Torgeir Dingsøyr, Finn Olav Bjørnson og Tor Sporsem

Organisering av digitaliseringsprosjekter

Concept arbeidsrapport 2021-1

Forord

Denne rapporten er skrevet for Concept-programmet og er basert på ekstra datainnsamling og analyse i etterkant av kompetanseprosjektet Agile 2.0, som hadde organisering og bruk av smidige utviklingsmetoder i store digitaliseringsprosjekt som et av hovedtemaene. Vi er veldig takknemlige for muligheten til å videreføre arbeidet med finansiering fra Concept. Vi gil også takke partnerne i Agile 2.0 som var DNV GL, Equinor, Kantega, Kongsberg Defence & Aerospace, Sopra Steria og Sticos samt forskningspartner NTNU. Prosjektet var ledet av SINTEF og støttet av Norges Forskningsråd (prosjekt 236759).

I arbeidet med rapporten ønsker vi å takke referansegruppen fra Concept som har bidratt med innspill både gjennom møter og i kommentarer på rapportutkast. Medlemmer av referansegruppen har vært Bjørn Andersen (NTNU), Sverre Berg (Direktoratet for økonomistyring), Therese Bjerke (Arbeids- og sosialdepartementet), Elisabeth Krogh (Atkins), Hans Gøran Nilsson (Digdir), Leif Erik Jordan (Forsvarsmateriell) og Jan Alexander Langlo (NTNU). Takk til Morten Welde og Gro Holst Volden fra Concept for mulighet til å skrive rapporten og også for kommentarer på utkast. Det har også vært verdifullt å jobbe i parallell med prosjektet «Hvordan lykkes med digitalisering? En undersøkelse av nyttestyring i IT-prosjekter i offentlig sektor» (Concept prosjekt 1266).

Videre ønsker vi å takke Kjetil Røe fra SopraSteria som var kontaktperson i prosjektet Agile 2.0 og som i likhet med Knut Rolland (Universitetet i Oslo) har kommentert på et tidligere utkast. Videre vil vi takke informanter fra Accenture, Sopra Steria og NAV som deltok på workshop i oktober 2020. Takk til stipendiat Kathrine Vestues fra NTNU for diskusjoner og materiale fra siste case, og takk til alle som har blitt intervjuet, observert eller på annen måte bidratt med datainnsamling på tre case.

Trondheim, mars 2021

Torgeir Dingsøyr, Finn Olav Bjørnson og Tor Sporsem

Innholdsfortegnelse

Sammendrag

Stadig flere prosjekter inkluderer grader av digitalisering, spesielt prosjekter med innovasjon i produkter, tjenester eller arbeidsmetoder. Slike prosjekter har hatt utfordringer med å levere på nytte, teknisk produktkvalitet, kostnadskontroll, tidskontroll eller effektivitet i prosjektarbeid. Et forbedringstiltak som er foreslått i en tidligere rapport er mer hensiktsmessig organisering av prosjekter.

Denne rapporten beskriver erfaringer og forskningsfunn om organisering av store digitaliseringsprosjekt, med funn fra tre vellykkede store prosjekt i norsk offentlig sektor. Vi beskriver endring i prosjektorganisering, fra det vi beskriver som en kombinasjon av råd fra prosjektledelse med råd fra programvareutvikling i «førstegenerasjons metoder for storskala smidig utvikling», til mer tilpassede måter å organisere digital produktutvikling i «andregenerasjon metoder for storskala smidig utvikling».

Vi beskriver organisering gjennom praksiser, roller og artefakter brukt i prosjektene. Videre beskriver vi erfaringer og forskningsfunn på spesielle utfordringer i store digitaliseringsprosjekt hvor mange team jobber i parallell med oppgaver som å definere behov og krav, utvikle og teste produkt. Spesielt ser vi på håndtering av kundebehov, hvordan en sikrer gode tekniske løsninger og hvordan arbeid koordineres og kunnskap deles på tvers av team.

Denne rapporten gir et innblikk i intern organisering på et felt hvor de finnes mange foreslåtte beste praksiser, men få studier internasjonalt på hva som faktisk gjøres i praksis. Rike beskrivelser av organisering viser erfaringsbaserte praksiser som kan være nyttige for kommende digitaliseringsprosjekt.

1 Innledning

Stadig flere prosjekter inneholder elementer av digitalisering, spesielt prosjekter med fokus på innovasjon i produkter, tjenester eller arbeidsmetoder. Organisering av prosjekter for digitalisering kan dermed være interessant for en rekke prosjekter selv om hovedmålet ikke er å utvikle en digital løsning. I det følgende bruker vi Finansdepartementets definisjon på digitaliseringsprosjekter, *«et utviklingsprosjekt eller endringsprosjekt hvor IKT utgjør en sentral del, og som endrer arbeidsprosesser, organisering, regelverk eller teknologi».*[1] Denne rapporten peker på at slike prosjekter er spesielle på grunn av raske endringer i teknologi, store muligheter for innovasjon og at digitalisering har preg av kontinuerlig tjenesteutvikling. I tråd med rapporten vil vi i det følgende bruke begrepet prosjekt som digitaliseringsprosjekt som kan være programmer eller enkeltprosjekt, men også arbeid med digitale tjenester og tiltak som er en del av helhetlig virksomhetsutvikling.

Digitaliseringsdirektoratets prosjektveiviser[2] beskriver tre hovedløp i digitaliseringsprosjekter: organisasjonsutvikling, programvareutvikling og IT-anskaffelser. I denne rapporten presenterer vi modeller for gjennomføringsfasen[3] for store digitaliseringsprosjekt. Vi fokuserer på prosjekter hvor programvareutvikling er den sentrale komponenten, og som er prosjekter i betydningen at det ikke er faste team som driver kontinuerlig tjenesteutvikling, men en større midlertidig prosjektorganisasjon. Slike gjennomføringsmodeller kan imidlertid også være relevant for andre typer kunnskapsarbeid hvor det er viktig å tilrettelegge for læring underveis i prosjektgjennomføring.

På 90-tallet skrev forskere innen programvareutvikling om en krise, på grunn av at mange prosjekter ikke leverte på tid, kost eller kvalitet. En oppsummerende rapport om suksess og fiasko i offentlige IKT-prosjekter (Jørgensen, 2015) fant at kostnadene ved feilslåtte IKT-prosjekter er vanskelig å beregne, «men er helt klart i milliardklassen». En undersøkelse på IKT-prosjekter i Norge fant at 50% av prosjektene hadde lav suksess på minst en av suksessfaktorene levert nytte, teknisk produktkvalitet, kostnadskontroll i prosjektet, tidskontroll i prosjektet eller effektivitet i prosjektarbeidet. Et av forbedringstiltakene som diskuteres i rapporten er «uhensiktsmessig organisasjon av prosjektet». Prosjekter med innebygd endringsevne med hyppige leveranser og prioritert «kø» av funksjonalitet lykkes oftere med å levere bra nytte enn tradisjonelle prosjekter med fastpriskontrakter. Denne måten å organisere prosjekter på kalles i programvarebransjen for «smidige metoder».

---

[1] Finansdepartementet: Digitaliseringsprosjekter i Statens prosjektmodell. Veileder, 31. januar 2020.

[2] Digitaliseringsdirektoratet: Prosjektveiviseren.
https://www.prosjektveiviseren.no/prosjekttyper/digitaliseringsprosjekter

[3] https://www.regjeringen.no/no/tema/okonomi-og-budsjett/statlig-okonomistyring/ekstern-kvalitetssikring2/id2523818/

I denne rapporten stiller vi spørsmålet: *Hvordan organisere store digitaliseringsprosjekter?* Vi fokuserer på utviklingsmetode for store digitaliseringsprosjekter, hvor «smidige metoder» har ført til store endringer i hvordan prosjekter gjennomføres i praksis. Gjennom rike beskrivelser av tre offentlige prosjekt, kombinert med forskningsfunn fra feltet søker vi å formidle innsikt i to hovedtyper for organisering av store digitaliseringsprosjekt, det vi vil kalle første- og andre-generasjons metoder for storskala smidig utvikling. Førstegenerasjon metoder kombinerte råd fra prosjektledelse med smidige metoder mens andregenerasjon er metoder med mer tilpassede råd for store digitaliseringsprosjekt. De offentlige prosjektene er valgt fordi de er prosjekter som har levert på tid, kost og nytte, og hvor de i stor grad har tatt i bruk praksiser fra smidige metoder.

Hvorfor fokus på rike beskrivelser av case? Det er få prosjekter som er så store, slik at det er vanskelig å gjennomføre for eksempel spørreundersøkelser om store digitaliseringsprosjekt. Det er også et felt hvor det er raske endringer i arbeidspraksis. Slike prosjekt egner seg da godt for kvalitative casestudier. Internasjonalt er det imidlertid få slike studier, trolig fordi det ofte er vanskelig for forskere å få tilgang til relevante case. Rike beskrivelser egner seg godt for å forstå kontekst for praksiser, bruk av roller og artefakter i prosjektgjennomføring. Vi håper de rike beskrivelsene kan være til inspirasjon for hvordan kommende digitaliseringsprosjekt i offentlig sektor kan håndtere vanlige utfordringer. I neste seksjon vil vi definere hva vi mener med store prosjekter, og også spisse problemstillingen i fire viktige områder på organisering og konsekvenser av organisering.

Mye av referansene til tidligere publiserte studier er hentet fra et spesialnummer i tidsskriftet *Project Management Journal* om «agile approaches» (Niederman *m.fl.*, 2018), samt magasinet *IEEE Software´*s spesialnummer om «large-scale agile» (Dingsøyr *m.fl.*, 2019b). Fokuset på første- og andre-generasjonsmetoder viser forskjeller i bruk av prosjekt- og delprosjekt, faser, hvilke roller som ligger på teamnivå og i prosjektorganisasjonen, samt grad av autonomi for team og tilrettelegging for kontinuerlige leveranser. Enkelte organisasjoner velger *bimodal* utvikling ved at noen team eller prosjekt jobber smidig mens andre jobber etter «tradisjonelle» metoder eller «fossefallsmetoden». Vårt siste case hadde en *bimodal* fase, men vi har valgt å presentere andre-generasjonsmetode i siste fase hvor de da gikk bort fra bimodal utvikling og jobbet smidig i alle team.

I det videre beskriver vi først bakgrunn på digitaliseringsprosjekter fra feltet programvareutvikling, fra metoder for små prosjekter i team til store prosjekter. Videre redegjør vi for forskningsmetode for deretter å beskrive funn fra to case med førstegenerasjon og ett case med andregenerasjons metoder. I beskrivelsen av case trekker vi også inn relevant faglitteratur. Til slutt konkluderer vi med funn som vil være relevante i planlegging av nye digitaliseringsprosjekter.

Programvareutvikling er et fagfelt med mye stammespråk som kan gjøre stoffet i denne rapporten vanskelig tilgjengelig. Vi har forsøkt å forklare de fleste begrepene med norske ord, og også gitt eksempler. For lesere som finner ukjente ord og utrykk fra området «smidige metoder» men som har bakgrunn i prosjektledelse anbefaler vi først å lese Project Management Institute´s *Agile Practice Guide* (2017).

Som bakgrunn presenterer vi først tidlige tanker om organisering av programvareutvikling, overgangen til smidige metoder på teamnivå og deretter det vi kaller for første- og andregenerasjonsmetoder for store digitaliseringsprosjekter. Til slutt skisserer vi noen sentrale utfordringer i slike prosjekter, som vi videre vil belyse med forskningsfunn og erfaringer fra rike casestudier.

1.1 Programvareutvikling: Fra fossefall til smidige metoder

En NATO-konferanse i 1968 ble starten på fagfeltet programvareutvikling, på engelsk *software engineering*. Den rådende oppfatningen var lenge at programvare bør utvikles etter ingeniørmessige prinsipper. Som når nye bygg prosjekteres, bør IT-løsninger planlegges nøye på forhånd. Gjennom detaljerte kravspesifikasjoner ble løsningen beskrevet, deretter ble viktige avgjørelser tatt om hvordan løsningen skulle konstrueres, før selve løsningen ble utviklet. Brukere ble involvert i starten for å spesifisere hva som skulle lages, og mot slutten for å teste produktet. Royce (1970) beskrev utviklingsmetoden som en «fossefallsmodell», som illustrert i Figur 1.1. Faser ble ofte forsøkt avsluttet før prosjektet tok for seg neste fase. Forbedringsarbeid fokuserte på detaljerte beskrivelser av arbeidsprosesser og arbeidsflyt og standardisering av arbeidsprosesser.

Figur 1.1: Fossefallsmodell for programvareutvikling beskrevet i SINTEF og R-direktoratets «Håndbok for systemutvikling» (Bræk m.fl., 1985).

En utfordring var at kravene til IT-systemer ofte endret seg underveis. Videre er teknologien som brukes er under konstant videreutvikling. Ofte kjente ikke brukerne igjen kravene de hadde formulert da de så den ferdige løsningen. Uforutsette endringer kan skape usikkerhet, dødtid og demotivasjon (Finne, 2019). En mye sitert studie av oppfatningen av risiko blant prosjektledere på slutten av 90-tallet (Keil *m.fl.*, 1998) viste at de fleste risikofaktorene var koblet til kunden i programvareutvikling som: Forankring hos toppledelse, forankring hos brukere, misforståelse av behov og krav, manglende evne til å svare på forventninger hos sluttbrukere, og manglende involvering av brukere.

Siden tidlig på 2000-tallet har prosjektgjennomføringsmodeller for digitale prosjekter vært preget av smidige metoder. Et «manifest» for «smidig» programvareutvikling[4] la vekt på: Individer og interaksjon i utviklingsteamene framfor prosesser og verktøy, fungerende systemer framfor omfattende dokumentasjon, kundesamarbeid framfor kontraktsforhandlinger, og det å kunne respondere på uforutsette hendelser framfor å følge en fastspikret plan. Disse prinsippene legger vekt på programvareutvikling som et håndverk som skiller seg fra andre ingeniørdisipliner. Utvikling av programvareprodukt er en problemløsningsprosess hvor innsatsen ligger i utvikling,

[4] https://agilemanifesto.org/iso/no/manifesto.html

ikke i produksjon. Kostnadene er primært arbeidstimer for prosjektmedarbeidere. Tabell 1.1 oppsummerer forskjeller på tradisjonell utvikling og smidig utvikling.

Den mest populære smidige metoden er i dag Scrum.[5] Scrum (Schwaber & Beedle, 2001; Schwaber & Sutherland, 2020) er et rammeverk for programvareutvikling hvor et team gradvis utvikler et produkt som «potensielt kan leveres» etter hver nye iterasjon. En produkteier er ansvarlig for å definere og prioritere krav i en «produktkø», og et utviklingsteam velger i samråd med produkteier hva som skal utvikles i neste iterasjon. Teamet lager en detaljert plan for denne iterasjonen i et møte som er fasilitert av en Scrumleder. Produktet demonstreres på slutten av hver iterasjon, og ellers fasiliterer Scrumlederen daglige møter for koordinering og kunnskapsdeling i teamet. En vanlig utfordring i programvareutvikling har vært at en oppdager feil når komponenter integreres. Tilnærmingen i smidige metoder er å gjøre dette ofte, og å automatisere testarbeid.

Et team består typisk av 5-9 medlemmer, og kunnskapsoverføring i teamet foregår primært muntlig. Produktet defineres som regel i produktkøen i form av «epos» som dekomponeres i «brukerhistorier» (Patton & Economy, 2014). Epos er overordnede beskrivelser av funksjonalitet, som «brukere trenger et system for å reservere hotellrom», som da kan dekomponeres i brukerhistorier på formen «som ansatt i reisebyrå vil jeg kunne se bilder av hotellrom som en del av vurderingen når jeg skal bestille rom for en kunde». Epos er gjerne så overordnede at de kan fungere til grov prioritering hos en produkteier men for lite spesifikke til at det gir mening å starte utvikling. Hver brukerhistorie dekomponeres i oppgaver for teamet under hver økt for iterasjonsplanlegging. Produkteier og/eller teamet har jevnlige møter for å raffinere produktkøen. Da Scrum ble introdusert var vanlig iterasjonslengde fire uker, mens det i dag er vanlig med to ukers iterasjoner.

Mange miljø kombinerer utvikling, vedlikehold og drift («DevOps») og trenden har vært mot hyppigere integrasjon av alle komponenter, «kontinuerlig integrasjon» og også mot å kunne lansere ny funksjonalitet ofte, «kontinuerlige leveranser» (Dingsøyr & Lassenius, 2016; Fitzgerald & Stol, 2017). I stedet for iterasjoner velger mange å gjøre en og en oppgave helt ferdig, såkalt «flytbasert utvikling» - uten felles planlegging og demonstrasjon som i Scrum. Et studie av forskjellen på demonstrasjoner av produkt til interessenter og tilbakemeldinger ved faktiske leveranser viser at det ofte er vanskelig for brukere å se konsekvensene av ny funksjonalitet før løsningen faktisk tas i bruk (Schmitz *m.fl.*, 2019). Kontinuerlige leveranser gir da bedre muligheter for utviklingsteamet til å få tilbakemeldinger mens de jobber med en brukerhistorie i stedet for å måtte vente kanskje i flere uker på en demonstrasjon eller i flere måneder på en at en leveranse er ferdig.

[5] Verktøyleverandøren VersionOne publiserer årlig en «State of Agile»-undersøkelse, som for 2020 rapporterte at 58% av utviklingsteam bruker Scrum mens 10% bruker en kombinasjon av Scrum og Kanban, 9% «andre metoder» og 8% en kombinasjon av Scrum og Ekstremprogrammering («XP»).

Tabell 1.1: Forskjeller på tradisjonelle og smidige utviklingsmetoder, utdrag fra Nerur m.fl. (2005).

	Tradisjonell	**Smidig**
Fundamental antakelse	Programvaresystemer kan spesifiseres og bygges gjennom detaljert og omfattende planlegging.	Programvaresystemer kan lages av små team med prinsipper om kontinuerlig forbedring av design og testes ved hyppige tilbakemeldinger.
Kontroll	Prosessorientert	Menneskeorientert
Kunnskapsdeling	Eksplisitt	Taus
Roller	Individuelle – mot spesialisering	Autonome team – mot generalister
Kommunikasjon	Formell	Uformell
Kunderolle	Viktig	Kritisk
Utviklingsmodell	Fossefall	Evolusjonær

Kritikken av smidige metoder har blant annet dreid seg om overdrevne påstander om økt effektivitet, at nytten av enkelte praksiser som å programmere i par («parprogrammering» fra metoden Ekstremprogrammering) også er overdrevet, og at metodene legger for lite innsats i arbeid med å definere behov og å planlegge overordnet design (Meyer, 2018). Noen har også hevdet at man med fossefallsmetoder «vet hva man får» ved å lage planer for arbeidet. Svaret fra smidig-miljøet har vært at iboende usikkerhet rundt teknologi og krav gjør at mange antakelser i planarbeidet ikke vil gjelde, videre at stort fokus på planer og andre artefakter rundt selve utviklingsjobben stjeler tid fra det faktiske arbeidet med å lage produktet. Det er heller ikke slik at en ved smidige metoder ikke skal fokusere på å lage planer, men det lages grove planer på lengre sikt og detaljerte planer på kort sikt. Smidige metoder beskriver få arbeidspraksiser, roller og artefakter, men det er elementer som brukes i praksis. Studier på prosjektsuksess indikerer at prosjekt med smidige metoder leverer bedre på nytte enn prosjekt som bruker fossefallsmodeller (Mohagheghi & Jørgensen, 2017).

Da smidige metoder kom mente de fleste at de egnet seg til små, samlokaliserte team som laget programvare som ikke er sikkerhetskritisk. Imidlertid har bransjen som nevnt over i stor grad tatt i bruk smidige metoder, og metodene er også i bruk i distribuert og global programvareutvikling (Ågerfalk & Fitzgerald, 2006; Smite *m.fl.*, 2010), i sikkerhetskritisk programvareutvikling (Hanssen *m.fl.*, 2018) og i store prosjekter («storskala smidig» (Dingsøyr *m.fl.*, 2019b)).

I 2017 lanserte Project Management Institute og Agile Alliance «Agile practice guide» (2017) som beskriver tanker bak smidige metoder, vanlige praksiser i smidige miljø og hva en organisasjon bør fokusere på for å tilrettelegge for smidige metoder. Potensielle organisatoriske utfordringer inkluderer eksisterende organisering som kan skape avhengigheter (i organisasjoner med «siloer»), innkjøpsstrategier med fokus på pris, ledelse som prioriterer lokal optimalisering heller enn prosjektleveranser og medarbeidere med stor grad av spesialisering (Project Management Institute and Agile Alliance, 2017). Videre beskriver praksisguiden råd om å utvikle en organisasjonskultur preget av tillit, råd om innkjøp og kontrakter. Guiden nevner rammeverk for storskala utvikling og anbefaler å starte med smidige metoder i enkelt-team og ikke skalere opp før smidige praksiser fungerer godt på teamnivå. Prosjektveiviseren anbefaler å undersøke om forutsetninger for bruk av smidige metoder er til stede ved blant annet å vurdere prosjektets omgivelser og om at berørte parter har forstått smidige hovedprinsipper.

Å endre organisasjoner fra «tradisjonelle» til «smidige» verdier, prinsipper og praksiser er omtalt i litteraturen som «smidig transformasjon». Fra programvareutviklingslitteraturen finner vi blant annet beskrivelser av en slik endring hos Ericsson (Paasivaara *m.fl.*, 2018). Dette er et eget område som vi ikke vil behandle mer i detalj i denne rapporten.

1.2 Digitaliseringsprosjekter og store prosjekter

Som nevnt i innledning kjennetegnes digitale prosjekter ved raske endringer i teknologi, store muligheter for innovasjon og at digitalisering har preg av kontinuerlig tjenesteutvikling. I det følgende vil vi fokusere på store digitaliseringsprosjekter og valg av metodikk i gjennomføring. I faglitteraturen er det brukt mange definisjoner på hva store digitaliseringsprosjekter er, fra å beregne størrelsen i:

- *kostnad* – for eksempel brukes 300 millioner kroner som terskelverdi for digitaliseringsprosjekter på grunn av kompleksitet og risiko i rundskriv om statens prosjektmodell[6]
- *størrelsen på produktet* - som antall millioner linjer programkode
- *funksjonalitet som skal utvikles* - for eksempel målt i brukerhistorier
- *varighet av prosjektet* - gjerne flere år
- *antall deltakere i prosjektet* - gjerne flere hundre deltakere.

For utviklingsmetode er det imidlertid forskjellige anbefalinger etter antall utviklingsteam. Tabell 1.2 viser en oversikt over størrelser organisert etter antall utviklingsteam (Dingsøyr *m.fl.*, 2014). Hvis hvert team består av 5-9 medlemmer vil da et storskala prosjekt ha minst 10 teammedlemmer i tillegg til prosjektledelse og andre roller, mens et «veldig stor skala» har minst 50 teammedlemmer. I praksis har team ofte vært større en vanlige anbefalinger i smidige metoder, ofte 12 medlemmer i hvert team som da gir minst 120 deltakere i et «veldig stor skala»-prosjekt.

Tabell 1.2: Størrelse på digitaliseringsprosjekter etter antall utviklingsteam.

Nivå	Antall team	Beskrivelse
Liten skala	1	Koordinering av team med praksiser som daglige møter, iterasjonsplanlegging, gjennomganger («demo») og retrospektiv-møter.
Stor skala	2-9	Koordinering av team med nye praksiser.
Veldig stor skala	10+	Enda flere praksiser nødvendig for koordinering.

[6] https://www.regjeringen.no/no/tema/okonomi-og-budsjett/statlig-okonomistyring/ekstern-kvalitetssikring2/endringer-i-statens-prosjektmodell/id2632848/

En litteraturgjennomgang av prosjektstyring trekker fram flere områder hvor prosjektstyring bør gjøres annerledes i smidige prosjekter (Lappi *m.fl.*, 2018). Mål og omfang kan være under endring i løpet av prosjektperioden, fokus dreier seg fra prosjektplan som viktigste verktøy for kommunikasjon og styring til koordinerings- og kommunikasjonsarenaer som gir sanntidsinformasjon om prosjektstatus. I smidige prosjekter er status på produktet den viktigste indikatoren på fremdrift.

1.3 Førstegenerasjons metoder for storskala smidig utvikling

Førstegenerasjon metoder for storskala smidig utvikling kombinerte tradisjonelle rammeverk for prosjektledelse med smidige metoder på teamnivå. I Norge ble prosjektveiviseren som er basert på Prince2 gjerne brukt for å dele inn et prosjekt i faser og å etablere delprosjekt og prosjektstyring (se Tabell 1.3). Fra USA har vi et studie av et stort prosjekt for et cruise-selskap som leverte på krav, tid og kost til tross for at 60% av kravene ble endret underveis (Batra *m.fl.*, 2010). Project Management Institutes *body of knowledge* (Duncan, 2017) ble kombinert med den smidige metoden Scrum (Schwaber & Sutherland, 2020). Studien peker på at struktur fra prosjektledelsesrammeverk var viktig fordi prosjektet var stort, strategisk, tidskritisk og distribuert - og at kombinasjonen med smidige metoder gjorde det mulig å håndtere uforutsette forandringer og endringer i krav.

Tabell 1.3: Komponenter i førstegenerasjonsmetoder for storskala smidig utvikling.

Metode / rammeverk	Beskrivelse
Prosjektledelse	Rammeverk fra Project Management Institute eller Digitaliseringsdirektoratets prosjektveiviser7 basert på Prince2 (Bentley, 2010) med fokus på prinsipper som kontinuerlig forretningsmessig forankring, læring av erfaring, definerte roller og ansvar, styring i faser, avviksledelse, produktfokus og tilpasning til prosjektomgivelser.
Smidig metode	Rammeverk som Scrum (Schwaber & Beedle, 2001; Schwaber & Sutherland, 2020) med fokus på evolusjonær utvikling i korte «sprinter», gjennomgang av produkt etter hver sprint, detaljert planlegging for neste sprint, læring av erfaring gjennom retrospektiver og daglige møter for koordinering og kommunikasjon internt i teamet. Møte mellom team, «Scrum of Scrums» i prosjekt med flere team.

1.4 Andregenerasjons metoder for storskala smidig utvikling

De siste årene har det kommet en rekke nye gjennomføringsmodeller som erstatter prosjektledelsesrammeverk med mer spesifikke råd for digitaliseringsprosjekt. Eksempler på modeller er Disciplined Agile Delivery, Large-Scale Scrum, Scaled Agile Framework og Spotify-modellen. Spotify-modellen har inspirert mange bedrifter ved at den viser hvordan en ren digital virksomhet er organisert, med fokus på «autonome team». Organisering av team og typer av team har vært fokus, blant annet i boka *Team Topologies (Skelton & Pais, 2019).*

[7] https://www.prosjektveiviseren.no/

Tabell 1.4: Eksempler på andregenerasjons metoder for storskala smidig utvikling. Se (Dingsøyr m.fl., 2019b) for utfyllende oversikt.

Metode	Beskrivelse
Disciplined agile delivery	Metode for ett til flere team. Viktigste kjennetegn: Rullerende planlegging, prognoser, dynamisk styring av porteføljer kombinert med vanlige smidige praksiser som daglige møter og retrospektiver. Etablerer et sett med kjerneverdier som fokus på nytte, kontinuerlig evaluering av portefølje, demonstrere verdi av portefølje. Oppfordrer til samarbeid og styrking av team.
Scaled Agile Framework (SAFe)	Passer for programmer og prosjekter fra 50-150 medarbeidere til hele organisasjoner. Viktigste kjennetegn: Omfattende metode som tar inn ideer fra smidig utvikling og fra lean produksjon. Forskjellige versjoner for prosjekt, porteføljer og organisasjoner. Et sentralt konsept er «smidig leveransetog» hvor fem til 12 team jobber med produkt-inkrement i lengder på åtte til 12 uker.
Spotify-modellen	En måte å organisere produktutvikling med mange team. Viktigste kjennetegn: Organisering i team og avdelinger. Praksisfellesskap for læring og standardisering på tvers av team og avdelinger. Egne roller for teknisk arkitektur som systemeier og en sjefsarkitekt. En modell som er i stadig endring.

1.5 Utfordringer i storskala smidige prosjekter

En tidligere rapport om erfaringer med design, styring og gjennomføring av store IKT-prosjekter (Finne, 2019) beskriver flere utfordringer i store prosjekter på grunn av kompleksitet som gjør det vanskelig å lykkes med fossefallsstrategi, som økt kompleksitet, varighet på prosjekter og manglende fleksibilitet i å håndtere endringer. Erfaringer fra tidligere store prosjekter beskriver prosjektledelse som usikkerhetsstyring og hvor «programvarefabrikken» er «mer eller mindre selvgående». Oppgaver for prosjektorganisasjon er da å håndtere forholdet til eier- og brukerorganisasjon men også prosjektorganisasjonen internt. Eksperter anbefaler «godt organiserte gripeflater» mot brukerorganisasjonen, og beskriver vilje hos brukerorganisasjon til å bruke tilstrekkelig tid og ressurser som en kritisk suksessfaktor.

Det kan være utfordrende å ta i bruk en ny metode. Et studie av 13 innføringer av rammeverk som Scaled Agile Framework, Large-Scale Scrum og Spotify-modellen (Conboy & Carroll, 2019) indikerer at mange organisasjoner undervurderer hvor store endringer som kreves ved nye modeller. Videre velger firma ofte modell uten å reflektere over valg, og konsentrerer seg også ofte mer om å sikre at prosjektet følger metoden enn at prosjektet gir nytte.

Fra internasjonale studier har vi eksempler på utfordringer med smidige metoder i store prosjekt (Bick *m.fl.*, 2017; Vlietland & van Vliet, 2015), og det har siden 2013 vært arrangert en årlig workshop med fokus på å identifisere problemstillinger for relevant forskning (Bass, 2019). Aspekter som har vært belyst har inkludert:

- *Behov og kravarbeid* – hvordan sikrer man at løsninger tilfredsstiller kundebehov når mange team jobber i parallell og hvor produktet ofte skal lanseres på mange brukere?

- *Arkitekturarbeid og autonomi* – hvordan sikre gode tekniske løsninger når mange team jobber i parallell, og hvordan sikre at team ikke sinker fremdrift hos andre team?

- *Koordinering og kunnskapsdeling* – hvordan sikre tilstrekkelig koordinering og kunnskapsdeling mellom team som jobber etter smidige prinsipper?

2 Metode

Vi undersøker spørsmålet *hvordan organisere store digitaliseringsprosjekter,* som vi vil dekomponere i fire underspørsmål basert på kjente utfordringer fra forskningslitteratur:

- *Hvordan er store digitaliseringsprosjekt organisert i førstegenerasjon og andregenerasjons metoder for storskala smidig utvikling?*

- *Hvordan sikrer man at løsninger tilfredsstiller kundebehov når mange team jobber i parallell og hvor produktet ofte skal lanseres på mange brukere?*

- *Hvordan sikre gode tekniske løsninger når mange team jobber i parallell, og hvordan sikre at team ikke sinker fremdrift hos andre team?*

- *Hvordan sikre tilstrekkelig koordinering og kunnskapsdeling mellom team som jobber etter smidige prinsipper?*

Vi har valgt tre case som har lyktes godt og gjort erfaringer som vi antar er nyttige for andre statlige digitaliseringsprosjekter, og som også kan være relevant for andre typer kunnskapsarbeid. Alle prosjektene[8] er offentlige, er avsluttede, og produktet som er utviklet har gått over i forvaltning. Prosjektene har utviklet løsninger som har gitt nye arbeidsprosesser for ansatte ved høyere grad av automatisering og også gitt nye tjenester til befolkningen. Alle prosjektene har utviklet løsninger innen *en* sektor og er da ikke tverrsektorielle, selv om alle løsningene har avhengigheter til andre systemer hos andre aktører. IT-prosjekter skiller seg fra for eksempel infrastrukturprosjekter i at endringer i krav og teknologi fører til behov for vedlikehold, videreutvikling og at forventet levetid for et produkt er kortere. Prosjektene har hatt lange faser med over 100 prosjektmedarbeidere og alle har tatt i bruk smidige metoder som har blitt tilpasset underveis. Kontraktsmodell har vært PS2000-Smidig/SOL. Antall team varierer fra fire til tolv, hvor case 2 var det minste.

Vi redegjør kort for hvordan vi har gjennomført casestudiene (Runeson & Höst, 2009). Se (Dingsøyr *m.fl.*, 2018a) for mer informasjon om metode og begrensninger i case 1. For alle case har vi gjennomført intervjuer og workshops for *datainnsamling,* og har også hatt tilgang på dokumenter fra prosjektene. I intervjuer som enten har vært individuelle eller gruppe-intervjuer har vi brukt intervjuguider som i vedlegg. I workshoper har vi brukt teknikker for idemyldring for å samle informasjon om hva deltakerne mener har fungert bra, og hva de synes kunne fungert bedre. I case 1 (Perform) er det skrevet en egen magasinartikkel som oppsummerer lærdommer fra prosjektet i tillegg til forskningsartikler. For case 3 (Foreldrepengeprosjektet) er en forskningsartikkel under arbeid og det er også planlagt en magasinartikkel i samarbeid med

[8] Vi bruker «prosjekter» om alle tre selv om noen kan omtales som «program».

Tabell 2.1: Beskrivelse av case som er brukt i denne studien.

	Case	Beskrivelse	Datamateriale	Dokumentert erfaring
1	Perform-prosjektet	175 deltakere, 4 år, førstegenerasjon storskala smidig.	11 fokusgruppeintervjuer, erfaringsrapport fra case, rapport fra ekstern kvalitetssikring. Workshop med nøkkelpersoner.	(Bjørnson m.fl., 2018) (Dingsøyr m.fl., 2019a) (Dingsøyr m.fl., 2018b) (Dingsøyr m.fl., 2018a)
2	Stort offentlig prosjekt	120 deltakere, 3,5 år, førstegenerasjon storskala smidig.	20 intervjuer, 2 workshops, dokumenter fra prosjekt.	(Rolland m.fl., 2016)
3	Foreldrepengeprosjektet	200 deltakere, 3 år, endring fra førstegenerasjon til andregenerasjon storskala smidig.	3 runder med intervjuer. Totalt 35 intervju. Workshop med nøkkelpersoner.	

nøkkelpersoner fra prosjektet. Dokumenter inkluderer erfaringsrapport fra case 1 og planer for bemanning og for reorganisering av case 3.

Dataanalyse: Opptak av intervjuer og workshoper er transkribert og analysert i et verktøy for kvalitativ analyse (Nvivo og HyperResearch). Vi har «kodet» materialet for eksempel ved klassifisering av utsagn om arenaer for koordinering og på praksiser for arbeid med behov- og krav.

Vi har videre presentert funn til deltakere på intervju og workshops for å sikre at vi har tolket materialet riktig. Selv om vi har mange informanter i studiene så har vi en overvekt av utsagn fra folk som har hatt ledende roller i prosjektene. Vi har videre ikke undersøkt prosjekter som har gitt liten nytteverdi eller har hatt lav prosjektsuksess.

I det videre vil vi presentere funn fra vår studie kombinert med relevante internasjonale funn, strukturert etter førstegenerasjon i kapittel 3 (brukt i alle tre case) med hovedvekt på funn fra Perform-prosjektet, supplert med enkelte funn fra case 2 som i hovedsak var organisert på samme måte. I kapittel 4 kombinerer vi funn fra siste fase i Foreldrepengeprosjektet (case 3) med relevante internasjonale funn fra organisasjoner som Ericsson og Spotify.

3 Førstegenerasjon storskala smidig metode

Førstegenerasjon storskala smidige prosjekter kombinerer metoder fra prosjektledelse som i Prince2 eller prosjektveiviseren med smidige metoder som Scrum i konstruksjonsfasene. Et eksempel er Perform-prosjektet[9] hos Statens pensjonskasse (case 1). Dette prosjektet ble startet for å lage en ny saksbehandlingsløsning til pensjonsreformen. Gammel løsning var på en plattform som skulle fases ut og det ble besluttet å utvikle ny løsning til reformen på en ny teknisk plattform. Da prosjektet måtte starte hadde reformen ennå ikke gått gjennom i Stortinget, og på grunn av usikkerhet rundt behov og krav fikk prosjektet lov å starte med utbredt bruk av smidige metoder. Pensjonskassen hadde 380 ansatte og ga tjenester til rundt 950.000 kunder.

Perform-prosjektet varte i fire år, involverte på det meste 175 personer (ca. 100 eksterne) og brukte rundt 800 000 timer på å utvikle rundt 300 epos med ca. 2500 brukerhistorier. Prosjektet hadde en kostnad på 994 millioner NOK og en kostnadsramme på 1287 millioner NOK.[10] Eposene ble delt inn i 12 produktleveranser. Prosjektet leverte på tid og kost og er regnet som vellykket:[11] Prosjektet har «gitt store læringseffekter til hele organisasjonen, og derigjennom effektiviseringsgevinster», «Statens pensjonskasse når alle sine resultatmål for kvalitet» og «måloppnåelsen i prosjektet har vært god, og fem av prosjektets seks effektmål er oppnådd».

I det følgende beskriver vi prosjektorganisering, faser i utvikling, samt roller på teamnivå med fokus på hvordan det var mot slutten av prosjektet. Vi vil videre beskrive hvordan prosjektet organiserte behov- og kravarbeid, arbeid med arkitektur, hvordan team ble koordinert, autonomi på teamnivå og hvordan prosjektet la til rette for kunnskapsdeling over tid. Samme gjennomføringsmodell har blitt brukt i flere andre store digitaliseringsprosjekter i Norge, og vi tar også med erfaringer fra et annet prosjekt, «case 2» (Rolland *m.fl.*, 2016).

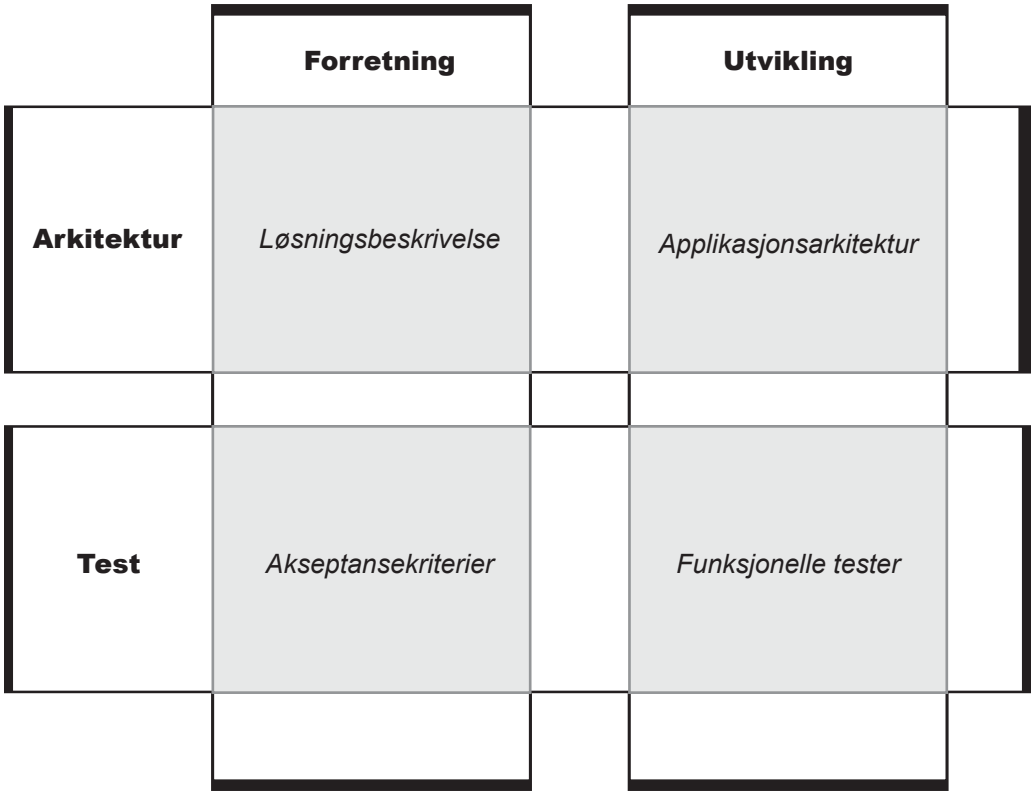
3.1 Organisering

Prosjektet ble organisert med hovedprosjekter for arkitektur, forretning, test og utvikling, samt prosjekter for kommunikasjon og innføring. Utviklingsprosjektet ble delt i tre, hvor Pensjonskassen hadde prosjektledelse på ett og eksterne leverandører hadde prosjektledelse på hvert av de andre, som i Figur 3.1.

[9] Vi bruker gjennomgående «prosjekt» i omtale av Perform selv om vi kunne beskrevet det som et «program».

[10] Tall i 2010-kroner.

[11] https://www.ntnu.no/concept/perform



Figur 3.1: **Prosjekt-***organisering med fire hovedprosjekter.*

Prosjektene hadde en prosjektleder og noen fulltidsressurser i tillegg til deltidsressurser fra utviklingsteamene med hovedansvar for:

- *Arkitektur:* Ansvarlig for å definere viktigste arkitekturbeslutninger, og bidro i å detaljere brukerhistorier i faser for behovsanalyse og løsningsbeskrivelse. Prosjektet bestod av sjefsarkitekt og tekniske arkitekter fra utviklingsteamene.

- *Forretning:* Definerte behov, beskrev løsning og prioriterte epos og brukerhistorier i produktkø. Prosjektet var bemannet med produkteier og opptil 30 ansatte fra linjeorganisasjonen. Funksjonelle og tekniske arkitekter fra utviklingsprosjektet bidro i prosjektet.

- *Test:* Ansvarlig for testprosedyrer og for å godkjenne leveranser fra utviklingsteamene. Bestod av prosjektleder og ressurser fra utviklingsteamene.

- *Utvikling:* Var i Perform inndelt i tre delprosjekter, i andre store digitaliseringsprosjekter organisert som ett prosjekt med mange team. I Perform jobbet det lenge 12 team i parallell med å utvikle brukerhistorier. Roller på team er gjengitt i Tabell 3.1. Teamene jobbet i tre ukers iterasjoner, med planlegging i start, daglige møter underveis, demonstrasjon av løsning (felles møte) og deretter retrospektiv på teamnivå før neste iterasjon.

Prosjektet leverte ny programvare 12 ganger i løpet av fireårsperioden. For hver ny leveranse ble det gjennomført følgende faser:

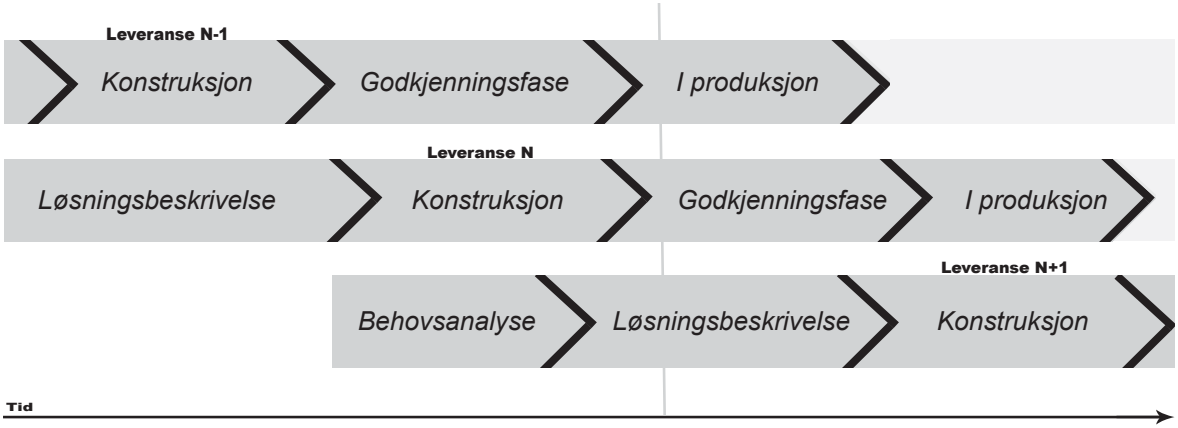
- *Behovsanalyse* – gjennomgang av hovedfunksjonalitet i leveransen, høynivå brukerhistorier (epos) ble definert i produktkø. Produkteiere fra forretningsprosjektet prioriterte behov.

- *Løsningsbeskrivelse* – høynivå brukerhistorier ble dekomponert i brukerhistorier og løsningsbeskrevet mer i detalj også med fokus på tekniske valg. Brukerhistorier ble estimert og tildelt utviklingsteam.

- *Konstruksjon* – Utvikling av brukerhistorier i iterasjoner på tre uker, fem til syv iterasjoner for hver leveranse av produktet.

- *Godkjenningsfase* – Funksjonelle og ikke-funksjonelle tester for å sikre at hele leveransen fungerte etter forventninger.

Prosjektet jobbet med flere leveranser i parallell. I et utviklingsteam ville noen da jobbe med oppgaver fra forrige leveranse i godkjenningsfase mens andre jobbet med konstruksjon og andre igjen med å definere behov eller løsningsbeskrive funksjonalitet til neste leveranse. Utviklingsteam demonstrerte funksjonalitet på en felles gjennomgang («demo») hver tredje uke, og hadde roller som beskrevet i Tabell 3.1. Teamene hadde 7-9 medlemmer.



Figur 3.2: Oversikt over faser – prosjektet jobbet med leveranser i parallell, en kunne være i produksjon, neste i godkjenningsfase, en annen i løsningsbeskrivelse.

Tabell 3.1: Roller *på teamnivå.*

Rolle	Beskrivelse
Funksjonell arkitekt	Ansvarlig for å detaljere brukerbehov, jobbet typisk 50% med behovsanalyse og løsningsbeskrivelse i prosjekt forretning og 50% med utvikling.
Scrumleder	Fasiliterte daglige møter, iterasjonsplanlegging, demonstrasjoner og retrospektiv.
Teknisk arkitekt	Ansvarlig for teknisk design jobbet 50% på prosjekt arkitektur og 50% på utvikling.
Testansvarlig	Ansvarlig for at testing ble gjennomført på teamnivå: enhetstester, integrasjonstester, systemtest og system-integrasjonstester. Deltok i å lage testkriterier i prosjekt test.
Utviklere	4-5 utviklere – junior og senior-utviklere.

3.2 Behov og kravarbeid

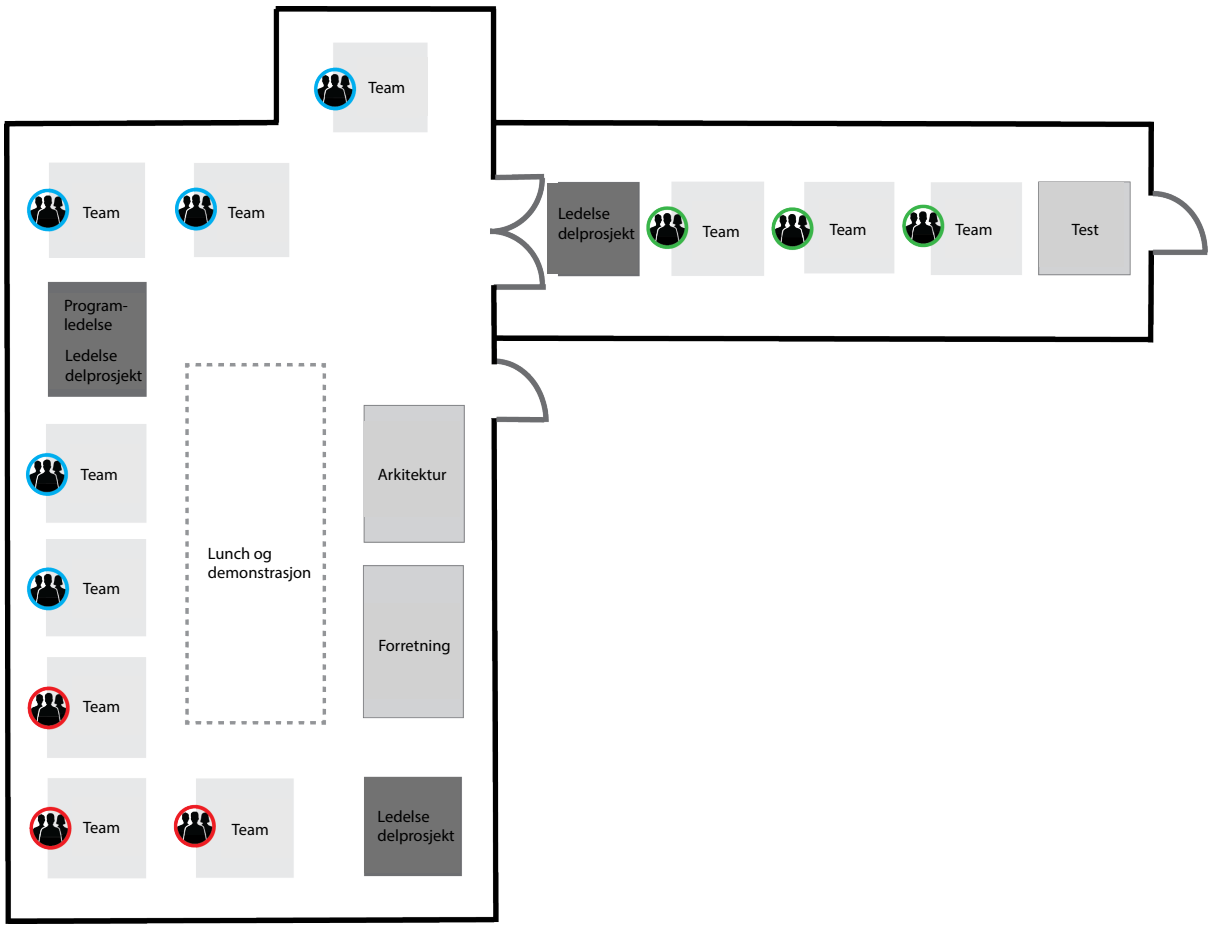
En informant beskrev arbeidet med behov og krav som «å ligge i forkant av utviklingsteamene» med å lage en funksjonell løsningsbeskrivelse med skjermbilder, feilhåndtering, tjenester, konseptuell og logisk datamodell før utviklingsteamene begynte iterasjonsplanlegging. Arbeidet kunne involvere deltakere fra én eller flere av underleverandørene. Ofte byttet utviklingsteamene på hvem som deltok i behovs- og kravarbeid slik at flere i teamet fikk innsikt i behov. Dette førte til at det ble mindre behov for å dokumentere behov og krav. I motsetning til tidligere store prosjekter flere hadde vært med i, så visste de som løsningsbeskrev hvem som ville lese løsningsbeskrivelsene og hva de hadde av bakgrunnskunnskap. Graden av detaljering ble redusert i løpet av prosjektet. Det var et trykk på å få løsningsbeskrevet nok funksjonalitet til at utviklingsteamene kunne gå i gang med arbeid som var av høy prioritet. Det varierte i hvor stor grad teamene løsningsbeskrev funksjonalitet før arbeidet startet. At prosjektet satt i et åpent landskap rundt team- og prosjekt-bord gjorde at det var lett å avklare med representanter for kunder eller på teknisk side. Etterhvert gikk behovs- og løsningsbeskrivelsesfasene over i hverandre, og løsningsbeskrivelser ble ferdige like før utviklingsteamene skulle starte nye iterasjoner, noe informanter beskrev som «kontinuerlig løsningsbeskrivelse».

Sentrale deltakere formulerte følgende læringspunkter om behovs- og løsningsbeskrivelse:

- *Løsningsbeskrivelsesprosess* – brukerhistorier ble beskrevet «akkurat nok» og var klare like før utvikling skulle starte.

- *Kontinuerlig løsningsbeskrivelse* – behovsanalyse og løsningsbeskrivelsesfasene ble slått sammen, noe som førte til at det bare var brukerhistorier som skulle implementeres som ble beskrevet, og de som løsningsbeskrev kjente allerede utviklet løsning og kompetansen i teamene som skulle utvikle.

- *Varierende detaljering av brukerhistorier* – grad av detaljering avhang av typen arbeid som skulle gjøres.



Figur 3.3: Skisse over kontorlandskap der alle team satt på samme plan. Utviklingsprosjektet var delt i tre (team per delprosjekt i egne farger), delprosjektledelse satt ved egne bord i nærheten av sine team.

3.3 Arkitekturarbeid og autonomi

Prosjektet laget en arkitektur som gjorde det mulig for mange team å kunne jobbe i parallell med å utvikle løsningen, avhengigheter til andre team ble forsøkt redusert. Prosjektet prioriterte fremdrift som gjorde at løsningen fikk utfordringer med ytelse, noe som ble fokus mot slutten av prosjektet. Rollen som arkitekt ble beskrevet som krevende, fordi den bestod av kontinuerlig koordinering og forhandling med mange interessenter, hvor det kunne være vanskelig å få aksept for langsiktige løsninger som kunne medføre økte kostnader. Noen informanter hadde ønsket mer eksperimentering og testing før store beslutninger ble tatt, men i hovedsak var arkitekturen stabil gjennom hele prosjektet.

Teamene kunne velge hvordan de løste oppgaver og hadde noe innflytelse på hvilke oppgaver de skulle løse, for eksempel ved at utviklingsteamene var involvert også i forretnings- og arkitekturprosjektet. Det var imidlertid klare begrensninger på teamenes autonomi, som at utviklingsteamene måtte jobbe etter Scrum, i samme takt (3 ukers iterasjoner), og avgi ressurser til arkitektur-, forretnings-, og testprosjektene. Det var rom for variasjoner i hvor detaljert team løsningsbeskrev oppgaver. Arkitekturarbeid blir ofte beskrevet i litteraturen om programvareutvikling som noe som besluttes sentralt og hvor utviklere har begrenset innflytelse. I Perform deltok representanter for alle team i diskusjonene om de viktige design-avgjørelsene.

En erfaringsrapport fra case 2 (Rolland *m.fl.*, 2016) anbefaler å eksperimentere med nye praksiser og beskriver to tilpasninger som ble gjort i prosjektet:

- *Arbeidsgrupper på tvers* – ble satt ned for å løse utfordringer som typisk var tekniske og dreide seg om ikke-funksjonelle krav. Eksempler var ytelse i løsningen, integrasjon med andre system og sikkerhet.

- *Champion-roller*[12] – en ny rolle som roterte mellom personer og som hadde ansvar for standardisering av arbeid innen områder som *java scripting* og *database*. Rollen ble oppfattet til å føre til mer enhetlig løsning og bedre teknisk koordinering.

3.4 Koordinering og kunnskapsdeling

I en lengre periode jobbet det 12 team i parallell med å utvikle løsningen. Prosjektet etablerte en rekke mekanismer for å sikre at teamene var koordinert. På teamnivå ble praksiser i Scrum fulgt, med detaljert planlegging av hver iterasjon, daglige møter, retrospektiver og demonstrasjon hver tredje uke. Demonstrasjonen var den eneste arenaen der alle deltakere i prosjektet var invitert. Hvert delprosjekt i utviklingsprosjektet hadde «Scrum of Scrums»-møter for å avdekke avhengigheter på tvers av team, i tillegg til fora som lunchseminar og andre arenaer for erfaringsdeling. Prosjektet koordinerte arbeidet i to ukentlige «metascrum»-møter hvor prosjektledere og delprosjektledere deltok. Oppgaver ble både skrevet på lapper og var synlige på tavler for hvert team, i tillegg til at de ble registrert i en «issue-tracker».[13] Løsningsbeskrivelser og rutiner for arbeid, arkitektur-retningslinjer, systemdokumentasjon og rapporter fra retrospektiver på teamnivå ble delt på en felles wiki. Det åpne landskapet gjorde at det var raskt å finne folk i andre team eller andre prosjekter for å gjøre avklaringer, og det var også synlig for andre team hvilke oppgaver som ble løst i hvert team. Tavler hadde oppgaver på én side, og den andre siden ble brukt i uformelle diskusjoner.

Flere informanter beskrev at det ble mer uformell koordinering over tid – «mer bruk av tavler, flere felles kaffepauser, flere felles luncher» mens flere av de formelle møtene ble avviklet. Mot slutten av prosjektet ble tema diskutert uformelt først for deretter å bli behandlet i formelle møter og hvor mange pekte på de daglige møtene, Scrum of Scrums-møtene og Metascrum som de viktigste.

[12] Merk at rollen her avviker fra vanlig forståelse i litteratur fra prosjektledelse hvor en *champion* er en som jobber for å fremme prosjektets interessser.

[13] Issue-tracker: Verktøy for oversikt over oppgaver. Muliggjør rask reprioritering og enkel oversikt over status på opgaver.

På teamnivå brukte prosjektet praksiser fra smidige metoder som parprogrammering og retrospektiver som arenaer for læring, mens det på delprosjekt- og prosjektnivå var en rekke møter som beskrevet under delkapittel om koordinering.

En sentralt funn fra studier av programvareutvikling er at prosjekter som har gode prosesser for å integrere kunnskap om domenet en løsning skal fungere i (behov- og krav-) med teknisk kunnskap leverer bedre på tid, kost og kvalitet (Tiwana, 2004). Det at prosjektene hadde roller på teamnivå var trolig viktig for å sikre at det ble god kunnskapsdeling mellom de som jobbet med å forstå behov, de som jobbet med å sikre at leveransene ble gode på kort og lang sikt (test, arkitektur) og de som laget løsningene (utvikling). Med smidige metoder skjer mye av kunnskapsdelingen muntlig, og er «ferskvare» i det man jobbet med å lage en enkelt brukerhistorie. Informanter nevnte eksempler fra tidligere prosjekter hvor kunnskapsdeling ble vanskelig enten på grunn av at brukerhistorier ble løsningsbeskrevet lenge før de skulle utvikles, eller at kunnskapsdeling ble vanskelig fordi prosjektorganisasjonen var spredt på mange etasjer i et kontorbygg som ga medarbeiderne liten innsikt i hva andre drev med i andre etasjer.

Studier av store IT-prosjekter med bruk av smidige metoder fra andre land beskriver også en rekke arenaer for koordinering og kunnskapsdeling (Hobbs & Petit, 2017). Anbefalinger fra Perform-prosjektet (Dingsøyr *m.fl.*, 2019a) er:

- *Ekstra roller* – ekstra roller på teamnivå hadde en rekke fordeler, som å spare tid med muntlig kommunikasjon, at det ikke ble overleveringer mellom faser og at det ble en følelse av at en jobbet sammen om leveransene.

- *Ekstra arenaer* – antall arenaer for koordinering og kunnskapsdeling var langt høyere enn anbefalinger i tidlige beskrivelser av smidige metoder. Erfaringen er at de ekstra arenaene sikret effektiv koordinering mellom teamene.

- *Retrospektiver* – referater fra alle retrospektiver på teamnivå ble lest av prosjektledelsen. Disse tilbakemeldingene førte til endringer og oppdateringer av risikovurdering i prosjektet. En intern evaluering viste at retrospektive ble vurdert som den viktigste praksisen for kontinuerlig læring.

- *Demonstrasjoner som læringsarena:* Korte presentasjoner fra hvert team var viktige for å kommunisere hva hvert enkelt team arbeidet med, og var eneste felles arena i prosjektet.

4 Andregenerasjon storskala smidig metode

Foreldrepengeprosjektet (case 3) varte fra oktober 2016 til juni 2019. Prosjektet mottok digitaliseringsprisen i 2019 for å ha «modernisert og digitalisert hele brukerreisen for kommende og nybakte foreldre».[14] Prosjektet var i første to faser organisert etter en førstegenerasjon storskala smidig modell, med en veldig stor leveranse produksjonssatt i slutten av hver fase. Ved høyeste aktivitet sysselsatte prosjektet nesten 200 medarbeidere fordelt på 10 team, hvor rundt 100 var innleid fra to konsulentfirmaer. Prosjektlederen var ansatt hos prosjekteier som var NAV. Prosjektleder satte ned en arbeidsgruppe for å vurdere organisering og gjennomføringsmodell før siste fase. På slutten av første fase hadde et team fått jobbe autonomt med gode erfaringer. Sentral IT-funksjon hadde også definert en ny måte å jobbe på som gikk bort fra førstegenerasjonsmetoden prosjektet hadde brukt og som var en modifisering av metoden presentert i forrige kapittel. Den nye metoden baserte seg på en ny teknisk plattform der mange ikke-funksjonelle krav ble håndtert i plattformen og hvor utviklingsteam kunne fokusere på funksjonalitet mot brukere. I tredje fase skulle prosjektet videreutvikle og forvalte de produksjonssatte løsningene samtidig som flere relaterte tjenester skulle utvikles. Metodikken fra første og andre fase, med en stor leveranse til slutt, egnet seg ikke i denne fasen. Gruppen fikk mandat til å foreslå endringer i arbeidsform som kunne gjøre at prosjektet kunne jobbe bedre, men som ikke ville øke risiko i fase to eller for å måtte skyve på tidspunkt for når produktet skulle tas i bruk. Både kunde og leverandører var representert i arbeidsgruppen. Arbeidsgruppen anbefalte en ny organisering med tverrfaglige team for alle områder av produktet. Disse teamene skulle være samlokalisert og ha ansvar for helhet og kvalitet. Graden av selvstyre i team skulle tilpasses kobling til andre team og andre avhengigheter, men mange team ville kontinuerlig kunne produksjonssette leveranser. Gruppen foreslo en overgang til kontinuerlige leveranser og også å etablere en kompetansepool med støttefunksjoner som teamene kunne bruke.

Forslaget fra arbeidsgruppen ble godkjent av prosjektleder, og dette førte til store endringer for siste fase. Prosjektet gikk da over til en andregenerasjons storskala smidig metode med autonome team. I denne fasen skulle prosjektet lage selvbetjeningstjenester som var integrert med en vedtaksløsning, samt støtte integrasjon mot andre offentlige aktører. Mål for fasen inkluderte å lage en fullstendig integrasjon mellom en planleggingskalender og løsning for dialog mellom bruker og saksbehandler. Forrige fase hadde laget en «minimumsløsning» av kjernefunksjonalitet som nå skulle videreutvikles.

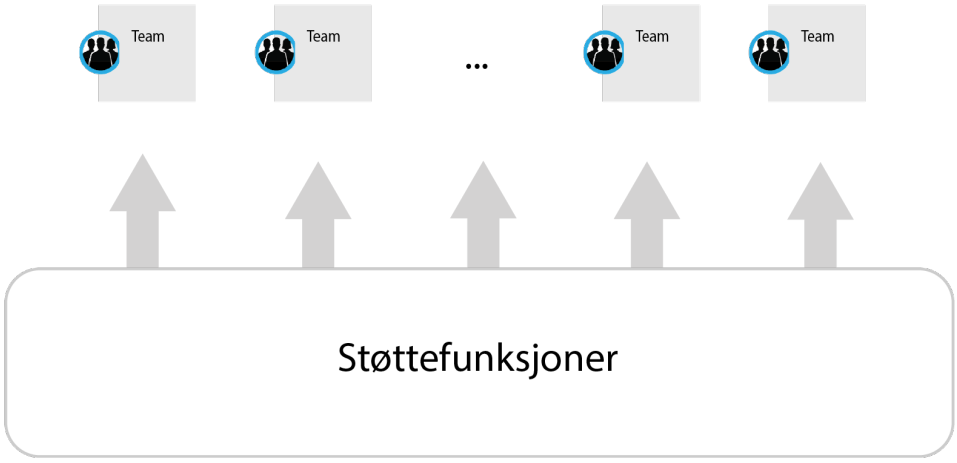
4.1 Organisering

Utvikling ble nå organisert etter en flytbasert modell som gjorde at mange roller i prosjektet forsvant, og det ble etablert nye tverrfaglige team med folk både fra kunde og fra to leverandører. Kontinuerlige leveranser startet første november 2018, dagen etter produksjonssetting av leveransen fra fase to. Mange møtearenaer forsvant. Nye støttefunksjoner

[14] https://www.cw.no/artikkel/priser-utmerkelser/digitaliseringsprisen-2019-til-nav

ble etablert (Figur 4.1 og Tabell 4.1). Kontraktsmodell ble endret slik at leverandører leverte ressurser til kunde.

De autonome teamene ble betegnet som tverrfaglige autonome *produktteam* og hadde rundt 12 medlemmer uten formell ledelse. Hvert team hadde en produkteier. Teamene ble fysisk samlokaliserte rundt et teambord. De fikk regelmessig råd om arbeidsmetoder av en smidig coach og løsningsarkitekter. Kompetansebehov for hvert team er beskrevet i Tabell 4.1.



Figur 4.1: Ny organisering med produktteam og støttefunksjoner. Fra forslag til ny leveransestrategi.

Tabell *4.1: Støttefunksjoner og kompetansebehov i team. Fra forslag til ny leveransestrategi.*

Støttefunksjoner:	**Kompetansebehov i team:**
• Forretningsansvarlig • Gevinstrealisering • Helhetlig arkitektur • Helhetlig design «UX»[15] • Koordinering mot «eksterne» team • Miljøkoordinering • Omstilling og kommunikasjon • Prosjektledelse • Project Management Office-funksjon (tilpasset ny modell) • Verdikjedetesting på tvers av team	• Brukerinteraksjon («UX») • Fag- og forretning • Funksjonell • Teknisk arkitektur • Utvikling • Test • Teamlead

[15] «UX» - vanlig forkortelse for User eXperience, altså hvordan bruker vil oppleve et digitalt produkt.

4.2 Behov og kravarbeid

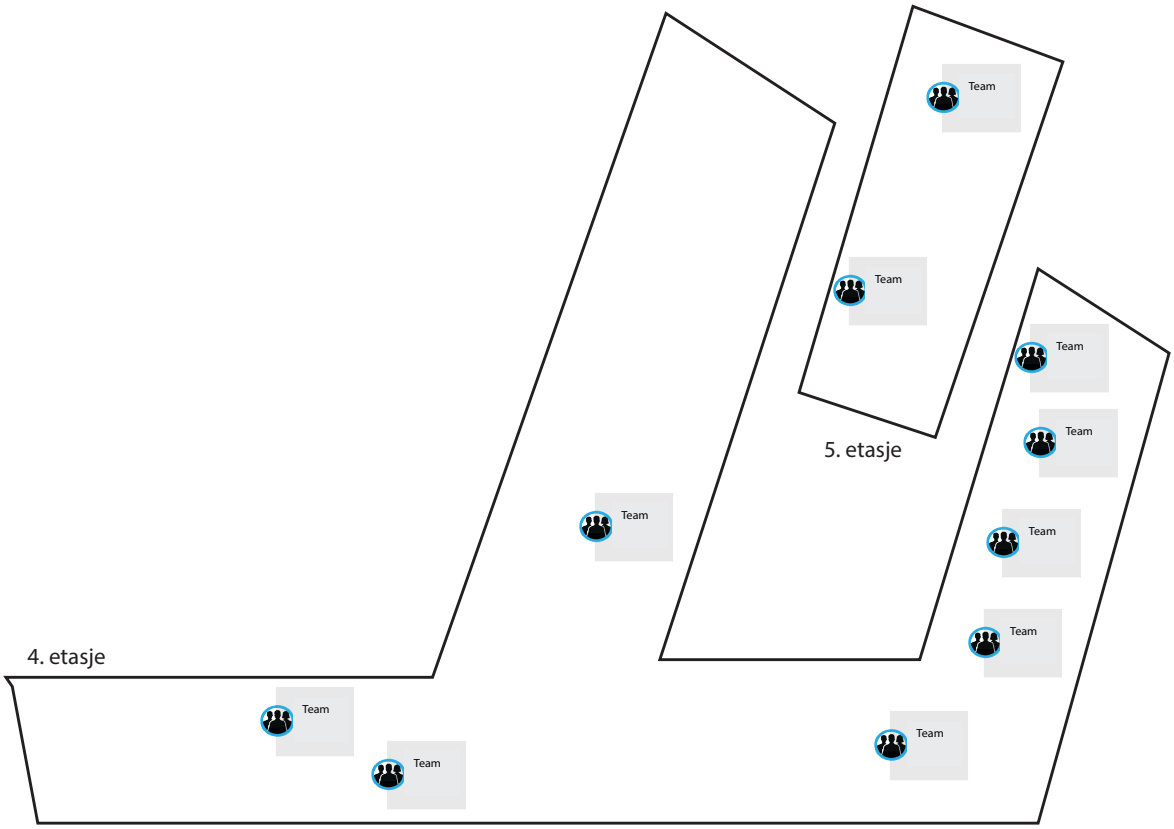
I første fase av prosjektet da de jobbet etter en førstegenerasjons storskala smidig metode, var det en underleverandør som jobbet med behovsanalyse og behovsbeskrivelse mens en annen hadde løsningsbeskrivelses-, konstruksjons- og vedlikeholdsfaser. Informanter beskrev at «… *både behovsdokumentasjon og løsningsbeskrivelsene var særdeles omfattende*» og at «*ting var veldig spekket*». Det ble overleveringer mellom mange, og informanter beskrev også at avklaringer var tidkrevende på grunn av det de opplevde som en møtekultur. Viktige interessenter for avklaringer var ofte opptatt i møter, og for å få avklaringer ble man derfor nødt til å invitere til nye møter som gjerne ble et stykke fram i tid hvis en skulle få med alle som var nødvendig for en avklaring. Det kunne gå lang tid fra en behov ble beskrevet av noen til tilhørende brukerhistorier ble konstruert av andre. Dette gjorde også avklaringer vanskeligere da de som jobbet med behov og konstruksjon ikke jobbet med de samme tingene til samme tid. Gjennomføringsmodellen skiller seg her fra Perform-prosjektet som hadde forretningssiden inkludert også i utviklingsteamene.

Ved overgang til autonome team ble forretningsressurser og utviklingsressurser satt sammen i team som var samlokalisert. Informanter beskrev endringen som at den førte til mer felles retning, økt tillit og «*veldig mye mindre dokumentasjon. Det var veldig deilig og jeg tror alle satte pris på det*». Mange av rollene forsvant og teamene fikk mer ansvar. Informanter rapporterer om at de jobbet med færre brukerhistorier men opplevde at det ble høyere kvalitet på leveransene. Det «*gikk litt ned hvor mange ting [brukerhistorier] jeg jobbet med til enhver tid. Men kvaliteten på det som ble gjort ble større fordi ting tok så veldig lang tid før det, så måtte man jo på en måte fylle opp med flere ting som man hadde i prosess hele tiden, mens nå var det mer sånn ´dette behovet får vi ut i løpet av uka´.*»

Teamene tok mer ansvar for at prosjektet skulle gi verdi, for eksempel ved at de tok inn oppgaver utover prosjektmandat ved å fornye eksterne websider, som var viktig for å få flere sluttbrukere til å velge digital løsning. En informant beskrev endringen som «*jeg opplevde jo vi fikk mye mer verdi ut i produksjon, si siste halve året enn før*».

For utviklerne ble det mindre detaljerte beskrivelser og nødvendig å snakke mer med forretningssiden. Noen team innførte en oppstartssamtale ved nye oppgaver, «*hver gang noen skulle ta en ny historie, så hadde vi en gjennomgang, hva er dette her, hvorfor skal det være sånn? Hva er det vi er ute etter? Så det gikk mye mer muntlig, men selve beskrivelsen av historien kanskje bare var en setning eller to*».

Nye roller som hadde ansvar for helhetlig løsning forsvant fort, «*Teamene begynte å ta ansvar selv*».



Figur 4.2: Skisse over kontorlandskap der prosjektet satt på to plan.

Studiet fra en Ericsson-enhet (Paasivaara & Lassenius, 2019) påpeker viktigheten av å involvere utviklere i beslutninger. «*...hvis produkteier eller en arkitekt bestemmer noe så er det ingen som tar det seriøst før de har organisasjonen bak seg. Beslutninger må tas av organisasjonen og være akseptert av organisasjonen*».

En oppsummerende studie av produkteier-rollen i andregenerasjons storskala smidig-metoder beskriver rollen som utfordrende i at den typisk består i å håndtere innspill fra mange interessenter med forskjellige behov (Bass & Haxby, 2019). Rollen er i praksis ikke én person, men et team med viktige oppgaver i å ha en langsiktig visjon for prosjektet, forhandle med interessenter og å prioritere arbeidsoppgaver og hva som skal lanseres når. Studien anbefaler produkteiere å ha fysiske møter for å bygge tillit – produkteiere som bygger gode nettverk med interessenter ser ut til å lykkes bedre. I Foreldrepengeprosjektet ble produkteier-forum re-introdusert litt ut i den siste fasen. Videre anbefaler forskerne fokus på hovedmål og å vise fremdrift mot hovedmål. I komplekse prosjekter anbefaler de å starte med et «minimumsprodukt». Studien understreker videre at behovs- og kravarbeid er en omfattende jobb som krever betydelig innsats.

4.3 Arkitekturarbeid og autonomi

En ny teknisk plattform muliggjorde mer autonomt arbeid for teamene. På grunn av mange avhengigheter mellom systemer hadde NAV tidligere fire årlige leveranser av ny funksjonalitet. En leveranse kunne inneholde 40 - 75 000 timer med utviklingsarbeid. Det kunne ta måneder fra en ny brukerhistorie var ferdig utviklet til den ble tatt i bruk (Vestues & Rolland, 2021).

Modularisering i ny plattform gjorde det mulig å gå fra fire årlige leveranser til at noen team kunne lansere så ofte de ville, og mange ikke-funksjonelle krav ble ivaretatt i den nye plattformen.

Før det ble bestemt å endre gjennomføringsmodell ble viktige tema diskutert i workshoper, som erfaringer fra forsøk med et autonomt team og arkitekturavklaringer. En informant beskrev disse workshopene som «*det jeg tror fungerte så enormt bra med de workshopene vi hadde der det var at vi ... gikk igjennom tema der folk fikk ta opp det som var bekymringer eller man trodde det kanskje kunne gå galt*». Arkitekturarbeid innebærer ofte å etablere arkitekturstiler, referanse-arkitekturer, design-avgjørelser og å ta valg av arkitektur-mønstre (Bass & Haxby, 2019). Med ny modell ble mange tekniske valg tatt av andre team med ansvar for den tekniske plattformen, men det innebar også mer ansvar overført til teamene, riktignok med støttefunksjon på helhetlig arkitektur.

Noen opplevde at det ble stor variasjon i teamene, «…*det jeg synes var en ulempe i starten, var at det var et så veldig sterkt ønske fra noen at teamene skulle selv få bestemme alt*» og at det også kunne være mer utfordrende å vite hvilke team som holdt i hva etter at roller og arenaer forsvant.

Ved innføring av kontinuerlige leveranser ble det etablert en ny arbeidsprosess som beskrevet i Figur 4.3. Teamene utviklet nye brukerhistorier og gjennomførte automatiske og manuelle tester. Ny funksjonalitet kunne slås av og på slik at ny kode kunne kjøre for å sjekke at alt fungerte teknisk før funksjonalitet ble tilgjengelig for brukere. Et viktig nytt møte som ble etablert var «go / no-go»-møtene med deltakere i Tabell 4.2. I disse møtene gikk deltakerne gjennom og bestemte hvilken funksjonalitet som skulle produksjonssettes. Teamene fylte ut release notes (hvilke brukerhistorier som er inkludert og kommentarer på testarbeid) i forkant, og møtet ble avholdt alle dager det var behov.

Figur 4.3: Prosess for produksjonssetting, fra dokument «kontinuerligere leveranser» v 14. (Forenkles og oversettes, tegnes på nytt)

Tabell *4.2: En rekke roller ble invitert til go / no-go-møtene i forkant av produksjonssetting.*

Deltakere på go / no-go møte	
• Forretningsansvarlig • Løsningsarkitekt • Omstilling • Team FP	• DP Test • En deltaker per utviklingsteam • Release manager (møteleder) • Andre ved behov

En studie av en enhet i Ericsson (Paasivaara & Lassenius, 2019) som forvalter en telecom-løsning på rundt 30 millioner linjer kode, og som har rundt 400 ansatte, beskriver hvordan

overgang til en andregenerasjons storskala smidig metoder (large-scale scrum) har ført til at team har fått større autonomi. Som i case 3 har team ansvar for design, implementasjon og testing av sine brukerhistorier. Hvis en brukerhistorie eller oppgave påvirker et annet team tas dette opp i et praksisfellesskap. Her tas beslutninger som tidligere ble tatt av ledelse, og dette er fora hvor alle i enheten kan delta. I tillegg til koordinering og kunnskapsdeling (se neste avsnitt), blir praksisfellesskap brukt til å ta avgjørelser om arkitektur. Ledelsens oppgaver har gått fra å overvåke og kontrollere til å støtte organisasjonen. Forslag til beslutninger blir presentert og diskutert i praksisfellesskapene. Møtene blir ledet av en fasilitator, og avgjørelser tas bare hvis ingen har alvorlige motforestillinger. Et utsagn fra en informant var at «*[møtene] fungerer veldig bra, du må ikke delta, men basert på om du tror du har noe å bidra med i diskusjonen eller det er noe som interesserer deg... vårt praksisfellesskap fungerer bra i den forstand at du alltid får hjelp når du spør*». Tidligere ble arkitekturarbeid gjort utenfor teamene, en produkteier beskrev endringen som «*vi deler informasjon og løser problemer sammen og tar beslutninger ... tidligere hadde vi en egen avdeling som ville skrive spesifikasjoner flere år i forkant og som var skrevet i stein og ble levert til utviklerne. Nå er det motsatt. Alle de tidligere systemekspertene og system-testerne er integrerte i teamene*». Forskerne hevder at praksisen med medvirkning i beslutninger har ført til høyere motivasjon blant de ansatte, bedre bruk av kunnskapen til ansatte og bedre beslutninger. De understreker imidlertid at det har tatt lang tid å komme dit organisasjonen er nå, praksisfellesskap ble innført fem år etter innføring av smidige metoder i hele enheten og de har nå tre års erfaring med denne måten å jobbe på.

4.4 Koordinering og kunnskapsdeling

I de første fasene med førstegenerasjonsmetode opplevde informanter at de var godt koordinert internt på kundeside og på leverandørside. Men, «*fra å sitte på kundesiden så var det helt umulig å få innsikt og forståelse av hva som skjedde og hvordan de var organisert, hvordan man jobber altså. Det var akkurat som om man leverte behov og så var det en black box det kom noe ut av*». Det ble større utfordringer med koordinering ettersom prosjektstørrelsen økte. Det var imidlertid mange med erfaring fra tidligere store IT-prosjekter som var klar over arbeidsoppgaver som kunne havne mellom team. Det var også en rolle som konstruksjonsansvarlig på utviklingssiden som tok tak i det som ble liggende mellom team.

Ved reorganisering til autonome team forsøkte de å ha så få avhengigheter som mulig mellom teamene, som ble oppdelt etter domener.[16] «*Tidligere koordinerte man seg rundt behov, løsning, utvikling, test, langs en sånn utviklingsprosess-akse*». Ved ny organisering ble koordinering tatt på teamnivå og fokusert mer på produkt. Det var bekymringer før endringen «*... hvordan vi skulle klare å holde oversikt over... hvordan skulle vi koordinere det her og hvordan passe på at ting henger sammen og at teamene snakker sammen og hvem har liksom det overordnede oppsynet her?*» Overgangen til autonome team ble oppfattet til å ha flere positive effekter på teamnivå: «*Man får mye tettere dialog med utviklerne, altså fra kundesiden når man sitter sammen i tverrfaglige team, det styrker utviklernes forståelse av domenet, og produkteiernes forståelse av det tekniske, du sparer masse tid og får levert mer*».

[16] Utviklingsteamene var også i første faser oppdelt etter domener.

Arenaer for koordinering ble imidlertid tatt bort ved overgang til autonome team. Dette var et bevisst valg for at teamene skulle ta ansvar for å etablere nye arenaer for koordinering der det var nødvendig. Noen opplevde at det var ulike praksiser i teamene og at det var vanskelig å vite hvilket team som holdt i hva. I ettertid mente noen at de burde beholdt flere arenaer med fokus på prosjektmål og erfaringsutveksling mellom team.

Etter en periode ble enkelte tidligere praksiser for koordinering og kunnskapsdeling på tvers av teamene gjeninnført: «…*rett før vi avsluttet prosjektet så hadde ting gått seg til veldig. Og da hadde vi fått mer sånn strømlinjeformet verktøybruk og samhandling, koordinering. Vi hadde også Scrum of Scrums»*. Dette inkluderte også møter mellom produkteiere og møter i et teknisk forum – de fleste arenaene rundt det som hadde vært roller i første fase. «*Så etter hvert, for det tok litt tid, så ble det sånn, ok, nå jobber alle litt mer likt».* Et møte som ble innført var «go/no go»-møtene som beskrevet i forrige avsnitt for å bestemme om ny funksjonalitet var klart for produksjonssetting. Likevel var det tatt forskjellige valg i team som ble en utfordring da produktet gikk over i en vedlikeholdsfase, «*ti autonome team som har tatt forskjellige valg på dokumentering og som skal smelte til et team... stor utfordring å overta»*.

En studie av kunnskapsdeling hos Spotify viser hvordan programvarefirmaet organiserer *praksisfellesskap* for både å dele kunnskap på tvers av team, samt som en arena for å koordinere arbeid mellom team. Organisasjonen har en rekke praksisfellesskap på tvers av team med frivillig deltakelse for ansatte som er interessert i et felles tema. Smite et al. (2019) fant fire hovedtyper av praksisfellesskap i sin studie, med fokus på «smidig metodikk», «kjerne-infrastruktur», «back-end utvikling» og «front-end utvikling». Studiet lister opp en rekke opplevde fordeler som større perspektiv på problemer, tilgang til ekspertise, mulighet til å forutse videre teknologisk utvikling, men rapporterer også utfordringer som at praksisfellesskap har uklare mandat, lite støtte i organisasjonen og at folk ikke har dedikert tid til å bidra. Forskerne mener imidlertid at praksisfelleskapene spiller en viktig rolle for å jobbe smidig i stor skala, at de bidrar til kunnskapsdeling og felles eierskap av programkode, samt at det bidrar til å jevne ut praksiser på tvers av team på forskjellige geografiske lokasjoner. En studie av praksisfelleskap hos Ericsson (Paasivaara & Lassenius, 2019) trekker også fram rollen disse har hatt både for koordinering og kunnskapsdeling.

5 Konklusjon

Vi har undersøkt spørsmålet *hvordan organisere store digitaliseringsprosjekter* gjennom å presentere erfaringer og forskningsfunn fra to måter å organisere digital produktutvikling, med eksempler fra det vi har kalt førstegenerasjon og andre-generasjons metoder for stor-skala smidig utvikling.

Generelt er råd både fra feltet prosjektledelse og fra feltet programvareutvikling å redusere størrelsen på digitaliseringsprosjekter (Ebert & Paasivaara, 2017; Jørgensen, 2015). Men ofte må prosjekter bli store for å lage produkter som støtter et helt område og det kan være nødvendig med prosjekter eller videreutvikling som strekker seg over lang tid og som involverer mange utviklingsteam.

Da smidige metoder kom på 2000-tallet mente de fleste at de egner seg for små team som er samlokaliserte og som lager programvare som ikke er sikkerhetskritisk. I dag brukes metodene i distribuert utvikling, i sikkerhetskritisk utvikling og også i økende grad i store prosjekter. For større prosjekter er det en generell anbefaling i «Agile Practice Guide» å sikre at organisasjonen først håndterer utvikling for et enkelt team. Digitaliseringsdirektoratets prosjektveiviser anbefaler å vurdere prosjektets omgivelser og om berørte parter har forstått smidige hovedprinsipper. Våre case er fra miljø som har lang erfaring med programvareutvikling og har forbedret praksiser over tid.

Overordnet er utfordringen i digitaliseringsprosjekter å omsette brukerbehov til løsninger i en kontekst hvor det ofte er usikkerhet om hva behovet egentlig er og hvordan behovet best kan løses ved stadige endringer i teknologi. Gode prosesser for å kombinere innsikt i forretning («domenet») og i løsningsrom («teknologi») er kritisk for gode digitaliseringsprosjekter.

Erfaringer med tre store vellykkede digitaliseringsprosjekter viser praksiser for hyppig integrering av komponenter og demonstrasjon av produkt. Prosjektene har stor involvering fra forretningssiden for avklaringer. Et annet fellestrekk ved prosjektene er stor innsats i koordinering og kunnskapsdeling mellom team.

Sentrale utfordringer i organisering av store digitaliseringsprosjekt er hvordan arbeidet med behov og krav, arkitektur og autonomi, samt koordinering og kunnskapsdeling organiseres. I det følgende oppsummerer vi funn på hvert av disse temaene førstegenerasjon og andre-generasjons metoder for stor-skala smidig utvikling:

- *Hvordan er store digitaliseringsprosjekt organisert i førstegenerasjon og andregenerasjons metoder for storskala smidig utvikling?*

- *Hvordan sikrer man at løsninger tilfredsstiller kundebehov når mange team jobber i parallell og hvor produktet ofte skal lanseres på mange brukere?*

- *Hvordan sikre gode tekniske løsninger når mange team jobber i parallell, og hvordan sikre at team ikke sinker fremdrift hos andre team?*

- *Hvordan sikre tilstrekkelig koordinering og kunnskapsdeling mellom team som jobber etter smidige prinsipper?*

Førstegenerasjon storskala smidig-metoder viser organisering av digitaliseringsprosjekter gjennom å kombinere råd fra prosjektledelse som å dele prosjekter i faser, etablere program som består av prosjekt og del-prosjekt og ekstra roller med råd fra smidige metoder om små team, få men faste møter, jevnlige integrasjoner og demonstrasjon av produkt og arenaer for læring og prosessforbedring. Viktige læringspunkter fra førstegenerasjon var at:

- Viktige praksiser for å sikre at løsningen tilfredsstilte kundebehov var «kontinuerlig løsningsbeskrivelse» og at prosjektet åpnet for å detaljere behov avhengig av typen arbeid.

- Prosjektene sikret involvering i avgjørelser om arkitektur gjennom at arkitekturprosjektet hadde deltakere fra alle utviklingsteam.

- Arkitektur ble laget for å muliggjøre arbeid i parallell og hvor en forsøkte å minimere avhengigheter på tvers av team.

- Case 2 gir innsikt i praksiser for å løse utfordringer på tvers av team gjennom *arbeidsgrupper på tvers* og bruk av *champion-roller*.

- Det trengs betydelig flere koordineringsarenaer enn i tidlige rammeverk: Ekstra roller og arenaer.

- Matrisemodell for å minimere overleveringer mellom faser, «kontinuerlig løsningsbeskrivelse».

- Praktikere påpeker viktighet av kontinuerlig forbedring og læring.

De siste årene har det kommet mange forslag til nye rammeverk – andregenerasjon storskala smidige metoder som erstatter råd fra prosjektledelses-rammeverk med mer tilpassede arbeidsprosesser og roller. Det er så langt få studier på slike rammeverk og hvordan de fungerer i praksis, men forskere har rapportert om at valg av rammeverk i mange bedrifter er tatt på ad-hoc basis, og at det ofte blir spørsmål når en kommer inn på problemstillinger som ikke er dekket i rammeverket. Erfaringer med en andregenerasjon med organisering i «autonome team» med kontinuerlige leveranser indikerer:

- Kontinuerlige leveranser fører til en annen type læring om bruk av produktet enn det som var mulig med sjeldne leveranser.

- Autonome team som har både forretnings- og teknisk kompetanse oppfattes å ha en rekke positive effekter som bedre evne til å håndtere endringer da prioriteringer og arbeid med behov, løsning og utvikling gjøres av teamet. En informant uttrykte det som *«det styrker utviklernes forståelse av domenet, og produkteiernes forståelse av det tekniske, du sparer masse tid og får levert mer».*

- Forskere understreker viktighet av fokus på fremdrift mot hovedmål, læring gjennom å starte med et «minimumsprodukt» og at behov- og kravarbeid er en omfattende jobb som krever betydelig innsats.

- Et ny viktig arena ved innføring av kontinuerlige leveranser var «go / no go»-møtene for å bestemme om ny funksjonalitet skulle settes i produksjon.

- Økt ansvar på team ga i Foreldrepenger en periode med manglende koordinering mellom team før nye arenaer ble etablert. Arbeidsinndeling etter produkt førte imidlertid til mindre behov for koordinering mellom team.

- Praksisfellesskap på tvers av team kan være en viktig arena for å koordinere og harmonisere arbeid i teamene.

Beskrivelsene av første- og andregenerasjonsmetoder for storskala smidig utvikling viser erfaringer som kan være nyttige for kommende digitaliseringsprosjekt. Ethvert prosjekt vil imidlertid være forskjellig fra tidligere prosjekter og det vil være nødvendig med tilpasninger til prosjektkontekst. En hovedanbefaling fra praktiker-miljøet vil være å kontinuerlig justere metode, for eksempel etter innspill gjennom praksiser som retrospektiver.

I dag er det mange forslag til hvordan en kan organisere digitaliseringsprosjekter, vi har foreløpig få empiriske studier på hele metoder eller hvordan praksisene i metodene fungerer. Det er stor avstand mellom de som argumenterer for store rammeverk som Scaled Agile Framework (SAFe) og andre som argumenterer for minimale rammeverk kombinert med kontinuerlig forbedring. Vi håper forskningsmiljø i fremtiden vil kunne gi anbefalinger basert på en større mengde studier, på hvordan praksiser kan tilpasses konteksten de skal brukes i for å sikre involvering av forretning, balansere autonomi og sikre koordinering og kunnskapsdeling. Kanskje kan vi om noen år se en tredje generasjon med metoder for storskala programvare utvikling som i større grad er basert på forskning.

Endringene vi har sett i digitaliseringsprosjekter tror vi vil påvirke både praksis og forskning innen prosjektledelse. Når fokus for arbeid endres fra plan til produkt vil det ha konsekvenser både for hvordan kvalitetssikring av digitaliseringsprosjekter kan gjøres, og for hva som er innholdet i rollen som prosjektleder. Kontinuerlige leveranser eller utvikling i iterasjoner endrer på forståelse av hvordan usikkerhet håndteres.

Vi håper de rike beskrivelsene av digitaliseringsprosjektene og funnene fra studier i fagfeltet programvareutvikling vil være nyttige inspirasjonskilder i planlegging av nye digitaliseringsprosjekter.

Referanser

Ågerfalk, P. J., & Fitzgerald, B. (2006). Flexible and distributed software processes: Old petunias in new bowls? *Communications of the Acm, 49*(10), 26-34. doi:10.1145/1164394.1164416

Bass, J. M. (2019). *Future Trends in Agile at Scale: A Summary of the 7th International Workshop on Large-Scale Agile Development*, Cham.

Bass, J. M., & Haxby, A. (2019). Tailoring product ownership in large-scale agile projects: managing scale, distance, and governance. *IEEE Software, 36*(2), 58-63.

Batra, D., Xia, W., VanderMeer, D., & Dutta, K. (2010). Balancing Agile and Structured Development Approaches to Successfully Manage Large Distributed Software Projects: A Case Study from the Cruise Line Industry. *Communications of the Association for Information Systems, 27*(1), 21.

Bentley, C. (2010). *PRINCE2 Revealed*: Butterworth-Heinemann.

Bick, S., Spohrer, K., Hoda, R., Scheerer, A., & Heinzl, A. (2017). Coordination challenges in large-scale software development: a case study of planning misalignment in hybrid settings. *IEEE Transactions on Software Engineering, 44*(10), 932-950.

Bjørnson, F. O., Wijnmaalen, J., Stettina, C. J., & Dingsøyr, T. (2018). *Inter-team Coordination in Large-Scale Agile Development: A Case Study of Three Enabling Mechanisms*. Paper presented at the XP2018, Porto, Portugal.

Bræk, R., Hygen, J., Høysæter, S., Johansen, T., & Scott, P. (1985). Håndbok i systemarbeid: TAPIR, Trondheim.

Conboy, K., & Carroll, N. (2019). Implementing large-scale agile frameworks: challenges and recommendations. *IEEE Software, 36*(2), 44-50.

Dingsøyr, T., Dybå, T., Gjertsen, M., Jacobsen, A. O., Mathisen, T.-E., Nordfjord, J. O., . . . Strand, K. (2019a). Key Lessons from Tailoring Agile Methods for Large-Scale Software Development. *IEEE IT Professional, 21*(1), 34-41. doi:10.1109/MITP.2018.2876984

Dingsøyr, T., Fægri, T., & Itkonen, J. (2014). What Is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development. In A. Jedlitschka, P. Kuvaja, M. Kuhrmann, T. Männistö, J. Münch, & M. Raatikainen (Eds.), *Product-Focused Software Process Improvement* (Vol. 8892, pp. 273-276): Springer International Publishing.

Dingsøyr, T., Falessi, D., & Power, K. (2019b). Agile Development at Scale: The Next Frontier. *IEEE Software, 36*(2), 30-38. doi:10.1109/MS.2018.2884884

Dingsøyr, T., & Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery. *Information and Software Technology*. doi:http://dx.doi.org/10.1016/j.infsof.2016.04.018

Dingsøyr, T., Moe, N. B., Fægri, T. E., & Seim, E. A. (2018a). Exploring Software Development at the Very Large-Scale: A Revelatory Case Study and Research Agenda for Agile Method Adaptation. *Empirical Software Engineering, 23*(1), 490-520. doi:https://doi.org/10.1007/s10664-017-9524-2

Dingsøyr, T., Moe, N. B., & Seim, E. A. (2018b). Coordinating Knowledge Work in Multi-Team Programs: Findings from a Large-Scale Agile Development Program. *Project Management Journal, 49*(6), 64-77. doi:DOI: 10.1177/8756972818798980

Duncan, W. R. (2017). *A guide to the project management body of knowledge* (Sixth Edition ed.). Newtown Square, PA: Project Management Institute.

Ebert, C., & Paasivaara, M. (2017). Scaling agile. *IEEE Software, 34*(6), 98-103.

Finne, H. (2019). *Styring og gjennomføring av store statlige IKT-prosjekter: Eksperters erfaringer og vurderinger*. Retrieved from Trondheim:

Fitzgerald, B., & Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software, 123*, 176-189. doi:http://dx.doi.org/10.1016/j.jss.2015.06.063

Hanssen, G. K., Stålhane, T., & Myklebust, T. (2018). *SafeScrum - Agile Development of Safety-Critical Software*: Springer.

Hobbs, B., & Petit, Y. (2017). Agile methods on large projects in large organizations. *Project Management Journal, 48*(3), 3-19.

Jørgensen, M. (2015). *Suksess og fiasko i offentlige IKT-prosjekter: En oppsummering av forskningsbasert kunnskap og evidensbaserte tiltak*. Retrieved from https://www.regjeringen.no/contentassets/9018344feae44c1f9a2a114e768ebd1b/suksess_fiasko_offentlige_ikt-prosjekter.pdf

Keil, M., Cule, P. E., Lyytinen, K., & Schmidt, R. C. (1998). A framework for identifying software project risks. *Communications of the ACM, 41*(11), 76-83.

Lappi, T., Karvonen, T., Lwakatare, L. E., Aaltonen, K., & Kuvaja, P. (2018). Toward an improved understanding of agile project governance: a systematic literature review. *Project Management Journal, 49*(6), 39-63.

Meyer, B. (2018). Making Sense of Agile Methods. *IEEE Software, 35*(2), 91-94. doi:10.1109/MS.2018.1661325

Mohagheghi, P., & Jørgensen, M. (2017). What Contributes to the Success of IT Projects? An Empirical Study of IT Projects in the Norwegian Public Sector. *JSW, 12*(9), 751-758.

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM, 48*(5), 72 - 78.

Niederman, F., Lechler, T., & Petit, Y. (2018). A Research Agenda for Extending Agile Practices In Software Development and Additional Task Domains. *Project Management Journal, 49*(6), 3–17. doi:https://doi.org/10.1177/8756972818802713

Paasivaara, M., Behm, B., Lassenius, C., & Hallikainen, M. (2018). Large-scale agile transformation at Ericsson: a case study. *Empirical Software Engineering, 23*(5), 2550-2596.

Paasivaara, M., & Lassenius, C. (2019). Empower Your Agile Organization: Community-Based Decision Making in Large-Scale Agile Development at Ericsson. *IEEE Software, 36*(2), 64-69.

Patton, J., & Economy, P. (2014). *User story mapping: discover the whole story, build the right product*: " O'Reilly Media, Inc.".

Project Management Institute and Agile Alliance. (2017). *Agile Practice Guide*: Project Management Institute.

Rolland, K. H., Mikkelsen, V., & Næss, A. (2016). *Tailoring agile in the large: Experience and reflections from a large-scale agile software development project.* Paper presented at the International Conference on Agile Software Development.

Royce, W. (1970). *The software lifecycle model (waterfall model).* Paper presented at the Proc. Westcon.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering, 14*, 131-164.

Schmitz, K., Mahapatra, R., & Nerur, S. (2019). User Engagement in the Era of Hybrid Agile Methodology. *IEEE Software, 36*(4), 32-40. doi:10.1109/MS.2018.290100623

Schwaber, K., & Beedle, M. (2001). *Agile Software Development with Scrum.* Upper Saddle River: Prentice Hall.

Schwaber, K., & Sutherland, J. (2020). *Scrum Guiden: Den definitive guide til Scrum: Spillets regler.* Retrieved from https://www.scrumguides.org

Skelton, M., & Pais, M. (2019). *Team Topologies: Organizing Business and Technology Teams for Fast Flow*: It Revolution.

Smite, D., Moe, N. B., & Ågerfalck, P. (2010). *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*: Springer Verlag.

Smite, D., Moe, N. B., Levinta, G., & Floryan, M. (2019). Spotify guilds: how to succeed with knowledge sharing in large-scale agile organizations. *IEEE Software, 36*(2), 51-57.

Tiwana, A. (2004). An empirical study of the effect of knowledge integration on software development performance. *Information and Software Technology, 46*(13), 899-906.

Vestues, K., & Rolland, K. H. (2021). Platformizing the Organization through Decoupling and Recoupling - A Longitudinal Case Study of a Government Agency. *Scandinavian Journal of Information Systems, To appear*.

Vlietland, J., & van Vliet, H. (2015). Towards a governance framework for chains of Scrum teams. *Information and Software Technology, 57*, 52-65.

Vedlegg: Eksempel på intervjuguide

Intervjuguide 1, case 3:

Intervjuguide: Undersøkelse om storskala smidig systemutvikling

Basert på intervjuguider fra Perform og <case2>-studiene. Intervjuguide prosjektledelse.

Oppstart - kontekst

- Forskningsprosjekt – anonymitet
- Lydopptak
- Fokus på hele prosjektet - fra innledende arbeid før prosjektet, <Fase1>, <Fase2>, <Fase3>, eventuelt etter prosjektet

- Tegne tidslinje for prosjektet
- Endringer: antall team, roller, teamstørrelse, andre større endringer?
- Avhengigheter mot andre prosjekt/systemer/prosesser i organisasjonen til kunden

Din Rolle

- Beskrive prosjektlederrollen
- Forskjell fra tilsvarende roller i tidligere prosjekt?
- Spesielle utfordringer?

Forfase:

- Kontraktsmodell
- Leveransemodell
- Prosjektplan
- Overordnet organisering

- Viktigste interessenter og arbeid mot interessenter
- Analyse av eksisterende system
- Risiko i prosjektet
- "Vanskelige avhengigheter" - i spesielle faser?
- Endring i sourcingmodell hos kunde
- Viktigste endringer i arbeidsprosesser? (koblet til fasene?)
- Viktigste endringer i roller (koblet til fasene?)
- Andre viktige endringer?
- Fysisk arbeidsmiljø: kontorlandskap, tavler, møteplasser (versus Perform)

<Fase1>:

- Hva var viktigste leveranser i <Fase1>
- Viktigste utfordringer i prosjektet?
- Hvordan ble det arbeidet med løsningsbeskrivelse?
- Utfordringer rundt løsningsbeskrivelse – konstruksjon – test
- Oppbygning med flere team: hvordan ble det håndtert / førte det til endringer i arbeidspraksis?
- I hvilke sammenhenger var koordinering mellom team nødvendig?
  Løsningsbeskrivelse - konstruksjon - test
- Hvordan ble team koordinert?
- Møter i prosjektet
- Ble det avholdt møter underveis som ikke var en del av de faste rammene? Behov for ekstra møter?
- Hvordan ble det arbeidet med arkitektur? (eksempler på viktige beslutninger?)
- Hvem initierte endringene i arbeidsprosess til <Fase2>?

- Hva skjedde i "teknisk review"?
- Hvordan var avhengighetene mellom teamene / hvilke avhengigheter var det mellom teamene?
- Hvordan ble avhengighetene håndtert? Hvilke arenaer ble brukt til koordinering?
- Hvordan foregikk kommunikasjonen mellom teamene? Var det noen endringer i denne fasen?
- Hvis det var endringer i teamstørrelse: hva var årsak til endringer, og påvirket det arbeidsfordeling mellom teamene eller kommunikasjonen mellom teamene?

<Fase2>:

- Hva var viktigste leveranser i <Fase2>?
- "Bi-modal" utvikling - hva fungerte bra og hva kunne fungert bedre?
- Hvordan ble team koordinert?
- Hvordan ble det arbeidet med arkitektur? (eksempler på viktige beslutninger?)
- Hvordan ble det arbeidet med løsningsbeskrivelse?
- Håndtering av mange team i koderepository
- Arbeid i gruppe som foreslå endret modell
- Endring i rapportering til kunde?
- Hvem initierte endringene i arbeidsprosess til <Fase3>?
- Informasjonsflyt: Tatt opp som utfordring i retrospektiv
- Hvordan var avhengighetene mellom teamene / hvilke avhengigheter var det mellom teamene?
- Hvordan ble avhengighetene håndtert? Hvilke arenaer ble brukt til koordinering?
- Hvordan foregikk kommunikasjonen mellom teamene? Var det noen endringer i denne fasen?
- Hvis det var endringer i teamstørrelse: hva var årsak til endringer, og påvirket det arbeidsfordeling mellom teamene eller kommunikasjonen mellom teamene?

<Fase3>:

- Hva var viktigste leveranser i <Fase3>?
- Hvordan ble team koordinert?
- Hvordan ble det arbeidet med arkitektur? (eksempler på viktige beslutninger?)
- Hvordan ble det arbeidet med løsningsbeskrivelse?
- Autonome team-modell: Hva ble konkrete endringer (roller, arbeidsprosess)
- Hvordan ble endringen oppfattet av teamene
- Hvordan var avhengighetene mellom teamene / hvilke avhengigheter var det mellom teamene?
- Hvordan ble avhengighetene håndtert? Hvilke arenaer ble brukt til koordinering?
- Hvordan foregikk kommunikasjonen mellom teamene? Var det noen endringer i denne fasen?
- Hvis det var endringer i teamstørrelse: hva var årsak til endringer, og påvirket det arbeidsfordeling mellom teamene eller kommunikasjonen mellom teamene?

Leveransemodell sammenlignet med Perform

Ville det vært lurt å bruke leveransemodell som i <Fase3> fra start?

Avslutning:

- Viktige faktorer for at prosjektet ble vellykket?
- Noe som kunne vært løst bedre?
- Er det noe vi ikke har diskutert som du synes er viktig