eksempler på dette kan være dokumentasjon av krav eller design. Dette blir også sett på som en klar fordel i lengre prosjekter hvor prosjektdeltakere kan forlate prosjektet og bli erstattet med andre i løpet av prosjektperioden (Petersen m. fl., 2009).

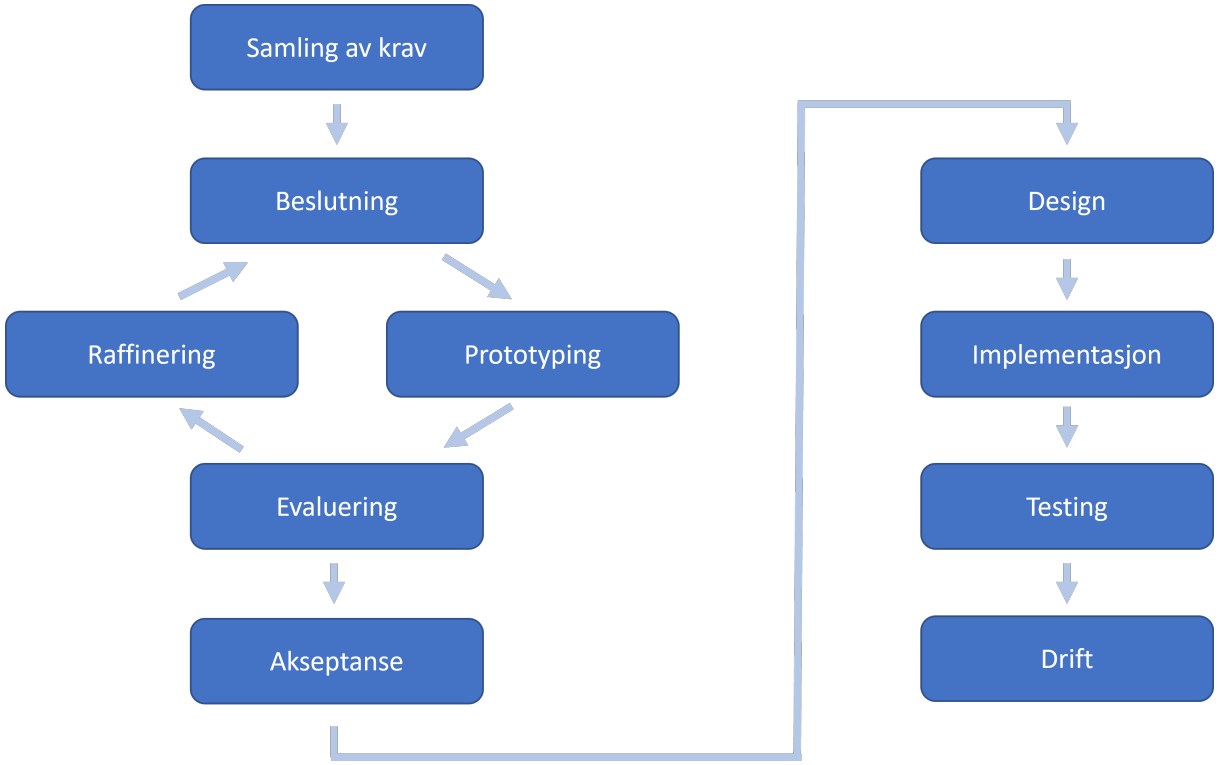
Fossefallsmetoden har også klare svakheter. I noen store systemutviklingsprosjekter som pågår over flere år, vil det kunne være en fare for at tidligere faser (eller deler av faser) i prosjektet blir utdatert før prosjektet er fullført – dette som følge av den rivende utviklingen på dette fagområdet de siste årene og at designere kanskje ikke er klar over fremtidige problemstillinger og krav som måtte dukke opp i løpet av prosjektperioden. Selve hensikten med fossefallsmetoden er å gjennomføre en sekvensiell prosess med gitte faser og da vil det være veldig vanskelig og kostbart å måtte gå tilbake og endre en tidligere fase (Alshamrani og Bahattab, 2015).

Flere påpeker også at kunder ikke nødvendigvis vet hva de egentlig vil ha eller ønsker med prosjektet – eventuelt at de ikke klarer å beskrive dette på en god nok måte. Dette i tillegg til kundenes begrensende mulighet til å se resultatet av prosjektet før det nærmer seg slutten av prosjektet gjør det vanskelig å forutsi om prosjekt faktisk blir vellykket eller ikke.

3.1.2 Prototype modellen

Det er relativt kjent at en av de største utfordringene ved programvare utvikling er at interessentene av et system ikke klarer å definere korrekt, konkret og detaljert informasjon om hva de egentlig ønsker. Dette vanskeliggjør arbeidet for designere og arkitekter som utarbeider produktspesifikasjoner og systemkrav med denne informasjonen som bakgrunnsmateriale i de tidlige fasene i et utviklingsprosjekt.

Prototype modellen ble derfor utviklet for å minimere problemet med begrensinger i forhåndsinformasjon til et systemutviklingsprosjekt. Modellen bygger på prinsippet med at systemutviklerne bygger en sterkt forenklet versjon av et system i de første fasene i prosjektet – dette for å kunne få tilbakemeldinger, innspill og kommentarer på dette samtidig som produktspesifikasjonen og systemkravene defineres og avgrenses (Muffatto, 2006). Figur 3.3 viser prototype modellen hvor sykluser med prototyping og evaluering gjentas til en eventuell akseptanse om å starte utviklingsprosjektet fattes.



Figur 3.3: Prototype modellen

Fordelene med prototype modellen er at brukeren får en følelse av systemet før det er ferdig utviklet og at brukermedvirkningen blir forbedret i forhold til prosjektgjennomføring med for eksempel fossefallsmodellen. Bruken av denne

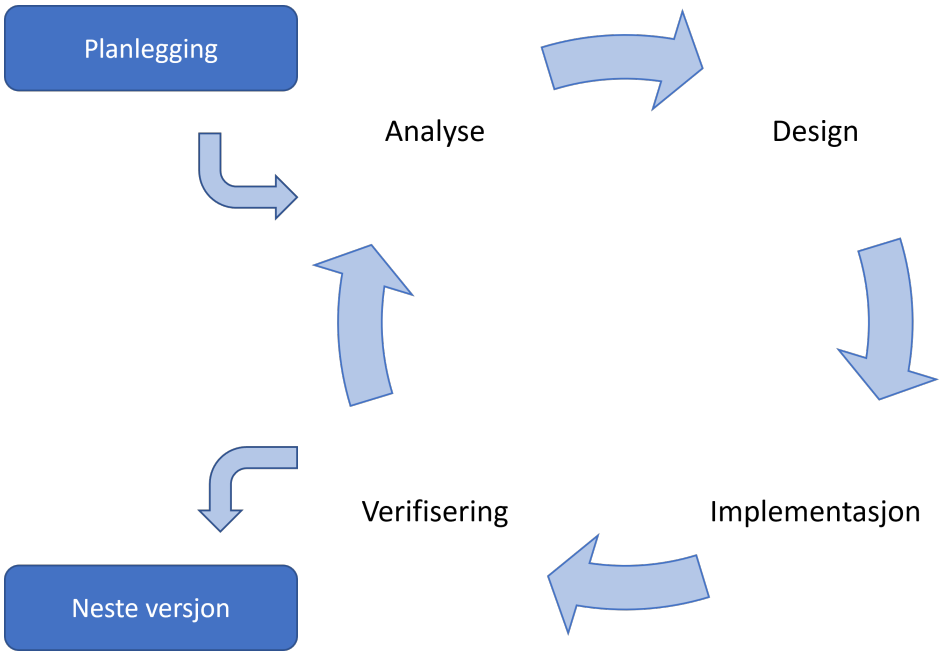
modellen øker også sannsynligheten for å produsere et system som kunden kan akseptere. Ulempene med prosjektgjennomføring etter denne modellen er at den øker sannsynligheten for et dårlig integrert og dokumentert system som kan være vanskelig å vedlikeholde over tid (Foster og Towle Jr, 2021).

3.1.3 Iterative og inkrementelle modeller

De iterative og inkrementelle utviklingsmodellene kombinerer elementer fra fossefallsmodellen med et iterativt design med inkrementell utvikling (Alshamrani og Bahattab, 2015). Denne iterative måten å utøve systemutvikling på ble ifølge Craig Larman og Victor R. Basili først beskrevet og anbefalt av F. W. Zurcher og B. Randell i sin rapport fra 1968 (Zurcher og Randell, 1968).

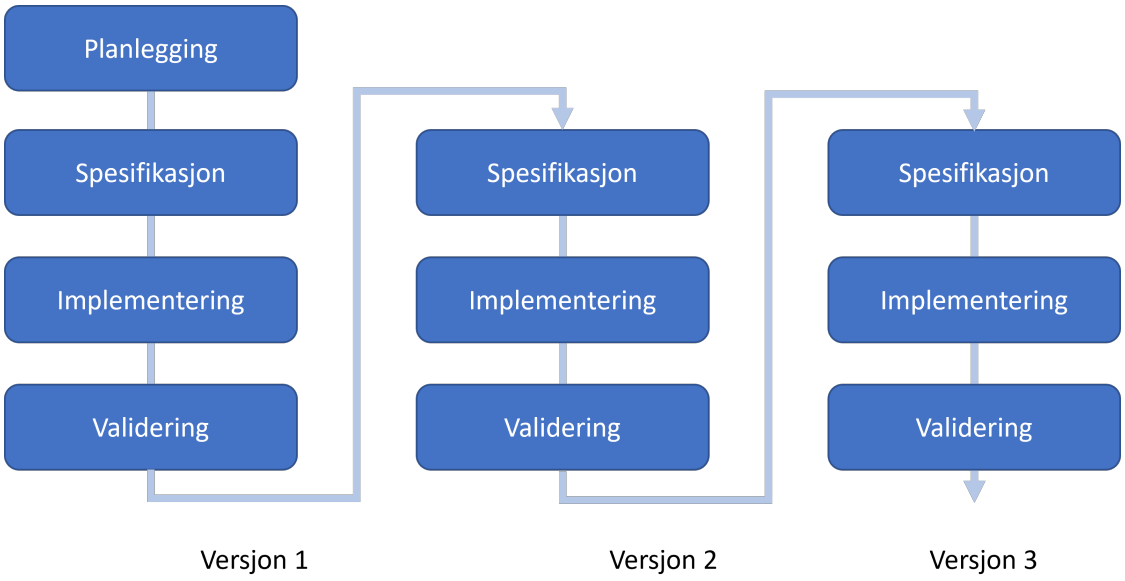
Ideen med disse metodene er å bryte ned prosjektet i flere gjentagende sykluser (iterative) og levere mindre deler om gangen (inkrementelle). Dette åpnet muligheten for mer konkret og jevnlig testing av systemet, og muliggjorde tilbakemeldinger og kommentarer på resultatet underveis i prosessen. Resultatet av en iterasjon i prosessen og et inkrement av systemet kan da også brukes som grunnlag for planleggingen av neste iterasjon.

Den iterative modellen produserer inkrementelle endringer på hele systemet – både kjernesystemet og tilhørende komponenter. Dette gjør at potensielt alle komponenter kan ha blitt endret fra en versjon til en annen. Slik vil hver iterasjon av denne prosessen lede utviklingsprosjektet videre mot målet hvor alle krav er oppfylt og all planlagt funksjonalitet er implementert (Alshamrani og Bahattab, 2015). Figur 3.4 viser en tradisjonell iterative prosess hvor hver enkelt iterasjon resulterer i en ny versjon av hele systemet og samtidig danner grunnlaget for planleggingen av neste iterasjon.



Figur 3.4: Den iterative utviklingsmodellen

Den inkrementelle modellen har mange likhetstrekk med den iterative modellen, men fokuserer på å fullføre en eller flere konkrete nye deler av systemet i hver enkelt iterasjon – i stedet for å forbedre alle parallelt som er tilfellet for den iterative modellen. Figur 3.5 viser den inkrementelle modellen og hvordan syklusene i denne modellen fører frem til neste versjon av systemet.



Figur 3.5: Den inkrementelle utviklingsmodellen

Colin Atkinson og Oliver Hummel mener også at denne måten å bryte opp et prosjekt i mange mindre små prosjekter som inkrementelt utvider et system med mer funksjonalitet vil gjøre det enklere å unngå problemer med integrasjoner mellom delkomponenter og forenkler den rigide dokumentasjonen fra fossefallsmetoden (Atkinson og Hummel, 2012).

Fordelene til de iterative og inkrementelle modellene er at man kan spre risiko i arbeidet over flere iterasjoner i stedet for å konsentrere seg om en stor utviklingsfase. Dette gjør også at man kan utvikle de mest kritiske komponentene eller delene i tidlige iterasjoner i prosjektet (Alshamrani og Bahattab, 2015). I disse modellene vektlegges det også av flere at man kan ta lærdom av tidligere iterasjoner og planlegge videre iterasjoner underveis i prosjektet – i tillegg til deres evne til å fremskaffe tidlige leveranser til kunder og interessenter.

Svakhetene til disse modellene er at det kreves god planlegging og design for å kunne dele opp prosjektet i avgrensende iterasjoner som til sammen skal resulterer i et fullt fungerende system etter siste iterasjon (Alshamrani og Bahattab,

2015). Andre peker på utfordringer som kan oppstå som følge av det sterke fokuset på ny funksjonalitet i hver iterasjon - og at dette i verste fall kan føre til at systemets arkitektur blir neglisjert som igjen kan føre til store vedlikeholdsproblemer over tid (Atkinson og Hummel, 2012).

3.2 Smidige utviklingsmetoder

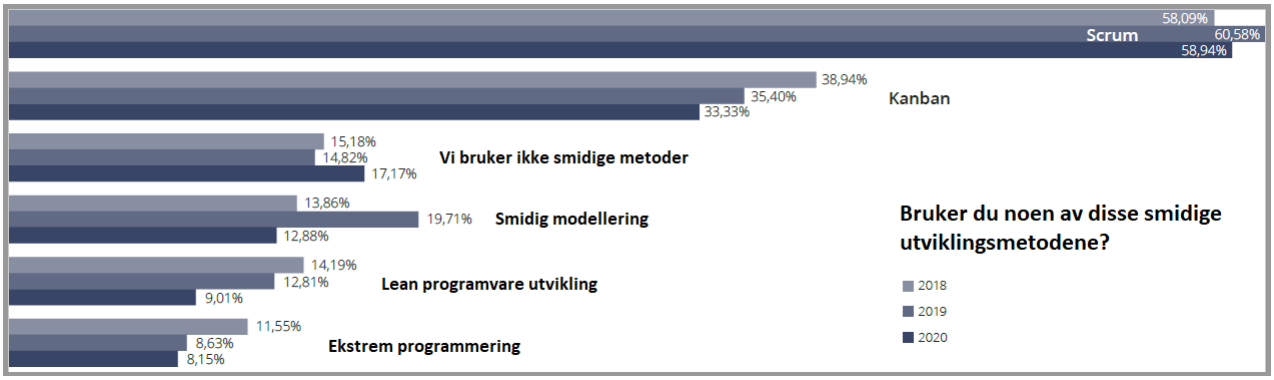
Siden slutten av 1990 -tallet har smidige utviklingsmetoder blitt mer og mer populært og er blitt en innarbeidet utviklingsstrategi for flere og flere bedrifter. Smidige utviklingsmetoder er en samlebetegnelse for flere utviklingsmetodikker som deler de samme prinsippene for god samhandling innen systemutviklingsprosjekter. Smidige utviklingsmetoder regnes som å være et konseptuelt rammeverk for programvareutvikling som starter med en planleggingsfase og videre fører til en leveransefase ved hjelp av iterative og inkrementelle interaksjoner gjennom hele prosjektperioden. Det opprinnelige målet for de smidige metodene er å redusere merarbeid i prosjektet med å håndtere endringer underveis i prosjektet uten å øke risikoen i prosjektet og måtte gjøre arbeid om igjen (Al-Saqqa m. fl., 2020). Smidige utviklingsmetoder innen programvareutvikling er blitt en reaksjon på de tradisjonelle måtene å utøve programvareutvikling på og fremhever behovet for en alternativ metode til fordel fra de andre tyngre og dokumentdrevne metodene som tradisjonelt har blitt brukt (Beck m. fl., 2001).

I begynnelsen av 2001, når de smidige metodene virkelig begynte å få rotfeste i programvareutviklingsbransjen, møttes representanter fra de forskjellige metodene i det kjente "Snowbird" -møtet i Utah. Resultatet av dette var *det smidige manifestet*. Dette manifestet inneholdt et sett med verdier og tolv prinsipper som setter søkelys på kundefokus, kontinuerlig læring, fleksibilitet og hyppige leveranser. Manifestets kjerneverdier ble definert som (Beck m. fl., 2001):

Individer og interaksjoner fremfor prosesser og verktøy
Fungerende programvare fremfor omfattende dokumentasjon
Kundesamarbeid fremfor kontraktsforhandling
Reagere på endring fremfor å holde seg til planen

I tiden etter at det smidige manifestet ble skrevet så har smidige utviklingsmetoder preget systemutviklingsbransjen i større og større grad. Figur 3.6 viser et

utdrag fra en statistikk over de mest brukte utviklingsmetodene de siste årene. Statistikken er hentet fra rapporten "State of software development" som er skrevet og publisert av programvare firmaet Coding Sans i 2020 (Wohlmuth, 2020). I undersøkelsen ble over 700 utviklere spurt om de benyttet seg av noen smidige utviklingsmetodikker (en eller flere).



Figur 3.6: Utdrag fra statistikk over bruk av smidige utviklingsmetodikker

Statistikken i figur 3.6 viser noe interessant – nemlig at stort sett alle de kjente smidige utviklingsmetodikkene synker i utbredelse, og at det blir flere som uttaler at de ikke bruker smidige utviklingsmetodikker.

3.2.1 Scrum

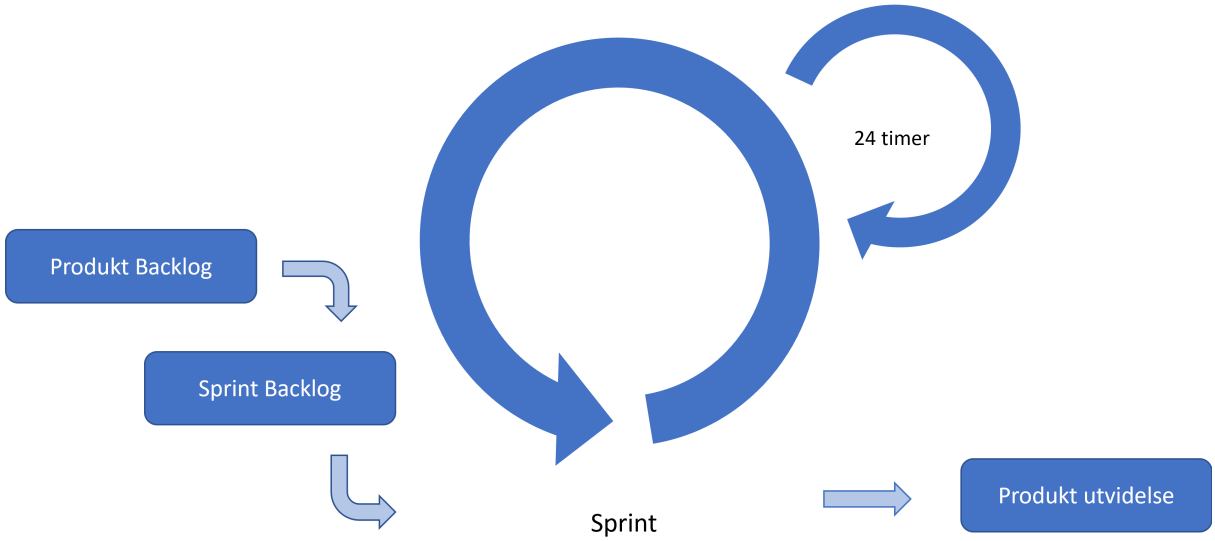
To av de som deltok på møtet som førte frem til *det smidige manifestet* var Ken Schwaber og Jeff Sutherland. Disse regnes som fedrene til Scrum etter at de i 1995 publiserte og presenterte et konsept de kalte *"Scrum Software Development Process"*. Navnet på denne utviklingsmodellen er hentet med inspirasjon fra ballspillet rugby hvor "Scrum" brukes som en betegnelse på en tett formasjon av spillere som fører spillet fremover på banen og hvor hver enkelt har gitte posisjoner og roller (Schwaber, 1997). Scrum ble utviklet som et svar på den såkalte fossefallsmetoden som krevde detaljert planlegging og kompleks arbeidsflyt som de mente var årsaken til en rekke prosjektkatastrofer på 1990 -tallet (Högstrand, 2019).

Scrum modellen krever at man arbeider i små, selvstendige og tverrfaglige grupper. Man arbeider i korte iterasjoner til fordel for langsiktig planlegging samtidig som man kontinuerlig kontrollerer resultatene underveis for å tilpasse den overordnede planen i prosjektet. Prosessen er fullstendig transparent og legger vekt på felles forståelse for mål og fremdrift (Högstrand, 2019).

Scrum regnes for å være et lettvekts rammeverk for å hjelpe systemutviklere, grupper og organisasjoner med å skape verdi igjennom adaptive løsninger for komplekse problemer. Kort fortalt så behøver denne metodikken en leder, "Scrum Master", som administrerer et miljø hvor (Schwaber og Sutherland, 2011):

1. Produkteier bestiller systemutviklingsarbeid via en "Product Backlog".
2. Scrum-gruppen gjør et utvalg og estimering av arbeidet i en "Sprint Backlog" som skal kunne gi produktet en verdiøkning.
3. Scrum-gruppen og øvrige interessenter verifiserer resultatene underveis og justerer før neste sprint.
4. Gjenta punkt 1-3 til "Product Backlog" er tom.

Figur 3.7 viser en grafisk fremstilling av denne utviklingsmetodikken hvor kortere iterasjoner (sprinter) gjentas til alt arbeid er utført.



Figur 3.7: Scrum-metodikken

De forskjellige elementene, rollene og ritualene i Scrum-metodikken beskrives videre – og er hentet fra Ken Schwaber og Jeff Sutherlands "The Scrum Guide" – hvor siste revisjon av denne guiden er fra 2020.

Scrum-gruppen er den fundamentale enheten i Scrum og utgjør en gruppe mennesker, inkludert "Scrum Master", produkteier og systemutviklere. Innenfor denne sammensatte gruppen av profesjonelle fagpersoner finnes det ingen andre undergrupper eller hierarkier – og alle skal være dedikerte og fokuserte på målsetningen med prosjektet. Gruppen skal også være tverrfaglig og kompetent nok til å kunne skape verdi innenfor hver iterasjon eller "Sprint". Gruppen skal også ha mandat til å være selvgående og skal konkret kunne bestemme hva som skal gjøres når og av hvem.

En "Sprint" består av et sett med ritualer eller arrangementer for å sikre fremdriften i prosjektet eller utviklingen. Alle disse hendelsene i løpet av Sprint-perioden er en mulighet til å inspisere og verifisere resultater og tilpasse arbeidet underveis i utføringen. Utformingen av disse hendelsene åpner for en gjennomsiktig og åpen prosess – hvor hensikten er å få minimert behovet for andre

møter. Disse hendelsene skal optimalt sett gjennomføres med faste mellomrom på samme sted for å minimere kompleksitet og bygge stabilitet. En Sprint starter med et planleggingsmøte hvor arbeidet som skal gjennomføres i neste sprint er i fokus. Det defineres et mål for sprinten i dette møtet og Scrum-gruppen tar inn relevante arbeidsoppgaver (brukerhistorier) fra "Product Backlog" som diskuteres og gjerne splittes opp i mindre oppgaver som estimeres ut ifra gruppens erfaring fra tidligere iterasjoner. En tommelfingerregel for tidsbruk i dette møtet er at det skal avsettes to timer for hver uke som sprinten skal vare, eksempelvis 4 timer for en sprint på 2 uker.

En annen viktig hendelse i Scrum-metodikken er det daglige Scrum møtet, eller *"Standup"* som det gjerne kalles. Dette møtet er et raskt daglig møte på omtrent 15 minutter hvor alle utviklerne presenterer status for deres oppgaver og eventuelle hindringer eller hendelser som har dukket opp det siste døgnet. Navnet *"Standup"* kommer av formatet på dette korte møte, hvor alle sammen skal reise seg fra sitt kontorsted og samles stående. Hensikten med at man absolutt skulle stå var å prøve å begrense tidsbruken i dette møtet til et minimum og bare rapportere det som kunne være av interesse for andre, samt en arena for raske avklaringer underveis i gjennomføringen.

Mot slutten av sprinten så gjennomføres "Sprint Review" og "Sprint Retrospective". Det førstnevnte møtet er en arena for demonstrasjon og inspeksjon av sprintens resultater hvor interessenter gjerne inviteres. Videre konkluderes det gjerne med hva som er neste steg på veien videre mot målsetningen i prosjektet. Estimert tidsbruk på dette møtet er omtrent en time per uke i sprinten. Hensikten med det siste møtet i sprinten ("Sprint Retrospective") er å planlegge justeringer for å øke kvalitet og effektivitet. Scrum-gruppen ser da sammen tilbake på forrige sprint for å se om de bør justere noe før neste sprint. Dette kan dreie seg om kompetanseheving, endringer i prosessen eller hvordan de kan unngå problemer som kan ha oppstått i forrige sprint. Dette møtet markerer avslutningen på en konkret sprint og estimert tidsbruk for dette møtet er i størrelsesorden tre timer for en sprint som varer i en måned.

Produkteieren i Scrum-gruppen er ansvarlig for å maksimere verdien av produktet som følge av det arbeidet som Scrum-gruppen utfører, og har også ho-

vedansvaret for administrasjonen av arbeidskøen, "Product Backlog". Denne administrasjonen omfatter definisjon av mål, konkretisering og prioritering av arbeidsoppgaver, samt ansvar for presentasjon og kommunikasjon av dette. Det er et viktig prinsipp at produkteierrollen er gitt til en bestemt person i organisasjonen – og at en eventuell endring av planlagt arbeid alltid gjøres med denne personens samtykke. Produkteieren representerer interessentene til produktet i prosjektet og skal derfor ha klare meninger om deres behov og krav.

I en Scrum gruppe skal det utnevnes en person som har rollen som "Scrum Master". Hans oppgave er å eie og drive Scrum-metodikken i gruppen. Dette inkluderer opplæring av Scrum teori og praksis til alle i gruppen og eventuelt andre i organisasjonen. "Scrum Master" er den som er ansvarlig for fremdrift og effektivitet i prosjektet og bruker Scrum-metodikken for å sikre og synliggjøre dette. "Scrum Master" er også den som har ansvaret for at overnevnte ritualer i Scrum-gruppen blir gjennomført og skal da kontinuerlig være behjelpelig med å eliminere eventuelle hindringer som måtte oppstå i løpet av gjennomføringen av prosjektet. Dette Scrum medlemmet er også en støttespiller for produkteieren gjennom hele prosessen og hjelper til med blant annet med definisjon av realistiske produktmål, planlegging av kommende arbeid og tilrettelegging for interessentsamarbeid. Ovenfor resten av organisasjonen skal denne rollen også være behjelpelig med opplæring i innføringsperioder av Scrum-metodikken og skal også prøve å fjerne avstand mellom Scrum-gruppen og andre interessenter for å strebe etter en transparent og åpen prosess.

Mange påpeker flere fordeler med å ta i bruk Scrum som en systematisk tilnærming til metodikk i sine systemutviklingsprosjekter. Kommunikasjonen – både den daglige mellom prosjektdeltakerne og jevnlige med de øvrige interessentene blir av flere påpekt som en av de klare styrkene til Scrum, i tillegg til den enkle tilpasningen til å håndtere uklare og skiftende krav (Despa, 2014; Mirza og Datta, 2019). Scrum setter kunden og kundens tilfredshet i fokus og legger så til rette for kvalitet i leveransene ut ifra et kundeperspektiv (Mirza og Datta, 2019) – noe som forsterkes av muligheten for å få delleveranser i sykluser for å få tilbakemeldinger og tilpasninger eller endringer underveis (Despa, 2014).

Flere påpeker også at Scrum er en teoretisk metodikk og at det finnes mange varianter av Scrum ute i bransjen (Hassani-Alaoui m. fl., 2020; Diebold m. fl., 2015). Det kan i mange tilfeller være nødvendig for å få en vellykket innføring av Scrum i forskjellige organisasjoner og prosjekter – men har også klare svakheter.

Størrelsen og rollesammensetningen av Scrum gruppen er en utfordring som flere har skrevet om. Selv om Scrum regnes for å være et lettvekts rammeverk for systemutvikling så kan det være ganske komplekst og vanskelig å innføre og gjennomføre i praksis. En gruppe på 3 til 9 personer er antatt å være et optimalt antall (Diebold m. fl., 2015) og at en gruppe på 10 eller flere bør vurderes å deles opp i flere grupper for å sikre god kommunikasjon i gruppen, men at dette krever et større fokus på koordinering mellom gruppene. En Scrum gruppe skal også være selvstendig i den grad det er mulig, noe som er utfordrende med tanke på rollesammensetning (Diebold m. fl., 2015). Produkteier og Scrum Master blir gjerne sett på som de viktigste rollene for å få Scrum-metodikken til å fungere optimalt i en organisasjon eller et prosjekt (Ereiz og Mušić, 2019; Diebold m. fl., 2015). Produkteier-rollen utpekes gjerne til en person som allerede er ansvarlig for eller kjenner til et fagområde og kundemasse og er avgjørende med tanke på kommunikasjon av målsetning og planlegging av arbeidsoppgaver i et Scrum-prosjekt (Diebold m. fl., 2015). Hvem som utpekes til å ta Scrum Master-rollen varierer veldig fra organisasjon til organisasjon hvor noen utpeker en utvikler i gruppen til å ta denne rollen, mens andre organisasjoner har mer dedikerte Scrum Mastere som deltar i flere prosjekter parallelt – ettersom denne rollen gjerne ikke er en fulltids jobb alene (Diebold m. fl., 2015). Utfordringene med dette kommer med tanke på planlegging av tid og parallelle prosjekter. Både produkteieren og Scrum Masteren er gjerne veldig opptatt med mange saker samtidig og dette kan skape utfordringer med gjennomføring. Ettersom disse to rollene regnes som de viktigste rollene så finnes det også eksempler på prosjekter som har brukt prosjektledere og linjeledere i slike roller - noe som kan føre til interessekonflikter og gå på bekostning av essensen med å jobbe effektivt og fokusere på produktet (Ereiz og Mušić, 2019).

Dokumentasjon eller mangel på dokumentasjon er en svakhet i Scrum-metodikken som nevnes av flere (Despa, 2014; Mirza og Datta, 2019). I større prosjekter over flere år er det naturlig at prosjektmedlemmer byttes ut over tid av forskjellige

årsaker. Mangelen på dokumentasjon og hyppige endringer og tilpasninger gjør det vanskeligere å ta inn nye medlemmer i et allerede pågående prosjekt.

Mange mener også at essensen i Scrum med stor vekt på funksjonalitet, kundebehov og hyppige endringer og leveranser blir for fokusert på funksjonalitet. Brukerperspektivet har spesielt blitt kritisert i Scrum – hvor man mener at Scrum fokuserer så mye på hvordan man fortest mulig får levert neste funksjonalitet at man ikke klarer å inkludere brukerne i tilstrekkelig grad (Cajander m. fl., 2013). I en slik smidig hverdag så er det rett og slett utfordrende å finne tid til brukervennlighetsaktiviteter og brukersentrert evaluering – i tillegg til å opprettholde oversikt over den totale brukeropplevelsen av produktet til Scrumgruppen (Cajander m. fl., 2013).

Programvarearkitekturen er et område som flere mener at ikke blir fokusert nok på i Scrum-metodikken. Det finnes flere definisjoner på programvarearkitektur, men jeg velger å støtte meg til Fritz Solms definisjon (Solms, 2012, s. 363):

> *"Programvarearkitektur er definert som programvareinfrastrukturen der applikasjonskomponenter som gir brukerfunksjonalitet kan spesifiseres, distribueres og kjøres".*

Selv om god programvarearkitektur anses som avgjørende for suksess i programvareprosjekter så blir ikke aktiviteter knyttet til arkitektur og programvarearkitekters rolle tatt i betraktning i Scrum-metodikken (Angelov m. fl., 2016). For å maksimere smidighet unngår eller minimerer utviklerne arkitektonisk planlegging. For lite planlegging kan imidlertid føre til en tilfeldig programvarearkitektur som på sikt kan føre til at gruppen får arkitekturproblemer og ikke klarer å levere ny verdi (Waterman m. fl., 2015). Som et tiltak for å minimere problemet med manglende eller utilstrekkelig programvarearkitektur som uansett vil endre seg i takt med de hyppige endringene i prosjekter er det foreslått å innføre en *"referanse arkitektur"* med mer generelle retningslinjer og standarder som kan fungere som et underlag for en programvarearkitektur for en Scrum-gruppe (Galster m. fl., 2017).

Scrum-metodikken fokuserer også veldig lite på vedlikehold av eksisterende programvare – selv om vedlikehold av programvare har en betydelig betydning

innen programvare utvikling. Dette på grunn av at det ofte er mye mer kostnadseffektivt og mindre tidkrevende å vedlikeholde et produkt til fordel for å erstatte det med et annet nytt produkt (Naz m. fl., 2016). Vedlikehold av allerede eksisterende programvare vil ofte være vanskeligere å planlegge enn tilfellet er hvis man skal starte på et nytt produkt. Behovet for ressurser vil sjelden være det samme over tid og man vet ofte ikke helt hva som kan oppstå av utfordringer eller behov for endringer frem i tid. Dette gjør at Scrum-metodikken vil være mindre passende i slike situasjoner og flere har introdusert modifiserte Scrum-metodikker for å bøte på dette problemet (Ashraf og Aftab, 2017; Naz m. fl., 2016). Som en løsning på dette problemet er det også foreslått et tillegg til Scrum-metodikken – hvor det innføres muligheten for å prioritere kortere krise-sprinter når nødsituasjoner oppstår. Innføringen av dette tillegget kompliserer prosessen, samtidig som at planlegging blir mer krevende ettersom man må planlegge og legge til rette for det ukjente (Rehman m. fl., 2018).

En annen generell utfordring med Scrum-metodikken er faren for at prosjektet blir for fokusert på å gjennomføre metodikken riktig – slik at dette går på bekostning av prosjektets virkelige målsetning. Dette har fått kallenavnet "Zombie-Scrum" - selv om noen omtaler Scrum som *"the best way to do software"* (Ozkan og Gök, 2021, s. 6), metoden som passer til alle prosjekter (Quick, 2023), eller metoden som kan redusere alt arbeid med 50% (Jakobsen og Sutherland, 2009).

3.2.2 Kanban

Kanban er en systemutviklingsmetode som har sin opprinnelse fra Japan og *"Lean manufacturing"* eller *"Just-in-time manufacturing"* som noen kaller denne metoden. Lean var opprinnelig utviklet for bilindustrien i Japan og ble fort veldig populær og er nå en av de raskest voksende metodikkene blant profesjonelle systemutviklere (Ahmad m. fl., 2013). Innen programvareutvikling så oppsto bruken av Kanban først i 2004 – når David J. Anderson assisterte en liten utviklingsgruppe hos Microsoft som ikke fungerte spesielt godt fra før (Anderson, 2010).

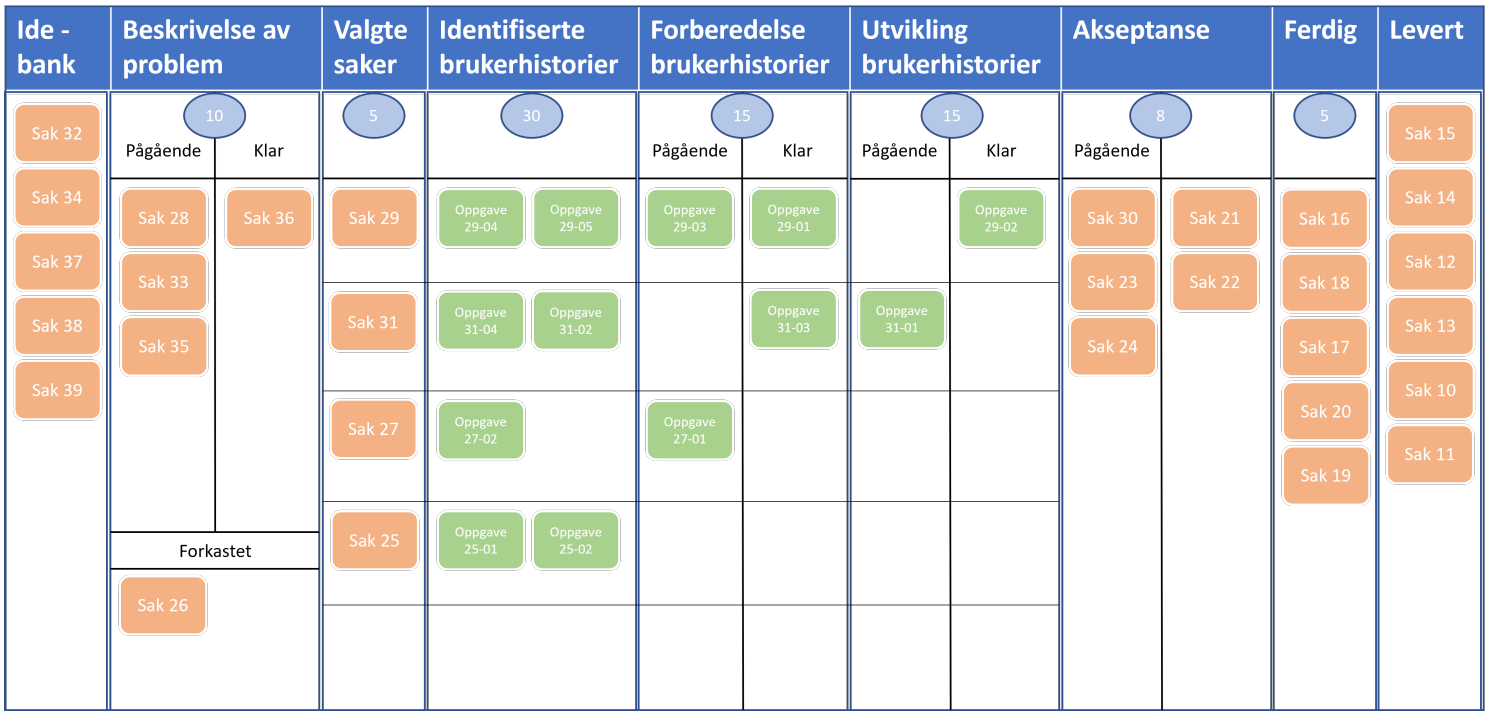
Essensen i Lean er å eliminere alt unødig arbeid, eller *"søppel"* som det kalles i metoden. For å ta metoden i bruk i forbindelse med programvareutvikling

ble det foreslått et sett med prinsipper for å håndtere søppelet (Poppendieck og Poppendieck, 2003):

1. **Bygge inn kvalitet**
2. **Skape kunnskap**
3. **Utsette avgjørelser**
4. **Levere raskt**
5. **Respektere mennesker**
6. **Optimalisere helheten**

Innen programvareutvikling så vil Kanban klassifisere alle saker som ikke produserer verdi for kunden som søppel. Dette inkluderer ekstra funksjoner, ekstra prosesser, delvis ferdigstilt arbeid og oppgavebytter (Poppendieck og Poppendieck, 2003).

Kanban metoden driver prosjektet frem ved å visualisere arbeidsflyten, begrense pågående arbeidet ("WIP – Work in progress") i hver fase i utviklingen og måler syklusene i tid (Ahmad m. fl., 2013). Samtidig så brukes gjerne en Kanban-tavle for å visualisere arbeidsflyten og status i prosjektet. Kanban-tavlen skaper en transparent programvareprosess og viser tildelt arbeid for hver utvikler, muliggjør kommunikasjon av tydelige prioriteringer og visualiserer flaskehalser. Hver fase i Kanban-tavlen tildeles en grense for hvor mange saker som skal kunne behandles samtidig ("WIP -limit") for å sikre konstant flyt av arbeidsoppgaver og at resultatene blir levert til kundene. De to viktigste nøkkelpunktene for å ta i bruk Kanban i programvareutviklingsprosjekter vil være fokus på flyt og eliminering av unødvendig arbeid (Ahmad m. fl., 2013). Figur 3.8 viser et eksempel på en typisk Kanban-tavle hvor de forskjellige fasene i et Kanban-prosjekt vises. Ideer beskrives som saker som igjen brytes ned til oppgaver som flyttes fra venstre mot høyre helt til saken er ferdig utviklet og er klar for akseptanse og endelig leveranse. Legg også merke til grensen ("WIP -limit") for hvor mange saker som samtidig kan befinne seg i hver fase som er markert med en lyseblå boble med hvitt tall.



Figur 3.8: En typisk Kanban tavle

De klare fordelene med å ta i bruk Kanban som utviklingsmetode innen programvareutvikling er at det først og fremst er en lett og intuitiv metode – uten spesielle roller eller ritualer. Kanban tavlen systematiserer og synliggjør arbeidsoppgavene og forbedrer arbeidsflyten med å identifisere og fjerne hindringer for effektivitet og fremdrift (Ahmad m. fl., 2018; Mirza og Datta, 2019) - og gjør det enklere å prioritere oppgaver underveis (Ahmad m. fl., 2018). Flere uttaler også at bruken av denne metoden vil føre til redusert gjennomføringstid ettersom unødig "søppel" elimineres for å maksimere verdien av prosjektet eller produktet (Mirza og Datta, 2019; Ahmad m. fl., 2018; Despa, 2014). Metoden vil også kunne føre til en mer motivert prosjektgruppe (Despa, 2014; Ahmad m. fl., 2018) med forbedret kommunikasjon og samarbeidsevne som til slutt vil ha en positiv innvirkning på kvalitet i leveransene og økt kundetilfredshet (Ahmad m. fl., 2018; Mirza og Datta, 2019).

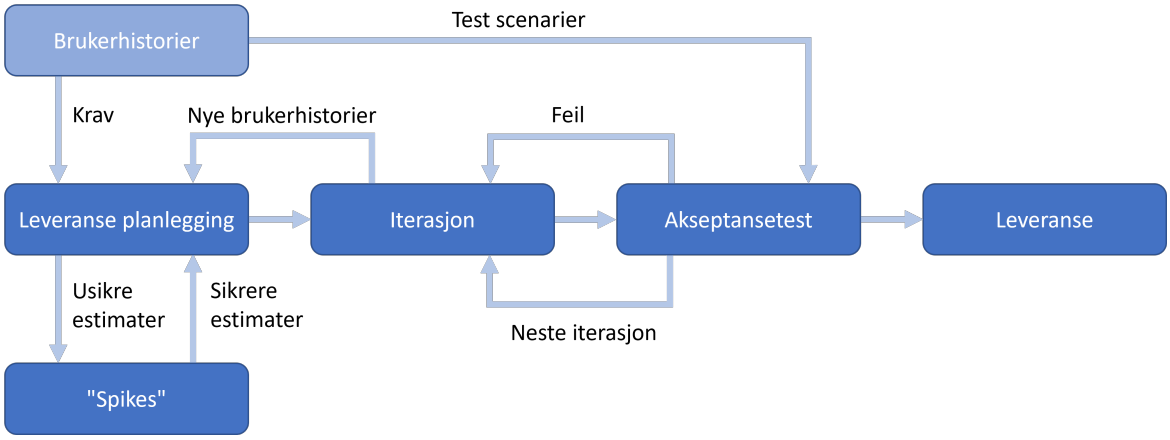
Noen mener også at bruk av Kanban metoden har klare svakheter ved bruk innen programvareutviklingsprosjekter. Gjennomføring av et effektivt Kanban prosjekt er svært avhengig av individuelle og kompetente gruppemedlemmer og i tillegg kreves det minst et gruppemedlem med sterke forretningsanalyseferdigheter (Despa, 2014). Andre trekker frem utfordringer med kommunikasjon mellom prosjektet og de øvrige interessentene - som ledelse og kunde (Ahmad m. fl., 2018; Mirza og Datta, 2019).

3.2.3 Ekstrem programmering

Ekstrem programmering er en programvareutviklingsprosess så vel som en metodikk. Denne prosessen definerer hvem som gjør hva, når og hvordan – og inneholder prinsipper, teknikker og praksis for effektiv, repeterbar og forutsigbar utvikling av programvare (Dudziak, 2000). Ekstrem programmering ble oppfunnet av Kent Beck i slutten av 1990 -årene. I begynnelsen av dette årtusen sprakk Dot com -boblen og denne metoden ble derfor populær på grunn av dens evne til å senke utviklingstiden før produktet kan tilbys i markedet (*"time-to-market"*) (Bell, 2001). Kent Beck definerer selv denne programvareutviklingsmetodikken slik (Beck, 2000, s. xv):

> *"Ekstrem programmering er en lettvekts-metodikk for små til mellomstore team som utvikler programvare i møte med vage eller raskt skiftende krav"*

Selve prosessen i ekstrem programmering er en iterativ og inkrementell prosess hvor prosjektet deles opp i mange små mini-prosjekter og hvor hver iterasjon resulterer i en økning i funksjonalitet og en leveranse (Dudziak, 2000). Som i flere andre utviklingsprosesser så brukes brukerhistorier for å beskrive ønsker, krav og behov som brukes i planleggingsfasen. For å minske usikkerhet i estimater så utnyttes prototyping i planleggingsfasen – hvor man utvikler såkalte "Spikes" for å test ut om antakelser er gyldige. Figur 3.9 viser et eksempel på et typisk prosess i ekstrem programmering.



Figur 3.9: Prosessen i ekstrem programmering

Ekstrem programmering blir altså klassifisert som en lettvekts-metodikk med høy frihetsgrad og hvor høy produktivitet og høy toleransefaktor er sentrale tema (Dudziak, 2000). Jevnlig uformell kommunikasjon, manglende dokumentasjon, samt små og hyppige leveranser er typiske kjennetegn på denne metodikken. Det er også definert fire kjerneverdier og tolv praksis prinsipper i ekstrem programmering (Newkirk, 2002) og disse er listet under.

Kjerneverdier:

1. **Kommunikasjon**
2. **Enkelhet**
3. **Tilbakemeldinger**
4. **Mot**

Praksis prinsipper:

1. **Planlegging**
2. **Mindre leveranser**
3. **Metaforer**
4. **Enkelt design**

5. **Testing**
6. **Re-faktorisering**
7. **Par-programmering**
8. **Kollektivt ansvar**
9. **Kontinuerlig leveranser**
10. **40 timers uker**
11. **Kundeinvolvering**
12. **Kodestandarder**

Ekstrem programmering fungerer egentlig som en mal med føringer for å gjennomføre prosjekter innen systemutvikling, og fungerer også som et prosessrammeverk fordi det åpner for å skreddersy prosessen etter de spesifikke behovene til prosjektet, organisasjonen eller gruppen (Dudziak, 2000).

Den største fordelen med ekstrem programmering er nok prosessens evne til å produsere ny funksjonalitet raskt (Despa, 2014; Mirza og Datta, 2019). Det spesielle praksis prinsippet par-programmering blir også trukket frem som en styrke i denne prosessen på grunn av dets evne til å heve kvalitet og kommunikasjon (Dalalah, 2014), men er også omtalt som en svakhet av flere grunner. Par-programmering krever en bredere kompetanse i gruppen hvor minst to innehar tilsvarende kompetanse (Mirza og Datta, 2019), senker effektiviteten (Dalalah, 2014) og kan motvillig oppleves umotiverende for utviklerne (Despa, 2014). Flere er derimot enige om at den største svakheten med denne systemutviklingsprosessen er mangelen på dokumentasjon (Despa, 2014; Mirza og Datta, 2019) og ergo behovet for et godt kommunikasjonsverktøy (Dalalah, 2014).

3.3 Oppsummering og utvikling av proposisjoner

I dette kapittelet har et utvalg av de mest kjente utviklingsmetodene blitt presentert sammen med uttalte fordeler og ulemper med disse. I tabell 3.1 er viktigste fordelene og ulempene oppsummert for hver av de utvalgte utviklingsmetodene.

Utviklingsmetode	***Styrker***	***Svakheter***
Fossefallsmetoden	*Enkel å forstå og god dokumentasjon av arbeidet.*	*Lite fleksibel og vanskelig å gjøre tilpasninger underveis i prosessen.*
Prototype modellen	*Tidlige tilbakemeldinger fra brukerne fører til økt sannsynlighet for et akseptabelt produkt.*	*Økt sannsynlighet for dårlige integrerte produkter med mangelfull dokumentasjon.*
Iterative og inkrementelle modeller	*Risiko spres over flere iterasjoner og man kan ta lærdom av tidligere iterasjoner.*	*Krever god planlegging og design og kan føre til dårlig systemarkitektur.*
Scrum	*God kommunikasjon i teamet og med interessenter. Tilpasningsdyktig og økt endringsevne.*	*Person/rolle -avhengig, ressurskrevende med tanke på planlegging og prosess, økt sannsynlighet for dårlig dokumentasjon og kan føre til dårlig systemarkitektur.*
Kanban	*Økt fokus på prioriterte arbeidsoppgaver og økt effektivitet og fremdrift.*	*Avhengig av individuelle og kompetente enkeltpersoner, samt kommunikasjonsutfordringer.*
Ekstrem programmering	*Enkel, effektiv og tilpasningsdyktig prosess.*	*Manglende dokumentasjon og økt behov for gode kommunikasjonsverktøy.*

Tabell 3.1: Oppsummering av fordeler og ulemper med de forskjellige utviklingsmetodene

Med bakgrunn i litteraturen som er presentert i dette kapittelet og delspørsmål 2 og 3 av forskningsspørsmålet vil det avslutningsvis i dette kapittelet utvikles to tilhørende proposisjoner. Delspørsmål 2 ble i kapittel 1 *Introduksjon* definert slik:

> *Hvilke alternative utviklingsmetoder er mest utbredt i bransjen - og hva er egentlig kjennskapen og bruken av disse i bransjen i dag?*

Proposisjon 2: Kjennskap til ulike utviklingsmetodikker

For å belyse omfanget og ulikhetene av forskjellige utviklingsmetoder og metodikker innen styring og gjennomføring av systemutviklingsprosjekter ble det i kapittel 3 *Utviklingsmetodikker* presentert et utvalg av både tradisjonelle og nyere smidige utviklingsmetodikker. Fossefallsmodellen regnes for å være den eldste kjente utviklingsmetodikken innen programvareutvikling (Awad, 2005), men med tiden har mer iterative og inkrementelle modeller blitt mer populære, se ansnitt 3.1.3 *Iterative og inkrementelle modeller*. Siden slutten av 1990 -tallet har nyere smidige utviklingsmetoder preget systemutviklingsprosjekter i større og større grad, noe som ble forsterket med *det smidige manifestet* som ble utviklet i begynnelsen av 2001, se delkapittel 3.2 *Smidige utviklingsmetoder*.

Flere av kildene til dette kapittelet påpeker at økende omfang og kompleksitet i dagens systemutviklingsprosjekter krever et økt fokus på samarbeid og styring i form av bruk av mer strukturerte utviklingsmetoder og metodikker, samtidig som at de forskjellige utviklingsmetodene har ulike styrker og svakheter med tanke på gjennomføring og resultat. Dette kapitelet prøver derfor å skape en overordnet oversikt over styrker og svakheter med de forskjellige utviklingsmetodene. Delspørsmål 2 løfter problemstillingen om kunnskapen til tilgjengelige og utbredte utviklingsmetodikker faktisk er god nok i bransjen. Hvis man skal vurdere ulike utviklingsmetodikker for et konkret prosjekt så må man faktisk kjenne til metoden og helst være klar over hvilke styrker og svakheter dette valget kan føre med seg. Med dette som bakgrunn og for å kunne underbygge svaret på delspørsmål 2 av forskningsspørsmålet har jeg formulert proposisjon 2 på følgende måte:

Dagens systemutviklere har en relativt god kjennskap til de mest utbredte og kjente utviklingsmetodikkene som benyttes i bransjen i dag og de kan også uttale seg om styrker og svakheter med disse.

Delspørsmål 3 ble i kapittel 1 *Introduksjon* definert slik:

Hva kjennetegner et programvareutviklingsprosjekt der man bør velge "Scrum" som utviklingsmetode?

Proposisjon 3: Valg av utviklingsmetodikk i systemutviklingsprosjekter.

Det er fordeler og ulemper med det meste – slik er det også med forskjellige utviklingsmetoder. For hver av utviklingsmetodene som er presentert i dette kapittelet så er det også presentert noen uttalte styrker og svakheter eller fordeler og ulemper. Disse egenskapene kan være til nytte ved valg av ønsket utviklingsmetodikk for en bedrift eller et konkret prosjekt. For øvrig så er det ikke valget av utviklingsmetodikk alene som egentlig betyr noe for en bedrift eller gruppe, men effektene valget kan føre til – gjerne tett knyttet opp imot strategien til bedriften. Ambisjoner om økt effektivitet, leveransekraft og lønnsomhet - samt målsetninger om økt samarbeid og markedsposisjon er typiske mål for mange bedrifter i denne bransjen i dag. Delspørsmål 3 etterspør hva som kjennetegner et programvareutviklingsprosjekt hvor man bør velge den populære utviklingsmetodikken "Scrum", men før dette kan besvares så ville det også vært interessant å finne ut hvorfor organisasjoner i økende takt innfører allerede dokumenterte utviklingsmetodikker i sine utviklingsprosjekter, samt hva de ønsker å oppnå med dette. Med dette som bakgrunn og for å kunne underbygge svaret på delspørsmål 3 av forskningsspørsmålet har jeg derfor formulert proposisjon 3 på følgende måte:

Organisasjoner som utøver stadig mer kompleks systemutvikling etterstreber økt bruk av dokumenterte utviklingsmetodikker i sine systemutviklingsprosjekter for å oppnå ønsket effekt.

3.4 Teoretisk rammeverk

Med bakgrunn av litteraturen som er presentert i kapittel 2 og dette kapittelet presenteres det her et teoretisk rammeverk som senere vil ligge til grunn for analysen og diskusjonen i dette forskningsprosjektet. Det teoretiske rammeverket består av de tre proposisjonene som jeg har utviklet og presentert avslutningsvis i disse kapitlene, og sammen vil de til slutt legge grunnlaget for å kunne svare på forskningsspørsmålet:

Hva kjennetegner programvareutviklingsprosjekter hvor det ikke vil være hensiktsmessig å benytte "Scrum" som utviklingsmetode?

I tabell 3.2 er alle de tre proposisjonene oppsummert. Den kronologiske nummereringen av proposisjonene er tilpasset nummereringen på delspørsmålene - proposisjon 1 er utledet fra delspørsmål 1 og så videre.

Nummer	***Proposisjon***
1	*Egenskaper ved prosjektet eller forskjellige typer av systemutviklingsprosjekter påvirker arbeidsflyten i prosjektet og følgelig hvilken utviklingsmetodikk man burde velge for å kunne oppnå suksess.*
2	*Dagens systemutviklere har en relativt god kjennskap til de mest utbredte og kjente utviklingsmetodikkene som benyttes i bransjen i dag og de kan også uttale seg om styrker og svakheter med disse.*
3	*Organisasjoner som utøver stadig mer kompleks systemutvikling etterstreber økt bruk av dokumenterte utviklingsmetodikker i sine systemutviklingsprosjekter for å oppnå ønsket effekt.*

Tabell 3.2: Oppsummering av proposisjonene

Alle de tre proposisjonene er utledet ved å kombinere tilgjengelig litteratur på dette området med min egen kunnskap og erfaring om bruk av forskjellige utviklingsmetoder i praksis.

Kapittel 4

Metode

I et forskningsprosjekt som dette så må man ta stilling til hvilken forskningsstrategi man skal velge og hvilket forskingsdesign og forskningsmetode man bør bruke - for å prøve å finne svar på forskningsspørsmålet som definerer forskningsprosjektet. I dette kapitelet vil valget av strategi, design og metode for dette forskningsprosjektet bli presentert og forklart. Videre beskrives prosessen med å komme i kontakt med kandidatene som er intervjuet i dette forskningsprosjektet - i tillegg til selve intervjuprosessen. Til slutt vil kvaliteten på denne studien samt etiske hensyn bli vurdert – før jeg avslutter med mine egne personlige refleksjoner.

Med bakgrunn i forskningsspørsmålet til dette forskningsprosjektet - *"Hva kjennetegner programvareutviklingsprosjekter hvor det ikke vil være hensiktsmessig å benytte "Scrum" som utviklingsmetode?"* ble det etter hvert valgt en kvalitativ tilnærming med semistrukturerte intervjuer for å prøve å finne noen funn knyttet til denne problemstillingen. Årsakene til dette valget og veien som førte dit vil også bli forklart i dette kapittelet.

4.1 Forskningsstrategi

Forskningsstrategien til et forskningsprosjekt definerer den overordnede retningen for forskningen og inkluderer prosessen som forskningen utføres med. Ved å ta stilling til en kvantitativ eller en kvalitativ forskningsstrategi kan man

enklere definere forskningsdesign og forskningsmetode. Hovedforskjellen mellom kvantitativ og kvalitativ forskningsstrategi er at kvantitative forskning benytter målinger, mens kvalitative undersøkelser ikke gjør det (Bryman, 2016). Kvalitativ forskning refererer til ikke-tallfestbare egenskaper (*"mykdata"*), mens kvantitativ refererer til kvantitet eller tall – eller *"harddata"* (Larsen, 2007). Det kan være mye som påvirker valget av forskningsstrategi og faktorer som tilnærming, formål, egne forutsetninger tid og ressurser er eksempler på egenskaper som kan ha betydning for dette valget.

I dette forskningsprosjektet hadde jeg opprinnelig tenkt å velge en kvantitativ forskningsstrategi med utsendelse av spørreskjema til et stort antall personer i programvare bransjen. Etter samtaler og diskusjoner med min veileder ble jeg derimot overbevist om at dette medførte en stor risiko både med tanke på at jeg ikke nødvendigvis trengte å få et tilstrekkelig antall svar, og at det kunne bli vanskelig å analysere disse dataene i etterkant med tanke på faktisk å kunne finne noen svar. Jeg valgte derfor en tilnærming med en kvalitativ forskningsstrategi i stedet. Fordelen med å bruke en kvalitativ forskningsstrategi vil være at jeg får muligheten til å gå i dybden til noen utvalgte kandidater i programvare bransjen – gjerne med forskjellige roller og erfaring.

4.2 Forskningsdesign

Et forskningsdesign er overbygningen som strukturer bruken av de valgte forskningsmetodene og hovedsakelig finnes det fem ulike typer forskningsdesign: eksperimentell, tverrsnitt, langsgående, case-studie og komparativ design (Clark m. fl., 2021). I dette forskningsprosjektet er det valgt å foreta en case-studie – så i dette avsnittet vil forskningsdesignet tilhørende forskningsprosjektet til denne masteroppgaven bli forklart. Dette inkluderer også styrker og svakheter med dette forskningsdesignet.

En case-studie forventes å omfavne kompleksiteten til en enkelt sak – eller "case" (Stake, 1995) og omfatter en detaljert og intensiv analyse av denne saken (Clark m. fl., 2021). Alan Bryman med flere uttaler også at en case-studie også er knyttet til et fellesskap eller en organisasjon – og i dette tilfellet er fellesskapet i forskningsprosjektet programvareutviklingsbransjen og organisasjonene

som opererer i denne bransjen. Felles for denne bransjen i dette tilfellet er at de utøver programvareutvikling i en eller annen form - og dette forskningsprosjektet er avgrenset av metoder og metodikker som de benytter seg av i denne forbindelse.

Det er mange som har definert forskjellige typer av casestudier og Arend Lijphart klassifiserer casestudiene innenfor kategoriene rene teoretiske, forklarende, hypotese-skapende, avvikende og teoribekreftende (Bennett, 2004). Harry Eckstein har presentert en tilsvarende klassifisering hvor han grupperer casestudier innenfor kategoriene ideografiske, disiplinerte, heuristiske og avgjørende (mest sannsynlige eller minst sannsynlige tilfeller) (Bennett, 2004). Sett bort fra de to først typene til begge to, som kan ses på som rene teoretiske casestudier så har de andre et forklarende eller teoribyggende formål. Forklarende eller disiplinerte casestudier bruker teoretiske variabler for å gi forklaringer på historiske tilfeller. Hypotese-skapende eller heuristiske casestudier søker å undersøke nye hypoteser induktivt, mens avvikende casestudier brukes til å finne avvik mellom eksisterende teorier og praksis ved å analysere bestemte tilfeller. Innenfor disse klassifiseringene kan vi plassere dette forskningsprosjektet både som en forklarende og avvikende casestudie.

Med tanke på om casestudiet er deduktivt eller induktivt – så er det i dette tilfellet valgt en abduktiv tilnæring. Den abduktive tilnærmingen starter med empirien (som ved induksjon) men hvor teorier og perspektiver spiller inn i forkant eller i løpet av forskningsprosessen (Tjora, 2012).

4.3 Forskningsmetode – kvalitative intervjuer

Forskningsmetoden forklarer bare hvilken teknikk som blir benyttet for å samle bakgrunnsinformasjonen til forskningsprosjektet (Clark m. fl., 2021). For dette forskningsprosjektet er det valgt å gjennomføre en kvalitativ undersøkelse og utføre kvalitative intervjuer. Når man gjennomfører kvalitative intervjuer så legges det veldig stor vekt på intervjuobjektets syn og refleksjoner rundt temaet som er i søkelyset, noe som er hensikten med denne undersøkelsen. Sakt på en annen måte så er det ikke så interessant å høre hva trender, bransjen eller en stor organisasjon mener – jeg er på utkikk etter den enkeltes oppfatning

av temaet. Kvalitative intervjuer vil uansett være mer personavhengige og noe mindre strukturerte enn hva for eksempel kvantitative undersøkelser vil fremstå (Clark m. fl., 2021).

Det finnes hovedsakelig to forskjellige typer av kvalitative intervjuer og disse klassifiseres enten som ustrukturerte intervjuer eller semistrukturerte intervjuer (Clark m. fl., 2021). I dette forskningsprosjektet er det valgt å gjennomføre semistrukturerte intervjuer hvor det på forhånd er definert en intervjuguide med spørsmål og oppfølgingsspørsmål som brukes som en tiltenkt kjøreplan for gjennomføring av intervjuet. Fordelene med å planlegge et semikonstruert intervju er at alle intervjuobjektene får mulighet til å svare på tilsvarende de samme spørsmålene – uavhengig av situasjon, rolle eller andre parametere, og at selve gjennomføringen av hvert enkelt intervju kan tilpasses situasjonen og intervjuobjektet. Selvfølgelig vil ingen svar i en kvalitativ intervjuprosess bli identiske, men bruken av en fast forhåndsdefinert intervjuguide vil muligens forenkle prosessen med å sammenligne intervjuobjektenes svar innenfor hvert hovedtema og spørsmål. Ulempene med å foreta semistrukturerte intervjuer er at det er veldig lett at samtalen sporer av noe som kan få konsekvenser for sammenlikning av resultatene - den som intervjuer må derfor ha god kontroll på hvilke kvalifikasjoner og erfaringer man faktisk vil vite mer om.

4.4 Datainnsamling

I dette avsnittet vil det forklares hvordan bakgrunnsinformasjonen til dette forskningsprosjektet ble innsamlet – dette innebærer hvordan intervjuobjekter ble identifisert, forsøkt kontaktet og hvordan endelig intervjuprosess ble gjennomført for de som sa seg villig til å bidra til dette forskningsprosjektet. Intervjuobjektene er anonymisert med tanke på navn og tilhørighet, men er kategorisert med tanke på rolle og ansiennitet i organisasjonen de er en del av i dag. For ytterligere detaljer knyttet til case-bedriften i dette forskningsprosjektet se delkapittel 5.1 *Presentasjon av case-bedrift.*

4.4.1 Samling og valg av kandidater

Fra mitt eget ståsted i skrivende stund og det faktum at jeg allerede er en del av en betydelig organisasjon sett i forhold til norske forhold i programvareutviklingsbransjen, så var jeg fra starten av ganske tydelig på at jeg først og fremst ønsket å gjennomføre mitt forskningsprosjekt utenfor min egen organisasjon. Årsakene til dette er først og fremst at en beslutning om valg av utviklingsmetodikk for våre utviklingsprosjekter allerede er tatt – før jeg var en del av denne organisasjonen, og at jeg er redd for at resultatene av en intern intervjuprosess ville blitt miljøpåvirket av dette faktum. Et alternativ var selvfølgelig å benytte mitt eget nettverk utenfor egen organisasjon for å rekruttere mulige intervjukandidater – men igjen ville dette kunne begrense utfallsrommet innenfor min egen deltakelse i prosjekter som jeg selv har deltatt i og eventuelle andre forbindelser jeg måtte ha i forbindelse med tidligere fullførte studier, kurs og liknende. Strategien min for samling og valg av informanter til dette forskningsprosjektet ble derfor å prøve å finne en case-bedrift innen programvareutviklingsbransjen hvor jeg kunne kontakte et passende utvalg av relevante informanter. For å starte dette arbeidet begynte jeg først å forhøre meg med relevante kontaktpersoner innen forskningsmiljøet på dette fagområdet, uten at dette ikke førte helt frem til min målsetning. Etter hvert forsøkte jeg videre med å kontakte ledelsen i konkrete bedrifter innen programvarebransjen – dette førte frem og på andre forsøk hadde jeg funnet en passende bedrift for mitt forskningsprosjekt uten å måtte forholde meg til hverken direkte eller indirekte påvirkning fra begynnelsen av forskningsprosjektet. Ifølge Tjora så vil det derimot skje påvirkning begge veier i større eller mindre grad i løpet av et observasjonsstudie – men at dette ikke bør være et hinder for ikke å gjennomføre det (Tjora, 2012).

4.4.2 Utvelgelse av informanter

Når jeg hadde knyttet kontakt med ledelsen til case-bedriften så startet arbeidet med å identifisere relevante intervjukandidater innenfor denne organisasjonen. Overfor ledelsen i denne bedriften la jeg vekt på at jeg gjerne ønsket tilgang til et representativt utvalg av intervjukandidater med litt ulik fartstid og ansiennitet i organisasjonen – og gjerne med litt forskjellige roller. Tjora beskriver dette som å søke etter subgrupper av mulige informanter for å kunne finne variasjoner i informasjonen (Tjora, 2012). Ledelsen i bedriften var veldig behjelpelig i denne

fasen og foreslo til slutt to senior ressurser med lang utviklingserfaring, samt to andre systemutvikler ressurser. Til slutt ble det gjennomført samtaler med alle disse fire systemutviklerne, samt en oppsummerende samtale med lederen for systemutviklerne i denne organisasjonen. Intervjukandidatene er oppsummert i tabell 4.1

Rolle	**Ansiennitet**
Utviklingsleder	Mer enn 15 år
Senior systemutvikler	Mer enn 15 år
Senior systemutvikler	Mer enn 15 år
Systemutvikler	Mindre enn 15 år
Systemutvikler	Mindre enn 15 år

Tabell 4.1: Informantene i forskningsprosjektet

4.4.3 Utvikling av intervjuguiden.

Utviklingen av intervjuguiden ble gjennomført relativt strukturert først med bakgrunn i problemstillingen til forskningsprosjektet og forskningsspørsmålet. Et viktig kriterium som lå til grunn for dette arbeidet var viktigheten med å få sikret de data som best mulig svarer på problemstillingen.

Intervjuguiden ble delt inn i to hovedtema *"Generelt om utviklingsmetoder* og *"Ulike typer av utviklingsprosjekter"*, og etter dette ble det utviklet et sett med spørsmål (med oppfølgingsspørsmål) under disse temaene. Alle disse spørsmålene ble underveis vektet opp mot relevansen til problemstillingen. Det var også viktig for meg å lage så åpne og nøytrale spørsmål som mulig – for å legge til rette for en god samtale og ikke påvirke kandidatenes svar. Dette arbeidet tok nok lengre tid enn forventet – og det ble laget en god del utkast og revisjoner før endelig versjon var klar. Veilederen fikk også et utkast av denne underveis i denne prosessen – som ga meg gode innspill på formulering og rekkefølge. Til slutt stilte jeg meg selv spørsmålet om teamene og spørsmålene var dekkende nok til å kunne samle så relevant data om problemstillingen som mulig.

I løpet av intervjuprosessen bestemte jeg i samråd med veileder at det kunne være interessant og nyttig å gjennomføre en oppsummerende samtale med lederen av utviklerne i case-bedriften. Det ville ikke være hensiktsmessig å gjennomføre dette med samme intervjuguide som de andre kandidatene – så derfor utviklet jeg en forenklet samtaleguide til dette formålet. Erfaringene med dette var veldig gode og gjorde at jeg kunne forankre og sikkerstille de funn som var observert i intervjuprosessen.

Før selve intervjuprosessen gjennomførte jeg to pilotintervju hvor jeg først intervjuet min kone for å verifisere flyt i intervjuet og til slutt en av mine egne kollegaer for å verifisere innhold og plausibilitet. Pilotintervjuer kan korrigere metodikkfeil før den opprinnelige studien (Thomas m. fl., 2022). Intervjuguiden ble tilpasset noe etter disse pilotintervjuene – men det ble ikke gjort store endringer på strukturen.

4.4.4 Intervjuprosessen

Intervjuprosessen ble gjennomført digitalt via Microsoft Teams. Hovedfordelen med bruk av et digitalt kommunikasjonsmiddel i et slikt forskningsprosjekt er mange og omfatter ingen reisetid, lite planlegging og administrasjon. Fra egen hverdag så vet jeg at tid er en begrenset ressurs og jeg tror det ville vært vanskeligere å gjennomføre intervjuene hvis digitale hjelpemidler ikke har blitt tatt i bruk. En digital gjennomføring av disse intervjuene muliggjorde også en enkel måte å ta opptak av intervjuene for å slippe å ta notater underveis og heller holde fokuset på samtalen. Opptakene ble senere brukt til transkribering av intervjuene og slettet når dette arbeidet var fullført. På forhånd hadde jeg beregnet varigheten på intervjuene til å være ca. 45 minutter - tre av intervjuene gikk noe over denne tiden, mens et av intervjuene ble gjennomført omtrent akkurat på tid.

Før intervjuprosessen hadde jeg forberedt meg så godt jeg kunne med tanke på bakgrunn, utdannelse og tilhørighet/rolle for hver av intervjuobjektene – ikke for å påvirke intervjuprosessen og dens utfall, men for å vite hvordan jeg skulle ordlegge meg i introduksjonen av intervjuet og hvor dypt jeg kunne gå i eventuelle oppfølgingsspørsmålene.

En av bakdelene med digitale intervjuer i forhold til fysiske intervjuer er selvfølgelig at det kan oppstå tekniske problemer både før og underveis i intervjuet (Clark m. fl., 2021). Heldigvis så opplevde jeg ingen større tekniske problemer underveis i gjennomføringen av mine intervjuer. Et par ganger ble samtalen noe forstyrret på grunn av svingninger i dataforbindelsen, men dette var så lite at det egentlig ikke påvirket flyten og samtalene i intervjuene.

4.4.5 Prosessering og analyse av innsamlet data

Når intervjuene var fullført, ble alle intervjuene transkribert og anonymisert. Videoopptakene av intervjuene ble slettet etter at transkriberingen var fullført. Selve transkriberingen av intervjuene var et langtekkelig arbeid, men samtidig en god måte å bli bedre kjent med de innsamlede data. Etter at transkriberingen av alle intervjuene var gjennomført startet arbeidet med kodingen av dataene. Dette innebærer å kategorisere innspill og svar i kategorier av temaet som ligger til grunn for intervjuet og se etter likheter og ulikheter i disse kategoriene. Bryman med flere beskriver dette som å bryte ned dataene i deler og sortere disse etter etiketter (Clark m. fl., 2021). I praksis så utførte jeg dette ved å definere fargekoder for de ulike kategoriene - og gruppere de innsamlede data på bakgrunn av disse fargekodede kategoriene. Dette innebar egentlig å se etter potensielle likheter og kategorisere utspill og utsagn på bakgrunn av dette. Tjora beskriver dette som en induktiv empiridrevet prosess hvor man prøver å slutte fra et enkelt tilfelle (eller et begrenset antall enkelttilfeller) til en generell regel (Tjora, 2012).

4.5 Vurderinger av kvaliteten på forskningen

Pålitelighet (relabilitet), gyldighet (validitet) og generaliserbarhet fungerer utmerket godt som kriterier for kvaliteten på kvalitativ forskning (Tjora, 2012). Pålitelighet handler om intern logikk gjennom hele forskningsprosjektet, mens gyldighet handler om en logisk sammenheng mellom prosjektets utforming, spørsmål og funn. Generaliserbarhet er knyttet til forskningens gyldighetsområde utover de enheter som faktisk er undersøkt. Videre har jeg vurdert forskningsprosjektet innenfor disse tre kriteriene for kvalitet.

4.5.1 Pålitelighet

Forskningsprosjektets pålitelighet forteller noe om hvor sannsynlig det er at funnene vil være gjeldende på andre tidspunkter (Clark m. fl., 2021). Med andre ord så sier pålitelighet noe om hvor sannsynlig det er at andre forskere ville funnet de samme funnene hvis de utførte den samme studien. Tjora vektlegger forskerens engasjement i teamet det forskes på i denne sammenheng, men er ganske klar på at forskerens kunnskap er en ressurs – men også at forskerens kunnskap bør brukes eksplisitt i forskningsprosjektets analyse (Tjora, 2012). I disse tilfellene er det altså viktig å gjøre rede for hvordan forskerens egen person kan prege forskningsarbeidet.

Prosessen og gjennomføringen av dette forskningsprosjektet bør være relativt god dokumentert i dette kapittelet omhandlende metodikk, noe som bør gjøre det mulig for en annen forsker å gjennomføre tilsvarende studie. Med hensyn på mitt eget engasjement i denne studien så er dette noe jeg bevisst har tenkt på igjennom hele forskningsprosjektet. Ettersom dette forskningsprosjektet er gjennomført som en del av mitt erfaringsbaserte masterprogram, så er det vanskelig å regne min egen person som helt nøytral og objektiv. Både problemstilling og forskningsspørsmål er basert på min egen kunnskap og erfaringer omhandlende temaet. Derfor har jeg forsøkt å holde mine egne erfaringer og meninger for meg selv – i den grad det faktisk er mulig i kvalitative studier, og valgt å presentere disse eksplisitt i kapittel 7 *Diskusjon* og kapittel 8 *Konklusjon* - samtidig med at forskningsspørsmålet blir besvart.

4.5.2 Gyldighet

Forskningsprosjektets gyldighet knyttes til spørsmålet om de svarene vi finner i forskningsprosjektet faktisk er svar på de spørsmål som stilles (Tjora, 2012). I dette forskningsprosjektet så ble intervjuene som sagt gjennomført digitalt ved hjelp av kommunikasjonsverktøyet Microsoft Teams. Dette gjorde det enkelt å ta opptak av intervjuene slik at jeg kunne fokusere på samtalen i stedet for å skrive referat underveis. Dette gjør meg også sikrere på at viktig og relevant informasjon ikke ble utelatt. Opptakene ble i etterkant transkribert så ordrett som mulig og nedskrevet på norsk (bokmål). Intervjukandidatene fikk også mulighet til å få tilsendt transkriberingen for verifisering – noe som enkelte av

informantene ønsket å benytte seg av.

4.5.3 Generaliserbarhet

Generaliserbarhet forteller noe om forskningsprosjektets gyldighet utover de tilfeller som faktisk er utforsket (Tjora, 2012). I denne sammenhengen snakkes det også mye om overførbarhet og om funnene også vil være gjeldende i andre kontekster (Clark m. fl., 2021). En åpenbar faktor i dette forskningsprosjektet er at case-studien er gjennomført i bare en bedrift. Ingen bedrifter eller prosjekter er identiske og både bransje, prosjekttyper og andre forutsetninger vil være påvirkende om funnene vil være gyldige eller overførbare til andre bedrifter – se for øvrig delkapittel 2.1 *Prosjekttyper* og 2.2 *Suksess i prosjekter* omhandlende ulike målsetninger med gjennomføring av prosjekter. For øvrig så mener jeg at funnene jeg har funnet gir meg god dekning for å påstå at disse vil være gjeldene for mange liknende bedrifter - som utvikler og forvalter sin egen programvare som tilgjengeliggjøres i markedet som produkter.

4.6 Etiske vurderinger

I intervjuene til dette forskningsprosjektet ble det gjennomført digitale opptak. Disse opptakene ble gjort med godkjenning av NSD *Norsk senter for forskningsdata*. Forskningsprosjektet ble meldt til NSD den 7. november 2022 via deres elektroniske meldeskjema og ble godkjent den 2. desember 2022 med lovlig grunnlag om samtykke (Personvernforordningen art. 6 nr. 1 bokstav a) og følgende konklusjon:

> *"Behandlingen av personopplysningene er lovlig så fremt den gjennomføres som oppgitt i meldeskjemaet. Det lovlige grunnlaget gjelder til 01.09.2023."*

I møteinnkallingene til intervjuene ble det vedlagt et informasjonsbrev hvor formålet med forskningsprosjektet, plan for gjennomføringen av intervjuene og ikke minst rettighetene til informantene var beskrevet. Brevet informerte også om mine planer om å foreta opptak av intervjuet og en begrunnelse for hvorfor dette var tilfellet. I starten av alle intervjuene (før opptak ble startet) ble denne informasjonen gjentatt for å sikre at informantene var klar over sine rettigheter og at samtykke til opptak av samtalen ble innhentet. Dette informasjonsbrevet er vedlagt som vedlegg I *Informasjonsbrev* i denne rapporten.

Når intervjuene var gjennomført og transkribert ble opptakene slettet – noe som var et av kravene fra NSD. Etter dette var alle de empiriske dataene i dette forskningsprosjektet helt anonymisert, kun gruppert etter rolle og ansiennitet som beskrevet i avsnitt 4.4.2 *Utvelgelse av informanter*.

4.7 Personlige refleksjoner

I dette delkapitlet vil jeg presentere mine egne erfaringer med gjennomføring av den kvalitative case-studien i dette forskningsprosjektet.

Som nevnt i delkapittel 4.1 *Forskningsstrategi* hadde jeg i utgangspunktet forestilt meg å gjennomføre en kvalitativ studie i dette forskningsprosjektet. En av hovedårsakene til dette var at jeg vet at ressurser innen systemutviklingsbransjen ofte er travle personer som gjerne jobber under et sterkt tidspress med korte leveringsfrister. Dette gjorde at jeg var redd for at arbeidet med rekruttering av informanter kunne bli vanskelig og tidskrevende. Rekrutteringsprosessen ble derfor prioritert høyt i arbeidet etter at godkjenningen for dette arbeidet ble mottatt fra NSD tidlig i desember 2022 – og gikk som fryktet litt tregt i startfasen, se avsnitt 4.4.1 *Samling og valg av kandidater*. Dette arbeidet løsnet veldig når jeg tilfeldigvis kom i kontakt og fikk presentert formålet med dette forskningsprosjektet for ledelsen til den bedriften som til slutt endte opp med å bli min case-bedrift. For å øke interessen for dette forskningsprosjektet hadde jeg da også gjort innkjøp av en trådløs høyttaler til en verdi av kr. 2690,- som ble overlevert en heldig informant etter trekning som ble foretatt i case-bedriftens lokaler etter at alle intervjuene var gjennomført. Resultatet av denne prosessen gjorde at jeg i løpet av februar 2023 hadde gjennomført alle de planlagte intervjuene – etter noen flyttinger av avtaler på grunn av sykdom og andre årsaker, og at jeg kunne erkjenne at rekruteringen av informanter og gjennomføringen av intervjuene hadde gått etter planen og kanskje bedre enn jeg først hadde fryktet.

I ettertid så har jeg også tenkt at jeg hadde en del flaks når jeg så tidlig kom i kontakt med bedriften som til slutt ble min case-bedrift, ettersom denne bedriften tilfeldigvis var en endringsprosess akkurat med tanke på arbeidsmetodikk og utviklingsprosesser. Dette kan nok ha vært en viktig faktor både for å skape

interesse for problemstillingen i mitt forskningsprosjekt og rekruteringen av relevante intervju kandidater.

Selve intervjuene ble gjennomført som semistrukturerte intervjuer med en forhåndsdefinert intervjuguide som er vedlagt som vedlegg II *Intervjuguide* i denne rapporten. Erfaringene mine fra gjennomføringen av disse intervjuene var at ingen av intervjuene ble helt slik jeg på forhånd hadde planlagt eller sett for meg. Når informantene begynte å snakke så berørte de gjerne temaer som jeg hadde tenkt å spørre om lenge før jeg fikk stilt selve spørsmålene og flere av oppfølgingsspørsmålene ble derfor overflødige. Tilsvarende gjaldt også for flere innspill og svar som var høyst relevante for temaet, men som jeg ikke direkte spurte om i spørsmålene. Alt i alt så ble dette veldig gode og interessante samtaler hvor jeg mot slutten også informerte litt om mine egne erfaringer omhandlende de forskjellige temaene. Til slutt gjennomførte jeg en oppsummerende samtale med lederen for systemutviklerne i case-bedriften. Temaet og innholdet i denne samtalen var på forhånd definert i en samtaleguide som er vedlagt som vedlegg III *Samtaleguide*.

Kapittel 5

Empiriske funn

I dette kapittelet så presenteres mine mest relevante empiriske funn. Funnene er gruppert i to hovedtemaer: generelt om utviklingsmetoder og ulike typer av systemutviklingsprosjekter. Denne grupperingen gjenspeiler temaene som ble diskutert i intervjuene, men er dels overlappende på noen områder. Avsnittene under disse hovedtemaene i dette kapittelet gjenspeiler også kategoriseringen som ble brukt under kodingen av de transkriberte intervjuene. Funnene presenteres her så nøytralt som mulig med minst mulig egne refleksjoner. Funnene vil videre bli analysert i kapittel 6 *Analyse*.

Før funnene presenteres vil case-bedriften hvor det empiriske grunnlaget er generert bli presentert. Dette for å gi en bedre innsikt i bedriftens nåsituasjon og at det skal bli lettere å forholde seg til funnene.

5.1 Presentasjon av case-bedrift

Det empiriske grunnlaget til dette forskningsprosjektet er generert gjennom kvalitative intervjuer i en middels stor norsk bedrift som driver med programvareutvikling. Bedriften har i underkant av 30 ansatte, hvor litt over halvparten av disse er ansatt som systemutviklere. Bedriften utvikler sine egne produkter (programvare) som benyttes av andre bedrifter knyttet til et konkret forretningsområde. Bedriften er organisert relativt tradisjonelt med egen avdeling for systemutvikling med tilhørende leder for denne avdelingen.

For å beholde anonymiteten til denne case-bedriften og bedriftens ansatte, samt å beholde funnene i dette forskningsprosjektet konfidensielt så vil det ikke være hensiktsmessig å beskrive denne bedriften ytterligere. For nærmere beskrivelse av informantene som deltok i de kvalitative intervjuene så vises det til 4.4.2 *Utvelgelse av informanter.*

5.2 Generelt om utviklingsmetoder

"Valg av utviklingsmetodikk for systemutvikling er som religion hos mange systemutviklere." – utviklingsleder

Bedriften har siden midten av 1990 -tallet utviklet sine produkter som har fått en suksess i markedet innenfor det konkrete forretningsområdet. Produktene er tilpasset over tid og ny funksjonalitet er utviklet etter hvert som markedet har utviklet seg og nye ønsker og krav har dukket opp. De siste årene har krav om økt brukervennlighet og tilgjengelighet gjort at bedriften er inne i en endringsfase hvor eksisterende produkter også gjøres tilgjengelig som lettere nettapplikasjoner for mer mobil bruk. Dette har ført til at bedriften nå utvikler nyere nettapplikasjoner samtidig som de vedlikeholder og forvalter de eksisterende produktene som har vært i markedet i mange år. Strategien til bedriften er at de nye nettapplikasjonene skal overta markedsposisjonen til de allerede eksisterende produktene på sikt, men at dette må gjøres i parallell ettersom dette kommer til å ta flere år å gjennomføre. Denne endringsprosessen har gjort at antallet systemutviklere har økt og at det har blitt ansatt flere nye systemutviklere de seneste årene. Økt kompleksitet og antall systemutviklere har videre ført til at bedriften nå har besluttet å innføre Scrum som utviklingsmetodikk i sine systemutviklingsprosjekter – dette til fordel for en egentilpasset inkrementell arbeidsmetodikk som hadde mange likhetstrekk med utviklingsmetoden Kanban.

5.2.1 Kjennskap til utviklingsmetodikken Scrum

Som et ledd i beslutningen om å innføre Scrum som utviklingsmetode i systemutviklingen i bedriften har alle systemutviklere nå gjennomført grunnkurs i Scrum, samt at noen utvalgte nøkkelpersoner også har gjennomført Scrum Master kurs. Flere av informantene nevnte for øvrig at hadde lært om forskjellige

utviklingsmetodikker igjennom skole og utdanning.

"Vi hadde et prosjektfag på skolen hvor vi skulle lære oss å jobbe med smidige metoder. " – systemutvikler

På grunn av nevnte kursing var kjennskapen til Scrum relativt stor hos informantene som ble intervjuet, samtidig som at flere påpekte at bruken av Scrum var relativt ny i bedriften og at de manglet erfaringer fra praktisk bruk av denne utviklingsmetoden. En av de yngre informantene utalte at bedriften nå hadde tilegnet seg en realativt passe god teoretisk forståelse for Scrum, men at denne var veldig fersk. Han utalte videre at i teorien så fremstår Scrum relativt overkommelig – men at dette ikke er det samme som å mestre det i praksis.

"Jeg har ikke så mye erfaring med Scrum, men vi har nylig startet med å ta i bruk Scrum – slik som det har blitt nå i moderne tid." – senior systemutvikler

To av informantene uttalte at han hadde noe erfaring med bruk av Scrum i jobbsammenheng. Denne erfaringen hadde de opparbeidet i tidligere jobber – før de startet i denne bedriften. Felles for begge to var at de snakket om en variant av Scrum eller at de plukket inn elementer fra Scrum i deres arbeidsprosesser.

"Det er ganske lenge siden, men tidligere har jeg jobbet med Scrum på mange forskjellige måter. Det er en del effekter med Scrum som er god." – senior systemutvikler

En av disse informantene uttalte blant annet at arbeidet ble delt opp i sprinter, men at møtene ikke ble gjennomført slik de er beskrevet i Scrum-metodikken.

"Der jeg jobbet sist så jobbet vi smidig – eller en variant av Scrum hvor vi plukket det som vi syntes hørtes bra ut. Vi hadde også sprinter på et vis, men ikke møtene." – systemutvikler

5.2.2 Kjennskap til alternative utviklingsmetodikker

Selv om Scrum naturlig nok var den utviklingsmetodikken som alle informantene kjente til og hadde en viss erfaring med, så var flere andre utviklingsmetodikker nevnt i intervjuene. Utviklingsmetodikken Kanban ble nevnt i alle intervjuene, noe som var naturlig ettersom arbeidsprosessen som tidligere hadde blitt brukt hadde flere likhetstrekk med denne metodikken.

"Vi har på en måte brukt Kanban her tidligere, men da er jeg egentlig usikker på om vi har brukt Kanban eller har vi hatt et Kanban-tavle..." – systemutvikler

Ettersom informantene var midt oppe i en endringsprosess akkurat med tanke på arbeidsmetodikk og prosess så var det flere som hadde ganske klare meninger angående valg av utviklingsmetodikk og valg av retning for deres systemutvikling.

"Kanban blir jo fort anarki, så Kanban er jo bare en unnskyldning for ikke å jobbe etter noen metodikk på mange måter. Du skal gjøre en oppgave ferdig og rett ut." – senior systemutvikler

Fossefallsmodellen ble nevnt av begge senior-informantene – og det var tydelig at de kjente til prinsippene og historikken med denne metoden. En av disse poengterte at metoden var utviklet som en metode for å gjennomføre spesialtilpassede prosjekter lenge før den ble benyttet i forbindelse med systemutvikling - og at den i så måte ikke egentlig var særlig tilpasset systemutviklingsprosjekter.

"For min del så startet det på ingeniørhøyskolen med at vi lærte om denne vannfallsmodellen og jeg syntes at det var så mange feil med den i forhold til hvordan programvare blir utviklet, eller kan utvikles – at jeg ble nesten kastet ut av klasserommet." – senior systemutvikler

Selv om begge senior-informantene ga uttrykk for at fossefallsmodellen var utdatert i dagens systemutviklingsprosjekter så mente den ene av disse at metoden fortsatt hadde noen fordeler sammenlignet med nyere og mer smidige metodikker. Informanten nevnte spesielt prosjektledelse og styring i denne sammenheng og mente at metoden hadde fordeler med tanke på forutsigbarhet, dokumentasjon og etterprøvbarhet som overgår andre metodikker.

"Fossefallsmodellen er jo den eldste og alle de som er prosjektledere ønsker seg tilbake til vannfallsmodellen." – senior systemutvikler

Den ene senior-informanten fortalte også at han tidligere hadde lest seg opp og tatt i bruk prinsipper fra Kent Becks "ekstrem programmering". Årsakene til at informanten hadde tatt i bruk denne metoden forklarte han med ren frustrasjon over fossefallsmodellen og at denne ikke var egnet systemutviklingsprosjekter. Han uttalte at "ekstrem programmering" hadde klare fordeler med tanke på

informasjonsflyt og utvikling av programvare og funksjonalitet i iterasjoner med jevnlig testing av mindre deler.

"Jeg var veldig frustrert over hvordan tingenes tilstand var og holdt på å lete etter noen andre som måtte ha fått liknende erfaring. Så kom jeg kom bort i Kent Beck og kompani og deres ekstrem programmering. Det var først da jeg følte at jeg hadde møtt noen som faktisk forstår. Jeg er ikke enig i alle små detaljer, men i all hovedsak så ... YES!" – senior systemutvikler

Felles for alle intervjuene var at alle informantene ga uttrykk for at det ville bli vanskelig å finne en utviklingsmetodikk som passet perfekt i alle sammenhenger og at det var vanskelig å faktisk forholde seg til denne i praksis over tid.

"Man plukker på en måte verktøyene, men så lever man ikke i metodikken eller gjennomfører den på en god måte." – systemutvikler

5.2.3 Tidligere arbeidsprosess

Som nevnt tidligere så hadde denne case-bedriften tidligere brukt en eget tilpasset metodikk med mange likhetstrekk med Kanban -metodikken. I forbindelse med dette så ble det også uttalt at programvareutvikling er et relativt nytt fagområde som har vokst veldig på kort tid – og at det derfor er naturlig at ikke alle prosesser er ferdig utviklet.

"Programvareutvikling er veldig nytt og ferskt. Vi har bare et par generasjoner på oss. Tidligere så ble det jo bare kalt koding uten å ha noe mer rundt det. Så det meste foregikk jo bare mellom ørene på en eller annen person." – senior systemutvikler

En av senior-informantene mente også at mangelen på struktur og god styring på systemutviklingsprosjekter i bransjen generelt har gjort det vanskelig å gjennomføre god programvareutvikling i prosjekter. Han mente at den teknologiske utviklingen har gått fortere enn utviklingen av gode prosesser for prosjektgjennomføring på dette området, men at dette gradvis er på vei til å bedres.

"Frustrasjonen har vært kjempestor, den blir gradvis mindre over at metodene og måtene vi mennesker (spesielt store organisasjoner) nærmer seg programvareutvikling ikke passer til programvare." – senior systemutvikler

Som nevnt innledningsvis i delkapittel 5.2 *Generelt om utviklingsmetoder* så er case-bedriften inne i en revitaliseringsfase hvor strategien er at nyere nettapplikasjoner skal overta markedsposisjonene til de eldre applikasjonene deres. Dette gjør at de i praksis bedriver nyutvikling samtidig som at de etablerte applikasjonene må vedlikeholdes. I sammenheng med dette så er det også ambisjoner om å forbedre brukeropplevelse og brukervennlighet i de nyere nettapplikasjonene – man ønsker ikke å utvikle alt på samme vis med de samme svakhetene som fantes i de gamle applikasjonene. Dette kan være et krevende og omfattende arbeid som krever ytterligere ressurser og behov for både styring og tillitt.

"Utfordringene hos oss har oppstått fordi det har vært de samme kontrollmekanismene som har foregått som tidligere - så det kjøres gammel metodikk på ny funksjonalitet og den detaljkontrollen går det ikke an å ha på nyutvikling" – senior systemutvikler

Når case-bedriften bare utviklet og vedlikeholdte den eksisterende applikasjonene som allerede er i bruk hos flere kunder – så er det ofte snakk om å endre lite eller minst mulig og levere ofte. Dette er naturlig for i en slik situasjon så ønsker man gjerne å møte kundenes behov samtidig som at man minimerer risikoen med å endre eller utvide det etablerte. Dette kan for eksempel være snakk om å endre litt på bruksmønsteret til applikasjonen eller endringer som følge av nye krav. Slik ny eller endret funksjonalitet medfører nødvendigvis ikke så veldig mye programmering, men krever forståelse og kjennskap til eksiterende domene og logikk, samt ressurser til testing av både eksisterende og ny funksjonalitet.

"Jeg tror vi hadde en teori om at ved å bruke Kanban så kan man holde veldig fokus på en og en oppgave om gangen og jobbe denne oppgaven helt ferdig og klar til produksjon . . . og hvorfor man valgte det – det virket fristende å bare ha et veldig fokus på å få en og en ting igjennom ikke sant." – systemutvikler

En periode inn i nyutviklingsprosjektet støttet de på utfordringer med tanke på dette. Lederen av utviklingsavdelingen uttalte at effektiviteten var for lav og at de produserte og leverte for lite for sjelden. En av de yngre informantene forklarte at de hadde utfordringer med samarbeidet i prosjektet med den nye nettapplikasjonen og at arbeidsmetodikken de jobbet etter kunne være en årsak til dette. Når man utvikler noe helt nytt så programmeres det gjerne veldig mye kildekode på mindre tid – samtidig som at bildet på horisont og retning gjerne er mindre klar enn hva tilfellet er ved vedlikehold av eksisterende kildekode.

Vedlikehold handler som nevnt ofte om å endre lite i noe som i utgangspunktet er veletablert. Flere av informantene, inkludert lederen, utalte at dette var hovedårsaken til at de nå måtte prøve å endre arbeidsmetodikk og prosess – dette i håp om å øke samarbeid og effektivitet og i deres interne nyutvikling.

"Vi bare jobbet hver for oss. Vi var en gruppe utviklere som satt under samme tak og møtte opp i de samme møtene - men vi jobbet ikke sammen." – systemutvikler

5.2.4 Kriterier for valg av utviklingsmetodikk

Som kriterier for valg av ny utviklingsmetodikk så ble samarbeid, effektivitet og produktivitet nevnt av flere av informantene. En endring for å prøve å forbedre dette har på en måte tvunget seg frem over tid.

"Vi har hatt problemer med produksjonen. Vi har produsert for lite. Vi har hatt store behov for å gjøre endringer. Jeg har tydelig sagt ifra for et år siden at vi må begynne å jobbe på en annen måte." – senior systemutvikler

I håp om å forbedre fremdrift og leveransekraft har case-bedriften nå en tanke om at Scrum-metodikken vil være en mer passende arbeidsmetodikk for deres systemutvikling.

"Det virker som at fremdrift er veldig sentralt i Scrum." – systemutvikler

En av senior informantene uttalte seg også kritisk til å ta i bruk Scrum i hele utviklingsavdelingen – og mente at dette var noe den nye ledelsen hadde besluttet uten egentlig å undersøke hvordan de jobbet tidligere. Han mente at Scrum-metodikken ville være vanskelig å gjennomføre ved forvaltning og vedlikehold av eksisterende kode.

"Det er endring i ledelse. De kommer med erfaringen av Scrum – eller jeg vet ikke hva egentlig . . . " – senior systemutvikler

Flere av informantene poengterte også at det ville være vanskelig å gjennomføre Scrum helt ut – og at noen tilpasninger til deres hverdag sannsynligvis ville bli nødvendig etter hvert. Samtidig så var det flere som mente at de burde prøve så godt som mulig fra start, for i det hele tatt ha mulighet for å lykkes. En av de yngre systemutviklerne mente også at det var en fordel å forholde seg til en godt dokumentert metode for å kunne unngå misforståelser eller ulike tolkninger.

"Alle kjenner jo til begrepet Agile eller smidig, men med Scrum så får man noe litt konkret som fortsatt ikke er for låst." – systemutvikler

5.2.5 Styrker og svakheter i utviklingsmetodikker

Når jeg utfordret informantene om de kunne uttale seg om styrker og svakheter i de forskjellige utviklingsmetodikkene så hadde de litt forskjellige meninger. En av de yngre informantene håpte at Scrum-metodikken kunne føre til økt kompetanseutveksling mellom utviklerne og at dette kunne være med å bygge bedre og mer slagkraftige team.

"Jeg tror at fokuset man får i Scrum er en styrke og det at man fokuserer på å bygge et godt team og forhåpentligvis så får man da også en del kompetanseutveksling." – systemutvikler

To av informantene vektla påvirkningskraft og autonomi som en styrke i Scrum – hvor prinsippet med at utviklingsteamene skal ha tillitt nok til å ta sine egne beslutninger og være selvstyrte ville være en fordel.

"Med å ta i bruk Scrum så blir utviklerne mer autonome, vi får større påvirkningskraft og vi kommer nærmere kundene" – senior systemutvikler

I forhold til samarbeid så mente en av informantene at Kanban ikke innbydde til godt samarbeid på samme måte som Scrum-metodikken gjør. Dette ble begrunnet med manglende planlegging og at bruken av Kanban kunne føre til "anarki" – hvor utviklerne bare plukket den oppgaven de ønsket seg eller ville gjøre uten å ta hensyn til prioriteringen av oppgavene.

"Kanban fører gjerne til mer individuelt arbeid i forhold til for eksempel Scrum hvor man skal samarbeide om å få til noe - så hvis man bruker Scrum på riktig måte så blir det mer samarbeid, mens med Kanban så blir det mer enkeltoppgaver." – senior systemutvikler

En av de yngre systemutviklerne mente at Scrum-metodikken muligens var litt for omfattende for deres bruk. Det er mange faste møter, roller og hendelser som er beskrevet i Scrum-metodikken – og informanten var spendt på om ambisjonen om å gjennomføre Scrum-metodikken til det fulle ville bli utfordrende med tanke på møtevirksomhet og tidsfrister. Videre mente han at Kanban nok hadde noen fordeler med tanke på administrasjonskostnader.

"Fordelen med Kanban så klart er vel at det er veldig lettvekt da – det er ikke så mye med det. I alle fall ikke slik som vi hadde implementert Kanban." – systemutvikler

Begge de erfarne informantene poengterte også at informasjonsflyten og kunnskapen til utfordringen eller problemet man skal løse er vel så viktig for suksess som utviklingsmetodikken man velger. Den ene informanten brukte "hviskeleken" som betegnelse på en ikke-fungerende organisasjon. Det vedkommende her adresserte er en kjent utfordring som gjerne blir større når organisasjoner vokser over tid.

"Jeg har gode erfaringer med Kanban metodikken – men det er ikke alt den fungerer like bra til. Og min konklusjon er at man må se på informasjonsflyten og organisere etter den." – senior systemutvikler

"Det som er det viktigste er at de som lager løsningen må ha førstehåndskunnskap til utfordringen eller problemet man skal løse. Jo flere ledd mellom der problemet oppstår eller den som bestiller en løsning på problemet til den som skal utvikle løsningen - desto dårligere blir produktet" – senior systemutvikler

5.3 Ulike typer av systemutviklingsprosjekter

Utviklerne i case-bedriften er i dag organisert i tre utviklingsteam – hvor to av teamene jobber med å utvikle de nye nettapplikasjonene for fremtiden, mens det siste teamet vedlikeholder og forvalter de eksisterende produktene som allerede er i drift hos flere kunder. Alle informantene virket innforstått med at behovet for prosjektgjennomføring med tanke på planlegging, styring og samarbeid kunne være forskjellige i disse to situasjonene.

"De eksisterende produktene skal gradvis fases ut og erstattes med de nye. Dette krever en bevissthet på prioritering av ressurser." – utviklingsleder

Alle informantene uttaler at Scrum-metodikken virker å være best tilpasset utvikling av nye applikasjoner og produkter. En av informantene brukte også uttrykket "greenfield" -prosjekter som en betegnelse på slike prosjekter. Argumentene for dette var også stort sett like hos informantene hvor de pekte på økt samarbeid mellom utviklerne, mer fokusert målsetning og bedre fremdrift

på mindre deler av prosjektet. En av senior utviklerne som hadde lang erfaring i forskjellige roller fra tidligere arbeidsforhold påpekte også at Scrum i utgangspunktet passet best for ordinær konsulentvirksomhet. Ordinær konsulentvirksomhet ble i denne forbindelse forklart som en gruppe utviklere som får oppgaven med å gjennomføre et systemutviklingsprosjekt fra A til Å innenfor en gitt tidsramme. Han mente videre at dette skilte seg veldig fra virkeligheten som case-bedriften befinner seg i, med eksisterende produkter i markedet, løpende aktiviteter mot eksisterende kunder, daglige henvendelser og håndtering av mye teknisk gjeld.

"Hvis det var opp til meg så ville jeg nok kanskje valgt Scrum i nye prosjekter. For hvis du sitter med masse teknisk gjeld så er det ikke så lett å være smidig" – systemutvikler

I forbindelse med ulikheter i systemutviklingsprosesser så løftet en av senior informantene også frem ulike behov for testing og verifisering av endringene som gjøres. Ved nyutvikling av et produkt så blir gjerne store deler av produktet til i løpet av prosjektet, og man vet gjerne ikke helt hvilke veier et slikt prosjekt vil ta. Det handler da gjerne om å forenkle arbeidsprosesser, tilgjengeliggjøring og visning av informasjon for en eller flere kunder. Forvaltning og vedlikehold handler gjerne om å endre lite på noe som allerede er etablert – noe som kan være forbundet med en mye høyere risiko ettersom man samtidig må forholde seg til det som allerede er etablert og tatt i bruk. Informanten påpeker at leveranser av små endringer oftere forenkler og muliggjør fokusert arbeid med testing og verifisering og minimerer risiko i leveransene.

"Når det gjelder vedlikehold av programvare, gjerne små nye funksjoner, så jobber man etter Kanban - og da har man gjerne et system hvor endringer kan gjøres i et miljø, testes i et annet og leveres videre til produksjon." – senior systemutvikler

Tre av informantene – de tre som hadde lengst erfaring med systemutvikling påpekte at suksess i alle systemutviklingsprosjekter avhenger av hvor god informasjon som er tilgjengelig for prosjektet og spesielt hvordan disse informasjonsstrømmene foregår i praksis. En av de mest erfarne hadde også erfaring med å sitte ute til kunden og observere hvordan kunden jobbet – samtidig som at han utviklet programvare for å forbedre og forenkle arbeidsprosessene deres. I dette tilfellet vil informasjonsstrømmen fra bruker til utvikler være minimal,

og man unngår gjerne mange misforståelser – samtidig som at det er lett å verifisere at programmet som utvikles faktisk møter kundes behov. Systemutviklingsprosjekter er som oftest mer kompliserte enn dette – hvor man gjerne vil møte mange kunders behov samtidig, og da blir man gjerne mer avhengig av styring og forskjellige roller med ulikt ansvar. Mange systemutviklingsprosjekter har her sviktet med at systemutviklerne har utviklet noe de trodde at kundene trengte – men som egentlig ikke løste kundes egentlige behov.

"I utviklingsprosjekt så må man organisere etter informasjonsstrømmene. Det er omtrent som å spørre en skipper vil du svinge mest til babord eller til styrbord? Tja ... Hvilket skip har du, hvor skal du og hvilken vei blåser det ...?" – senior systemutvikler

Som nevnt tidligere så var det akkurat besluttet at Scrum skulle innføres i systemutviklingsprosjektene til organisasjonen. Under samtalene med systemutvikler-informantene så tegnet det seg et bilde av at dette i utgangspunktet skulle gjelde alle prosjektene – men under samtalen med utviklingslederen noen uker senere så var dette endret. Under denne innføringsprosessen så fikk jeg nå høre at bare to av teamene nå skulle innføre Scrum og at det var besluttet at teamet som videreutviklet og forvaltet de eksisterende produktene heller skulle prøve å rendyrke Kanban metodikken.

5.3.1 Egenskaper ved systemutviklingsprosjektet

Systemutvikling handler mye om teknologiske muligheter, men også veldig mye om bakgrunn og domenekunnskap på områder hvor systemene skal brukes eller anvendes. De aller fleste systemer har en datamodell som på ett eller annet vis systematiserer en veldig forenklet virkelighet. I et lagersystem så vil man for eksempel kunne finne ut hvor en vare er på lageret og hvor mange enheter man har tilgjengelig – men lagersystemet inneholder ikke nødvendigvis alle egenskapene til alle varer. Når man skal lage et nytt system så starter man gjerne med å modellere denne datamodellen og utvider denne etter hvert som at systemet skal utvides, mens i et eksisterende system så finnes denne datamodellen allerede, noe som krever en oversikt over historie og bruksmønster for ikke å innføre feil ved endringer av denne datamodellen. En av senior informantene mente at dette var et av de viktigste kriteriene for å lykkes med et systemutviklingsprosjekt. For systemutviklere som har utviklet et system over lengre tid, kanskje

også vært med helt siden systemet bare var en ide, så vil denne datamodellen være kjent i utgangspunktet. Nye systemutviklere må først lære seg og skjønne denne datamodellen før de kan bidra med endringer av den – for ikke å innføre endringer som skader eller ødelegger for allerede eksisterende funksjonalitet.

"Når man holder på med nyutvikling så er det datamodellen man holder på med – den er mye mer flytende. Man har ikke så mye regresjonssaker å ta hensyn til." – senior systemutvikler

Når man har etablert et produkt i markedet som har blitt så populært at kunder er villige til å betale for å bruke det så starter gjerne forvaltning og vedlikeholdsfasen til et produkt. Verden er i stadig endring og både eksisterende og nye brukere på produktet vil gjerne påvirke produktet til sin egen fordel med innspill eller nye krav. Feilsituasjoner grunnet feilaktig eller endret bruk eller andre omliggende omstendigheter kan også oppstå, samtidig som kravene til stabilitet og ytelse stadig blir høyere. En av informantene påpeker i denne forbindelse viktigheten med å bevare de kundene man har og pleie disse relasjonene så godt som mulig. Brukere av systemet bruker gjerne systemet i sitt daglige arbeid og alle vet vel hvor frustrerende det er hvis dens eget arbeid blir forstyrret og utsatt på grunn av andres utfordringer. Informanten mener derfor at i denne fasen så handler det om å løse disse utfordringene så raskt som mulig og at Scrum-metodikken ikke er tilpasset denne virkeligheten.

"I et vedlikeholds-perspektiv så er Kanban overlegen i forhold til Scrum fordi man får minimert omfanget." – senior systemutvikler

Som nevnt tidligere så hadde case-bedriften en egenutviklet systemutviklingsprosess som hadde mye til felles med Kanban - og hvor denne metoden hadde fungert godt i lang tid. Når de startet med nyutviklingen av de nyere nettapplikasjonene så hadde de i utgangspunktet ikke planlagt å endre på denne prosessen. Behovet for en endring med hensyn på prosess og metodikk kom som et resultat av en periode med for lav produktivitet og leveranseevne i ettertid. De ønsket å styrke fokuset på arbeidsoppgavene og forbedre samarbeidet.

"Vår metodikk fungerte egentlig veldig bra for vedlikehold – derimot for nyutvikling så fungerte det dårlig fordi at man måtte håndtere så små biter i gangen" – senior systemutvikler

Planlegging av arbeid er en viktig del i de fleste prosjekter – også systemutviklingsprosjekter. Som de fleste er klar over så vil det ganske ofte oppstå endringer eller avvik fra det som er planlagt, og det kan være utallige årsaker til at dette forekommer. En av de yngre informantene trakk dette frem som en mulig utfordring ved bruk av Scrum-metodikken. Han beskrev at det daglig ble kommunisert problemstillinger fra resten av organisasjonen via forskjellige kommunikasjonskanaler, og at det var veldig vanlig at systemutviklerne plutselig hoppet over på noen av disse sakene – selv om dette overhodet ikke var en del av det planlagte arbeidet. Informanten var redd for at det ville bli vanskelig å kunne nå sprint målene som de da hadde forpliktet seg til.

"Altså vi har jo system hvor vi bare tar det litt på sparket – i alle fall foreløpig..." – systemutvikler

Tre av informantene var også ganske tydelig på at Scrum-metodikken krevde at teamet fikk rom til å fokusere på oppgavene i sprintene og sprintens målsetning hvis den skulle fungere – og ikke måtte håndtere mange andre prosjekter eller andre forstyrrelser samtidig. En av disse informantene trakk også frem det å legge til rette for samarbeid, og mente at Scrum-metodikken kunne være passende i de tilfellene der de faktisk fikk muligheten til at flere kunne jobbe med den samme oppgaven med en felles målsetning.

"Jeg synes Scrum-metodikken høres upassende ut hvis man skal håndtere mange baller i lufta samtidig." – systemutvikler

5.3.2 Foretrukket utviklingsmetodikk

I forhold til hvilken utviklingsmetodikk informantene ville gått for i neste prosjekt – helt uten å vite om hverken omfang eller innhold i dette prosjektet så svarer flere av informantene at de ville valgt Scrum. Dette gjaldt spesielt de yngste informantene. Samtidig så poengterer alle det faktum at de nå er inne i en periode med kursing i denne metodikken og at denne derfor er friskt i minne og en spennende tanke. Samtidig så nevner flere at de mangler kjennskap og erfaring med alternative metoder – unntagen hvordan de hadde jobbet tidligere med systemutvikling.

"Kanban – jeg vet egentlig ikke så mye om denne metodikken - annet enn at man flytter rundt på oppgaver. Jeg har ikke så stort grunnlag der egentlig." – systemutvikler

Begge de yngste informantene uttaler også at Scrum-metodikken kan bli utfordrende å få til i praksis – og at bruken av metodikken sannsynligvis må tilpasses deres bruk over tid for å få den til å fungere. De uttrykker også viktigheten med å faktisk prøve ut Scrum-metodikken til sitt fulle i starten – for å prøve å innarbeide gode rutiner og ikke komme skeivt ut fra start. Samtidig så argumenteres det for at Scrum-metodikken er godt dokumentert og fremstår som en ryddig og tilpasningsdyktig prosess. Tilgangen på Scrumguiden og ulike kurs for forskjellige roller innen Scrum-metodikken trekkes også frem som fordeler med denne metodikken til fordel for de alternative utviklingsmetodene.

Begge informantene med lengre erfaring er mer tvilende til å svare på hvilken utviklingsmetodikk de ville valgt uten å vite mer om forutsetninger og hva utviklingsprosjektet faktisk ville inneholdt. Bakgrunnskunnskapen til deltakerne i prosjektet trekkes her frem som en viktig faktor, og i denne forbindelse så nevnes det også at erfaring og kjennskap til et konkret prosjekt eller produkt gjør det mye enklere å vurdere hvor komplisert eller omfattende en endring faktisk trenger å være for å kunne løfte produktet slik at det neste kravet eller behovet også kan møtes.

"Valget av passende utviklingsmetodikk avhenger helt av prosjektet – det er forskjellig hva vi kan, det er avhengig av folkene som er med, det er avhengig av størrelsen på prosjektet, det er avhengig av hvilken implementasjon som er der og hvem som skal gjøre noe med det." – senior systemutvikler

5.3.3 Suksessfaktorer

Selve utviklingsarbeidet eller programmeringen i et systemutviklingsprosjekt er i utgangspunktet en individuell jobb. I systemutviklingsprosjekter med mer enn en utvikler så implementerer hver utvikler deler av systemet individuelt – noe som også krever samarbeid for at hele systemet skal fungere sammen som et produkt. Når produkter eller prosjekter vokser så kommer også ofte behovet for forskjellige ansvarsområder og mer strukturert samarbeid. En av senior informantene trekker nettopp kommunikasjon og samkjøring som en av de viktigste suksessfaktorene i et større systemutviklingsprosjekt.

"Etter hvert som du involverer flere mennesker i jobben så er det behov for samkjøring og kommunikasjon." – senior systemutvikler

Som et svar på dette trekker en av de yngre informantene frem styrken med å utnevne roller med forskjellig ansvar og spesielt egne produkteiere i Scrum-metodikken. Han påpeker at ved å ta i bruk Scrum-metodikken på nyutviklingsprosjekter så blir veien til litt på løpende bånd - og at behovet for at flere personer da er med på planlegging og forankring av arbeidet blir høyere.

Som nevnt innledningsvis i delkapittel 5.3 *Ulike typer av systemutviklingsprosjekter* så hadde en av senior informantene erfaring med konsulentvirksomhet – hvor han til daglig var lokalisert sammen med kunden. Han mente at nærhet til brukerne og førstehåndskunnskap om deres utfordringer var en viktig suksessfaktor for et vellykket systemutviklingsprosjekt.

"Hvis man sitter sammen med brukerne og er nært de som har utfordringene så får man til å løse problemene mye lettere – man får en helt annen forståelse for hva man egentlig skal løse." – senior systemutvikler

God fremdrift og effektivitet blir nevnt av flere av informantene i sammenheng med hva som kjennetegner et godt systemutviklingsprosjekt. En av informantene mener at resultatet av valget av egnet utviklingsmetodikk avhenger av motivasjonen bak valget – og uttaler at å ta i bruk Scrum for å skape fortgang fort kan virke mot sin hensikt hvis masse teknisk gjeld hindrer for å skape fortgang. Han var redd for at de på sikt bare ville få mer av de problemene de allerede hadde med hensyn på fremdrift og effektivitet. I samme forbindelse så nevner en av de eldre informantene at effektivitet og fremdrift i stor grad handler om å fokusere slik at man får levert noen få endringer oftere, i stedet for å levere mange endringer sjeldnere.

"Tidligere så kunne vi ha en ny versjon ferdig testet på et par dager – og vi kunne ha to, tre kanskje fire leveranser i løpet av en uke." – senior systemutvikler

Informanten som var mest kritisk til innføringen av Scrum i hele organisasjonen mente at resultatet av dette valget ikke var avhengig av folks erfaring eller følelser for bruk av forskjellige utviklingsmetoder. Han mente at arbeidshverdagen og de daglige henvendelsene uansett ville påvirke de interne prosessene omhandlende endringer i deres produkter. Informanten påpekte at dette var spesielt gjeldende i en forvaltning/vedlikeholdsfase hvor det handlet om å rette kritiske feil eller hjelpe brukere videre med sine arbeidsprosesser.

"Det ligger så naturlig i saken – altså man må bare prøve å bruke Scrum på eksisterende kode og du vil få en masse frustrasjon, overflødig arbeid og bli veldig ineffektiv. " – senior systemutvikler

Kapittel 6

Analyse

I slutten på kapittel 3 *Utviklingsmetoder* ble det presentert et teoretisk rammeverk med tre proposisjoner som var utledet fra utvalgt litteratur om prosjektledelse, forskjellige utviklingsmetoder, samt mine egne erfaringer på området. De empiriske funnene som ble innhentet fra intervjuene med informantene i casebedriften som ble presentert i forrige kapittel blir i dette kapittelet analysert for å vurdere i hvilken grad proposisjonene kan støttes eller ikke.

Proposisjonene vil bli gjennomgått kronologisk og det vil bli utført en analyse av de relevante empiriske funnene knyttet til hver enkelt proposisjon. Tabell 6.1 viser oversikten over proposisjonene som er hentet fra delkapittel 3.4 *Teoretisk rammeverk*.

Nummer	***Proposisjon***
1	*Egenskaper ved prosjektet eller forskjellige typer av systemutviklingsprosjekter påvirker arbeidsflyten i prosjektet og følgelig hvilken utviklingsmetodikk man burde velge for å kunne oppnå suksess.*
2	*Dagens systemutviklere har en relativt god kjennskap til de mest utbredte og kjente utviklingsmetodikkene som benyttes i bransjen i dag og de kan også uttale seg om styrker og svakheter med disse.*
3	*Organisasjoner som utøver stadig mer kompleks systemutvikling etterstreber økt bruk av dokumenterte utviklingsmetodikker i sine systemutviklingsprosjekter for å oppnå ønsket effekt.*

Tabell 6.1: Oppsummering av proposisjonene (repetisjon)

De tre proposisjonene vil avslutningsvis i dette kapittelet bli omformulert og tilpasset det empiriske grunnlaget i dette forskningsprosjektet hvis det viser seg at dette er hensiktsmessig eller nødvendig.

6.1 Proposisjon 1: Forskjeller og ulikheter i systemutviklingsprosjekter

Som presentert i delkapittel 2.4 *Oppsummering og utvikling av proposisjon* ble proposisjon 1 formulert slik:

Egenskaper ved prosjektet eller forskjellige typer av systemutviklingsprosjekter påvirker arbeidsflyten i prosjektet og følgelig hvilken utviklingsmetodikk man burde velge for å kunne oppnå suksess.

Proposisjonen bygger på prinsippet med at ingen prosjekter er like og at gitte egenskaper for et konkret prosjekt vil kunne ha påvirkning på hvilken utviklingsmetodikk man burde tilnærme seg for å oppnå ønsket resultat i prosjektet. I kapittel 2. *Prosjektledelse* vises det til eksempler på slike egenskaper som kan være tidsfrister, budsjett, antall deltakere eller omfang. Som andre prosjekter så er også systemutviklingsprosjekter ulike og selve hensikten med prosjektet

kan også være med å påvirke gjennomføringen av prosjektet i stor grad. Det antas derfor at typen prosjekt vil være førende for arbeidsmetodikken i prosjektet. I delkapittel 2.1 *Prosjekttyper* så presenteres nyutviklingsprosjekt og et vedlikeholdsprosjekt som to klare eksempler på slike typer av systemutviklingsprosjekter – hvor forutsetningene for å lykkes vil være delvis helt forskjellig.

6.1.1 Argumenter for at ulike prosjekter krever forskjellige arbeidsflyt

Det sies i delkapittel 5.3 *Ulike typer av systemutviklingsprosjekter* at man må organisere etter informasjonsstrømmene i et prosjekt – samtidig så uttales det i avsnitt 5.3.3 *Suksessfaktorer* at nærhet til brukerne øker forståelsen for deres egentlige utfordringer og problemet man egentlig skal løse. Systemutvikling handler mye om teknologiske muligheter for systematisering, problemløsning eller forenkling av arbeidsoppgaver hos vanlige mennesker, og ofte så handler det mye om først å skjønne hva behovet til disse menneskene er og hva de egentlig trenger. Når omfanget på prosjekter eller produkter øker så øker også behovet for ressurser, styring og roller. Administrative ledere, selgere, prosjektledere, konsulenter og andre roller utnevnes gjerne over tid i slike situasjoner, noe som kan føre til at systemutviklerne blir sittende lengre og lengre unna de virkelige brukerne av sitt system eller produkt. Det er naturlig av dette blir vektlagt av informantene i denne forbindelse – ettersom dette er en kjent utfordring i bransjen og noe jeg selv har erfart. Daglige brukere av et system krever en helt annen responstid på sine egne problemer med egne arbeidsoppgaver – enn hva tilfellet er hvis prosjektet skal føre frem til et nytt produkt som skal løse arbeidsoppgaver til brukeren på en bedre eller smartere måte en eller annen gang i fremtiden.

Etter hvert som prosjekter eller produkter vokser i omfang så øker også kompleksiteten. I avsnitt 5.3.1 *Egenskaper ved systemutviklingsprosjektet* så trekker en av informantene frem datamodellen til et system som et bilde på at kompleksiteten vokser over tid – og at fartstid og erfaring med det konkrete systemet blir mer og mer essensielt når systemet skal endres eller videreutvikles. I forbindelse med dette så uttales det også at risikoen med å endre slike etablerte systemer øker i takt med omfanget og antall brukere med potensielt ulik bruk av systemet. I dette lyset så synes jeg det vil være unaturlig å legge opp det planlagte arbeidet etter fastsatte tidsbestemte milepæler som Scrum-metodikken tilsier

med sin planlegging av sprinter med forankret innhold og fastsatt ferdigstillelsesdato på forhånd. Fallgruvene her vil være at de små endringene som brukerne egentlig trenger for selv å komme seg videre blir for lite vektlagt til fordel for større planlagte endringer – og at arbeidsoppgaver fort kan bli liggende halvferdige på ferdigstillelsesdato, noe som igjen kan medføre en risiko ved leveranse til eksisterende brukere.

For systemutviklingsprosjekter som skal skape et nytt system så vil prosjektet ha en helt annen inngang og flyt i arbeidet. Et nytt system lages gjerne på bakgrunn av ideer om forenklinger eller forbedringer av hverdagen og stiller helt andre krav til systemutviklerne enn tilfellet er ved vedlikehold av et allerede eksisterende system. I et nytt system så har man ingen eller lite teknisk gjeld å forholde seg til, ingen eksisterende data eller datamodeller å forholde seg til. Man starter på en måte med blanke ark og må prøve å finne ut hvordan systemet skal fungere ved hjelp av informasjon fra fremtidig brukere igjennom krav, møter, intervjuer og liknende. Man har gjerne her muligheten til å teste ut ulike løsninger (prototyper) og endre datamodellen hyppigere – uten å måtte ta hensyn til eksisterende brukere og risikoen for å ødelegge eller forstyrre for disse. I slike situasjoner så produseres det gjerne mye ny kode på kort tid – noe som gjerne også tilrettelegger for styrket samarbeid mellom systemutviklerne. Det er både er interessant og spennende at flere av informantene trekker frem denne forskjellen som den viktigste egenskapen til et systemutviklingsprosjekt når det gjelder å velge egnet arbeidsflyt eller utviklingsmetodikk. I delkapittel 5.3 *Ulike typer av systemutviklingsprosjekter* så går det også frem at case-bedriften hadde endret sin beslutning om å innføre Scrum i alle sine systemutviklingsprosjekter – og heller innføre Kanban som utviklingsmetode for teamet som forvaltet de eksisterende produktene. Dette bekrefter både at de ulike utviklingsprosjektene behøver ulike tilnærminger med tanke på gjennomføring og at det også er ulikt syn på dette behovet.

6.1.2 Argumenter mot at ulike prosjekter krever forskjellige arbeidsflyt

Det finnes nok helt klart mange gode argumenter for at en organisasjon bør tilstrebe å bruke samme utviklingsmetodikk for alle sine utviklingsprosjekter. Fra et ledelsesperspektiv så er ett eksempel på dette styring og omrokering av

ressurser mellom ulike prosjekter. For deltakerne i et prosjekt så vil det nok oppleves enklere å bytte prosjekt hvis arbeidsflyten i prosjektet er kjent fra før – selv om hensikten med prosjektet kan være en helt annen. Dette kan også være med på å dempe risikoen for interne misforståelser innad i prosjektgruppen.

Som nevnt i avsnitt 5.2.1 *Kjennskap til utviklingsmetodikken Scrum* så har alle systemutviklerne også gjennomført opplæring i utviklingsmetodikken Scrum ved at alle har deltatt på eksternt kurs i bruk av denne metoden. Felles kursing og opplæring av fagpersoner i en organisasjon åpner gjerne for rabattordninger hos kursholdere og mange andre stordriftsfordeler. Felles opplæring fører også gjerne til økt fokus på et område og tilrettelegger for fremtidig samarbeid.

Styring og leveransekraft i organisasjoner forbindes også gjerne med forutsigbarhet i arbeidet. For en organisasjon som gjennomfører flere like eller ulike prosjekter samtidig så vil det selvfølgelig være enklere å måle og sammenlikne fremdrift og resultater hvis arbeidsflyten og gjennomføringen av disse prosjektene er så lik som mulig. I avsnitt 5.2.4 *Kriterier for valg av utviklingsmetodikk* så uttales det at effektivitet, produktivitet og fremdrift er viktige argumenter for å begynne å jobbe på en annen måte – noe som jeg opplever som hovedårsaken til at organisasjonen nå har tatt valget om å prøve ut utviklingsmetodikken Scrum for systemutviklingsprosjekter i organisasjonen.

6.1.3 Oppsummering og konklusjon

Forskjellige systemutviklingsprosjekter og hvor prosjektet befinner seg i livsløpet til systemet vil kunne sette helt forskjellige krav til kompetanse, erfaring og mange andre egenskaper. Kravene til oppetid, responstid og levereansehyppighet vil også være helt forskjellig i ulike typer av systemutviklingsprosjekter noe som påvirker hverdagen og arbeidsdagen til deltakerne i prosjektet. I forhold til risiko i forbindelse med endringer av systemer så uttaler også informantene at den største forskjellen her vil være om prosjektet handler om å lage noe nytt eller endre noe allerede eksisterende – noe som jeg vektlegger høyt i min egen vurdering av dette tema. I forhold til at organisasjonen nå har tatt valget om å prøve ut utviklingsmetodikken Scrum for systemutviklingsprosjekter i organisasjonen så finner jeg dette spesielt interessant. Min egen erfaring i forhold til dette er at dette nødvendigvis ikke trenger å hjelpe organisasjonen med for-

bedret effektivitet, produktivitet og fremdrift i alle prosjekter - i alle fall ikke på kort sikt. I avsnitt 3.2.1 *Scrum* så beskrives denne metodikken som relativt omfattende med tanke både på rollesammensetning og ritualer – samtidig som at planlegging frem i tid er essensielt for å lykkes med denne metodikken. Men i hvilken grad det faktisk er mulig å planlegge godt nok i et utviklingsprosjekt avhenger også i stor grad av hvilken fase i livsløpet produktet som utvikles befinner seg i. Valget vil nok helt sikkert hjelpe for å skape forbedret samarbeid og fokus på felles arbeidsoppgaver – men dette trenger nødvendigvis ikke å bety at de blir så mye mer leveransedyktige, spesielt med tanke på eksisterende kunder og brukere. Med dette som bakgrunn så støtter denne studien proposisjonen, og dette underbygges av de delvis forskjellige funnene og argumentasjonen med hensyn på ulikheter i systemutviklingsprosjekter.

Proposisjon 1 er basert på eksisterende litteratur og mine egne erfaringer, og er ikke tilpasset de empiriske funnene som ble presentert i kapittel 5 *Empiriske funn*. Basert på analysen av de empiriske funnene tilknyttet denne proposisjonen gir dette ikke grunnlag for å endre proposisjonen veldig, men jeg har videreutviklet og spisset proposisjonen noe, slik at proposisjon 1.1 nå kan presenteres slik:

Forutsetninger for prosjektet og typen av systemutviklingsprosjekt påvirker informasjonsflyten i prosjektet - og følgelig hvilken utviklingsmetodikk eller arbeidsflyt man burde velge for å kunne oppnå suksess.

6.2 Proposisjon 2: Kjennskap til ulike utviklingsmetodikker

Som presentert i delkapittel 3.3 *Oppsummering og utvikling av proposisjoner* ble proposisjon 2 formulert slik:

Dagens systemutviklere har en relativt god kjennskap til de mest utbredte og kjente utviklingsmetodikkene som benyttes i bransjen i dag og de kan også uttale seg om styrker og svakheter med disse.

Proposisjonen tar for seg selve grunnlaget for utnyttelse av ulike utviklingsmetodikker i gjennomføring og styring av systemutviklingsprosjekter. Hvis ikke

kjennskapen til de ulike utviklingsmetodikkene er til stede, vil det heller ikke være mulig å utnytte disse for å oppnå den ønskede effekten man prøver å oppnå ved å ta de i bruk. Det antas derfor at de mest brukte utviklingsmetodikkene i bransjen er kjent for dagens systemutviklere og at de har en viss kjennskap til styrkene og svakhetene til de forskjellige metodene.

6.2.1 Argumenter for god kjennskap til ulike utviklingsmetoder

Først og fremst så var det interessant at case-bedriften var midt inne i en endringsprosess med tanke på arbeidsmetodikk med tanke på sine systemutviklingsprosjekter. Som en følge av dette så hadde alle informantene gjennomført kursing i Scrum – så akkurat denne utviklingsmetodikken hadde de alle god og fersk kjennskap til, selv om noen nevner at de bare har teoretisk kjennskap til denne og at de fortsatt mangler praktisk erfaring.

Kanban ble også nevnt av alle informantene, selv om kunnskapen til denne metoden virket mer varierende på flere av informantene. I avsnitt 5.3.2 *Foretrukket utviklingsmetodikk* så uttales det rett ut at informanten egentlig ikke vet så mye om denne metoden – annet enn at lapper (oppgaver) utføres og flyttes (endrer status). Dette er nok naturlig ettersom case-bedriften ikke hadde jobbet etter noen fast uttalt utviklingsmetodikk tidligere – men at slik de hadde jobbet hadde klare likhetstrekk med akkurat Kanban metodikken.

Fossefallsmetoden nevnes av begge de to mest erfarne informantene, se avsnitt 5.2.2 *Kjennskap til alternative utviklingsmetodikker*. Informantene forklarer at de først ble gjort oppmerksomme på denne metoden på sine egne studier - og hvor den ene av disse uttaler at han allerede da så klare ulemper med å ta i bruk denne metoden for systematisert systemutvikling. Den andre informanten var også kritisk til metoden, men uttalte videre at prosjektledere gjerne ønsket seg tilbake til denne metodikken blant annet på grunn av dens forutsigbarhet. Dette finner jeg særdeles interessant. I avnsitt 5.2.2 *Kjennskap til alternative utviklingsmetoder* uttaler informanten at fossefallsmetoden har klare fordeler med tanke på blant annet dokumentasjon og etterprøvbarhet som klart overgår andre metoder og at dette er mangelvare i mange prosjekter i dag.

En av senior informantene nevnte også ekstrem programmering i avsnitt 5.2.2 *Kjennskap til alternative utviklingsmetodikker*. Han forklarte at han hadde satt seg inn i denne metodikken i frustrasjon over at fossefallsmetodikken ikke fungerte i hans utviklingsprosjekter – men at han ikke hadde veldig utstrakt erfaring og bruk av denne metoden, selv om elementer av denne sannsynligvis hadde påvirket hvordan organisasjonen så langt hadde jobbet.

Prototype modellen som er beskrevet i avsnitt 3.1.2 *Prototype modellen* ble ikke nevnt som en egen metode av noen av informantene – selv om flere uttalte at dette var en nyttig og hensiktsmessig måte å utvikle ny funksjonalitet på. Dette er også naturlig ettersom prototyping kanskje er mer kjent som et hjelpemiddel under utvikling og uttesting – enn som en egen prosjektmetodikk i bransjen i dag.

6.2.2 Argumenter mot god kjennskap til ulike utviklingsmetoder

Organisasjonen hadde frem til nylig jobbet etter en egentilpasset metodikk som de selv hadde tilpasset etter behov over tid. Dette fungerte godt i en periode – helt til at større endringer måtte gjøres og flere nye utviklere måtte bidra. Denne måten å jobbe på var ikke effektiv nok med tanke på utvikling av nye produkter og overfor de nye utviklerne så førte dette til misforståelser og dårlig samarbeid – se avsnitt 5.2.3 *Tidligere arbeidsprosess*. Organisasjonen valgte altså å ta i bruk Scrum for å prøve å forbedre dette og derfor så var alle informantene godt kjent med denne metodikken og i tillegg nylig kurset i denne. Samtidig så uttalte flere av informantene og spesielt de yngre at de bare hadde teoretisk kjennskap til denne metodikken og manglet erfaring om praktisk bruk. Dette peker imot at organisasjonen har god praktisk kjennskap til denne metodikken.

Det var bare en av de eldre informantene som hadde erfaring fra konsulentvirksomhet fra tidligere arbeidsforhold. Informanten uttalte i delkapittel 5.3 *Ulike typer av systemutviklingsprosjekter* at han faktisk hadde dokumentert erfaring med bruk av forskjellige metodikker - og delte gjerne hans meninger angående styrker og svakheter med disse, samtidig som at han presiserte at dette var for en tid tilbake og at mye er endret siden den gang.

6.2.3 Oppsummering og konklusjon

Organisasjonen virker relativt godt informert om de vanligste utviklingsmetodikker som brukes i bransjen i dag – selv om flere av informantene uttaler at de mangler praktisk erfaring om bruk av disse. Det at organisasjonen er i en endringsfase med tanke på utviklingsmetodikk og fokuset rundt dette taler for at informantene er oppdatert spesielt med tanke på Scrum-metodikken – mens den manglende erfaringen hos flere av informantene taler imot. Foruten Scrum-metodikken så var det stort sett bare senior informantene som kunne uttale seg om erfaringer eller tanker rundt bruk av andre utviklingsmetodikker og hvordan disse ville påvirke arbeidsflyten og leveransene i prosjektene til organisasjonen. Med dette som bakgrunn så støtter denne studien proposisjonen bare delvis. Hovedargumentene for dette er at mange av utviklerne mangler praktisk bruk og erfaring med bruk av forskjellige dokumenterte metoder og at det tidligere ikke hadde vært noe stort fokus på å ta i bruk kjente metoder til fordel for metodikken som de selv hadde utviklet med tanke på gjennomføring av systemutviklingsprosjekter.

Proposisjon 2 er basert på eksisterende litteratur og mine egne erfaringer, og er ikke tilpasset de empiriske funnene som ble presentert i kapittel 5 *Empiriske funn*. Basert på analysen av de empiriske funnene tilknyttet denne proposisjonen har jeg endret og videreutviklet proposisjonen, slik at proposisjon 2.1 nå kan presenteres slik:

Dagens systemutviklere har varierende kjennskap til de mest utbredte og kjente utviklingsmetodikkene som benyttes i bransjen i dag, samt styrker og svakheter med disse – og noen er i tillegg kjent med historikken til de mer tradisjonelle metodene.

6.3 Proposisjon 3: Valg av utviklingsmetodikk i systemutviklingsprosjekter

Som presentert i delkapittel 3.3 *Oppsummering og utvikling av proposisjoner* ble proposisjon 3 formulert slik:

Organisasjoner som utøver stadig mer kompleks systemutvikling etterstreber økt bruk av dokumenterte utviklingsmetodikker i sine systemutviklingsprosjekter for å oppnå ønsket effekt.

Proposisjonen omhandler en kjent problemstilling i bransjen som driver med systemutvikling – nemlig at en stadig økende kompleksitet og omfang krever mer omfattende styring av prosjektene. De ulike utviklingsmetodikkene som er nevnt i kapittel 3 *Utviklingsmetoder* er utviklet med tanke på denne problemstillingen og hjelper prosjektene med å møte denne utfordringen med å innføre faste regler, roller og verktøy. Hensikten med dette er å skape struktur, forutsigbarhet og gjennomføringsevne for å kunne oppnå målsetningene de har med sine systemutviklingsprosjekter.

6.3.1 Argumenter for å ta i bruk dokumenterte utviklingsmetoder

Et ønske om økt samarbeid og kompetanseheving nevnes i avsnitt 5.2.3 *Tidligere arbeidsprosess* som et argument for å ta i bruk Scrum-metodikken i organisasjonens nyutviklingsprosjekter. Under arbeidet med å utvikle nye lettere nettbaserte applikasjoner som på sikt skal ta over for de gamle applikasjonene hadde det dukket opp en utfordring med tanke på å fortsette å jobbe etter eksisterende arbeidsprosess. Det kommer frem i avsnitt 5.3.3 *Suksessfaktorer* at tidligere arbeidsprosess dreide seg mye om å levere små endringer og forbedringer til kundene så hurtig som mulig og samtidig redusere risikoen for at endringene skapte utilsiktede sideeffekter. Dette passet dårlig for nyutvikling hvor mange gjerne store endringer skjer kontinuerlig og behovet for testing og verifisering er mindre på grunn av at systemet enda ikke er tatt i bruk. Å ta i bruk Scrum-metodikk i slike oppdrag vil gjerne øke samarbeidet på grunn av at man da først må planlegge/estimere arbeidet som skal gjøres og da settes det gjerne et større fokus på akkurat disse oppgavene samtidig. Samtidig så finner jeg det særdeles interessant at kriteriene som lå til grunn for beslutningen om å

innføre Scrum også omfattet et ønske om økt effektivitet og produktivitet slik det fremgår i avsnitt 5.2.4 *Kriterier for valg av utviklingsmetodikk*. Det at informantene hadde ulike meninger angående dette, og at noen også mente at dette bare var noe ledelsen hadde bestemt gjør akkurat dette bare mer interessant.

I og med at organisasjonen hadde ansatt flere nye systemutviklere de siste årene så kan en endring av utviklingsmetodikk være positivt i seg selv. Det å lære noe nytt sammen med andre – hvor ingen egentlig har noe spesiell erfaring med området fra før er veldig ofte mer motiverende enn tilfellet er hvis noen må lære seg noe andre kan fra før. Det går frem i avsnitt 5.2.1 *Kjennskap til utviklingsmetodikken Scrum* at alle utviklerne nå har fått deltatt på kursing i Scrum – noe som kan gjøre at de fremover styrker felleskapet og den enkeltes eierskapsfølelse til deres produkter hvor denne metodikken blir brukt.

Å ta i bruk en allerede dokumentert utviklingsmetodikk kan minske risikoen for misforståelser. Det er god tilgang til både dokumentasjon og andres erfaringer med bruk av de mest relevante metodene. Dette kan dreie seg om rutiner for hva som skal skje når en oppgave er fullført eller hva som skal gjøres i forbindelse med planlegging av fremtidig utvikling. Dette kan skape forutsigbarhet i prosjektene og gjør også prosjektene mer sammenliknbare med hverandre. Økonomisk vil en allerede dokumentert utviklingsprosess som er allment tilgjengelig også være mye billigere å innføre enn tilfellet er hvis man skal utvikle disse metodene og rutinene selv.

6.3.2 Argumenter mot å ta i bruk dokumenterte utviklingsmetoder

En vanlig problemstilling med tanke på å endre en arbeidsprosess til fordel for en annen dokumentert prosess er den faktiske forskjellen mellom teori og praksis. Prosessene er gjerne dokumentert generelt slik at de skal kunne innføres i så mange sammenhenger som mulig – noe som alltid reiser spørsmålet om dette er riktig valg for et konkret prosjekt. I avsnitt 5.2.1 *Kjennskap til utviklingsmetodikken Scrum* så nevnes akkurat dette. Alle prosjekter er unike, og dette gjør at det alltid vil være forutsetninger eller andre forhold i prosjektet som krever en ulik gjennomføring av prosjektene.

I avsnitt 5.2.5 *Styrker og svakheter i utviklingsmetodikker* så sies det at man må organisere arbeidet etter informasjonsflyten. Det sies også i denne sammenheng at desto flere ledd denne informasjonen skal passere før den presenteres til den som utvikle løsningen – desto dårligere blir produktet. Dokumenterte utviklingsmetoder har alle til felles at de prøver å generalisere systemutviklingsprosjektene med et sett med regler, rutiner og verktøy – og fordeler gjerne ansvarsområder til forskjellige roller slik for eksempel Scrum-metodikken gjør. Denne ansvarliggjøringen av enkeltpersoner setter helt andre krav til den interne kommunikasjonen i form av møter og planlegging. Spørsmålet da er om man faktisk har de rette personene som kan ta ansvar for dette og at de faktisk har tid til å gjøre det. Slik intern kommunikasjon og møtevirksomhet er gjerne ensartet med bruk av interntid og administrasjonskostnader – noe som samtidig gjerne vil begrenses fra ledernes perspektiv. Akkurat dette vil være helt avgjørende for å lykkes med å ta i bruk dokumenterte utviklingsmetodikker i prosjektene.

Det er viktig å presisere at ulike utviklingsmetodikker og -metoder bare er et verktøy for å gjennomføre systemutviklingsprosjekter. Innholdet i prosjektet er det som er den egentlige verdiskapningen for kundene og det som betyr noe for dem. Det er derfor viktig å passe på at utviklingsmetoden som velges ikke er til hinder for organisasjonens virkelige hensikt og virke. I avsnitt 5.3.3 *Suksessfaktorer* så nevner informantene litt om hvordan de tidligere har oppnådd suksess, og det er viktig at dette også blir hensyntatt i vurderingen av valget av metodikk.

6.3.3 Oppsummering og konklusjon

Profesjonell systemutvikling blir stadig mer kompleks og omfangsrik - og krever derfor både ansvarliggjøring i form av forskjellige roller - men også rutiner og strukturer for gjennomføring av prosjekter. Det at organisasjonen faktisk var i en endringsprosess med tanke på utviklingsmetodikk taler i utgangspunktet for seg selv og levner liten tvil om organisasjonens fokus på dette området. Det at de enkelte systemutviklerne ikke var enige om hvilken retning man burde ta med tanke på utviklingsmetodikk tyder også på at interessen for dette temaet er stort. Jeg finner det både interessant og naturlig at dette faktisk er tilfelle – noe som mine egne erfaringer bekrefter. Jeg har selv bidratt til mange ulike utviklingsprosjekter og vet selv hvor frustrerende det kan være at interne regler og

prosedyrer blir et hinder for å oppfylle kundenes virkelige behov. Med dette som bakgrunn så støtter denne studien proposisjonen. Hovedargumentet for dette er organisasjonens faktiske endringsprosess med tanke på gjennomføring av systemutviklingsprosjekter, men kan også underbygges av en stadig mer kompleks arbeidshverdag og behovet for forutsigbarhet over tid.

Proposisjon 3 er basert på eksisterende litteratur og mine egne erfaringer, og er ikke tilpasset de empiriske funnene som ble presentert i kapittel 5 *Empiriske funn*. Basert på analysen av de empiriske funnene tilknyttet denne proposisjonen har jeg endret og konkretisert proposisjonen noe, slik at proposisjon 3.1 nå kan presenteres slik:

Organisasjoner som utøver stadig mer kompleks systemutvikling etterstreber økt bruk av smidige utviklingsmetodikker som for eksempel Scrum og Kanban i sine systemutviklingsprosjekter for å oppnå forutsigbarhet, bedre samarbeid og økt effektivitet.

6.4 Oppsummering og empirisk tilpasset rammeverk

I analysen har jeg i kronologisk rekkefølge analysert de tre proposisjonene som er utviklet på bakgrunn av de tre delspørsmålene i forskningsspørsmålet til denne studien. For hver proposisjon så er det på bakgrunn av denne analysen tatt stilling til om denne studien enten støtter proposisjonen eller ikke. Jeg har endt opp med at proposisjon 1 og 3 kan støttes, men at studien bare delvis kan støtte proposisjon 2.

Under analysen av den enkelte proposisjonen så har også proposisjonene blitt tilpasset det empiriske datagrunnlaget i kapittel 5 *Empiriske funn* og følgelig blitt omformulert med dette som underlag. I tabell 6.2 er de empirisk tilpassede proposisjonene oppsummert. Det har vært interessant å teste generaliserbarheten i disse proposisjonene i en organisasjon som jeg ikke selv er en del av.

Nummer	***Proposisjon***
1.1	*Forutsetninger for prosjektet og typen av systemutviklingsprosjekt påvirker informasjonsflyten i prosjektet - og følgelig hvilken utviklingsmetodikk eller arbeidsflyt man burde velge for å kunne oppnå suksess.*
2.1	*Dagens systemutviklere har varierende kjennskap til de mest utbredte og kjente utviklingsmetodikkene som benyttes i bransjen i dag, samt styrker og svakheter med disse - og noen er i tillegg kjent med historikken til de mer tradisjonelle metodene.*
3.1	*Organisasjoner som utøver stadig mer kompleks systemutvikling etterstreber økt bruk av smidige utviklingsmetodikker som for eksempel Scrum og Kanban i sine systemutviklingsprosjekter for å oppnå forutsigbarhet, bedre samarbeid og økt effektivitet.*

Tabell 6.2: Oppsummering av de empirisk tilpassede proposisjonene

Det opprinnelige teoretiske rammeverket som ble presentert i delkapittel 3.4 *Teoretisk rammeverk* er nå blitt tilpasset funnene i denne case-studien. I det neste kapittelet blir disse empirisk tilpassede proposisjonene diskutert med tanke på hvilke teoretiske og praktiske implikasjoner de empiriske funnene har i forhold til litteraturen som ble presentert i kapittel 2 *Prosjektledelse* og kapittel 3 *Utviklingsmetoder* – samt mine egne erfaringer fra deltakelse i utallige forskjellige systemutviklingsprosjekter over mange år. Avslutningsvis vil også disse proposisjonene bli brukt i kapittel 8 *Konklusjon*, hvor jeg bruker disse proposisjonene til å besvare det endelige forskningsspørsmålet i denne studien.

Kapittel 7

Diskusjon

I dette kapittelet blir de empirisk tilpassede proposisjonene fra forrige kapittel gjennomgått med en kort oppsummering av analysen og en diskusjon om hvilke teoretiske og praktiske implikasjoner funnene har for teorien som ble presentert i kapittel 2 *Prosjektledelse* og kapittel 3 *Utviklingsmetoder*, samt mine egne erfaringer. Det vil bli trekket paralleller til teorien som proposisjonen bygger på, og basert på diskusjonen så kommenterer jeg om funnene bekrefter eller avviker fra teorien.

7.1 Proposisjon 1.1: Forskjeller og ulikheter i systemutviklingsprosjekter

I avsnitt 6.1.3 *Oppsummering og konklusjon* ble proposisjon 1 tilpasset det empiriske datagrunnlaget i kapittel 5 *Empiriske funn* og omformulert til proposisjon 1.1:

> *Forutsetninger for prosjektet og typen av systemutviklingsprosjekt påvirker informasjonsflyten i prosjektet - og følgelig hvilken utviklingsmetodikk eller arbeidsflyt man burde velge for å kunne oppnå suksess.*

Denne proposisjonen omhandler hvordan forutsetninger som omfang, kompleksitet og grad av teknisk gjeld påvirker arbeidsflyten og den praktiske gjennomføringen av prosjektet. Videre oppsummeres analysen fra delkapittel 6.1

Proposisjon 1: Forskjeller og ulikheter i systemutviklingsprosjekter – som bygger på de empiriske funnene.

7.1.1 Oppsummering av proposisjonen

Ulike målsetninger og forventninger til prosjekter styrer arbeidsflyten og gjennomføringen av et systemutviklingsprosjekt. Forskjellene mellom å utvikle noe nytt til fordel for å endre noe allerede eksisterende virker til dels store, og dette har også påvirkning på hvilken utviklingsmetodikk det vil være mest hensiktsmessig å benytte i prosjektet. Forskjeller i forventet responstid, leveransehyppighet og risiko i leveranser trekkes frem som viktige argumenter for dette. Av denne grunn støttes proposisjonen.

7.1.2 Teoretiske og praktiske implikasjoner

Mine funn støtter i stor grad litteraturen som presenteres i avsnitt 3.2.1 *Scrum* – spesielt med tanke på planlegging og samarbeid mot en felles målsetning. I systemutviklingsprosjekter hvor et nytt system skal utvikles så er veldig mye uklart både med tanke på muligheter teknologisk – men også med tanke på prioriteringer. God planlegging fra kompetente fagpersoner (Produkteiere) som kjenner behovene og som evner å kommunisere disse vil være essensielt for sikre et godt samarbeid og at slike prosjekter lykkes med å oppfylle kundene og brukernes egentlige behov. Dette sammenfaller veldig med Mirza og Datta (2019) sine funn både med tanke på intern kommunikasjon og kvalitet i leveransene sett fra et kundeperspektiv og Despa (2014) sine funn med tanke på jevnlige delleveranser med hyppige tilbakemeldinger. Omfang og kompleksitet i prosjekter ses også gjerne i sammenheng med hvor mye planlegging som er nødvendig i et konkret prosjekt. I delkapittel 2.3 *Kompliserte eller komplekse prosjekter* så beskrives det hvordan Project Management Institute (2017) mener at mer kompliserte og komplekse prosjekter øker behovet for adaptive tilnærmelser. Dette fører oss tilbake til hensikten med de iterative og inkrementelle modellene som er beskrevet i avsnitt 3.1.3 *Iterative og inkrementelle modeller* - og som igjen danner grunnlaget for bruken av dagens smidige utviklingsmetoder. Dette støttes også av mine funn både med tanke på målsetningen om økt bruk av mer dokumenterte utviklingsmetoder og informantenes argumentasjon både for og imot bruken av Scrum eller Kanban i deres utviklingsprosjekter.

I avsnitt 3.2.1 *Scrum* så presenteres det utfordringer i Scrum-metodikken med tanke på planlegging av prosjekter som omfatter vedlikehold av eksisterende produkter som allerede er tatt i bruk. Dette på grunn av daglige henvendelser og hyppigere og mindre leveranser med minimert risiko. Mine funn om forskjeller i systemutviklingsprosjekter bekrefter altså uttalelsene fra Ashraf og Aftab (2017) og Naz, Khan og Aamir (2016). Argumentasjonene mot innføringen av Scrum i systemutviklingsprosjekter som omfatter vedlikehold av eksisterende programvare sammenfaller også veldig med funnene til Ahmad, Kuvaja, Oivo og Markkula (2016) som er omtalt i avsnitt 3.2.2 *Kanban*, hvor to vedlikeholdsteam som tidligere hadde brukt Scrum som utviklingsmetodikk opplevde klare fordeler etter at de i stedet gikk over til å benytte Kanban i sine prosjekter.

Ønsket om en økt effektivitet og produktivitet på kort sikt – og at innføringen av Scrum i bedriftens utviklingsprosjekter skulle være løsningen på dette, stemmer derimot mindre med teorien som ble presentert i kapittel 3 *Utviklingsmetoder*. I min oppsummering av styrker og svakheter hos de forskjellige utviklingsmetodene i delkaptittel 3.3 *Oppsummering og utvikling av proposisjoner* så fremgår det at Kanban og ekstrem programmering er de utviklingsmetodene som først og fremst kjennetegnes av dette. Men i denne forbindelse så må skepsisen til to av informantene også nevnes. En av de yngste informantene var redd for at Scrum-metodikken kunne bli for omfattende og fort kunne overskygge den virkelige målsetningen i prosjektene, noe som kan ses i klar sammenheng med *"Zombie-Scrum"* som omtalt i avsnitt 3.2.1 *Scrum* - og en av de mer erfarne informantene som var redd for at oppfølgingen av eksisterende kunder (vedlikeholdsarbeidet) ville bli dårligere.

Mine egne erfaringer med bruk av Scrum – både som systemutvikler og Scrum Master er også varierende med tanke på innholdet i de forskjellige prosjektene som jeg har deltatt i. Prosjekter som er blitt gjennomført på bestilling fra en konkret kunde er nok de prosjektene som jeg selv føler har vært gjennomført best med tanke på Scrum-metodikken. Disse prosjektene er gjerne bedre avgrenset både i omfang og tid, samt at ambisjonen og hensikten gjerne er klarere på bakgrunn av bestillingsprosessen. I andre mer interne vedlikeholdsprosjekter så blir det gjerne fokusert litt for lite på planleggingsfasen – både på grunn av

manglende innsikt i allerede eksisterende kompleksitet og på grunn av mangelen på forutsigbarhet i arbeidsoppgavene som kommer. Dette har gjort at arbeidet i alle sprinter stort sett bare er halvferdig i forhold til forpliktelsen til sprinten – og at det i stedet har blitt tatt inn arbeidsoppgaver som man ikke viste at eksisterte på det tidspunktet hvor arbeidsmengden i sprinten ble planlagt og estimert. Dette kan fort føre til "evigvarende" sprinter og fører veldig ofte til forsinkelser i leveranser og utfordringer med tanke på administrasjonen av prosjekter.

7.2 Proposisjon 2.1: Kjennskap til ulike utviklingsmetodikker

I avsnitt 6.2.3 *Oppsummering og konklusjon* ble proposisjon 2 tilpasset det empiriske datagrunnlaget i kapittel 5 *Empiriske funn* og omformulert til proposisjon 2.1:

Dagens systemutviklere har varierende kjennskap til de mest utbredte og kjente utviklingsmetodikkene som benyttes i bransjen i dag, samt styrker og svakheter med disse – og noen er i tillegg kjent med historikken til de mer tradisjonelle metodene.

Denne proposisjonen omhandler at kjennskap og kunnskap til de forskjellige utviklingsmetodikkene er selve grunnlaget for å kunne utnytte styrkene til disse i gjennomføring og styring av systemutviklingsprosjekter. Videre oppsummeres analysen fra delkapittel 6.2 *Proposisjon 2: Kjennskap til ulike utviklingsmetoder* – som bygger på de empiriske funnene.

7.2.1 Oppsummering av proposisjonen

Delvis på grunn av pågående endringsprosess med tanke på utviklingsmetodikk og noen av informantenes tidligere erfaringer tilsa at case-bedriften var relativt godt informert om de vanligste utviklingsmetodikker som brukes i bransjen i dag. Varierende erfaring og kjennskap til styrker og svakheter med disse tilsa at proposisjonen bare delvis kan støttes i denne studien.

7.2.2 Teoretiske og praktiske implikasjoner

Funnene med tanke på kjennskap til fossefallsmetoden støttes i stor grad av litteraturen som ble presentert i avsnitt 3.1.1 *Fossefallsmetoden*. Dette med tanke på at de sekvensielle og ikke overlappende fasene kan være vanskelig å gjennomføre i dagens systemutviklingsprosjekter – som gjerne omfatter hyppige endringer i løpet av prosjektperioden. Funnene tilsier også at prosjektledere gjerne ønsker seg tilbake til fossefallsmetoden – noe som tolkes i sammenheng med forutsigbarhet og dokumentasjon av arbeidet i prosjektet. Prosjektledere har gjerne ansvar for fremdrift, milepæler og målsetninger – men mangler gjerne en dypere teknisk innsikt i prosjektet. Disse funnene stemmer godt med sammenlikningsstudiene til Munassar og Govardhan (2010) hvor kvalitetssikring igjennom planlegging og dokumentasjon av arbeidet fremstår som en av metodens styrker.

Scrum-metodikken var mye omtalt i funnene som ble presentert i kapittel 5 *Empiriske funn*. Informantenes ønske om økt samarbeid og fokus på felles arbeidsområder og målsetninger støttes av litteraturen som presenteres i avsnitt 3.2.1 *Scrum* både med tanke på avgrensning av arbeidsoppgaver i Sprint'er og teamets forpliktelse til dette arbeidet. Funnene viser også at effektivitet, produktivitet og fremdrift ble nevnt som de viktigste argumentene for beslutningen om å prøve ut Scrum-metodikken i systemutviklingsprosjektene til case-bedriften. Dette støttes ikke av litteraturen som ble presentert i kapittel 3 *Utviklingsmetoder* – hvor andre smidige utviklingsmetoder som Kanban og ekstrem programmering gjerne trekkes frem som de beste alternativene for å optimalisere effektivitet og fremdrift. Diebold, Ostberg, Wagner og Zendler (2015) og Ereiz og Mušić (2019) peker alle på utfordringene med rollesammensetningen og viktigheten med gode og ansvarsfulle personer og roller innad i et Scrum team, og dette i tillegg til all den tiden som må benyttes til jevnlige møter og ritualer som Schwaber og Sutherland (2011) beskriver er nok ikke først og fremst forbundet med en snarlig økt effektivitet og fremdrift.

Funnene viser at Kanban-metodikken også var en kjent utviklingsmetodikk hos informantene i dette forskningsprosjektet. Metodikken eller prosessen de tidligere hadde brukt for gjennomført sine systemutviklingsprosjekter minnet mye om denne metodikken – selv om de ikke omtalte metodikken som Kanban. Funnene tilsier at denne måten å jobbe på hadde fungert veldig bra over lang tid

og at denne over tid var blitt optimalisert for å betjene gjennomførbare kunde ønsker og henvendelser så fort som mulig og få dette testet og levert til bruk i løpet av dager. Dette samstemmer bra med litteraturen omhandlende Kanban metodikken som er presentert i avsnitt 3.2.2 *Kanban* – spesielt med tanke på fordelene som Mirza og Datta (2019) og Ahmad, Dennehy, Conboy og Oivo (2018) beskriver med tanke på redusert gjennomføringstid, fjerning av hindringer og maksimering av prosjektverdien. Når nyutviklingsprosjektene startet og antallet systemutviklere økte så beskriver funnene også relativt store problemer med bruk av denne prosessen eller metodikken med tanke på samarbeid – arbeidet gikk tregere og effektiviteten og produktiviteten gikk ned. En av svakhetene som er beskrevet av Despa (2014) avslutningsvis i avsnitt 3.2.2 *Kanban* omhandler akkurat dette med at metoden er svært avhengig av individuelle og kompetente gruppemedlemmer. Dette vil klart kunne føre til utfordringer med tanke på samarbeid når antall prosjektdeltakere vokser. For meg personlig så er ikke dette overaskende – spesielt ettersom det var en av de yngre informantene som uttrykte de største utfordringene med tanke på akkurat dette.

Funnene viser at metoden ekstrem programmering bare ble nevnt av en av informantene. Informanten hadde ikke selv brukt denne metoden aktivt i sitt utviklingsarbeid – men kunne fortelle at elementer av denne metoden hadde påvirket utviklingen av den egenutviklede metodikken som tidligere hadde blitt benyttet i case-bedriftens systemutviklingsprosjekter. Fordelen med bruken av denne metoden som er beskrevet i funnet stemmer godt overens med litteraturen som er beskrevet i avsnitt 3.2.3 *Ekstrem programmering* – spesielt i forhold til Dudziak (2000) med tanke på effektiv, repeterbar og forutsigbar utvikling av programvare og Newkirk (2002) med tanke på metodens evne til forbedret kommunikasjon, testing og tilbakemeldinger.

For min egen del så startet jeg karrieren som systemutvikler på et tidspunkt hvor prosjektgjennomføring enda omfattet elementer fra fossefallsmetoden – selv om de smidige metodene hadde begynt å etablere seg i bransjen. De smidige metodikkene har etter dette selvfølgelig fått mer og mer fokus og har på mange vis tatt over med tiden, men når jeg tenker tilbake på den første tiden i min karriere så gikk dette stort sett ut på å lese dokumentasjonen som allerede var produsert i prosjektet som jeg skulle bidra inn i – det fantes faktisk dokumenta-

sjon om både målsetningen med prosjektet, samt relativt fersk dokumentasjon om status. For min del var dette en klar fordel med tanke på å komme inn i et allerede pågående prosjekt – med målsetning om å bli produktiv så fort som mulig. De siste årene har jeg tatt imot utallige nye og ferske utviklere hvor de ofte må bidra inn i prosjekter som allerede er pågående - og i dagens smidige metoder (hvor det meste av informasjon om prosjektet har blitt avtalt muntlig og stort sett befinner seg i hodene på folk) har dette ofte bydd på utfordringer.

Scrum har helt klart vært den metodikken som har vært mest brukt i de prosjektene som jeg har bidratt til i løpet av min karriere. Årsaken til dette tror jeg er at dette er den mest omtalte og brukte metodikken – noe som har bred seg som en slags standard for gjennomføring av systemutviklingsprosjekter. Når det er sagt og med tilbakeblikk på dette så mener jeg kanskje at metodikken ikke skulle blitt brukt i mange av prosjektene. Flere av disse prosjektene har vært relativt små prosjekter med to til tre deltakere – hvor innvesteringen med å ta i bruk en relativt omfattende metodikk har vært for stor i forhold til omfanget i prosjektet. Denne realiteten har også ført til mange delte og uklare roller – noe som igjen har ført til dårlig eller manglende prioriteringer med følgende dårlige og forsinkende leveranser til kunder. I ettertid så tenker jeg at disse prosjektene ville vært mye enklere å gjennomføre hvis en enklere metodikk hadde blitt valgt – med mer fokus på prioriteringer til fordel for metodikk. I andre prosjekter med et mer omfattende omfang og klarere roller føler jeg derimot at vi har hatt mye mer igjen for innvesteringen med å innføre Scrum. Omfanget av prosjektet og hvor klare forutsetningene faktisk er før oppstart av programmering i slike systemutviklingsprosjekter er derfor det jeg i dag tenker på ved valg av utviklingsmetodikk – i tillegg til kompetansen til de konkrete prosjektdeltakerne i forhold til ansvarsområder og rollefordeling.

7.3 Proposisjon 3.1: Valg av utviklingsmetodikk i systemutviklingsprosjekter

I avsnitt 6.3.3 *Oppsummering og konklusjon* ble proposisjon 3 tilpasset det empiriske datagrunnlaget i kapittel 5 *Empiriske funn* og omformulert til proposisjon 3.1:

Organisasjoner som utøver stadig mer kompleks systemutvikling etterstreber økt bruk av smidige utviklingsmetodikker som for eksempel Scrum og Kanban i sine systemutviklingsprosjekter for å oppnå forutsigbarhet, bedre samarbeid og økt effektivitet.

Proposisjonen omhandler en kjent problemstilling i bransjen som bedriver med systemutvikling – nemlig at en stadig økende kompleksitet og omfang krever mer omfattende styring og forutsigbar gjennomføring av prosjektene. Videre oppsummeres analysen fra delkapittel 6.3 *Proposisjon 3: Valg av utviklingsmetodikk i systemutviklingsprosjekter* – som bygger på de empiriske funnene.

7.3.1 Oppsummering av proposisjonen

Case-bedriften var som sagt midt inne i en endringsprosess med tanke på bruk av smidig utviklingsmetodikk når dette forskningsprosjektet ble gjennomført. Argumentene for endringen var ønske om økt effektivitet og fremdrift og funnene viser også at samarbeidet internt i noen av utviklerteamene måtte forbedres for å oppnå ønsket effekt. Med dette som grunnlag kunne proposisjonen støttes av denne studien.

7.3.2 Teoretiske og praktiske implikasjoner

Funnene viser at case-bedriften i løpet av endringsprosessen hadde besluttet å ta i bruk Scrum som utviklingsmetodikk i sine to utviklingsteam som bedriver nyutvikling av nye produkter, mens Kanban ble vurdert som mer hensiktsmessig for utviklerteamet som vedlikeholder og videreutvikler de eksisterende produktene. Både Scrum og Kanban er to velkjente utviklingsmetodikker i systemutviklingsbransjen – og statistikken over de mest brukte utviklingsmetodene de siste årene som presenteres i delkapittel 3.2 *Smidige utviklingsmetoder* stemmer god overens med disse funnene. Statistikken viser for øvrig en nedgang i utbredelse hos nesten alle de kjente smidige utviklingsmetodikkene fra 2018 til 2020 – noe som isolert sett taler imot proposisjonen. Dette kan dog forklares med utbruddet av Covid-pandemien, noe som førte til økt bruk av hjemmekontor, mer individuelt arbeid og utfordringer med tanke på samarbeid og samhandling. Det ville vært interessant å se en mer oppdatert statistikk angående dette og dens utvikling i etterkant av denne pandemien.

Funnene som presenteres i delkapitlene 5.2 *Generelt om utviklingsmetoder* og 5.3 *Ulike typer av systemutviklingsprosjekter* viser at det er ulike meninger både om styrker og svakheter i utviklingsmetodikkene Scrum og Kanban. Flere uttalte at et ønske om økt samarbeid og kompetanseheving var argumenter som veide tungt i valget om å innføre Scrum som utviklingsmetodikk i bedriftens nyutviklingsprosjekter. Dette argumentet samsvarer veldig med fordelene til Scrum som ble presentert i avsnitt 3.2.1 *Scrum*. Et ønske om økt effektivitet og produktivitet ble også uttalt som et av argumentene for innføringen av Scrum. Dette samsvarer ikke med beskrivelsen av Scrum slik den fremgår i avsnitt 3.2.1. *Scrum* – hvor Scrum kjennetegnes av økt kundetilfredshet, klar rollefordeling og omfattende planlegging. Det at bedriften hadde valgt å rendyrke Kanban som utviklingsmetode for de prosjektene som vedlikeholdt eksisterende programvare stemmer derimot bedre overens med beskrivelsen av Kanban i avsnitt 3.2.2 *Kanban* både med tanke på effektivitet som et resultat av fokus på mindre prioriterte oppgaver og økt behov for kompetente individualister til fordel for samarbeid mellom prosjektdeltakere.

For min egen del så har jeg fulgt utviklingen med hensyn på utviklingsmetodikk via deltakelse og gjennomføring av utallige systemutviklingsprosjekter fra 2006 til skrivende stund. I starten på denne perioden så var det fortsatt vanlig med gjennomføring av systemutviklingsprosjekter med elementer fra fossefallsmetoden som omfattet betydelige krav til beskrivelser og dokumentasjon – men Scrum-metodikken fikk relativt raskt fotfeste i organisasjonene som jeg har vært en del av. De aller fleste systemutviklingsprosjektene av en viss størrelse har etter dette blitt gjennomført mer eller mindre etter Scrum – med alle fordelene og ulempene dette har medført. Jeg har også selv deltatt i noen systemutviklingsprosjekter hvor Kanban har blitt benyttet – men da har prosjektet gjerne bestått av større eller mindre "ad-hoc" oppgaver som har vært vanskelig å planlegge. Mine egne erfaringer angående bruk av dokumenterte utviklingsmetoder viser altså at det ikke er noe som tyder på at fokuset på bruk av disse blir noe særlig mindre i fremtiden – men at den stadig økende teknologiske utviklingen og kompleksiteten i slike prosjekter gjør at fokuset på valg av mest hensiktsmessig metodikk for et konkret utviklingsprosjekt kanskje blir høyere.

7.4 Oppsummering

I dette kapittelet er de empirisk tilpassede proposisjonene gjennomgått i kronologisk rekkefølge og diskutert med tanke på hvilke teoretiske og praktiske implikasjoner funnene har for hver enkelt proposisjon. Diskusjonen av proposisjon 1 bidrar til litteraturen som trekker frem ulikheter mellom forskjellige typer av systemutviklingsprosjekter, samt styrker og svakheter med bruken av de forskjellige utviklingsmetodikkene for gjennomføring av disse prosjektene. Proposisjon 2 bidrar til litteraturen med den faktiske kjennskapen og utbredelsen av de kjente og dokumenterte utviklingsmetodikkene, mens proposisjon 3 synliggjør fokuset på bruken av disse metodikkene fra et organisasjonsperspektiv.

Igjennom analysen og diskusjonen ser vi at funnene i dette forskningsprosjektet stort sett støtter mange av synspunktene og uttalelsene som presenteres i kapittel 2 *Prosjektledelse* og kapittel 3 Utviklingsmetoder. Dette med hensyn på den faktiske kjennskapen av de ulike utviklingsmetodikkene, - samt styrker og svakheter med disse i praktisk bruk. Uttalelsene som derimot ikke støttes av litteraturen er innføringen av Scrum-metodikken med tanke på økt effektivitet. Scrum-metodikken kan gjerne føre til økt forutsigbarhet og kundetilfredshet – men ikke økt effektivitet på kort sikt. Økt effektivitet kjennetegnes heller av andre smidige metoder som Kanban og ekstrem programmering – se avsnittene 3.2.2 *Kanban* og 3.2.3 Ekstrem programmering.

I neste kapittel formuleres det et svar på forskningsspørsmålet før mine egne personlige betraktninger blir presentert. Avslutningsvis i dette kapittelet vil tema som kan være interessant for videre forskning også bli diskutert.

Kapittel 8

Konklusjon

Formålet med dette kapittelet er å oppsummere masteroppgaven og besvare forskningsspørsmålet:

Hva kjennetegner programvareutviklingsprosjekter hvor det ikke vil være hensiktsmessig å benytte "Scrum" som utviklingsmetode?

I kapittel 6 *Analyse* analyseres de tre opprinnelige proposisjonene med bakgrunn i de empiriske funnene som ble presentert i kapittel 5 *Empiriske funn* og ble videre omformulert med bakgrunn av disse funnene. I det neste delkapittelene vises det hvordan de empirisk tilpassende proposisjonene er behjelpelig med å besvare forskningsspørsmålet. I delkapittel 8.3 *Avslutning* presenteres mine egne personlige betraktninger og avslutningsvis i delkapittel 8.4 *Videre forskning* diskuteres videre forskning som kan være interessant på dette området.

8.1 Konklusjon på delspørsmålene

Forskningsspørsmålet er delt opp i tre delspørsmål hvor det er utviklet og videreutviklet en proposisjon til hver av disse. Se kapittel 1 *Introduksjon* for oppdeling av delspørsmål og delkapittel 6.4 *Oppsummering og empirisk tilpasset rammeverk* for oppsummeringen av de empirisk tilpassede proposisjonene.

For å besvare forskningsspørsmålet blir hvert delspørsmål gjennomgått i kronologisk rekkefølge for å se på hvilken måte proposisjonene underbygger konklusjonen på delspørsmålet med bakgrunn i funnene fra forskningsprosjektet.

8.1.1 Konklusjon på delspørsmål 1

I kapittel 1 *Introduksjon* så formuleres delspørsmål 1 slik:

Finnes det typer av programvareutviklingsprosjekter som favorittseirer en utviklingsmetode fremfor en annen?

I delkapittel 6.1.3 *Oppsummering og konklusjon* så konkluderes det med at forutsetninger og andre egenskaper som riktig kompetanse og erfaring sammen med andre viktige egenskaper som oppetid, responstid og levereansehyppighet påvirker arbeidsflyten og gjennomføringen til systemutviklingsprosjekter. Typen av prosjekt er også meget aktuell i denne sammenheng og for systemutviklingsprosjekter så viser de empiriske funnene at den største forskjellen i systemutviklingsprosjekter kommer an på om prosjektets hensikt er å utvikle ny programvare eller bedrive vedlikehold og tilpasninger i allerede eksisterende programvare.

I hvilken grad det er mulig å planlegge et systemutviklingsprosjekt avhenger av mange faktorer. I større prosjekter så kan også nye faktorer dukke opp i løpet av prosjektperioden – som man ikke hadde oversikt over i starten på prosjektet. Disse ukjente faktorene er det som gjør planlegging av alle prosjekter vanskelig – ikke bare systemutviklingsprosjekter. For å kunne gjennomføre en god planlegging av et systemutviklingsprosjekt så kreves det relativt stor innsikt og oversikt over både behov og teknologiske muligheter, og hvis man skal videreutvikle noe som allerede finnes så kreves det selvfølgelig en spesiell innsikt i det som skal videreutvikles. Hvis systemet i tillegg benyttes av eksisterende kunder og brukere som samtidig må følges opp av de samme prosjektdeltakerne så sier det seg selv at det blir vanskelig å planlegge og estimere arbeid i tid. Man vet stort sett ikke hvilke henvendelser eller feil som måtte dukke opp i den eksisterende programvaren frem i tid. Dette er stort sett enklere når prosjektet omfatter nyutvikling – ettersom man da ikke trenger å forholde seg til eksisterende programvare eller den daglige oppfølgingen av eksisterende kunder.

Delspørsmålet konkluderes med at forskjellige typer av programvareutviklingsprosjekter vil ha klare ulikheter med tanke på informasjonsflyt, planlegging, gjennomføring og sluttføring, og at dette følgelig setter føringer for hvilken utviklingsmetodikk som burde benyttes. I hvilken grad arbeidet faktisk kan plan-

legges, avvik og avbrytelser fra det planlagte utviklingsarbeidet, samt forventet leveransehyppighet og risiko i leveranser er viktige faktorer som bør veies før denne beslutningen tas.

Et spesielt interessant funn som klart kan underbygge denne konklusjonen er at utviklingslederen som hadde vært med helt fra når organisasjonen begynte med systemutvikling uttalte at de begynte å ta i bruk Scrum i begynnelsen – men at det ble mer og mer vanskelig å gjennomføre denne metodikken etter hvert som at organisasjonens kundemasse, bruken av systemet og antall daglige henvendelser vokste. Når de nå skulle starte med nyutvikling igjen så hadde de i utgangspunktet besluttet å innføre Scrum i alle prosjekter igjen – men at denne beslutningen ble omgjort i løpet av prosessen og at prosjektet som var konsentrert rundt oppfølging av eksisterende kunder heller skulle prøve å rendyrke Kanban som metodikk for sin prosjektgjennomføring.

8.1.2 Konklusjon på delspørsmål 2

I kapittel 1 *Introduksjon* så formuleres delspørsmål 2 slik:

> *Hvilke alternative utviklingsmetoder er mest utbredt i bransjen - og hva er egentlig kjennskapen og bruken av disse i bransjen i dag?*

I avsnitt 6.2.3 *Oppsummering og konklusjon* og avsnitt 7.2.1 *Oppsummering av proposisjonen* så ser vi at Scrum og Kanban er de mest omtalte og aktuelle smidige utviklingsmetodikkene i denne studien, og at dette er veldig omforent med statistikken over bruk av smidige utviklingsmetodikker som ble presentert i delkapittel 3.2 *Smidige utviklingsmetoder*.

Studien viser også at organisasjoner gjerne tilpasser dokumenterte utviklingsmetodikker slik at de passer til organisasjonens hverdag. Case-bedriften hadde jobbet etter en egenutviklet metodikk som minnet veldig om Kanban i utgangspunktet – men hadde også elementer som omtales både i beskrivelsen av Scrum og ekstrem programmering.

Fokuset på bruk av allerede dokumenterte og tilgjengelige utviklingsmetodikker var selvfølgelig stort i case-bedriften på det tidspunktet at studien og forskningsprosjektet ble gjennomført, spesielt med tanke på bedriftens pågående endrings-

prosess som presenteres i delkapittel 5.2 *Generelt om utviklingsmetoder* – men store deler av organisasjonen manglet fortsatt inngående kjennskap og erfaring med bruk av dokumenterte metodikker.

Fossefallsmetoden ble også nevnt i mange sammenhenger i forbindelse med gjennomføring av intervjuene uten at denne metoden ble fremstilt som et reelt alternativ for case-bedriftens prosjektgjennomføring. Det er uansett interessant at metodens fokus på å dele prosjektet inn i faser og dokumentasjon trekkes frem som styrker med tanke på styring av prosjekter – og at dette er til fordel for prosjektlederen. Mangelen på styring og dokumentasjon trekkes gjerne frem som svakheter i de mer moderne smidige metodene, se for eksempel avsnitt 3.2.1 *Scrum* og 3.2.3 *Ekstrem programmering*. Selv om fossefallsmetoden ikke regnes som et reelt alternativ i bransjen i dag så kan dette være noe å tenke på spesielt med tanke på planlegging og dokumentasjon før utviklingsdelen i et systemutviklingsprosjekt, samt dokumentasjon i forbindelse med sluttføring.

Delspørsmålet konkluderes altså med at Scrum, Kanban og til dels ekstrem programmering er de mest utbredte utviklingsmetodikkene i bransjen i dag. Alle disse metodikkene er godt dokumentert og lett tilgjengelige for bruk. Kunnskap om disse metodikkene er også mulig å bygge ved hjelp av både kursing og sertifiseringer. Erfaringer med tanke på styrker og svakheter på de forskjellige metodikkene er også uttalt i mange forum – men for en organisasjon så er det den praktiske erfaringen som vil kunne avgjøre hvilken utviklingsmetodikk som passer best for et konkret prosjekt.

8.1.3 Konklusjon på delspørsmål 3

I kapittel 1 *Introduksjon* så formuleres delspørsmål 3 slik:

Hva kjennetegner et programvareutviklingsprosjekt der man bør velge "Scrum" som utviklingsmetode?

I avsnitt 7.1.2 *Teoretiske og praktiske implikasjoner* så diskuteres det utfordringer i Scrum-metodikken med tanke på planlegging av prosjekter som omfatter vedlikehold av eksisterende produkter som allerede er tatt i bruk – dette på grunn av den samtidige kundeoppfølgingen og avbrytelsene som kommer som et følge av dette. Flere av informantene til forskningsprosjektet uttaler også

at Scrum-metodikken passer best for nyutvikling og gjerne som del av en avgrenset konsulelentvirksomhet med mer avgrenset og klarlagt prosjektomfang. Scrum krever også ganske mye med tanke på rollefordeling og forpliktelser til sitt ansvarsområde – sett i forhold til de andre smidige utviklingsmetodikkene som er omtalt i denne studien.

Planlegging av et systemutviklingsprosjekt og i hvilken grad det faktisk er mulig å planlegge og verifisere arbeidet er også en viktig faktor hvis Scrum-metodikken vurderes som metodikk for et prosjekt. Scrum-metodikken krever at det er mulig å dele opp prosjektet i mindre arbeidspakker som utføres i tidsbestemte sprinter – noe som følgelig krever at det må være mulig å planlegge og estimere utviklingsarbeidet før utviklingen faktisk kan utføres.

Delspørsmålet konkluderes med at programvareutviklingsprosjekter der det vil kunne være hensiktsmessig å benytte Scrum som utviklingsmetode først og fremst kjennetegnes av at dette er et prosjekt av en viss størrelse og at organisasjonen er robust nok til å fylle de viktigste rollene i Scrum-metodikken - og at disse faktisk har tid til å ta dette ansvaret på alvor. Denne studien viser at den mest avgjørende faktoren for om Scrum -metodikken vil fungere i et konkret prosjekt er graden av daglig oppfølging av eksisterende kunder og andre uforutsette henvendelser. Dette gjør at jeg også kan konkludere med at større nyutviklingsprosjekter med godt dokumenterte behov og liten grad av avbrytelser eller avvik fra det planlagte arbeidet vil passe best for prosjektgjennomføring med Scrum-metodikken.

Med prosjektgjennomføring med bruk av Scrum så vil man kunne oppleve et økt fokus på delleveranser og forbedret samarbeid på disse delene av prosjektet – men man kan ikke være avhengig av å omprioritere oppgaver og ressurser jevnlig. Prosjektet kan heller ikke være avhengig av leveranser utenom de planlagte og avsatte sprintene for ikke å forstyrre gjennomføringen både med tanke på omfang og tid. Dette underbygges med case-bedriftens endrede beslutning om å ikke benytte Scrum i sitt prosjekt som videreutvikler og forvalter de eksisterende produktene som er presentert i kapittel 5.3 *Ulike typer av systemutviklingsprosjekter*.

8.2 Konklusjon på forskningsspørsmålet

I dette kapittelet har jeg gjennomgått de tre delspørsmålene i kronologisk rekkefølge og konkludert på disse med bakgrunn av innhold i denne studien og funn i forskningsprosjektet som denne studien omfatter. Videre vil jeg konkludere på forskningsprosjektet med bakgrunn i konklusjonen på de tre delspørsmålene.

Denne masteroppgaven konkluderes med at det ikke vil være hensiktsmessig å benytte Scrum som utviklingsmetodikk i systemutviklingsprosjekter som omfatter vedlikehold av eksisterende programvare og oppfølging av daglige kundehenvendelser.

Argumentene for denne konklusjonen er at det gjerne vil være vanskelig å gjennomføre en god planlegging frem i tid i slike prosjekter – rett og slett fordi man ikke kan forutse hva som måtte dukke opp i løpet av den neste perioden. Når feil og utfordringer hos allerede eksisterende kunder og brukere oppstår så må dette veldig ofte løses så fort som mulig og passer ofte ikke med beskrivelsen av Scrum - med omfattende planlegging, estimering og prioritering før utvikling starter og tidligst kan leveres etter neste planlagte sprint er gjennomført. Avbrytelser og avvik fra det opprinnelige planlagte arbeidet vil også kunne føre til at tidspunktet hvor sprinter fullføres flyttes frem i tid som følge av at arbeidsomfanget endres. Dette vil også kunne føre til at endelige leveranser og milepæler trekker ut i tid noe som kan ha både økonomiske og strategiske konsekvenser. Hvis kort leveransetid, effektivitet og produktivitet er de viktigste argumentene for valget av utviklingsmetodikk så vil det være mer hensiktsmessig å heller velge en utviklingsmetodikk som faktisk kjennetegnes av dette – som Kanban eller ekstrem programmering.

Denne konklusjonen samsvarer også mye med IEEEs publikasjon fra 2015 hvor utfordringene med å gjennomføre komplekse programvareprosjekter med smidige utviklingsmetodikker beskrives. I denne publikasjonen så konkluderer de også med at utviklingsmetodikken *ekstrem programmering* er best egnet i prosjekter hvor man må akseptere hyppige endringer og leveranser i løpet av gjennomføringsperioden (Wingo og Tanik, 2015).

Denne konklusjonen samsvarer derimot ikke med uttalelser som at Scrum-metodikken passer for alle prosjekter (Quick, 2023), og at Scrum-metodikken alene kan redusere alt arbeid med 50% (Jakobsen og Sutherland, 2009).

Funnene som har ført frem til denne konklusjonen viser at systemutviklingsprosjekter i bransjen i dag tenderer til å legge for mye vekt på metodikk og prosesser i prosjektene – noe som går på bekostning av den virkelige kundeverdien som er grunnlaget for disse prosjektene. Fremover ville jeg foreslått at bransjen burde fokusere mer på forutsetninger, målsetninger og resultat i prosjektene – fremfor metodikk, rutiner og prosess.

8.3 Avslutning

I dette delkapittelet så oppsummerer jeg mine egne personlige betraktninger med gjennomføring av dette forskningsprosjektet og skrivingen av denne masteroppgaven.

Problemstillingen for denne oppgaven er utarbeidet på bakgrunn av mine egne erfaringer og utfordringer så var selvfølgelig motivasjonen for forskningen og temaet veldig stort – samtidig så har ikke mine tidligere studier eller jobbforhold omfattet akademisk skriving i særlig stor grad – og ikke i nærheten av hvordan denne oppgaven er strukturert. Dette har derfor vært en læringsprosess for min egen del, som til tider har vært relativt tung og frustrerende – men hvor veilederen for denne oppgaven har vært til stor hjelp.

Med tanke på forskingsprosjektet, og gjennomføringen av den kvalitative undersøkelsen med semistrukturerte intervjuer så var jeg i utgangspunktet spent på utfallet av dette. Mine tidligere tekniske studier har heller ikke omfattet forskning og undersøkelser med slik metodebruk. Før jeg kom i kontakt med ledelsen til firmaet som til slutt ble case-bedriften i dette forskningsprosjektet så var jeg redd for at det ville bli vanskelig å få samlet nok relevant data om teamet for oppgaven – aller helst fra en organisasjon som jeg selv ikke er en del av. Jeg vet selv hvor vanskelig det er å sette av tid til slikt i en ellers travel hverdag med korte tidsfrister. Denne delen av forskningsprosjektet gikk som tidligere nevnt over all forventning – og jeg hadde nok en del flaks med tanke

på endringsprosessen som case-bedriften faktisk sto overfor akkurat med tanke på mitt tema for oppgaven. Antallet semistrukturerte intervjuene som til slutt ble gjennomført kunne helt sikkert vært flere – men disse var de som stilte opp etter noen runder med påminnelser og flyttinger av avtaler. Samtidig så må det nevnes at de fem intervjuene et relativt høyt tall med tanke på størrelsen på case-bedriften - og at sammensetningen av informantene ga meg et relativt klart bilde av case-bedriftens utfordringer sett fra forskjellige sider. Dette gjør at jeg ikke er sikker på om det faktisk hadde dukket opp noe nytt hvis jeg bare hadde gjennomført flere intervjuer i denne organisasjonen. Tidsaspektet for gjennomføringen av denne masteroppgaven er selvfølgelig også en begrensende faktor i denne forbindelse.

En annen betraktning som må nevnes er knyttet til endringsprosessen som case-bedriften var inne i med tanke på metodikkbruk i forbindelse med deres systemutvikling. Når jeg gjennomførte de første intervjuene med systemutviklerne så holdt de på med å ta kurs i bruk av Scrum – på bakgrunn av en beslutning om at Scrum skulle innføres i alle systemutviklingsprosjektene til denne organisasjonen. Når jeg hadde den oppsummerende samtalen med utviklingslederen en god stund etter så var denne beslutningen omgjort til bare å gjelde nyutviklingsprosjektene. I forhold til forskningsprosjektet så har jeg prøvd å forholde meg så objektiv og nøytral som mulig – men jeg kan ikke garantere at mitt forskningsprosjekt og fokuset på alternative metodikker for systemutvikling ikke har hatt påvirkning på denne siste beslutningen til organisasjonen. Som Tjora beskriver så vil det forekomme påvirkning begge veier i en observasjonsstudie – men at dette ikke bør være et hinder for ikke å gjennomføre det (Tjora, 2012).

8.4 Videre forskning

For denne studien var det opprinnelig en tanke om å foreta en kvantitativ undersøkelse. Årsakene til dette var at jeg da potensielt kunne nå frem til et bredere spekter av bransjen for å få belyst større deler av utfordringene som forekommer i forbindelse med gjennomføring av profesjonelle utviklingsprosjekter. Det ville derfor vært spennende å utføre dette – for å se om resultatene av en kvantitativ undersøkelse avviker eller bekrefter funnene som jeg har gjort i dette forskningsprosjektet.

Av tidshensyn er denne case-studien gjennomført i bare en organisasjon. En bredere studie som omfatter flere andre organisasjoner ville derfor også vært spennende – ettersom man ved en case-studie med bare en case-bedrift får et relativt avgrenset situasjonsbilde av organisasjonen og dets utfordringer – men disse trenger nødvendigvis ikke å være omforent med andre organisasjoners utfordringer og tanker. Dette er beskrevet i litteraturen som en *multi-case studie* som forsøker å finne ulikheter mellom flere forskjellige case (Clark m. fl., 2021). I denne sammenheng ville det kanskje vært spesielt spennende å sammenlikne tilsvarende resultater med en bedrift som ikke bedriver egen produktutvikling, men utfører konsulentvirksomhet på oppdrag for andre. Forskjellene her er at slik konsulentvirksomhet ofte er mye mer variert, samt at kravet til formalisert kompetanse er høyere. Jeg blir ikke overasket om resultatene i en slik studie ville vært merkbart annerledes enn de som er presentert i form av denne studien.

Referanser

Ahmad, M. O., Dennehy, D., Conboy, K., og Oivo, M. (2018). Kanban in software engineering: A systematic mapping study. *Journal of Systems and Software*, 137:96–113.

Ahmad, M. O., Markkula, J., og Oivo, M. (2013). Kanban in software development: A systematic literature review. I *39th Euromicro conference on software engineering and advanced applications*, sider 9–16, Oulu. IEEE.

Al-Ahmad, W., Al-Fagih, K., Khanfar, K., Alsamara, K., Abuleil, S., og Abu-Salem, H. (2009). A taxonomy of an it project failure: root causes. *International Management Review*, 5(1):93–104.

Al-Saqqa, S., Sawalha, S., og AbdelNabi, H. (2020). Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies*, 14(11):246–270.

Alliance, A. (2017). *Agile Practice Guide*. Project Management Institute, Pennsylvania.

Alshamrani, A. og Bahattab, A. (2015). A comparison between three sdlc models waterfall model, spiral model, and incremental/iterative model. *International Journal of Computer Science Issues (IJCSI)*, 12(1):106–111.

Anderson, D. J. (2010). *Kanban: successful evolutionary change for your technology business*. Blue Hole Press, Washington.

Angelov, S., Meesters, M., og Galster, M. (2016). Architects in scrum: What challenges do they face? I *European Conference on Software Architecture*, sider 229–237, København. Springer.

Ashraf, S. og Aftab, S. (2017). Iscrum: An improved scrum process model. *International Journal of Modern Education & Computer Science*, 9(8):16–24.

Atkinson, C. og Hummel, O. (2012). Iterative and incremental development of component-based software architectures. I *Proceedings of the 15th ACM SIGSOFT symposium on component based software engineering*, sider 77–82, Mannheim. Software Engineering Group.

Atkinson, R. (1999). Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International journal of project management*, 17(6):337–342.

Awad, M. A. (2005). A comparison between agile and traditional software development methodologies. *Honours Programme of the School of Computer Science and software Engineering*, 30:1–69.

Bailey, K. D. (1994). *Typologies and taxonomies: An introduction to classification techniques*. Sage, Los Angeles.

Beck, K. (2000). *Extreme programming explained: embrace change*. Addison-Wesley professional, Boston.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., m. fl. (2001). Manifesto for agile software development. *Resultat fra møte på Snowbird ski resort i Wasatch fjellene i Utah*, sider 1–10.

Bell, J. T. (2001). Extreme programming. *Thinking for Innovation*, sider 1–19.

Bennett, A. (2004). Case study methods: Design, use, and comparative advantages. *Models, numbers, and cases: Methods for studying international relations*, 2(1):19–55.

Berg, H. (2021). *Hvordan lykkes med digitalisering? : en undersøkelse av nyttestyring i IT-prosjekter i offentlig sektor*. Concept-rapport nr. 64 (trykt utg.). Ex ante akademisk forlag, Trondheim.

Briner, W., Hastings, C., og Geddes, M. (2000). *Prosjektledelse*. Gyldendal akademisk, Oslo.

Brustad, R. og Jarle, I. (2001). *Prosjektstyring*. Gyldendal yrkesopplæring, Oslo.

Bryman, A. (2016). *Social research methods*. Oxford university press, Oxford, 5. utgave.

Cajander, Å., Larusdottir, M., og Gulliksen, J. (2013). Existing but not explicit-the user perspective in scrum projects in practice. I *IFIP Conference on Human-Computer Interaction*, sider 762–779, Cape Town. Springer.

Clark, T., Foster, L., Bryman, A., og Sloan, L. (2021). *Bryman's social research methods*. Oxford University Press, Oxford, 6. utgave.

Crawford, L., Hobbs, J. B., og Turner, J. R. (2005). *Project categorization systems : aligning capability with strategy for better results*. Project Management Institue, Pennsylvania.

Dalalah, A. (2014). Extreme programming: Strengths and weaknesses. *Computer Technology and Application*, 5(1):15–20.

Despa, M. L. (2014). Comparative study on software development methodologies. *Database Systems Journal*, 5(3):37–56.

Diebold, P., Ostberg, J.-P., Wagner, S., og Zendler, U. (2015). What do practitioners vary in using scrum? I *International Conference on Agile Software Development*, sider 40–51, Cham. Springer.

Dudziak, T. (2000). Extreme programming an overview. *Methoden und Werkzeuge der Softwareproduktion WS*, 1999:1–28.

Ereiz, Z. og Mušić, D. (2019). Scrum without a scrum master. I *IEEE International Conference on Computer Science and Educational Informatization (CSEI)*, sider 325–328, Kunming. IEEE.

Foster, E. C. og Towle Jr, B. A. (2021). *Software engineering: a methodical approach*. CRC Press, Abingdon, 2. utgave.

Galster, M., Angelov, S., Martínez-Fernández, S., og Tofan, D. (2017). Reference architectures and scrum: friends or foes? I *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, sider 896–901, New York. Association for Computing Machinery.

Hansson, K. W. (2013). Objektorientert analyse og design med uml. *HiBu: Arbeidsnotater fra Høgskolen i Buskerud*.

Hassani-Alaoui, S., Cameron, A.-F., og Giannelia, T. (2020). "we use scrum, but...": Agile modifications and project success. I *Proceedings of the 53rd Hawaii International Conference on System Sciences*, sider 6257–6266.

Högstrand, J. (2019). *E-bok: Det smidige landskapet*. Metier OEC, Oslo.

Institute, P. M. (2001). *The standard for project management and A guide to the project management body of knowledge : (PMBOK® guide)*. Project Management Institute Inc, Pennsylvania, 7. utgave.

Jakobsen, C. R. og Sutherland, J. (2009). Scrum and cmmi going from good to great. I *Agile Conference*, sider 333 – 337, Chicago. IEEE.

Kerzner, H. (2017). *Project management: a systems approach to planning, scheduling, and controlling*. John Wiley & Sons, New Jersey, 12. utgave.

Kwak, Y. H. (2005). A brief history of project management. *The story of managing projects: An interdisciplinary approach*, 9(1):1–10.

Larsen, A. K. (2007). *En enklere metode: veiledning i samfunnsvitenskapelig forskningsmetode*. Fagbokforlaget, Bergen, 1. utgave.

Lucky, E. O.-I., Adegoke, O., og Nordin, N. (2014). Project management challenges and difficulties: A case study of information system development. *International Postgraduate Business Journal*, 6(1):99–113.

McConnell, S. (1996). *Rapid development: taming wild software schedules*. Microsoft Press, Washington.

Mirza, M. S. og Datta, S. (2019). Strengths and weakness of traditional and agile processes-a systematic review. *Journal of Software*, 14(5):209–219.

Morris, P. W. G., Pinto, J., og Söderlund, J. (2011). *The Oxford Handbook of Project Management*. Oxford University Press, Oxford.

Muffatto, M. (2006). *Open source: A multidisciplinary approach*. World Scientific, London.

Munassar, N. M. A. og Govardhan, A. (2010). A comparison between five models of software engineering. *International Journal of Computer Science Issues (IJCSI)*, 7(5):94–101.

Naz, R., Khan, M., og Aamir, M. (2016). Scrum-based methodology for product maintenance and support. *International Journal of Engineering and Manufacturing (IJEM)*, 6(1):10–27.

Newkirk, J. (2002). Introduction to agile processes and extreme programming. I *Proceedings of the 24th International Conference on Software Engineering. ICSE 2002*, sider 695–696, Chicago. IEEE.

Ozkan, N. og Gök, M. S. (2021). How scrum inhibits agility. I *15th Turkish National Software Engineering Symposium (UYMS)*, sider 1–6, Kocaeli. IEEE.

Petersen, K., Wohlin, C., og Baca, D. (2009). The waterfall model in large-scale development. I *International Conference on Product-Focused Software Process Improvement*, sider 386–400, Berlin. Springer.

Poppendieck, M. og Poppendieck, T. (2003). *Lean software development: an agile toolkit*. The Agile software development series. Addison-Wesley, Boston.

Quick, L. (2023). Does scrum apply to all types of projects? https://www.knowledgehut.com/blog/agile/does-scrum-apply-to-all-types-of-projects. Lest: 2023-08-14.

Rehman, F. u., Maqbool, B., Riaz, M. Q., Qamar, U., og Abbas, M. (2018). Scrum software maintenance model: Efficient software maintenance in agile methodology. I *21st Saudi Computer Society National Computer Conference (NCC)*, sider 1–5, Riyadh. IEEE.

Rolstadås, A., Olsson, N., Johansen, A., og Langlo, J. A. (2014). *Praktisk prosjektledelse : fra idé til gevinst*. Fagbokforlaget, Bergen, 1. utgave.

Royce, W. W. (1970). Managing the development of large software systems: concepts and techniques. *Proceedings of IEEE WESTCON, Los Angeles*, sider 1–9. Utgitt på nytt i *Proceedings* of the Ninth International Conference on Software Engineering, Mars 1987, sider 328–338.

Samset, K. F. (2014). *Prosjekt i tidligfasen: valg av konsept*. Fagbokforlaget, Bergen, 2. utgave.

Schwaber, K. (1997). Scrum development process. I *Business Object Design and Implementation: Workshop Proceedings*, sider 117–134, Texas. Springer.

Schwaber, K. og Sutherland, J. (2011). The scrum guide. *Scrum Alliance*, sider 1–16.

Solms, F. (2012). What is software architecture? I *Proceedings of the south african institute for computer scientists and information technologists conference*, sider 363–373, New York. Association for Computing Machinery.

Stake, R. E. (1995). *The art of case study research.* Sage, California.

Sundnes, S. L. (2002). Fou-og innovasjonsstatistikk 2001. *U-notat nr. 4*, sider 1–110.

Thomas, J. R., Martin, P., Etnier, J., og Silverman, S. J. (2022). *Research methods in physical activity.* Human kinetics, Champaign, 8. utgave.

Tjora, A. H. (2012). *Kvalitative forskningsmetoder i praksis.* Gyldendal akademisk, Oslo, 2. utgave.

Waterman, M., Noble, J., og Allan, G. (2015). How much up-front? a grounded theory of agile architecture. I *IEEE/ACM 37th IEEE International Conference on Software Engineering*, sider 347–357, Wellington. IEEE.

Wingo, R. S. og Tanik, M. M. (2015). Using an agile software development methodology for a complex problem domain. I *SoutheastCon 2015*, sider 1–8, Florida. IEEE.

Wirth, N. (2008). A brief history of software engineering. *IEEE Annals of the History of Computing*, 30(3):32–39.

Wohlmuth, M. (2020). State of software development. *Coding Sans*, sider 1–53.

Zurcher, F. og Randell, B. (1968). Iterative multi-level modelling: A methodology for computer system design. I *Proceedings IFIP Congress*, sider 867–871, New York. Citeseer.

Vedlegg

I Informasjonsbrev

Vedlegget på neste side viser informasjonsbrevet som ble sendt ut til alle informantene til forskningsprosjektet som tilhører denne masteroppgaven. Informasjonsbrevet inneholder informasjon om formålet for forskningsprosjektet/intervjuene, redegjørelse av informantenes rettigheter, samt informasjon om den praktiske gjennomføringen av intervjuene.

NTNU
Fakultet for økonomi
Institutt for industriell økonomi og teknologiledelse

07.12.22

Forespørsel om deltakelse i forskningsprosjekt: «ERFARINGER OM METODER OG METODIKKER INNEN PROGRAMVAREUTVIKLING I PRAKSIS»

Formål

I forbindelse med min masteroppgave på Institutt for industriell økonomi og teknologiledelse på Norges teknisk-naturvitenskapelige universitet gjennomføres det nå et forskningsprosjekt hvor jeg ønsker å kartlegge erfaringer og bruk av forskjellige metoder og metodikker for gjennomføring og styring av systemutviklingsprosjekter (AGILE / Smidige metoder). Videre er jeg også veldig interessert i å finne eventuelle kriterier for valg av metodikk som måtte finnes i forskjellige organisasjoner. For å gjennomføre denne studien har jeg valgt en kvalitativ tilnærming med semikonstruerte intervjuer/samtaler for å innhente data som grunnlag til denne masteroppgaven.

Målsetningen med samtalen er å lytte til deres erfaringer innen prosjektgjennomføring i denne sammenheng, samt styrker og svakheter med valg som du eller din organisasjon/gruppe har foretatt i både ferdigstilte og/eller pågående prosjekter som du deltar i eller har kjennskap til. Jeg er også spesielt interessert i om typen systemutviklingsprosjekt eller andre egenskaper ved prosjektet kan påvirke dette valget.

Informasjon om samtalen/intervjuet

- Samtalen gjennomføres fortrinnsvis digitalt ved bruk av Microsoft Teams eller liknende.
- Jeg ønsker å foreta opptak av samtalen for å sikre at relevant informasjon ikke går tapt. Opptaket slettes så fort samtalen er transkribert/nedskrevet. Informasjon og eventuelle personopplysninger fra intervjuet anonymiseres og vil ikke bli nevnt i transkriberingen.
- Om ønskelig vil du få tilgang til transkriberingen i etterkant av samtalen slik at du kan påpeke endringer på denne - hvis du mener at noe er feilaktig eller bør endres.
- Det er helt frivillig å delta og du kan trekke deg når som helst uten å oppgi noen grunn.

Behandling av personopplysninger

Forskningen utføres i tråd med forskningsetiske retningslinjer. Det stilles strenge krav til behandling av personopplysninger og forskningsprosjektet er meldt til Norsk senter for forskningsdata (NSD), som har gjort en vurdering av hvordan data blir samlet inn og videre om

Postadresse
7491 Trondheim
Norge

Org.nr. 974 767 880
E-post:
info@adm.ntnu.no
http://www.ntnu.no/adm/info

Besøksadresse
Hovedbygningen
Høgskoleringen 1
Gløshaugen

Telefon
+ 47 73 59 55 40
Telefaks
+ 47 73 59 54 37

hvordan opplysninger vil bli behandlet. Behandlingen av eventuelle personopplysninger vil skje på bakgrunn av den enkeltes samtykke og avsluttes så raskt som mulig - senest i sluttfasen av forskningsprosjektet som er i utgangen av august 2023. En eventuell klage for brudd på behandling av personopplysninger i dette forskningsprosjektet kan rettes til Datatilsynet.

Andre opplysninger

Har du spørsmål til studien eller samtalen, eller ønsker å benytte deg av dine rettigheter, ta kontakt med:

- NTNU – enten via meg: Magne Tøndel: magne@toendel.com
  eller veileder: Ola Edvin Vie: ola.edvin.vie@ntnu.no
- Vårt personvernombud: Thomas Helgesen: thomas.helgesen@ntnu.no
- NSD – Norsk senter for forskningsdata, på epost (personverntjenester@nsd.no) eller telefon: 55 58 21 17.

Format på samtalen/intervjuet

Samtalen er konstruert som et semistrukturert intervju – hvor jeg vil innlede samtalen med noen spørsmål og vil kunne komme med oppfølgingsspørsmål underveis. Varigheten på samtalen vil være omtrent 45 minutter, og hovedtemaene i intervjuet vil være:

- Generelt om utviklingsmetodikker
- Ulike typer av systemutviklingsprosjekter

Med vennlig hilsen

Magne Tøndel

Magne Tøndel

Masterstudent – NTNU

Utviklingssjef – Norconsult Informasjonssystemer AS

II Intervjuguide

Vedlegget på neste side viser informasjonsguiden som ble brukt i de ordinære intervjuene i forskningsprosjektet som tilhører denne masteroppgaven. Intervjuguiden viser inndelingen av hovedtema og hvilke spørsmål/oppfølgingsspørsmål som var planlagt for de semistrukturerte intervjuene som ble gjennomført.

NTNU
Fakultet for økonomi
Institutt for industriell økonomi og teknologiledelse

22.11.22

Intervjuguide tilhørende forskningsprosjekt:
«ERFARINGER OM METODER OG METODIKKER INNEN PROGRAMVAREUTVIKLING I PRAKSIS»

Introduksjon til intervjuet

Formålet med dette studien og dette intervjuet er å kartlegge kjennskap, erfaringer og bruk av forskjellige metoder og metodikker for gjennomføring og styring av ulike programvareutviklingsprosjekter – samt eventuelle kriterier for valg av disse.

I hovedsak så består intervjuet av to hovedtema:

- Generelt om utviklingsmetoder
- Ulike typer av systemutviklingsprosjekter

Bakgrunn:

Alder:

Utdanning:

Stillingstittel:

Tidligere stillingstitler:

Ansiennitet:

Postadresse
7491 Trondheim
Norge

Org.nr. 974 767 880
E-post:
info@adm.ntnu.no
http://www.ntnu.no/adm/info

Besøksadresse
Hovedbygningen
Høgskoleringen 1
Gløshaugen

Telefon
+ 47 73 59 55 40
Telefaks
+ 47 73 59 54 37

NTNU
Fakultet for økonomi
Institutt for industriell økonomi og teknologiledelse

22.11.22

Tema 1: Generelt om utviklingsmetoder

"Scrum" er en relativt kjent og utbredt utviklingsmetodikk innen systemutvikling.

1. Hva er din kjennskap til "Scrum"?

2. Har du kjennskap til noen andre utviklingsmetodikker?
    a. Hvordan har du fått denne kjennskapen? - prosjektgjennomføring, studier, kursing eller via andre læringsarenaer?
    b. Har du selv gjennomført noen kurs eller sertifiseringer innen bruk av utviklingsmetodikker?
    c. Har du noen gang tenkt at du skulle visst mer om de forskjellige utviklingsmetodikkene?

3. Hvilke utviklingsmetodikker har vært brukt i prosjekter som du selv har deltatt i, eller som du har kjennskap til i egen organisasjon?
    a. Har du gjort deg noen tanker om hvorfor disse utviklingsmetodene har blitt valgt?
    b. Har du noen eksempler på prosjekter hvor valget av utviklingsmetodikk har føltes riktig og fungert godt?
    c. Har du noen eksempler på prosjekter hvor valget av utviklingsmetodikk har føltes feil og fungert dårlig?

4. Hvilke kriterier legger du eller din organisasjon til grunn for valget av utviklingsmetodikk for et nytt utviklingsprosjekt?
    a. Tror du at trender eller valg i bransjen påvirker deg og din organisasjons valg av utviklingsmetodikk for et nytt prosjekt?
    b. I hvilken grad tror du at tidligere gjennomførte prosjekter påvirker valget av utviklingsmetodikk for et nytt prosjekt?

5. Hvilke styrker (eventuelt svakheter) vil du trekke frem som de viktigste i de utviklingsmetodikkene som du har kjennskap til?
    a. Hvordan håndteres avvik fra planlagt arbeid i deres organisasjon?
    b. Hvordan fungerer fordelingen av forskjellige roller i deres organisasjon?
    c. Hvordan fungerer planleggingen av arbeidet?

Postadresse
7491 Trondheim
Norge

Org.nr. 974 767 880
E-post:
info@adm.ntnu.no
http://www.ntnu.no/adm/info

Besøksadresse
Hovedbygningen
Høgskoleringen 1
Gløshaugen

Telefon
+ 47 73 59 55 40
Telefaks
+ 47 73 59 54 37

NTNU
Fakultet for økonomi
Institutt for industriell økonomi og teknologiledelse

22.11.22

Tema 2: Ulike typer av systemutviklingsprosjekter

Alle prosjekter er unike både i omfang, antall prosjektdeltakere og ambisjonsnivå. Innen systemutvikling er også prosjektene forskjellige - og et forskningsprosjekt eller nyutviklingsprosjekt kan være relativt ulikt et vedlikeholdsprosjekt over flere år.

1. Har dere en fast utviklingsmetodikk som dere bruker for alle utviklingsprosjekter?

2. Opplever du at din organisasjon vektlegger typen prosjekt eller andre egenskaper ved prosjektet før valg av utviklingsmetodikk gjøres?
    a. Hvem er det som tar dette valget? – ledelsen, sterke personligheter eller andre nøkkelpersoner?
    b. Er det enkeltpersoner som tar dette valget?

3. Har du noen eksempler på prosjekter hvor typen prosjekt eller egenskaper ved prosjektet har påvirket arbeidsmetodikken – og hvordan synes du det fungerte?
    a. Hva tror du var årsaken til at dette fungerte så godt/mindre godt i disse prosjektene?

4. Uavhengig av prosjekter og typer prosjekter – hvilken utviklingsmetodikk tror du at du ville valgt for det neste systemutviklingsprosjektet, og hvorfor?

Postadresse
7491 Trondheim
Norge

Org.nr. 974 767 880
E-post:
info@adm.ntnu.no
http://www.ntnu.no/adm/info

Besøksadresse
Hovedbygningen
Høgskoleringen 1
Gløshaugen

Telefon
+ 47 73 59 55 40
Telefaks
+ 47 73 59 54 37

III Samtaleguide

Vedlegget på neste side viser samtaleguiden som ble brukt til den oppsummerende samtalen jeg hadde med utviklingslederen i case-bedriften. Samtalen omhandlet de samme hovedtema som de ordinære intervjuene – men ble gjennomført noe mindre formelt. Denne avsluttende samtalen ble gjennomført dels for å bekrefte eller avkrefte funnene som jeg hadde funnet på dette tidspunktet og for å få svar på de spørsmålene som jeg satt med etter de ordinære intervjuene.

NTNU
Fakultet for økonomi
Institutt for industriell økonomi og teknologiledelse

13.02.23

Samtaleguide tilhørende forskningsprosjekt:
«ERFARINGER OM METODER OG METODIKKER INNEN PROGRAMVAREUTVIKLING I PRAKSIS»

Introduksjon til samtalen

Formålet med dette studien og dette intervjuet er å kartlegge kjennskap, erfaringer og bruk av forskjellige metoder og metodikker for gjennomføring og styring av ulike programvareutviklingsprosjekter – samt eventuelle kriterier for valg av disse.

I hovedsak så består samtalen av to hovedtema:

- Generelt om utviklingsmetoder
- Ulike typer av systemutviklingsprosjekter

Tema 1: Generelt om utviklingsmetoder

Når jeg kom i kontakt med dere så ble det i starten sagt at *"Valg av utviklingsmetodikk for systemutvikling er som religion hos mange systemutviklere"*. Jeg vil bare si at jeg skjønner veldig godt hva du mener - og kanskje enda bedre etter at jeg har snakket med noen av deres systemutviklere.

1. Hvordan har innføringen av "Scrum" gått?

2. Dere er nå godt i gang med å ta i bruk "Scrum" som utviklingsmetodikk. Kan du si noe om kriterieriene for at dere valgte å gå for "Scrum"?
    a. Hvilke fordeler ønsker dere å oppnå?

3. I forhold til slik dere jobbet før – kan du si noe om hva den/de største forskjellene?

4. Har det dukket opp noen overraskelser så langt?
    a. Eventuelt hva tror du blir den største ulempen/utfordringen?

Postadresse
7491 Trondheim
Norge

Org.nr. 974 767 880
E-post:
info@adm.ntnu.no
http://www.ntnu.no/adm/info

Besøksadresse
Hovedbygningen
Høgskoleringen 1
Gløshaugen

Telefon
+ 47 73 59 55 40
Telefaks
+ 47 73 59 54 37

NTNU
Fakultet for økonomi
Institutt for industriell økonomi og teknologiledelse

13.02.23

Tema 2: Ulike typer av systemutviklingsprosjekter

Hverdagen til en systemutvikler kan være relativt forskjellig. Når man skal utvikle noe nytt så skriver man gjerne mye kode i en periode og leverer sjelden – mens hvis man forvalter eksisterende kode som allerede er i produksjon så koder man gjerne mindre og leverer oftere.

1. Har dere noen tanker om hvordan eksisterende kunder også skal betjenes i lys av "Scrum" -metodikken? – de krever gjerne mindre leveranser oftere og raskere.

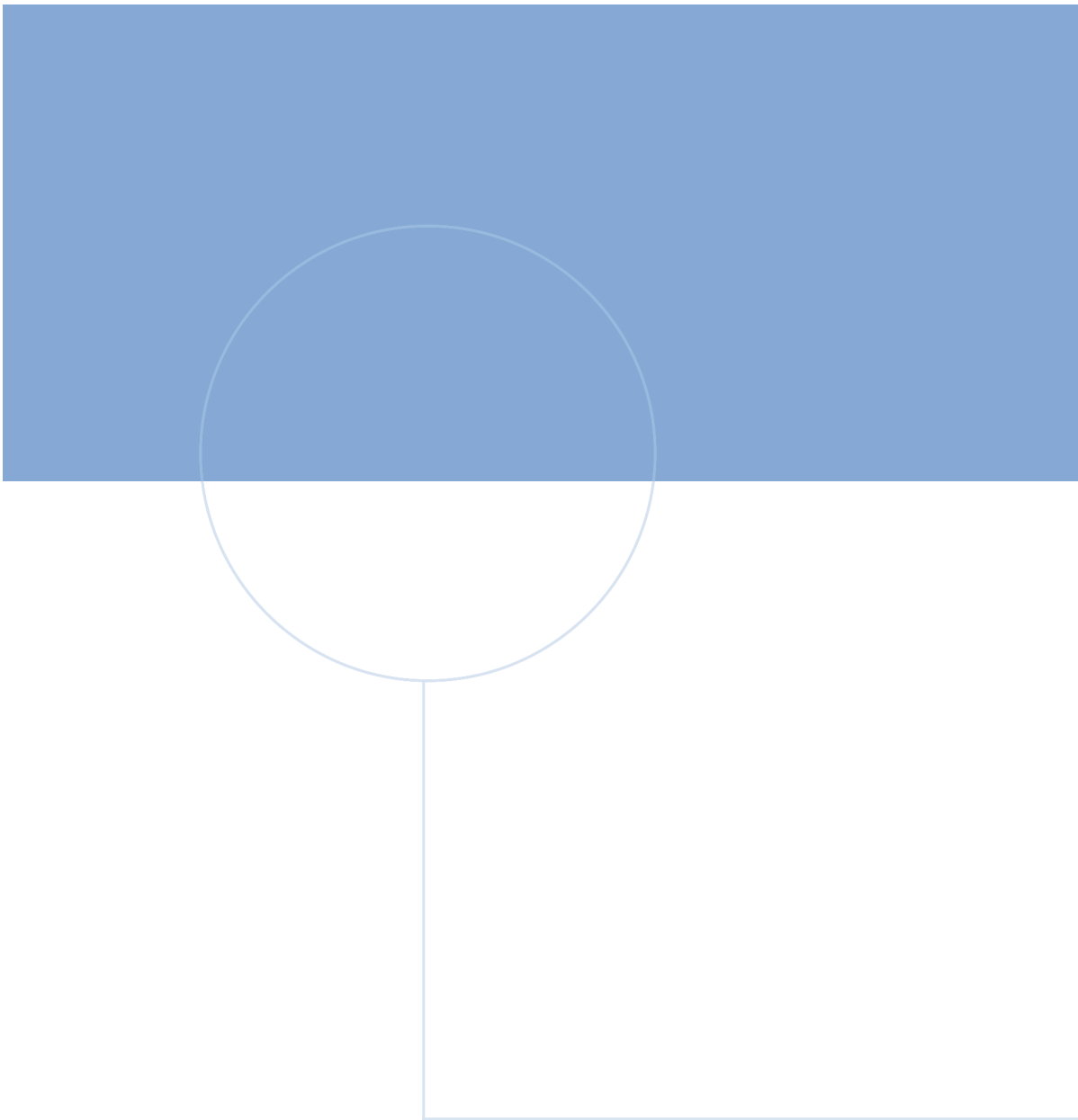
2. Tror du at dere kommer til å tilpasse "Scrum" -metodikken i lys av deres hverdag og eventuelt hvilke tilpasninger tror du kommer til å bli aktuelt?

Postadresse
7491 Trondheim
Norge

Org.nr. 974 767 880
E-post:
info@adm.ntnu.no
http://www.ntnu.no/adm/info

Besøksadresse
Hovedbygningen
Høgskoleringen 1
Gløshaugen

Telefon
+ 47 73 59 55 40
Telefaks
+ 47 73 59 54 37



NTNU

Kunnskap for en bedre verden