En smidigere arkitekturprosess

- fra «hviskeleken» til felles arkitekturforståelse?

Ida Kristine Dørum

Masteroppgave
IT og ledelse
30 studiepoeng

Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

UNIVERSITETET I OSLO

Januar 2017

© Ida Kristine Dørum

2017

En smidigere arkitekturprosess

Ida Kristine Dørum

http://www.duo.uio.no/

Trykk: Reprosentralen, Universitetet i Oslo

Sammendrag

IT-arkitekturfaget har tradisjonelt hatt fokus på grundige analyser, basert på målbilder, for å sikre at IT-systemer implementeres på en måte som understøtter langsiktige behov på en kostnadseffektiv måte. Når rammebetingelsene endrer seg, og systemeiere ønsker seg raskere leveranser av endringer, kan denne tilnærmingen komme til kort.

I oppgaven undersøkes hvordan arkitekturprosessen i NAVs IT-avdeling kan balansere behovene for kontroll på over helheten i en omfattende systemportefølje, og en smidigere systemutviklingsprosess. Normative og empiriske kilder beskriver mulige løsninger på mange utfordringer, men tilbyr ikke noe helhetlig rammeverk som dekker behovene for smidige arkitekturprosesser i store organisasjoner.

Teorien som er undersøkt i denne oppgaven er oppsummert i et rammeverk som tar utgangspunkt i modeller som beskriver arkitekturprosessen (Tang, Avgeriou, Jansen, Capilla, & Ali Babar, 2010) og arkitekturaktiviteter (Li, Liang, & Avgeriou, 2013). Rammeverket skiller arkitekturprosessen og -aktivitetene fra konteksten de utføres i. (Kruchten, 2013). Rammeverket utgjør et *forventet mønster*, som sammenlignes med det *empiriske mønsteret* som dannes av funnene (Yin, 2013).

Funnene viser at arkitekturarbeidet i NAV ikke oppleves som smidig. Hindringene for en smidig tilnærming er dels knyttet til konteksten arkitekturprosessen utføres i, dels til praksis blant arkitektene og i utførelsen av selve arkitekturarbeidet. Behovet for å håndtere risiko i en omfattende systemportefølje har ført til rigide kontrollfunksjoner og praksis. Det kan også være utfordrende å få finansiert tiltak som er nødvendige for å sikre at systemene er smidige. Interessenter i prosessen kommuniserer lite direkte og muntlig, noe som gjør det vanskelig å etablere en felles forståelse av arkitekturen..

Utfordringene som beskrives av informantene i NAV er i stor grad sammenfallende med utfordringer som beskrives i litteraturen. Basert på dette foreslås tiltak for å forbedre arkitekturprosessen, gjennom å tilpasse innsats og omfang av arkitekturarbeidet etter risiko-nivå, etablere mer direkte, muntlig kommunikasjon mellom sentrale interessenter, og delegering av myndighet fra arkitektene til utviklingsmiljøene. I tillegg til med forbedringer av selve prosessen, bør arkitekturmiljøet sikre forståelse i organisasjonen for nødvendigheten av tiltak som sikrer systemenes smidighet.

Innhold

Sammendrag 3
Innhold 5
Forord 9
1 Innledning 11
1.1 Problemformulering 12
1.2 Sentrale begreper 13
1.3 Avgrensinger 14
1.4 Leseveiledning 14
2 Teori 15
2.1 Normative kilder 15
2.2 Empirisk forskning 22
2.3 Oppsummering av innsikter fra forskningen 33
3 Metode 35
3.1 Forskningsdesign 35
3.2 Teori 35
3.3 Datainnsamling 36
3.4 Data-analyse 41
3.5 Validitet og reliabilitet 42
3.6 Rapport 42
3.7 Etiske problemstillinger 43
4 Presentasjon av case 46
4.1 Systemlandskapet 46
4.1 Organisering 48
4.2 Arkitekturroller 49
4.3 Arkitekturutvikling 50
4.4 Arkitekturstyring 50
4.5 Prosjekter og leveranser 51
4.6 Utfordringer 53
4.7 Forsøk med mer smidig tilnærming 54
4.8 Nye målsetninger for IT-avdelingen 54

5 Funn 55
5.1 Kontekst for prosessen 55
5.2 Arkitekturprosessen 59
5.3 Arkitekturbeslutninger 63
5.4 Arkitekturbeskrivelse 64
5.5 Arkitekturforståelse 66
5.6 Andre arkitekturaktiviteter 67
5.7 Oppsummering av funn 69
6 Diskusjon 71
6.1 Kontekst for prosessen 72
6.2 Arkitekturprosessen 76
6.3 Arkitekturbeslutninger 78
6.4 Arkitekturbeskrivelse 79
6.5 Arkitekturforståelse 81
6.6 Andre arkitekturaktiviteter 82
6.7 Mulige forklarende mekanismer 84
6.8 Oppsummering og anbefalte tiltak 85
7 Konklusjon 87
7.1 Hva sier forskningen om smidige arkitekturprosesser i store organisasjoner? 87
7.2 Hindringer og muligheter i IT-avdelingen 88
7.3 Forslag til videre undersøkelser 90
8 Litteratur 92
8.1 Interne dokumenter fra NAV 95
9 Vedlegg 97
Vedlegg 1: Intervjuguide 99
Vedlegg 2: Innsikter og svar fra informanter 101
Vedlegg 3: Presentasjon fra gjennomgang av funn for informanter 151
Vedlegg 4: Referat fra gjennomgang av funn 157
Vedlegg 5: Oversikt over hindringer og muligheter 163
Vedlegg 6: Sammenligning av forventet og empirisk mønster 167
Vedlegg 7: Informasjonsskriv til informanter 173
Vedlegg 8: Godkjenning fra NSD 175

Tabeller

Tabell 1 - Faser i arkitekturprosessen 16
Tabell 2 - Arkitekturaktiviteter (Li et al, 2013) 16
Tabell 3 - Det smidige manifest (Beck et al, 2001) 18
Tabell 4 - Innsikter fra forskningen 34
Tabell 5 - Informanter 38
Tabell 6 - Steg i data-analysen 41
Tabell 7 - Beskrivelse av delprosessene i SLEM 52
Tabell 8 - Grunnpilarer fra nye NAV IT (hentet fra NAVs intranett) 54
Tabell 9 - Oppsummering av funn 70
Tabell 10 - Forenklet oppsummering av forventet og empirisk mønster 72
Tabell 11 - Anbefalte tiltak for forbedringer av arkitekturprosessen. 86
Tabell 12 - Mulige tema for videre undersøkelser 91

Figurer

Figur 1 - En generisk arkitekturprosess (Tang et al., 2010) 15
Figur 2 - TOGAF ADM (The Open Group, 2009) 17
Figur 3 - Rammeverk for oppsummering av teori 33
Figur 4 - Kart over sentrale applikasjoner i NAV (fra intern wiki) 47
Figur 5 - Forenklet skisse av avhengigheter fra et sentralt fagsystem (fra intern wiki) 48
Figur 6 - Organisasjonskart for IT-avdelingen i 2016 49
Figur 7 - Førende prinsipper for SLEM 51
Figur 8 - Delprosesser i SLEM – når deltar forskjellige roller (forenklet) 52
Figur 9 - Faktorer i arkitekturprosessens kontekst 55
Figur 10 - Forklarende mekanismer 85

Forord

Jeg vil gjerne takke veileder Bendik Bygstad for god og engasjert hjelp med arbeidet med oppgaven, og for å gi meg ideen som førte til valg av tema og problemstilling for denne oppgaven gjennom forskningsagendaen som er beskrevet i *"Arkitektur handler om praktisk arbeid i organisasjonen, ikke en tegning"* (Bygstad & Pedersen, 2012).

Takk også til sjefsarkitekt Petter Hafskjold i NAV for fri tilgang til nødvendig informasjon og informanter i IT-avdelingen.

Etter at jeg valgte temaet for denne masteroppgaven har det skjedd mye i IT-avdelingen i NAV. Målsetninger, organisasjonskart og arbeidsprosesser er under endring. Det har vært interessant, men også utfordrende, å gjennomføre undersøkelser og analyse samtidig som rammebetingelsene for arkitekturprosessen har vært i bevegelse. Forhåpentligvis kan denne oppgaven også være et praktisk bidrag i retning av en smidigere arkitekturprosess.

Sist men ikke minst, takk til engasjerte informanter som har bidratt med sine erfaringer, og til engasjerte kollegaer som har gitt meg lesetips og andre innspill underveis.

Oslo 31. januar 2017

Ida Kristine Dørum

1 Innledning

NAV har en omfattende IT-portefølje, som understøtter saksbehandling og utbetaling av ytelser til store deler av Norges befolkning. Stabil drift er svært viktig, og endringer i IT-løsningene er underlagt streng styring. Prosessene for kvalitetssikring og planlegging skal sikre kvalitet, men gjør det krevende å realisere nye ideer og behov i IT-løsningene.

De siste årene behovet for endringer og nyutvikling økt, og leveransene må gjennomføres både i større omfang og hurtigere enn tidligere. Dette gjør at IT-avdelingen ser etter nye arbeidsmetoder, som kan gjør organisasjonen bedre i stand til å levere endringer hurtig. Gjennom 2016 er det gjennomført flere mindre endringer av rutiner, og fra 1.1.2017 omorganiseres hele avdelingen. Målet er at IT-avdelingen skal bli i stand til å levere IT-tjenester raskere og rimeligere. Dette gjør at også IT-arkitektene må vurdere hvordan arbeidsprosessene kan forbedres, slik at de bedre understøtter nye mål og prioriteringer.

IT-arkitektur-disiplinen har tradisjonelt hatt fokus på at grundig analyse basert på langsiktige målbilder skal være gjennomført i forkant av utviklingsprosjekter. Denne tilnærmingen dekker ikke nødvendigvis behovene når organisasjonens målsetninger og behov er skiftende (Bygstad & Pedersen, 2012).

For lite arkitekturarbeid og analyse i forkant av prosjekter kan føre til tidkrevende og kostbare endringer underveis i prosjektet, mens for omfattende analyseprosesser kan føre til forsinkelser og unødvendige kostnader (Boehm, 2011). Det er altså viktig å prioritere hvordan arkitektene skal bruke sin arbeidsinnsats, slik at den bidrar til å redusere risiko, og tilfører prosjektene verdi.

Smidige metoder legger til rette for hyppige leveranser av programvare. Disse metodene har imidlertid ofte lite fokus på arkitektur (Waterman, Noble, & Allan, 2015). Flere av opphavsmennene bak *The Agile Manifesto* har imidlertid pekt på behovet for arkitektur i smidige metoder (Abrahamsson, Babar, & Kruchten, 2010). En tilnærming der alle beslutninger tas underveis i prosjektet kan resultere i en arkitektur med manglende endringsevne, som vanskeliggjør senere endringsbehov (Brown et al., 2010). En smidig arkitekturprosess, må ha smidige systemer som et sentralt mål, ellers vil systemet kunne bli en hindring for fremtidige endringer. (Bloomberg, 2013)

Yang, Liang og Avgeriou (2016) oppsummerer forskningsfunn knyttet til forsøk på å kombinere smidige teknikker og arkitektur, og peker på en spenning mellom de iboende egenskapene i tradisjonelt arkitekturarbeid og smidig tilnærming, blant annet:

- Hvordan balansere behovet for planlegging og risikokontroll opp mot det smidige prinsippet om å «omfavne endring»?
- Hvordan kan man balansere behov for sentral styring og oversikt opp mot behovet for å distribuere beslutningsmyndighet og ansvar til selvstyrte team?
- Hvordan utformes arkitekturbeskrivelser, og hvor mye innsats skal legges i dette?

(Yang, Liang, & Avgeriou, 2016, p. 173)

1.1 Problemformulering

En arkitekturprosess tilpasset organisasjonens nye behov må balansere mellom å understøtte hurtige leveranser, og å sikre at løsningene ikke bygges på en måte som reduserer evnen til videre endringer. Med utgangspunkt i dette formuleres et teoretisk og et praktisk forskningsspørsmål:

1. *Kan forskningen peke på smidige arkitekturprosesser og -teknikker som understøtter hurtige leveranser av IT-løsninger i store organisasjoner?*
2. *Hvilke faktorer i IT-avdelingen er til hinder eller gir muligheter for en mer smidig arkitekturprosess?*

1.2 Sentrale begreper

Arkitektur defineres av ISO som *«fundamentale konsepter og egenskaper for et system i sitt miljø, konkretisert gjennom systemets bestanddeler, hvordan disse relaterer seg til hverandre, prinsipper for hvordan systemet designes og utvikles»* (ISO, 2011). Bloomberg (2013) har utvidet denne definisjonen, til å omfatte system**er** i flertall. Arkitekturen beskriver altså relasjonene mellom systemer, ikke bare den interne oppbygningen av det enkelte system.

Virksomhetsarkitektur *«dreier seg om hvordan en virksomhet er organisert, hvordan arbeidsprosesser er satt sammen og hvordan IT-løsninger utnyttes. En virksomhetsarkitektur består av prinsipper, metoder og modeller som til sammen beskriver dette i en helhet. […] Formålet er å sikre god sammenheng mellom arbeidsprosesser og IT-løsninger, og å unngå at det etableres informasjonssystemer som ikke snakker sammen, eller såkalte siloer.»* (DIFI, 2014)

IT-arkitektur *«utgjør en delmengde av virksomhetens helhetlige arkitektur og handler om hvordan IT-systemer er bygget opp ved hjelp av komponenter og relasjonene mellom disse.»* (DIFI, 2014)

Med **arkitekturprosess** menes *architecting*, som defineres av ISO som *«Prosessen med å unnfange, definere, uttrykke, dokumentere og kommunisere en arkitektur, å sikre dens implementasjon og forbedre den gjennom systemets livssyklus»* (ISO, 2011). Arkitekturprosessen er altså en del av hele systemsyklusen, ikke bare design-fasen. Hofmeister et al. (2007), Tang et al. (2010) og Li et al. (2013) konkretiserer begreper knyttet til arkitekturprosessen og tilhørende teknikker. Dette er nærmere beskrevet i kapittel 2.1.1.

Smidig er den vanlige norske oversettelsen av begrepet *agile*, som ble kjent gjennom *The Agile Manifesto*, eller *Det smidige manifest* (Beck et al., 2001). Manifestet ble utarbeidet av opphavsmenn og talsmenn for flere forskjellige eksisterende metodeverk som stilte spørsmål ved nytten av mange vanlige praksiser innen systemutvikling. Smidige metoder vektlegger samhandling mellom personer, kjørende programvare, samarbeid mellom kunde og leverandører og evne til å reagere fortløpende på endringer. Smidig-manifestet og videre beskrivelse av smidige metoder finnes i kapittel 2.1.2.

1.3 Avgrensinger

Temaet som undersøkes er i utgangspunktet omfattende. I litteraturen brukes *agile architecture*-begrepet både om prosesser for å utforme arkitektur (arkitekturprosess), og om hvordan systemer skal utformes for å understøtte smidig håndtering av endringer. Disse to perspektivene er til dels overlappende, gjennom at prosessene kan påvirke grad av smidighet man oppnår i systemene, samtidig som systemene kan utgjøre et hinder for en smidig prosess.

Problemstillingen er avgrenset til å undersøke arkitekturprosessen, og se på systemenes utforming som kontekst. Hva som er god arkitektur, eller hvilke typer design som gir smidighet i systemene behandles derfor bare overfladisk i denne oppgaven.

Begrepet *arkitektur* brukes i denne oppgaven om *IT-arkitektur*. Andre deler av virksomhetsarkitekturen behandles ikke.

Undersøkelsen gjennomføres i NAVs IT-avdeling, og situasjonen det tas utgangspunkt i er nærmere beskrevet i kapittel 4. Informantene har pekt på flere forhold utenfor IT-avdelingen, og til dels utenfor NAV, som påvirker arkitekturprosessen, men problemstillingen er avgrenset til forhold internt i IT-avdelingen, og spesifikt til prosesser som er knyttet til utvikling av programvare.. Teknologinære tema som infrastruktur, drift og teknologivalg behandles ikke i detalj.

1.4 Leseveiledning

Kapittel 2 inneholder en gjennomgang av relevant teori om området «smidig arkitektur», inndelt i en del for normative kilder, og en del for bidrag fra empirisk forskning. Innsikter fra teorien oppsummeres til slutt i kapittelet, i et rammeverk som benyttes for strukturering av funn og analyse videre i oppgaven.

Kapittel 3 beskriver metoden for undersøkelse og analyse.

Kapittel 4 beskriver caset som undersøkes, arkitekturprosessen i IT-avdelingen i Arbeids- og velferdsdirektoratet.

Kapittel 5 oppsummerer funn fra intervjuer og dokumentundersøkelser i IT-avdelingen.

Funnene diskuteres opp mot teorien i kapittel 6, som også gir anbefalinger om mulige tiltak for å oppnå en mer smidig arkitekturprosess. Oppgaven oppsummeres og konkluderes i kapittel 7.

2 Teori

I dette kapittelet oppsummeres teori knyttet til arkitekturprosesser og smidig metodikk, og hvordan arkitekturarbeid og smidig tilnærming kan kombineres. Teorigjennomgangen er delt opp i normativ og empiriske kilder. Til slutt i kapittelet oppsummeres teorien i et rammeverk, som benyttes som utgangspunkt for videre strukturering og analyse av funn.

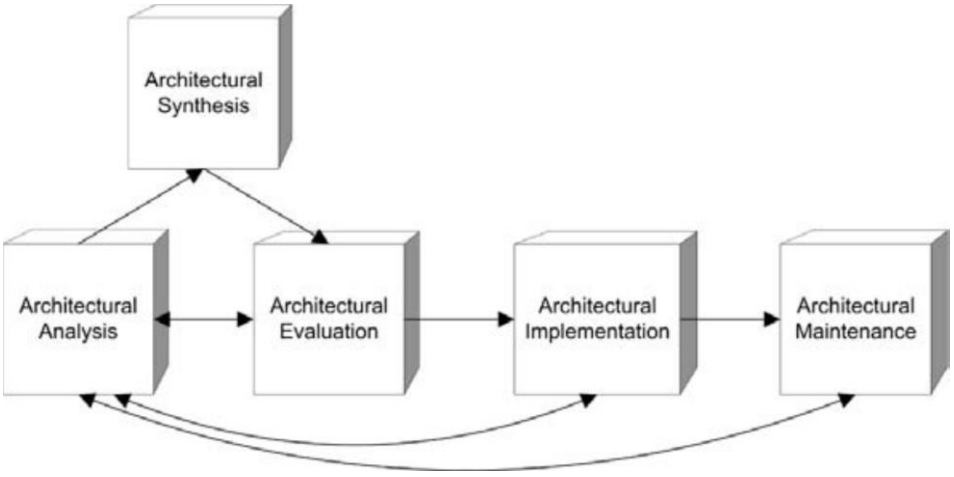
2.1 Normative kilder

De normative kildene beskriver både referansemodeller for arkitekturprosessen, prinsipper og teknikker fra smidige metoder, og bruk av smidige prinsipper i arkitekturarbeid.

2.1.1 Arkitekturprosessen

Hofmeister et al. (2007) beskriver en generell modell for design av programvarearkitektur, basert på sammenstilling av flere etablerte modeller fra IT-industrien. Modellen beskriver en prosess der man først formulerer problemstilling, og identifiserer relevante krav (*Analyse*). Deretter utarbeides en eller flere mulige løsninger på problemstillingen (*Syntese*). I *Evaluere*-steget tar man stilling til om hvilke foreslåtte løsninger som best møter kravene.

Modellen har senere har senere blitt videreutviklet og utvidet av andre forskere med ytterligere to faser, for å ta høyde for utviklingsprosessen og systemets livssyklus, *Implementasjon* og *Vedlikehold* (Tang et al., 2010). Den samlede prosessdefinisjonen er beskrevet i Figur 1 og Tabell 1.



Figur 1 - En generisk arkitekturprosess (Tang et al., 2010)

Tabell 1 - Faser i arkitekturprosessen

Fase	**Beskrivelse**
Analyse	Definere problemstilling og identifisere krav som har en vesentlig påvirkning på arkitekturen (Hofmester et al, 2007)
Syntese	Design av en eller flere mulige løsninger («kandidat-arkitekturer») for å ivareta et krav eller et sett med krav som er identifisert i analysen. (Hofmester et al, 2007)
Evaluering	Vurdere om designet understøtter kravene, sikre at de riktige arkitekturbeslutningene blir tatt (Hofmester et al, 2007)
Implementasjon	Realisering av arkitekturen i form av et design. (Tang et al. 2010)
Vedlikehold	Vedlikehold vil si endring av arkitekturen for å håndtere feil (Tang et al. 2010), eller for å håndtere nye krav. (Yang et al. 2016)

Li et al. (2013) har tatt utgangspunkt i denne prosessen, og definert seks arkitekturaktiviteter som kan utføres uavhengige av prosessfaser (Tabell 2).

Tabell 2 - Arkitekturaktiviteter (Li et al, 2013)

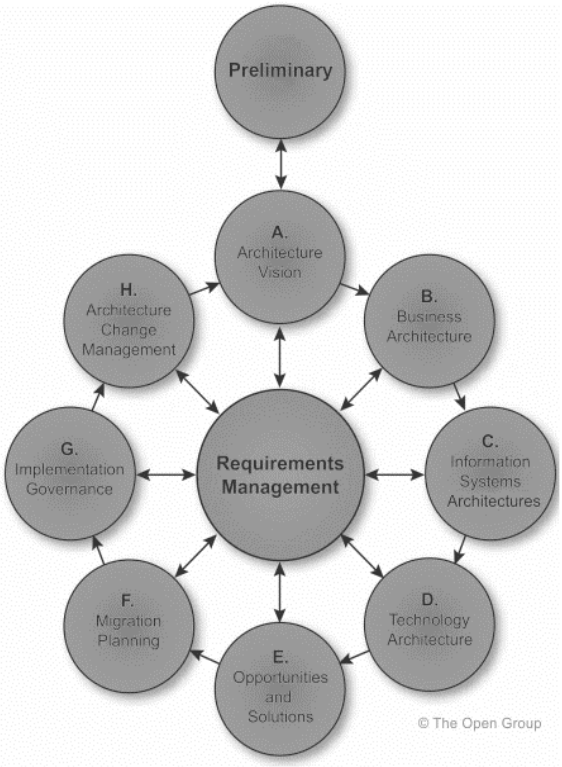
Aktivitet	**Beskrivelse**
Arkitekturbeskrivelse	Å beskrive arkitekturen ved hjelp av et sett med arkitekturelementer som bidrar til at interessenter kan forstå systemet, og understøtter kommunikasjon og samarbeid.
Arkitekturforståelse	Å oppnå forståelse av arkitekturelementene (f.eks. arkitekturbeslutninger) og deres relasjoner i et arkitekturdesign.
Analyse av påvirkning	Å analysere hvilke arkitekturelementer som påvirkes i et endringsscenario
Arkitekturgjenbruk	Å gjenbruke eksisterende arkitekturdesignelementer (beslutninger, rammeverk, patterns mm) i arkitekturen til et nytt system.
Arkitekturgjenvinning	Å ekstrahere den nåværende arkitekturen fra en systemimplementasjon.
Arkitektur-refaktoring	Å endre arkitekturstrukturen uten å endre ekstern oppførsel.

Disse referansemodellene og begrepene er brukt som rammeverk av flere forskere, blant annet studien til Yang et al. (2016), som beskrives nærmere i kapittel 2.2.3.

En annen referansemodell for arkitekturprosessen er Architecture Development Method (ADM), som er en del av virksomhetsarkitekturrammeverket TOGAF. TOGAF har fått stor utbredelse i IT-bransjen de siste årene.

ADM beskriver en syklisk prosess der man utvikler arkitektur for utvalgte områder (domener) gjennom først å etablere en visjon, så analysere henholdsvis nåsituasjon og målbilder for

forretningsarkitektur, systemarkitektur og teknologiarkitektur. Basert på målbildene som etableres analyserer man mulige løsninger, før man legger en plan for hvordan arkitekturen implementeres gjennom et eller flere utviklingsinitiativ. (The Open Group, 2009)



Figur 2 - TOGAF ADM (The Open Group, 2009)

Bloomberg (2013) kritiserer ADM og tilsvarende rammeverk for virksomhetsarkitektur for å være for omstendelige. Han mener det er umulig å beskrive alle endringer som må utføres for å nå et mål på forhånd, og at målet uansett vil endre seg før man er ferdig med alle de planlagte endringene. Istedenfor å se på virksomhetsarkitekturen som en «matoppskrift», kan man se på den som en eikenøtt som inneholder DNAet som beskriver hvordan et tre vokser fram. Virksomhetsarkitekturen betraktes som et dynamisk styringsrammeverk, som består av krav og prinsipper som kontinuerlig røktes. Prinsipper som bidrar til å oppnå forretningsmålene tas vare på, mens prinsipper som viser seg å være til skade må forkastes. (Bloomberg, 2013, pp. 248–250)

Waterman (2015) definerer arkitektur som «et sett med vesentlige beslutninger om systemets oppførsel og høynivå struktur, der «vesentlig» måles ut fra hva kostnaden ved endring vil

være. I arkitekturarbeidet bør man prioritere områder der endringer senere i prosessen vil bli kostbare. Beslutninger som er enkle å endre senere, kan utsettes.

Poort (2012) tar også utgangspunkt i at arkitektur består av en samling designvalg, og beskriver en tilnærming til arkitekturarbeid som tar utgangspunkt i å styre risiko og kostnad. Metoden tar utgangspunkt i at før en beslutning er tatt vil risiko for at man tar feil beslutning synke over tid, etter hvert som mer informasjon er tilgengelig og beslutningsgrunnlaget blir rikere. Prosjektene bør lage en liste over arkitekturspørsmål, som sorteres etter risiko og kost. Listen må bearbeides aktivt underveis i prosjektet, siden risikoprofilen kan endres over tid. Dersom risiko og kostnad knyttet til problemstillingen er lav, bør man ikke bruke tid på arkitekturarbeid. (Poort, 2012, pp. 2000–2001)

2.1.2 Smidige metoder og teknikker

Det smidige manifest (Beck et al., 2001) og smidige metoder kan sies å være en reaksjon på planbaserte, tradisjonelle metoder med røtter i ingeniørfaget (Dybå & Dingsøyr, 2009). Den mest kjente smidige metodikken er Scrum, som legger vekt på hurtige tilbakemeldinger («feedback loops»), selvorganiserende team og inkrementell utvikling i form av korte «sprinter» (Dybå & Dingsøyr, 2008).

Tabell 3 - Det smidige manifest (Beck et al, 2001)

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

Dette vil si: Selv om punktene som står til høyre har verdi,
så verdsetter vi punktene til venstre enda høyere.

Noen vanlige smidige praksiser er å beskrive krav og behov i form av «**brukerhistorier**», som legges i en «**backlog**» eller «kø», slik at man kan gjøre **løpende prioritering** hvilke behov som skal løses i hver utviklingsiterasjon. Teamene skal være **selvorganiserende**, og samles daglig i korte stand-up-møter for å dele informasjon og løse hindringer. **Kunden skal være til stede** der utviklingen skjer, og helst være en del av teamet. Erfaringer fra hver

iterasjon oppsummeres i **retrospektiv**-møter, og brukes til forbedringer av metoden. Programmererne skal ha et **kollektivt eierskap til koden**, og ta felles ansvar for at den holder god kvalitet. **Enkelt design** vektlegges. (Yang et al., 2016)

Kruchten (2013) påpeker at for at smidige praksiser skal lykkes, må konteksten være riktig. Smidige metoder har hatt stor suksess i mange prosjekter, men er ikke egnet i alle sammenhenger. For å forstå hvilken praksis som er egnet for formålet, er det viktig å forstå konteksten prosessen skal implementeres i. Organisasjonen utgjør et miljø, som driver eller begrenser prosjektenes kontekst-attributter, som igjen påvirker valg og innføring av praksiser i den faktiske prosessen. (Kruchten, 2013, p. 353)

Eksempler på faktorer på organisasjonsnivå som påvirker prosessens evne til smidighet er antall instanser av systemet, organisasjonens modenhet innen programvareutvikling, innovasjonsgrad og organisasjonskultur. Slike faktorer påvirker igjen prosjektenes åtte kontekst-attributter: Størrelse, arkitekturens stabilitet, forretningsmodell, team-distribusjon, endringshyppighet, systemets alder, systemets kritikalitet og prosjektets styringsmodell. (Kruchten, 2013, pp. 352–354)

Bloomberg (2013) peker på at et dogmatisk syn på smidig metodikk er en selvmotsigelse. Dersom det viser seg å være vanskelig å følge den valgte metoden, bør ikke betraktes som «feil», men som en mulighet. Utfordringen kan brukes for å undersøke om reglene kan endres så de bedre understøtter formålet metoden skal oppfylle, eller *meta-metoden*.

2.1.3 Kombinasjonen arkitektur og smidig

Smidige utviklere forsøker ofte å minimere omfanget av arkitekturplanlegging som utføres, for å legge til rette for å levere mest mulig verdi tidlig. For lite planlegging kan imidlertid resultere i *Accidental architecture*[1] som krever omfattende endringsarbeid og merkostnader. Arkitektur definerer vanligvis egenskaper på systemnivå, som kan være vanskelige å endre etter at utvikling starter. Smidige metodikker er basert på strategien om å håndtere endringer fortløpende underveis, og sier lite konkret om hvordan man skal jobbe med arkitektur. (Waterman et al., 2015)

[1] Begrepet ble først tatt i bruk av Grady Booch

Alistair Cockburn, en av smidig-manifestets opphavsmenn peker i likhet med Kruchten (2013) på at metoder må velges på grunnlag av prosjektets egenskaper. Han mener prosjekter med flere enn 14 deltakere kan ha behov for en dedikert arkitektur-rolle. (Nord & Tomayko, 2006, p. 48).

Blair (2010) og Faber (2010) beskriver arkitekten som en *tjenestetilbyder*, som må tilføre prosjektet eller utviklingsteamet nytte, ikke bare belaste utviklingsteamene med omfattende krav. Rollen som *tjenestetilbyder* er et balansepunkt mellom ytterpunktene der arkitekten er for detaljstyrende på den ene siden, og for passiv og ettergivende på den andre siden.

Madison (2010) skriver at arkitekten må ha tillit til at teamene som sitter nærmest implementasjonen har kunnskap og kompetanse nok til å ta nødvendige beslutninger om design. Før prosjektet starter bør arkitekten som hovedregel skal definere hva som er det akseptable handlingsrommet er, snarere enn hva som er den spesifikke løsningen. Det er viktig at arkitekten har gode kommunikasjonsevner, og kunnskap om kvalitetskriterier, *design patterns* og teknologien som brukes.

Buschmann & Henney (2013) argumenterer for at en smidig prosess som fører til løsninger som er vanskelige å endre, bryter med grunnleggende kriterier for smidighet, og henviser til at *Det smidige manifest* (Beck et al., 2001) understreker at *«kontinuerlig oppmerksomhet på teknisk fortreffelighet og godt design forsterker smidighet».* En smidig prosess som resulterer i programvare som ikke understøtter smidig utvikling er altså ikke en «ekte» smidig prosess. Bloomberg (2013) argumenterer også for at smidige systemer er et viktigere mål enn en smidig prosess, og definerer smidighet som et *meta-krav* som må være førende for alle andre krav til systemet. Faber (2010) mener arkitekten må ta en aktiv rolle som eier av de ikke-funksjonelle kravene til løsningen, for å balansere produkteieren, som har ansvar for de funksjonelle kravene.

Tjenesteorientering er et sentralt virkemiddel for å oppnå smidighet i løsninger. Dette forutsetter en tett dialog mellom forretningsside og IT. (Bloomberg, 2013, pp. 8–10). DevOps- tilnærmingen, der utviklere og driftspersonell jobber sammen for å automatisere konfigurasjon og oppsett som understøtter effektiv produksjonssetting, er også en sentral forutsetning. (Bloomberg, 2013, p. 225)

Leffingwell (2011) beskriver en arkitekturvisjon som understøtter forretningens visjon. Visjonen er mindre omfattende enn arkitekturmålbildene som beskrives i TOGAF ADM, men setter retning for arkitekturarbeidet. Ved å undersøke virksomhetens planer, og se dem i sammenheng med arkitekturvisjonen, kan man lage en *backlog* av arkitekturepos. Arkitekturepos beskriver infrastruktur og felleskomponenter virksomheten har behov for. Elementene implementeres trinnvis, i tide for prosjektene som har behov for dem, og utgjør en «arkitektur-rullebane» (architecture runway) de enkelte prosjektene kan «lette» fra.

Arkitektureposene dekker behov som går ut over hva det enkelte smidige utviklingsteamet kan håndtere på egen hånd, og som dermed ikke «vokser fram» av seg selv. Enten fordi utfordringene går på tvers av flere systemer, eller fordi de er for omfattende til å kunne håndteres av et enkelt team eller i løpet en sprint. Tilnærmingen forutsetter at det finnes styringsprosesser på plass, der man kan prioritere og beslutte IT-behov på tvers av prosjekter og avdelinger i organisasjonen. (Leffingwell, 2011)

Det kan også være behov for å endre større deler av systemet gjennom refaktorering, på grunn av endrede behov, teknologiendringer, endring i bruksmønstre eller fordi designet har forvitret på grunn av tidspress i utviklingsteamet. For visse områder kan det også være fordelaktig å etablere felleskomponenter eller gjenbruke komponenter, for å understøtte brukerbehov eller mer økonomisk utvikling. (Leffingwell, 2011)

Leffingwell fastholder at utviklingsteamene er den mest sentrale aktøren i å designe systemet, men at det er et samarbeid mellom flere roller. Arkitekturen skal være *«the simplest thing that can possibly work».* Kode og prototyping bør brukes for å utvikle designet, supplert med modeller der problemstillingene er for omfattende til at design gjennom koding er egnet.

(Selic, 2009) mener at detaljert designdokumentasjon som utarbeides før koding i mange tilfeller er bortkastet arbeid. Designet overlever sjelden første møte med virkeligheten, det vil si realiseringen i form av kode. I tillegg er designdokumentasjon på «kode-nivå» så godt som mulig å holde oppdatert. Dokumentasjonen bør holdes på et høyt abstraksjonsnivå, teknologiuavhengig, med en tydelig kobling til applikasjonens krav og konsepter, og må omfatte begrunnelser for designvalgene som er gjort.

«Det skal finnes så mye dokumentasjon det er behov for, ikke mer, og den skal alltid følges av direkte kommunikasjon», skriver Faber (2010) i sin oppsummering av erfaringer fra Siemens.

(Madison, 2010) argumenterer for at programvare og funksjonalitet som ikke er dokumentert ikke skal regnes som ferdig, men at en arkitekt også bør undersøke den faktiske koden etter hver sprint.

Erder og Pureur (2016) fremhever at arkitektens arbeidsprodukt er ikke modeller eller prototyper, men faglig funderte, dokumenterte beslutninger, tatt i samarbeid med sentrale interessenter. Arkitekten er en prosessdriver og fasilitator for disse beslutningene. Arkitekten må ta på seg ansvaret for det langsiktige perspektivet, og ivareta interessentenes behov, framfor å være styrt av kortsiktige tidsplaner og budsjett. Beskrivelsene av arkitekturen skal være kortfattet, og kontinuerlig ta inn over seg læring fra den realiserte arkitekturen, i form av kode som kjører på fysisk infrastruktur.

2.2 Empirisk forskning

Empirien som er undersøkt omfatter både arkitekturarbeid i større og mindre prosjekter eller produktutviklingsteam, men sier lite om hvordan man koordinerer arkitektur på tvers av flere prosjekter eller produkter.

2.2.1 Arkitekturprosessen

Jørgensen (2015) oppsummering relevant forskning om årsaker til store offentlige IT-prosjekter. Blant en rekke faktorer for hvorfor prosjektene mislykkes, basert på undersøkelser i USA, England og Australia, finner vi tre som er relaterte til arkitekturprosessene eller kvaliteten på arkitekturarbeidet i prosjektet:

- Lav forståelse av kompleksiteten i systemet som skulle lages
- Manglende fokus på integrasjon og god system/portefølje-arkitektur
- Liten evne til å beskrive og evaluere ikke-funksjonelle krav

Faktorene omtales som trolig relevante også for norske forhold. (Jørgensen, 2015)

Arkitekturarbeid er altså en viktig faktor for at prosjekter lykkes. Et viktig spørsmål er hvor omfattende dette arbeidet skal være for å ivareta risikohåndteringen i prosjektet, men samtidig ikke bli så omfattende at det blir en forsinkende faktor.

Boehm (2011) oppsummerer forskning på data om prosjekter samlet gjennom 40 år, og konkluderer med at omfanget av nødvendig arkitekturarbeid er avhengig av størrelsen på

systemet som skal utvikles. Jo større, mer kritisk og mer stabilt i målsetning prosjektet er, dess sterkere er behovet for å sikre at arkitekturen er realiserbar før prosjektet settes i gang. For større utviklingsprosjekter kan riktig mengde arkitekturarbeid tilsvare så mye som en tredjedel av utviklingsinnsatsen. Mindre prosjekter kan redusere dette behovet gjennom bruk av smidige teknikker. (Boehm, 2011, p. 182).

Andre årsaker til at offentlige IT-prosjekter feiler er for stort omfang og for høye ambisjoner. Forskning viser på at oppdeling av store oppgaver i kortere prosjekter, og å dele opp prosjekter i flere, hyppige leveranser, er et viktig tiltak for å redusere risiko (Jørgensen, 2015, p. 7). En slik tilnærming vil også kunne redusere omfanget av arkitekturarbeidet som må utføres før oppstart.

2.2.2 Smidige prosesser og teknikker

I en omfattende litteratur-review fra 2008 viste de fleste undersøkte studiene at smidige utviklingspraksiser er enkle å innføre, og fungerer godt. Flere komparative studier av smidig og tradisjonell metodikk mener å påvise gevinster gjennom at endringer håndteres mer effektivt, og at det er lettere å vise til forretningsverdi for det som utvikles. (Dybå & Dingsøyr, 2008, p. 18)

Studiene viser at menneskelige og sosiale faktorer er viktige for å lykkes med smidig. Team som lykkes mestrer en balanse mellom individuell autonomi og team-autonomi, og føler også et ansvar for resten av virksomheten. (Dybå & Dingsøyr, 2008, p. 20)

Før prosjekter tar i bruk smidige metoder bør de imidlertid vurdere prosjektets egenskaper opp mot egenskapene som understøttes av metoden. Det er ikke gitt at smidige metoder egner seg for store prosjekter. (Dybå & Dingsøyr, 2008, p. 20). For mindre prosjekter med skiftende eller ukjente funksjonelle krav, kan en mer smidig tilnærming der arkitekturen utvikles underveis være mer effektiv enn omfattende arbeid før oppstart. (Boehm, 2011, pp. 179–181). Dersom prosjektenes omfang reduseres, i henhold til anbefalingene fra Jørgensen (2015), vil det også kunne legge til rette for en mer smidig tilnærming.

2.2.3 Kombinasjonen arkitektur og smidig

Både forskningsstudier og faglitteratur trekker fram manglende fokus på design- og arkitekturspørsmål som en utfordring i smidig utvikling (Dybå & Dingsøyr, 2008). En litteratur-review fra 2010 om sammenhengen mellom smidig og arkitektur peker på

manglende empiriske bevis for vanlige påstander knyttet til smidig og arkitektur. Studien fant empiriske bevis for at enkelte smidige praksiser kan bidra til forbedret arkitektur, men bevisene ble vurdert som svake. (Breivold, Sundmark, Wallin, & Larsson, 2010).

En studie fra 2016 som går gjennom av forskning om arkitektur og smidighet, viser at det interessen for temaet siden har vært økende, særlig de siste tre årene. Denne studien fant heller ikke sterke empiriske bevis på påstander rundt smidig og arkitektur, men strukturerer og oppsummerer funn fra studier av temaet. Det er verdt å merke seg at de ikke fant tilstrekkelig med data om feilslåtte forsøk på å kombinere smidig tilnærming og arkitekturarbeid, siden de fleste studiene anså forsøkene som vellykket. (Yang et al., 2016, p. 180). Hoveddelen av studiene som er undersøkt var industri-baserte (ca 80%), de øvrige akademiske studier. (Yang et al., 2016, p. 165). Studien oppsummerer erfaringer basert på forskningen. Det tas imidlertid forbehold om at disse ikke nødvendigvis er generaliserbare, og de omtales derfor som «lessons learned», ikke «best practices». (Yang et al., 2016, p. 178).

Kjente utfordringer ved kombinasjonen arkitektur og smidig

Prosjektets størrelse påvirker evnen til smidighet, også i arkitekturarbeidet, siden konsekvensene av arkitekturendringer øker som følge av prosjektet størrelse. I prosjekter med høy kritikalitet, for eksempel systemer der feil kan føre til tap av liv, vil man også han behov for stor grad av sikkerhet om løsning før oppstart. (Yang et al., 2016, p. 174).

Organisasjonens kultur og modenhet som helhet er også en faktor. Det er svært utfordrende å drive med smidig arkitektur i en organisasjon som er uten erfaring med smidig utvikling, eller har umodne utviklingsprosesser. Den tekniske infrastrukturen for smidig utvikling, som automatisering av test og andre utviklingsprosesser er også en viktig forutsetning. (Yang et al., 2016, p. 174)

En del av utfordringene er knyttet til modenhet og forståelse blant deltakere i teamene og andre interessenter. For eksempel opplever mange at arkitektens rolle ikke er tydelig definert i en smidig kontekst. Nesten en tredjedel av studiene som er undersøkt peker på utfordringen med at «smidig arkitektur»-tilnærmingen mangler en aktiv «forkjemper» som bidrar til å innføre de nye teknikkene. (Yang et al., 2016, p. 173)

Andre utfordringer er knyttet til mangel på etablerte teknikker for å utføre arkitekturarbeid, for eksempel hvordan man utfører evalueringer i en smidig kontekst. Etablerte teknikker for

arkitekturarbeid oppleves som tungrodde og unødig omfattende av mange smidig-praktikere. (Yang et al., 2016, p. 173)

Det er også viktig at man ikke overdriver omfanget av arkitekturegenskaper som utvikles, for eksempel kan overdrevent fokus på fleksibilitet i en løsning føre til at man utfører overflødig arbeid, og at løsningen blir mer omfattende enn nødvendig. Det kan være utfordrende å utvikle arkitekturen inkrementelt, men å refaktorere kode for å endre arkitektur i etterkant kan også være svært tidkrevende og kostbart. (Yang et al., 2016, p. 173)

Hvor mye arkitektur?

For lite planlegging i forkant kan føre til omfattende endringsbehov, og at prosjektet feiler. For mye planlegging og arkitekturarbeid kan på den andre siden føre til store forsinkelser, og hindre at prosjektet leverer nødvendig funksjonalitet. (Waterman et al., 2015)

Waterman et al. (2015) har basert seg på *grounded theory* for å etablere en modell med seks drivere og fem strategier. Driverne gir muligheter eller begrensninger for hvilke strategier som kan brukes. Modellen kan brukes for å vurdere hvor mye arkitekturarbeid som bør gjøres på forhånd.

Strategien ***Big design up front*** (BDUP) innebærer at de fleste arkitekturvalg gjøres før utviklingen starter. Dette er en tradisjonell tilnærming, kjent fra såkalt fossefallsmetodikk (Waterman et al., 2015). Jo større, mer kritisk og mer stabilt i målsetning prosjektet er, dess sterkere er behovet for å bevise at arkitekturen er gjennomførbar på forhånd, slik at man kan unngå risiko og behov for kostbare endringer underveis. (Boehm, 2011). På den annen side gir «Big design up front» lite rom for nødvendige endringer underveis i prosjektet (Waterman et al., 2015).

Emergent arkitektur (framvoksende arkitektur) som strategi innebærer at man ikke utarbeider arkitektur på forhånd, men lar den vokse fram som en del av systemutviklingen. Behov for endringer løses ved å refaktorere kode. Dette kan være effektivt for mindre prosjekter med skiftende eller ukjente funksjonelle krav. (Waterman et al., 2015)

Et balansepunkt mellom disse to strategiene er ***Respondere på endringer***-strategien. Den innebærer at teamet bruker en smidig tilnærming, basert på fem taktikker:

- Et enkelt design, som ikke tar høyde for alle mulige framtidige behov.

- Bekrefte arkitekturen og designet gjennom å teste kjørende kode basert på design.
- Utsette arkitekturbeslutninger til det finnes nok informasjon om kravene.
- Bruke gode design-praksiser som legger til rette for å kunne utføre endringer.
- Planlegge for flere muligheter og bygge inn muligheter i designet.

De tre første bidrar til å redusere behovet for arkitekturarbeid før oppstart. De to siste kan bidra til å øke omfanget av arkitekturarbeid, men vil ofte redusere det totale omfanget av innsats. Teamkultur, kundens smidighet og erfaringen i teamet øker evnen til å følge strategien *Respondere på endringer.* Tilsvarende vil fravær av disse elementene kunne føre til *Big design up-front*. Ukjente eller ustabile krav driver fram et behov for å *Respondere på endring*. Der tidlig uthenting av gevinst er sentralt, vil det drive fram behovet for en *Emergent architecture.* (Waterman et al., 2015)

Strategien *Respondere på endring* kan komme i konflikt med behov for å adressere (teknisk) risiko. Det vil da være sentralt å finne ut av hvilken risiko man må prioritere å redusere tidlig i prosjektet. Strategien *Adressere risiko* innebærer å håndtere teknisk risiko i forkant, for eksempel å ta viktige teknologivalg som er vanskelige å reversere. Ved å aktivt redusere risiko til et akseptabelt nivå på forhånd, kan man legge til rette for å benytte *Respondere på endring*-strategien underveis i prosjektet. (Waterman et al., 2015).

Bruk av rammeverk og standardarkitektur kan redusere behovet for beslutninger i prosjektet. Waterman presiserer likevel at denne tilnærmingen ikke kan løse alle problemer. Har man spesielle utfordringer i prosjektet, er det ikke gitt at det finnes gode standardløsninger på disse. Bruk av rammeverk og standardarkitektur kan likevel bidra til å redusere teknisk risiko, og øke evnen til å *Respondere på endring*.

Smidige teknikker

Backlog og *iterativ tilnærming* er smidige teknikker som er mye brukt i arkitektursammenheng. Ved å utforme arkitekturbehov og ikke-funksjonelle krav som brukerhistorier, kan de vurderes som en del av den samme prosessen som de funksjonelle kravene. (Yang et al., 2016, p. 169).

Det kan være en utfordring at produkteiere som ikke har tilstrekkelig forståelse av verdien av arkitekturarbeid nedprioriterer brukerhistorier som ivaretar arkitektur og tekniske aspekter

(Yang et al., 2016, p. 173). Det er sentralt at arkitekten klarer å formidle hvilken forretningsverdi arkitektturelementene tilfører programvaren, for eksempel å øke vedlikeholdbarheten slik at kostnader ved senere endringer kan holdes nede. Madison (2010) anbefaler at prosjektets arkitekt vedlikeholder en egen «arkitekturkø», adskilt fra prosjektets produktkø. I dialog med teamet og produkteieren flyttes elementer fra denne køen inn i prosjektet produktkø.

Cleland-Huang et al. (2013) beskriver erfaringer med en tilnærming der *Arkitekturkyndige personas* benyttes for å beskrive viktige arkitekturkrav i form av brukerhistorier. Tilnærmingen gjør arkitekturkravene forståelige for ikke-tekniske interessenter, og gjør det mulig å diskutere hvilke arkitekturkrav som tilfører reell verdi. Arkitekter, sammen med utviklere og driftspersonell, kan betraktes som interessenter til programvaren, på lik linje med forretningssiden (Cleland-Huang, Czauderna, & Keenan, 2013).

Forskningen viser at arkitekter som lykkes i smidige prosesser er skolerte og i stand til å kommunisere med «techies». Det er viktig at rollen er tydelig definert og forstått. Arkitekter kan ikke sitte i «elfenbenstårn», men delta i utviklingsteamene. Arkitekter må anerkjenne utviklere som viktige interessenter for arkitekturen, og involvere utviklere i relevante beslutningsprosesser. Kommunikasjon bør foregå direkte og kontinuerlig for å understøtte ide-utvikling. (Yang et al., 2016, p. 177)

Arkitektursentriske teknikker

Nord og Tamayko (2006) argumenterer for at «tradisjonelle» arkitektursentriske teknikker kan benyttes sammen med smidige metoder, for å utnytte styrkene ved begge tilnærminger.

Teknikken «Quality Attribute Workshop» kan benyttes for å identifisere krav i en tidlig fase, før arkitekturdesign og utvikling påbegynnes. Deltakerne utarbeider scenarier som representerer viktige kvalitetsattributter. Et scenario kan for eksempel være at det skal være mulig å legge til en ny web-klient til løsningen på under 90 dager, uten å påvirke eksisterende klienter. Scenariene brukes videre i utviklingsprosessen, dokumenteres som en del av arkitekturdokumentasjonen, og kan brukes for å definere akseptansekriterier for test. (Nord & Tomayko, 2006, pp. 48–50)

Architecture Trade-off Analysis (ATAM) og Cost-Benefit Analysis Method (CBAM) er teknikker for å prioritere kvalitetsattributter. Nord og Tomayko har beskrevet hvordan disse

kan benyttes sammen med den smidige metoden XP. Andre opplever disse tilnærmingene for omfattende, og har utviklet lettvekts-alternativ. *Attribute Driven Design* baserer designet på prioriterte kvalitetsattributt-scenarier, og bidrar dermed til at de viktigste kravene møtes. Man etablerer et grovkornet design, som kan detaljeres iterativt av teamet. (Nord & Tomayko, 2006, pp. 50–51).

I *Decision Centric Architecture Reviews* (DCAR), undersøkes arkitekturen i et system eller prosjekt gjennom en workshop der sentrale interessenter deltar. Fokus er på dokumenterte og udokumenterte beslutninger tatt i prosjektet. Gjennom en gruppe-prosess identifiseres og evalueres beslutningene, og det defineres oppfølgingspunkter for forbedring. Prosessen mindre tidkrevende enn tradisjonelle review-prosesser, og gjennomføres normalt i løpet av en dag. (van Heesch, Eloranta, Avgeriou, Koskimies, & Harrison, 2014)

Arkitekturbeslutninger

Det er viktig at de faktiske arkitekturbeslutningene dokumenteres eksplisitt. (Yang et al., 2016, p. 177). Arkitekturbeslutninger utgjør normalt bare en liten andel av de beslutningene som tas i løpet av et prosjekt. Til gjengjeld er de ofte vanskelige å reversere, endre eller refaktorere. Det er en fare for at «alt» betraktes som arkitekturbeslutninger, med en konsekvens at «alt» må besluttes tidlig, og ved hjelp av omfattende prosesser. I realiteten har de fleste designbeslutninger ikke konsekvenser for arkitekturen, og mange beslutninger kan og bør tas i forbindelse med design eller i forbindelse med koding. (Abrahamsson et al., 2010, p. 18)

Blair et al. (2010) beskriver en tilnærming kalt *Responsability Driven Architecture* (RDA). RDA definerer beslutningsprosess som er ment som et kompromiss mellom *Big Design up-front* (BDUF) og *Emergent design*. Tilnærmingen er basert på teorien om *real-opsjoner*, som stammer fra finans-sektoren. En real-opsjon er en identifisert mulighet, som er tilgjengelig fram til et gitt tidspunkt. Bedriften kan velge å utsette beslutningen fram til dette gitte tidspunktet. Dette handlingsrom til å konsentrere seg om oppgaver som er viktige på kort sikt.

Overført til arkitekturarbeid, innebærer dette å først identifiser problemstillinger knyttet til arkitektur og design som må løses i prosjektet. Det er sentralt å identifisere interessenter og plassere ansvar for den enkelte beslutning. Deretter identifiseres første mulige tidspunktet det kan være fornuftig å begynne å jobbe med problemet, og det siste tidspunktet beslutning om

løsning kan tas for å ikke forsinke framdrift eller øke risiko. Metoden muliggjør at beslutninger som ikke er nødvendige å ta før oppstart kan utsettes, slik at man har mer kunnskap tilgjengelig når beslutningen skal tas. (Blair, Watt, & Cull, 2010, p. 27)

(Brown et al., 2010) trekker også fram real-opsjoner som en tilnærming for å konstruere «just-in-time» arkitektur. I tillegg til å kartlegge funksjonelle behov som skal implementeres må man kartlegge arkitekturegenskaper som skal implementeres, og kartlegge avhengigheter mellom disse. Kartlegging av avhengigheter, sammen med vurderinger av hvilken teknisk gjeld som eventuelt oppstår om implementeringen utsettes, kan brukes som grunnlag for vurdering av realopsjonene.

Arkitekturbeskrivelse

Selv om muntlig kommunikasjon er sentralt i smidige prosesser, er det også behov for skriftlig dokumentasjon av viktige elementer i arkitekturen. Omfanget av disse arkitekturbeskrivelsene er et mye omdiskutert tema.

Bruk av *Arkitekturbeskrivelser* i kombinasjon med smidig er den tilnærmingen som er mest undersøkt og diskutert i forskningen som er undersøkt av Yang et al. (2016). Hvor mye arkitekturdokumentasjon som er «passe» varierer avhengig av en rekke faktorer, knyttet prosjektets omfang og tidsrammer, teamet og interessentenes kompetanse og motivasjon, og systemets og arkitekturens kvalitet. (Yang et al., 2016, p. 178).

I en smidig arkitekturdrevet modelleringsprosess kreves lite arkitekturplanlegging på forhånd, og bare de artefaktene som bidrar til iterasjonen som pågår utvikles. Prinsippet som følges for å oppnå arkitektursmidighet er «akkurat nok design», basert på «akkurat nok forventninger». Det er viktig at forutsetningene man legger til grunn er informerte, slik at man ikke overvurderer eller undervurderer arkitekturspørsmål og framtidige behov. (Yang et al., 2016, p. 169)

Rendell (2009) beskriver erfaringer fra et prosjekt der arkitektens rolle ble endret fra å inspisere designdokumenter, til å kvalitetssikre kode, i samarbeid med utviklingsteamene. Teamet og arkitekten identifiserer områder som er kompliserte nok eller viktige nok til å dokumenteres. En utvikler som kjenner området godt får ansvaret for å gjennomføre en white-board-sesjon med teamet. Foto av whiteboardet sammen med viktige konklusjoner legges inn i prosjektets wiki. For spesielt viktige områder kan det også etableres UML-diagram eller

annen mer formell dokumentasjon. Ved behov eksporterer teamet innholdet i wikien til en dokument-mal, slik at det kunne arkiveres og distribueres i henhold til formelle krav i virksomheten. (Rendell, 2009, pp. 184–185)

(Hadar, Sherman, Hadar, & Harrison, 2013) gjennomførte en case-studie, der resultatet er en dokumentmal som setter begrensninger på omfanget av beskrivelsene. Arkitekturen skal presenteres som en «elevator pitch», altså så kort som mulig, og framheve det som er viktigst for interessentene. Dokumentene orienteres rundt planlagte leveranser, og beskriver sentrale trekk ved systemet, systemets tilstand før endring, hvilke endringer som er planlagt, og hva som er viktige mål og ikke-funksjonelle krav.

Interessenter kan ha et ønske om at design som ivaretar deres krav er beskrevet på forhånd, slik at de kan godkjenne løsningen, og føle seg trygge på at kravene de representerer blir ivaretatt. I RDA (Blair et al., 2010) avtaler teamet og interessentene en overordnet tilnærming før oppstart, sammen med et løfte om når designet skal være på plass. For eksempel må krav knyttet til utrulling være på plass før første release, men teamet kan love at det meste er klart etter noen sprinter. Dokumentasjonen kommer på plass i den iterasjonen kravet implementeres. Interessentene godkjenner designet fortløpende. Ved prosjektets slutt finnes en grundig dokumentasjon av de ikke-funksjonelle kravene, som er utarbeidet løpende, og inkrementelt.

Arkitekturforståelse

«*Arkitekturforståelse*» diskuteres også i mange av studiene som er undersøkt i (Yang et al., 2016). Siden smidig metodikk forutsetter en inkrementell tilnærming, der arkitekturen utvikles gjennom kommunikasjon mellom aktørene, kan dårlig kommunikasjon påvirke hele utviklingsprosessen. (Yang et al., 2016, p. 174)

Det er viktig at teamet forstår sammenhengen mellom arkitekturspørsmål og de funksjonelle behovene som skal løses, og at arkitekturen har tydelige og kommuniserte «boundaries». (Yang et al., 2016, p. 177).

En av teknikkene som ble av Rendell (2009) benyttet var å generere UML-diagrammer basert på koden, for å avdekke om den er i henhold til det avtalte designet. Kravene til arkitekturen ble i stor grad uttrykt som prinsipper, og ble formidlet muntlig til utviklerne i en felles gjennomgang. Rendell peker på paradokset at tekniske arkitekter som utelukkende inspiserer

dokumenter, avskjæres fra å nyttiggjøre seg av erfaringene og kompetansen som gjorde dem kvalifiserte til å bli arkitekter. (Rendell, 2009, p. 180)

Faber (2010) beskriver erfaringer fra Siemens, der man gradvis beveget seg vekk fra en arkitekturprosess der arkitektene formidlet design og beslutninger gjennom dokumenter, til å samhandle tett med systemutviklere. Det er viktig med kommunikasjon mellom arkitekter og utviklere fra et tidlig stadium i prosjektet, og gjennom hele prosjektets livsløp. For å få til dette er det en viktig forutsetning at begge parter har nytte av kommunikasjonen. Arkitekten må innta en rolle som *tjenestetilbyder* til utviklingsteamet, ikke bare ensidig pålegget utviklingsteamet føringer. Arkitekten er en *mesterprogrammerer* som bidrar med kildekode, ikke bare dokumenter. Arkitekten kan dermed formidle viktige elementer av arkitekturen på utviklernes eget språk, i tillegg til at han bidrar direkte til framdriften i prosjektet. For organisasjoner med større arkitekturmiljøer, er det tilstrekkelig at arkitekturteamet som helhet kan dekke disse behovene. (Faber, 2010)

(Blair et al., 2010, pp. 28–30) beskriver en lignende arkitektrolle, der arkitekten skal være en tilrettelegger. Arkitektur skal oppleves som en tjeneste, ikke som en belasting for utviklingsteamet. I en tradisjonell rolle fungerer arkitekten ofte som en fasade mellom teamet og resten av organisasjonen, og ender lett som en flaskehals. Informasjon kan også gå tapt, siden krav formidles via arkitekten til teamet, istedenfor at interessentene som eier kravene kommuniserer med teamet selv. Arkitekten bør heller fungere som en tilrettelegger som «kobler» interessenter med teamet og legger til rette for god kommunikasjon.

Andre arkitekturaktiviteter

Rammeverket til (Li et al., 2013) beskriver ytterligere fire arkitekturaktiviteter. Disse er ifølge studien til (Yang et al., 2016) ikke undersøkt i like stor grad som *Arkitekturbeskrivelse* og *Arkitekturforståelse*.

Analyse av arkitekturpåvirkning

(Yang et al., 2016) peker på at det er lite litteratur om *Analyse av arkitekturpåvirkning* i sammenheng med smidig, og at det kan være fordi dette er tidkrevende prosess som ikke enkelt lar seg flette inn i smidige prosesser (Yang et al., 2016, p. 176).

(Diaz, Perez, Garbajosa, & Yague, 2013) beskriver erfaringer med en tilpasning til Scrum (*agile sprint architecting*) hvor man kartlegger påvirkningen endringen de prioriterte brukerhistoriene vil ha på den realiserte arkitekturen, og dokumenterer nødvendige beslutninger og framtidige behov for beslutninger. Prosessen som foreslås av Diaz et al framstår også som forholdsvis omfattende, men har den fordelen at den understøtter en kontinuerlig revurdering av arkitekturen, og oppdatert arkitekturdokumentasjon.

Arkitekturgjenbruk

(Durdik, 2011) peker på at en svakhet ved smidige metoder er at man i liten grad legger til rette for gjenbruk, siden man unngår arkitekturmodellering og andre teknikker som ikke gir umiddelbar nytte. På den annen side gir refaktorering mulighet for å gjøre tilpasninger slik at komponenter, kode eller patterns kan gjenbrukes.

Denne typen gjenbruk beskrives også av Waterman et al. (2015), i strategien *Bruk av rammeverk og standardarkitektur.* Manglende investeringer i IT-infrastruktur som sikkerhet, grunndata og samhandlingskanaler kan også redusere fleksibilitet, og gir tregere utvikling av nye eller endrede funksjoner. (Weill, Subramani, & Broadbent, 2002, s. 19)

Arkitekturgjenvinning

Arkitekturgjenvinning er ikke omtalt i noen av artiklene som inngår i Yangs studie.

Arkitekturrefaktorering

Refaktorering er et sentralt virkemiddel i smidige metoder, og innebærer å endre strukturen på koden, for å forbedre kvaliteten, uten at systemets funksjonalitet påvirkes. Arkitektur-refaktorering er et sentralt virkemiddel for å utvikle arkitektur i en smidig prosess. Arkitekturdesignet optimaliseres gradvis underveis i utvikling, og kan være et viktig virkemiddel for å oppnå viktige arkitekturegenskaper som vedlikeholdbarhet (Yang et al., 2016).

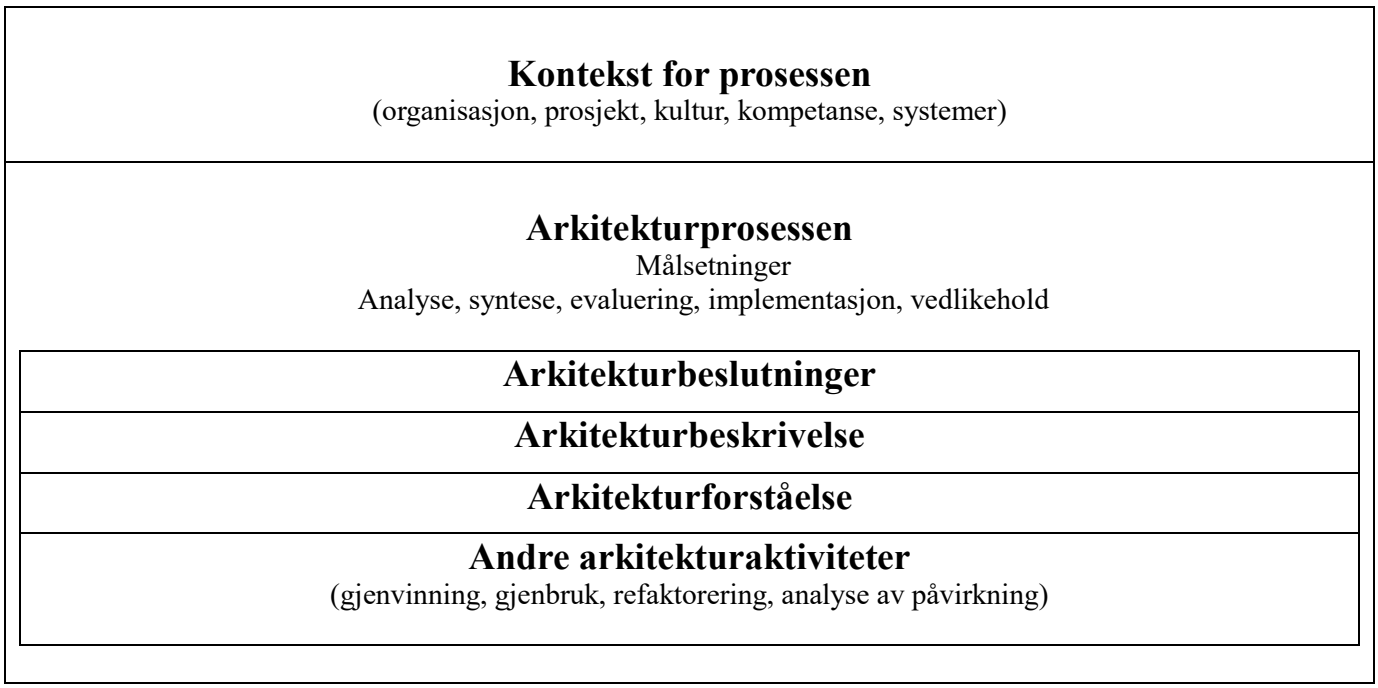
(Sharifloo, Saffarian, & Shams, 2008) beskriver lettvektsteknikker for å identifisere arkitekturforbedringer, slik at man kan gjennomføre arkitektur-refaktoring. Arkitekten kan undersøke modeller av koden ved slutten av hver iterasjon, og identifiserer forslag til forbedringer som overleveres til det smidige teamet. Forslagene kan evalueres av team og

interessenter i en workshop, hvor man også identifiserer nye forbedringsforslag knyttet til prioriterte kvalitetsattributter.

2.3 Oppsummering av innsikter fra forskningen

Kruchten (2013) beskriver hvordan organisasjonen og prosjektenes egenskaper utgjør en kontekst som påvirker prosessen, og hva som utgjør god praksis. I oppsummeringen av teori skilles det derfor mellom kontekst (omgivelser) og selve prosessen. Arkitekturprosessen som er beskrevet først av Hofmeister et al. (2007), siden utvidet av blant annet Tang et al. (2010) og Li et al. (2013) brukes som rammeverk for å undersøke selve prosessen. Noe av litteraturen som er undersøkt peker på hva som er formålet med en mer smidig praksis. Dette har fått et eget tema, Målsetninger.

Arkitekturbeslutninger er ikke er et eget tema i Hofmeisters modell, eller noen av utvidelsene, men berøres av mange av artiklene i teorigjennomgangen. Hofmeister omtaler *Analyse* som et verktøy for å kunne ta de riktige beslutningene. Men arkitekturbeslutninger kan også tas i forbindelse med implementasjon eller vedlikehold, *Arkitekturbeslutninger* betraktes derfor som et tverrgående tema, på linje med aktivitetene beskrevet av (Li et al., 2013)



Figur 3 - Rammeverk for oppsummering av teori

I Tabell 4 oppsummeres teorien som er gjennomgått, sortert inn i modellen. Strukturen brukes videre som utgangspunkt for å oppsummere empiri og analyse.

Tabell 4 - Innsikter fra forskningen

Tema	Innsikt
Kontekst for prosessen	**Prosessens kontekst er avgjørende for evnen til smidighet** • Organisasjonens modenhet påvirker prosjektenes, og dermed prosessenes, egenskaper (Kruchten, 2013) • Store prosjekter begrenser muligheten til smidig tilnærming, og øker behovet for å uttrykke arkitektur på forhånd. (Dybå & Dingsøyr, 2009), (Nord & Tomayko, 2006), (Boehm, 2011) • Kompetanse og kultur hos teamdeltakere og interessenter er viktig for å lykkes med smidige prosesser. (Waterman et al., 2015), (Yang et al., 2016), (Dybå & Dingsøyr, 2008). • Arkitektene må ha god teknisk kompetanse. (Rendell, 2009), (Faber, 2010) • Systemenes alder, størrelse eller risikoprofil kan være til hinder for smidighet. Riktig valg av infrastruktur og systemutforming kan gi bedre støtte for smidighet. (Bloomberg, 2013), (Waterman et al., 2015), (Kruchten, 2013)
Arkitekturprosessen - målsetning	**Arkitekturarbeidet bør konsentreres om risiko-områder** • Arkitekturprosessen kan balansere behovene for smidighet i prosessen og risikohåndtering, gjennom å prioritere områder med høy risiko tidlig. (Poort, 2012), (Waterman et al., 2015). **En smidig prosess må resultere i et smidig system** • En smidig prosess må resultere i et smidig system, ellers vil systemet være til hinder for videre *Vedlikehold og videreutvikling*. (Beck et al., 2001) (Bloomberg, 2013), (Buschmann & Henney, 2013).
Analyse, syntese, evaluering, implementasjon og vedlikehold	**Smidige teknikker og lettvektsmetoder kan understøtte en smidig arkitekturprosess** • En smidig arkitekturprosess kan understøttes av smidige teknikker, og av lettvektsvarianter av arkitektursentriske metoder. (Yang et al., 2016), (Madison, 2010), (Cleland-Huang et al., 2013), (Nord & Tomayko, 2006).
Arkitektur-beslutninger	**Arkitekturbeslutninger bør tas trinnvis, og involvere viktige interessenter** • Beslutningsbehov og interessenter bør kartlegges tidlig i prosjektet, og legges inni en plan som angir perioden hver enkelt beslutning må tas. (Poort, 2012), (Blair et al., 2010).
Arkitekturbeskrivelse	**Arkitekter skal ikke detaljstyre design, men definere rammer og handlingsrom for utviklingsteamene** • Arkitekturvisjon, -prinsipper og -krav må understøtte forretningens mål. (Leffingwell, 2011), (Bloomberg, 2013) • Arkitektene må ikke detaljstyre design, men definere handlingsrommet til utviklingsteamet, og ha tillit til utviklingsteamets kompetanse. (Erder & Pureur, 2016), (Madison, 2010), (Abrahamsson et al., 2010) • Designdokumentasjon bør være høynivå, og dokumentere rasjonalet bak sentrale designvalg. (Selic, 2009). Kode kan brukes for å uttrykke design (Leffingwell, 2011), (Faber, 2010)
Arkitekturforståelse	**Direkte muntlig kommunikasjon mellom interessenter er vesentlig for arkitekturforståelse** • Arkitekten må delta i utviklingsteamet, og kunne bidra i tekniske diskusjoner. Arkitektene må tilrettelegge for kommunikasjon mellom teamet og viktige kravstillere. (Faber, 2010), (Madison, 2010), (Blair et al., 2010)
Andre arkitekturaktiviteter	**Det er lite empiri om øvrige arkitekturaktiviteter i en smidig kontekst** • ***Analyse av arkitekturpåvirkning*** kan være tidkrevende.(Yang et al., 2016) • Smidige metoder legger ikke godt til rette for ***arkitekturgjenbruk***. (Durdik, 2011) • Behov for ***arkitekturrefaktorering*** kan identifiseres gjennom inspeksjoner av arkitekturen etter hver sprint.. (Sharifloo et al., 2008)

3 Metode

Oppgaven er en case-studie, med intervjuer som viktigste informasjonsskilde. Analysen er basert på «pattern-matching» (Yin, 2013), der teori og funn innenfor hvert tema i et teorirammeverk sammenlignes.

3.1 Forskningsdesign

Oppgaven har et deduktivt design, der funnene testes opp mot teorirammeverket som er beskrevet i kapittel 2.3, for å finne fram til ny kunnskap og praktiske implikasjoner.

Undersøkelsen er en case-studie, som tar utgangspunkt i et positivistisk perspektiv. Et positivistisk perspektiv forutsetter at det finnes en virkelighet som er uavhengig av observatøren (Yin, 2013, p. 17). Case studier kan benyttes for finne ny kunnskap og empiriske bevis fra «ekte mennesker i ekte organisasjoner». (Myers, 2013, p. 76). Case studier er spesielt egnet når man skal finne svaret på «hvordan» og «hvorfor»-spørsmål i samtiden, og dersom forskeren har liten eller ingen kontroll over variablene som undersøkes (Yin, 2013, pp. 12–14).

3.2 Teori

Den første delen av problemstillingen er å avdekke om det finnes forskning som kan peke en arkitekturprosess som understøtter hurtige leveranser av IT-løsninger. Dette teoretiske grunnlaget er viktig for å få en forståelse for temaet for undersøkelsen, og for å gi rammer for datainnsamling og analyse (Yin, 2013, pp. 37–38).

Litteraturen som er gjennomgått er identifisert gjennom søk i Google Scholar og Oria. I tillegg til at jeg har fått tips til litteratur fra veileder og fra kollegaer. Referanselistene i artikler er brukt for å identifisere nye relevante kilder. Spesielt har de forholdsvis ferske studiene av Waterman et al. (2015) og Yang et al. (2016) vært nyttige kilder for å finne fram til mer litteratur.

Innsikter fra litteraturen er oppsummert i kapittel 2.3. Disse innsiktene utgjør et forventet mønster (*expected pattern*) som gir grunnlag for videre datainnsamling, -analyse og diskusjon (Yin, 2013, p. 143).

(Yin, 2013, p. 34) peker på at begrepsbruken som benyttes i en case-studie ikke bør være *idiosynkratisk*, det vil si særegen for studiet. Idiosynkratisk begrepsbruk gjør det vanskelig å sammenligne funn med tidligere forskning, og analyse-enhetene bør derfor ta utgangspunkt i det som tidligere er studert. Teorirammeverket tar derfor utgangspunkt i begrepsbruk definert av Hofmeister et al. (2007), Tang et al. (2010) og Li et al (2013). Gjennom å ta utgangspunkt i begrepsbruk som er definert i tidligere teoristudier blir det enklere å koble funn til tidligere forskning.

3.3 Datainnsamling

Det å benytte flere kilder er en av de fremste styrkene til case-studien som metode. Studier med flere informasjonskilder regnes som mer pålitelige, og gir grunnlag for triangulering, det vil si at funn fra en type kilde kan styrkes av funn fra en annen type kilde. (Yin, 2013, pp. 119–121).

En case-studie kan baseres på kvalitative eller kvantitative data, eller på en kombinasjon av disse. (Yin, 2013, p. 19). Jeg har valgt en kvalitativ tilnærming, siden forskningsspørsmålet forutsetter forståelse av kontekst for arkitekturprosessen. En kvalitativ tilnærming, der informasjon i hovedsak hentes inn gjennom å intervjue folk med relevant erfaring, er bedre egnet for å forstå kontekst enn spørreundersøkelser eller andre former for kvantitative undersøkelser.

Hovedkilden for informasjon i denne undersøkelsen er intervjuer, som er supplert av undersøkelser av relevante dokumenter fra IT-avdelingen. Dokumentene som er brukt som kilder er listet i kapittel 8.1. Uttrekk fra intervjuene er dokumentert i Vedlegg 2

Datainnsamling for denne oppgaven ble gjennomført første halvår 2016, og intervjuer og dokumenter som er undersøkt beskriver det som da var gjeldende organisering og rutiner.

3.3.1 Intervjuer

Intervjuene i denne undersøkelsen ble gjennomført som semi-strukturerte intervju, med en intervjuguide som tok utgangspunkt i teorien og problemstillingen. (se Vedlegg 1)

Fordelen med intervjuer er at datainnsamlingen kan fokuseres direkte på temaet for case-studien, og at man kan få forklaringer og personlige synspunkter fra personer som kjenner materien som undersøkes godt. Svakheten med intervjuer som kilder, er at informanten kan

huske feil, eller påvirkes av bias. Bias kan for eksempel oppstå hvis spørsmål er dårlig formulert, eller hvis informanten svarer «det han tror du vil høre» (refleksivitet) (Yin, 2013, p. 106). Et tiltak for å veie opp mot disse svakhetene var at det ble gjennomført en felles gjennomgang av foreløpige funn med informantene, se kapittel 3.3.1.

Informanter

Myers (2013) vektlegger viktigheten av å identifisere nøkkelinformanter, det vi si de personene som har mest kunnskap og beslutningsmyndighet knyttet til temaet, men samtidig unngå *elite-bias,* det vil si at man bare intervjuer personer med høy status. Informantene (Tabell 5) ble valgt for å belyse problemstillingen fra flere forskjellige synspunkt, gjennom at de har forskjellige roller i prosessen, og i varierende grad har påvirkning på hvordan prosessen fungerer. Informantene har også tilhørighet i forskjellige deler av IT-avdelingen, og er en blanding av ansatte og eksterne konsulenter.

Siden undersøkelsen skulle kartlegge hvilke hindringer og muligheter som er knyttet til dagens praksis, var det ikke relevant å ha med deltakere som helt nylig har kommet til NAV, og ikke er kjent med hvordan prosessene fungerer. Alle informantene har jobbet for NAV over lengre tid, fra to til ti år. Åtte av informantene har teknisk utdanning, den siste har økonomiutdanning. De har mellom 6 og 30 års arbeidserfaring fra IT-området, både privat og offentlig sektor. De fleste informantene har erfaring fra smidige prosesser.

Miljøer i IT-avdelingen som i dag ikke er direkte involvert i utvikling av programvare er ikke representert blant informantene. Dette er først og fremst et grep å ikke øke bredden i temaene som blir behandlet i oppgaven til et uhåndterlig nivå.

De første 4 informantene ble valgt ut basert på egen kunnskap om deres erfaringer med arkitektur og smidig tilnærming, de øvrige ble valgt ut basert på innspill fra de første informantene. Denne tilnærmingen bidro til at utvalget av informanter ikke bare ble «de jeg først kom på».

De resterende informantene ble kontaktet i «bolker», for å løpende kunne vurdere om det var behov for flere intervjuer, og hvilket perspektiv informantene burde ha for å supplere tidligere intervjuer. (Kvale & Brinkmann, 2015, p. 148) viser til at antall intervjuer må vurderes ut fra hva man har behov for å finne ut av, og tid man har tilgjengelig. Videre vil det etter et visst

punkt tilføres stadig mindre ny kunnskap gjennom nye intervjuer, informantene vil i større grad gjenta ting som allerede er sagt av andre. En tommelfingerregel er «15 +/- 10» intervjuer.

Etter ni intervjuer dukket det ikke lenger opp så mye nytt i intervjuene, og det viste seg at informantene på pekte på mange av de samme hindringene og mulighetene, tross av forskjellig synspunkt.

Tabell 5 - Informanter

	Rolle	Bakgrunn	Ansatt/ konsulent	Erfaring i NAV
1	Løsningsarkitekt	Tilknyttet Seksjon for IKT-arkitektur	Begge	3 år
2	Løsningsarkitekt	Tilknyttet prosjekt som tester ut alternativ smidig metodikk.	Begge	4 år
3	Områdearkitekt	Ansvarlig for et delområde i systemporteføljen.	Ansatt	5 år
4	Løsningsarkitekt	Erfaring fra hovedleveranser.	Begge	2 år
5	Løsningsarkitekt	Jobber for en av utviklingsleverandørene i IT-avdelingen.	Konsulent	3 år
6	Rådgiver	Tilknyttet forvaltningskontor i IT-avdelingen.	Begge	2 år
7	IT-prosjektleder	Erfaring fra smidige prosjekter, blant annet i NAV.	Konsulent	5 år
8	Utvikler	Java-utvikler og arkitekt.	Begge	7 år
9	Løsningsarkitekt	Erfaring fra flere større prosjekter i NAV.	Konsulent	10 år

Intervjuguide

(Kvale & Brinkmann, 2015, s. 137) anbefaler at intervjuer baseres på en intervjuguide. Semi-strukturerte intervju gir rom for at spørsmål kan improviseres underveis i intervjuet, samtidig som strukturen i intervjuguiden gir en viss konsistens mellom de forskjellige intervjuene (Myers, 2013, p. 122).

Intervjuguiden tok utgangspunkt i problemformuleringen, og var orientert rundt informantenes opplevelse av muligheter og hindringer i forbindelse med arkitekturprosessen. Guiden inneholder både direkte og indirekte spørsmål orientert om dette. Spørsmålene ble forsøkt formulert som åpne spørsmål som begynner med spørreord, for å unngå lite informative ja/nei-svar (Myers, s. 130-131). I noen av intervjuene ble oppfølgingsspørsmål improvisert.

Spørsmålene ble bevisst ikke knyttet «strengt» opp til modeller eller begreper fra teorigjennomgangen, jeg ønsket å få informantenes perspektiv på hva arkitekturprosessen består av, uavhengig av om de har formell kunnskap om standardmetoder, referansemodeller o.l.

Etter de to første intervjuene ble det gjort justeringer av intervjuguiden. For de resterende intervjuene ble samme mal benyttet (versjon 1.2).

Gjennomføring av intervjuene

I innledningen til hvert intervju ble det informert om at begrepene *smidig* og *arkitekturprosess* skulle tolkes vidt. Begrepet arkitekturprosess har i innledningen til intervjuene blitt forklart som «det vi faktisk gjør når vi jobber med arkitektur, ikke nødvendigvis det samme som formelle prosesser som er beskrevet i Arkitekturportalen»

Det ble skrevet referat fortløpende under intervjuene. Fortløpende referatføring skapte en del naturlige tankepauser for informantene, der de kom på litt mer de ville si.

Det ble i tillegg tatt lydopptak av intervjuene. Intervjuene ble ikke transkribert ord for ord fra opptakene, men ble brukt som «backup» der notatene var ufullstendige, slik at referatene kunne ferdigstilles.

Erfaringer fra intervjuene

De som stilte opp til intervju var engasjerte i problemstillingen, og hadde mye på hjertet. Det var satt av 90 minutter til hvert intervju, det korteste intervjuet tok 45 minutter, det lengste tok 80. I de lengste intervjuene ble et par spørsmål kuttet, snarere enn å avbryte eller stresse informanten.

I noen tilfeller «unnskyldte» informantene seg når de sa noe kritisk om Seksjon for IKT-arkitektur eller eksisterende arkitekturprosesser. Dette har nok sammenheng med at jeg er ansatt i Seksjon for IKT-arkitektur, og har deltatt i utformingen av deler av eksisterende prosesser, dette er nærmere drøftet i kapittel 3.7.3

3.3.2 Dokumentanalyse

Dokumenter kan være gode kilder, blant annet fordi de er stabile, og dermed etterprøvbare, og fordi de ikke er utarbeidet som en del av case-studien. De kan imidlertid avspeile ukjent bias hos forfatteren, eller gi bias gjennom at utvalget er ufullstendig. (Yin, 2013, p. 106). I en case-

studie kan dokumenter gi en støtte til å forsterke eller bekrefte informasjon fra andre kilder (triangulering). Man skal likevel være forsiktig med å stole blindt på dokumenter som kilde – de vil aldri være en helt korrekt beskrivelse av hva som har skjedd (Yin, 2013, p. 107)

Dokumenter er undersøkt både for å sette sammen case-beskrivelsen (kapittel 4), og som en del av datainnsamlingen. Dokumenter som er gjennomgått er listet i kapittel 8.1. Dokumentene har vært nyttige blant annet fordi de beskriver intensjoner bak utforming av prosessene, som ikke alltid er sammenfallende med informantenes opplevelse av praksis.

3.3.3 Strukturering av funn

Analysen ble påbegynt etter tre intervjuer. De renskrevne intervjureferatene ble «klippet opp», og utsagn plassert inn i en tabell (Vedlegg 2) basert på temaene fra teorirammeverket.

Hvert utsagn som ble lagt inn i tabellen hadde referanse tilbake til intervjureferatet det var hentet fra, så det skulle være enkelt å gå tilbake for å finne mer kontekst. Nye intervjuer ble bearbeidet inn i oversikten etter hvert som de var ferdigstilt.

I etterkant av datainnsamlingen ble det tydelig at strukturen som ble brukt i intervjuguiden ikke var godt egnet for analyse. Rammeverket som er beskrevet i kapittel 2.3 ble etablert for å binde sammen teori, analyse og drøfting. Tabellen med utsagn ble sammen med dokumentasjonen fra IT-avdelingen brukt som grunnlag for å beskrive funnene, innenfor temaene som i de forventede mønsteret (se kapittel 5.7).

3.3.1 Deltakervalidering av funn

Deltakervalidering er en tilnærming der funn og eventuelt også analyser og konklusjoner legges fram for informanter, slik at de kan gi tilbakemeldinger. I positivistisk, kvalitativ forskning bruke deltakervalidering først og fremst til å verifisere at forskerens forståelse av det undersøkte fenomenet er objektivt «korrekt», og normalt ikke til å kvalitetssikre analyser og konklusjoner. (Bygstad & Munkvold, 2007).

Informantene ble invitert til et møte, der en første oppsummering av forskningsinnsikter og funn ble presentert (vedlegg 3), og de kunne komme med innspill. Seks av ni informanter hadde anledning til å stille. De øvrige fikk presentasjonen fra møtet tilsendt på epost. Alle informantene fikk også tilsendt referatet fra gjennomgangen, og kunne gi innspill på epost.

Informantene bekreftet mange av funnene som ble lagt fram, men bidro også med presiseringer og detaljering. Blant annet ble den første oppsummeringen rundt temaet *Arkitekturbeslutninger* ble oppfattet som for positivt og unyansert av respondentene.

Referatet fra gjennomgangen (Vedlegg 4) og eposter som med kommentarer som ble mottatt i etterkant behandlet som data på linje med intervjuene, og utsagn er lagt inn i samme tabellstruktur i Vedlegg 2.

3.4 Data-analyse

En kvalitativ undersøkelse innebærer normalt at man samler inn store mengder data. Det er derfor svært viktig at man finner en systematisk tilnærming for å redusere dataene. (Myers, 2013, s. 165-166).

Tabell 6 oppsummerer stegene i analysen. Stegene er ikke gjennomført rent sekvensielt. Tidlige iterasjoner omfattet bare de første stegene, etter hvert ble prosessen utvidet med flere steg. Ved behov har ett eller flere steg blitt gjentatt.

Funnene er oppsummert i kapittel 5.6, strukturert etter samme mønster som teorirammeverket (*expected pattern*), og utgjør et empirisk mønster (*empirical pattern*) som ble brukt sammen med det forventede mønsteret som grunnlag for analysen (kapittel 6). Metoden der man sammenligner et forventet mønster med et empirisk mønster kalles *pattern matching* (Yin, 2013, p. 143).

Tabell 6 - Steg i data-analysen

#	**Aktivitet**	**Resultat**
1	Etablere rammeverk for analyse (forventet mønster).	Tabell 4
2	Sortere intervjusvar etter tema i rammeverket.	Vedlegg 2
3	Oppsummere hindringer og muligheter innenfor hvert tema.	Vedlegg 5
4	Gjennomgang av foreløpige funn med informantene	Vedlegg 3 og 4
5	Oppsummering av funn (empirisk mønster)	Tabell 9
6	Sammenligne empirisk og forventet mønster	Tabell 9 og Vedlegg 6
7	Dokumentere analyse, forklarende mekanismer, forslag til tiltak.	Kapittel 6
8	Beskrive teoretiske og praktiske implikasjoner	Kapittel 7

3.5 Validitet og reliabilitet

3.5.1 Intern validitet (reliabilitet)

Intern validitet (reliabilitet) innebærer å at caset gjengitt på en troverdig, objektiv måte, og at resultatene er reproduserbare. Det må være mulig å gjenta studien, og få samme resultat. Derfor er det viktig å dokumentere tydelig hvordan dataene er samlet inn. Dataene som samles inn, f.eks. intervjuopptak, dokumenter, notater fra observasjoner skal tas vare på i en case studie database. (Yin, 2013, pp. 45–49). I tillegg til representasjoner av data som er lagt ved denne rapporten som vedlegg, er alle lydopptak fra intervjuene, intervjureferat, og interne dokumenter fra NAV som er undersøkt lagret. Lagrede data, sammen med undersøkelsesinstrumentene, som intervjuguiden (vedlegg 1), teorirammeverket, sammen med metodebeskrivelsen i dette kapittelet bør gjøre det mulig å reprodusere studien.

3.5.2 Ekstern validitet

Ekstern validitet innebærer å definere hvilket område funnene kan generaliseres innenfor, gjennom bruk av teori. Det er mulig å generalisere funn fra case-studier, men generaliseringene har ikke samme form som statistiske bevis, man utvider eller generaliserer teori. Det er mer vanlig at man dokumenterer erfaringer, «lessons learned» basert på case-studien. (Yin, 2013, pp. 40–41). Yang et al. (2016) skriver i sin oppsummering at området *smidig arkitektur* er forholdsvis ferskt, og at forskning innenfor området så langt bare kategoriseres som «lessons learned».

Funnene i denne oppgaven er også å anse som «lessons learned», som kan være relevante for andre store IT-organisasjoner.

3.6 Rapport

Dette dokumentet har blitt utarbeidet gradvis gjennom hele arbeidsprosessen. F.eks. er ble første versjon av teorikapittelet et produkt av den første fasen med litteraturgjennomgang. Metodekapittelet ble først formulert som en teoretisk plan, men ble oppdatert underveis med erfaringer fra arbeidet. Skriveprosessen har vært nyttig underveis, for å holde framdrift, og for å holde en rød tråd i arbeidet.

3.7 Etiske problemstillinger

Etikk kan defineres som *«moralske prinsipper som styrer eller påvirker oppførsel»,* og overført til forskning innebærer dette at planlegging, gjennomføring og rapportering av resultater følger slike moralske prinsipper. (Myers, 2013, s. 48-49)

Mer konkret finnes det både lovfestede, formelle og uformelle regler for hvordan et forskningsprosjekt skal gjennomføres på en etisk forsvarlig måte. Kapittelet oppsummerer regler som er relevante for undersøkelsen, og hvilke konsekvenser de har for gjennomføring. De etiske retningslinjene fra Association for Information Systems (AIS Research Conduct Comittee, 2015), er brukt som utgangspunkt, siden de beskriver prinsipper og regler som er relevante for forskning knyttet til informasjonssystemer.

3.7.1 Ærlighet og åpenhet

Plagiering innebærer å gjengi andres arbeid uten å referere til sine kilder, og dermed utgi det for å være egne tanker. Dette regnes som *«en av de verste synder i akademia».* Forskningen må være transparent og etterprøvbar, det innebærer at forskeren må være ærlig i beskrivelsen av data, funn og metode (Myers, 2013, side 50). Forbudet mot plagiering står øverst på AIS liste over etiske regler for forskning, som framhever at data og analyser ikke må diktes opp eller forfalskes. Det er heller ikke akseptabelt å gi en feilaktig framstilling av forskningsmetoden som er brukt. (AIS, 2015).

3.7.2 Beskyttelse av personer som deltar i undersøkelsen

(Ringdal, 2014, s. 454-460) gjengir sju punkter for beskyttelse av personer i forskning, hentet fra Den nasjonale forskningsetiske komite for samfunnsvitenskap og humaniora. Problemstillinger knyttet til skade og belastninger, privatliv og nære relasjoner og tredjepart vurderer jeg ikke som relevant, gitt temaet for oppgaven. Disse er heller ikke tema i AIS Code of Research Conduct. De følgende avsnittene oppsummerer håndtering av de tre resterende punktene.

Krav om konfidensialitet

Informasjon som samles inn i et forskningsprosjekt skal som hovedregel behandles konfidensielt. Personer som har deltatt skal ikke kunne identifiseres i det ferdige arbeidet. I mange undersøkelser gjennomført i organisasjoner anonymiseres organisasjonen undersøkelsen er gjennomført i, for å redusere muligheten for at informanter identifiseres.

Det er ikke mulig i mitt tilfelle, uten å komme i konflikt med prinsippet om transparens (se 3.7.1). Jeg anser åpenhet om egen rolle til å være viktigere enn å anonymisere organisasjonen.

I noen av intervjuene nevnes personer i organisasjonen, enten ved navn eller på andre måter som gjør dem enkelt identifiserbare. Dette er anonymisert etter beste evne i referatene.

(Myers, 2013, s. 51) anbefaler at en passende person fra organisasjonen leser gjennom oppgaven gir kommentarer og til slutt sin godkjenning, dersom organisasjonen ikke skal anonymiseres. Rapportutkastet ble gjennomlest av sjefsarkitekten i IT-avdelingen, som har gitt samtykke til at den ikke klausuleres.

Konsesjon og meldeplikt

Siden det skal samles inn informasjon som indirekte kan brukes til å identifisere enkeltpersoner, er undersøkelsen meldepliktig til Norsk samfunnsvitenskapelig datatjeneste (NSD). Oppgaven er meldt inn til NSD, som har gitt sin godkjenning. (Vedlegg 8).

Informert og fritt samtykke

De som deltar i undersøkelsen skal informeres om hva formålet er, hvordan undersøkelsen skal gjennomføres, og eventuelle negative konsekvenser som kan følge av deltakelsen. Det skal komme tydelig fram at det er frivillig å samtykke til deltakelse. (Ringdal, 2014, s. 356-457). Det er også viktig at informantene er klare over sin rett til å trekke tilbake sitt samtykke når som helst i undersøkelsen. (Myers, 2013, s. 51)

Alle informanter fikk et standardskriv som oppsummerte relevant informasjon om datainnsamlingen, og rettighetene de har til å trekke tilbake samtykke. (Vedlegg 7)

3.7.3 Egen rolle

En del etiske spørsmål er knyttet til min egen rolle, mine holdninger, og at undersøkelsen er gjennomført i den organisasjonen jeg jobber i. Det har vært spesielt viktig å være bevisst rundt problemstillinger knyttet til bias. Bias oppstår når man (bevisst eller ubevisst) søker etter å bekrefte sitt opprinnelige standpunkt, snarere enn å utfordre det (Tversky & Kahnemann, 1974, s. 1128-1130). Ved å aktivt oppsøke flere informasjonskilder og søke etter andre tilnærminger, kan man motvirke denne effekten (Hammond, Keeney, & Raiffa, 1998).

Gjennomgangen av funn sammen med informantene (kapittel 3.3.1) har vært nyttig for å få en motvekt mot dette. For eksempel oppsummerte jeg i første omgang utsagnene fra

informantene som at de var positive til arkitekturbeslutningsprosessen. Dette kan ha vært en form for bekreftelsesbias, siden jeg selv i utgangspunktet hadde et positivt syn på denne prosessen. Tilbakemeldingene i deltakervalideringen viste at jeg ikke hadde tillagt innvendingene som fram i intervjuene nok vekt.

Det har vært viktig at informantene ikke skulle underslå kritikk av etablerte metoder eller praksis ut fra høflighet, eller fordi det oppleves utrygt. Jeg har forsøkt å kommunisere tydelig at motivet (ut over å produsere en masteroppgave) er forbedringer, og at alle innspill i den sammenhengen er nyttige. Det har likevel skjedd i noen av intervjuene at informantene har «unnskyldt seg» når de har kommet med kritikk, eller i første omgang har drevet en form for «selvmoderasjon», for eksempel i form av utsagn som «dere gjør vel så godt dere kan». Jeg har forsøkt å møte dette med å kommunisere tydelig at det er greit å si hva man mener, og respondere positivt på utsagnene, f.eks. ved å uttrykke at de er interessante.

Dobbeltrollen min kan likevel ha en negativ effekt på validiteten i funnene, ved at tema jeg assosieres med har fått «penere behandling» av informantene enn andre tema.

4 Presentasjon av case

IT-avdelingen i Arbeids- og velferdsdirektoratet har ansvar for å utvikle, drifte og forvalte IT-løsningene i NAV, som støtter 19.000 medarbeidere i å utføre sine oppgaver. (NAV, 2016). Avdelingen har omtrent 500 ansatte, som vedlikeholder og drifter ca 300 applikasjoner.

4.1 Systemlandskapet

Applikasjonsporteføljen er sammensatt av flere generasjoner med løsninger, fra stormaskinløsninger, til nyere web-orienterte Java-løsninger, i tillegg til standardsystemer som støtter for eksempel regnskap, lønn og dokumentproduksjon.

Da NAV ble etablert i 2006, var det resultatet av en fusjon av Aetat og Trygdeetaten. De to etatene hadde ved sammenslåingen hvert sitt omfattende sentrale saksbehandlingssystem, på forskjellig teknologiplattform.

- Trygdeetatens fagsystem Infotrygd er et stormaskin-system etablert i 1978.
- Aetats fagsystem Arena er et databasesentrisk system, etablert i 2001.

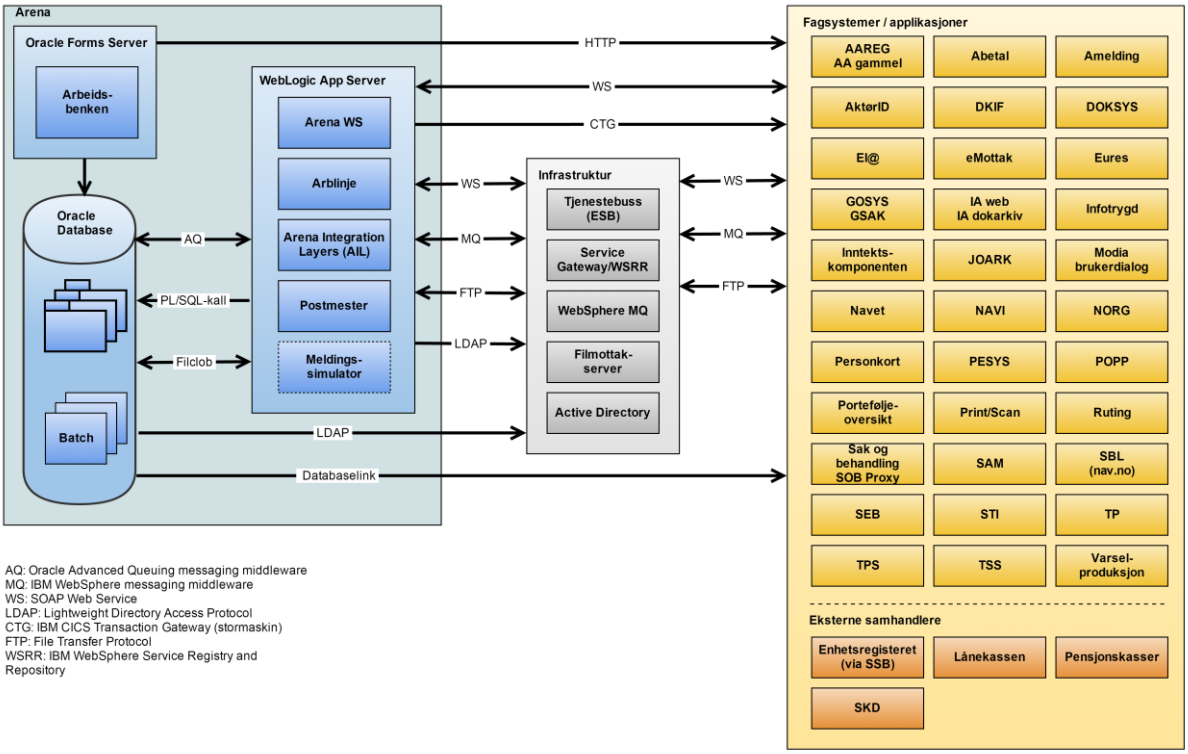
Den nye etaten valgte en tjenesteorientert arkitektur basert på Java som sin strategiske plattform, og løsninger som er utviklet etter 2006 er i all hovedsak basert på dette. Det mest omfattende systemet etablert på denne plattformen er Pensjonsløsningen, som håndterer alderspensjoner og uføretrygd.

I tillegg til saksbehandlingssystemene kommer en lang rekke registre, systemer som understøtter utbetaling av ytelser til brukere, selvbetjeningsløsninger, dokumentløsninger som støtter opp under tjenestene NAV leverer. Figur 4 viser en forenklet oversikt over sentrale applikasjoner.

Figur 4 - Kart over sentrale applikasjoner i NAV (fra intern wiki)

Figur 5 gir en illustrasjon på kompleksiteten antallet systemer og heterogen teknologiplattform kan føre til. Saksbehandlingssystemet Arena samhandler med 33 andre interne systemer i NAV, i tillegg til eksterne systemer. Systemene det integreres mot er basert på flere forskjellige teknologiplattformer, og for å understøtte integrasjonene brukes flere forskjellige protokoller. I tillegg til at Arena må forholde seg til felles infrastruktur for blant annet sikkerhet. Endringer i Arena kan potensielt påvirke de andre systemene, og omvendt. Tilsvarende problemstillinger gjelder også de andre store saksbehandlingssystemene.

Kompleksiteten rundt forvaltning og endringer i de sentrale saksbehandlingssystemene har vært en viktig driver for utforming av organisasjon og prosesser i IT-avdelingen i NAV.

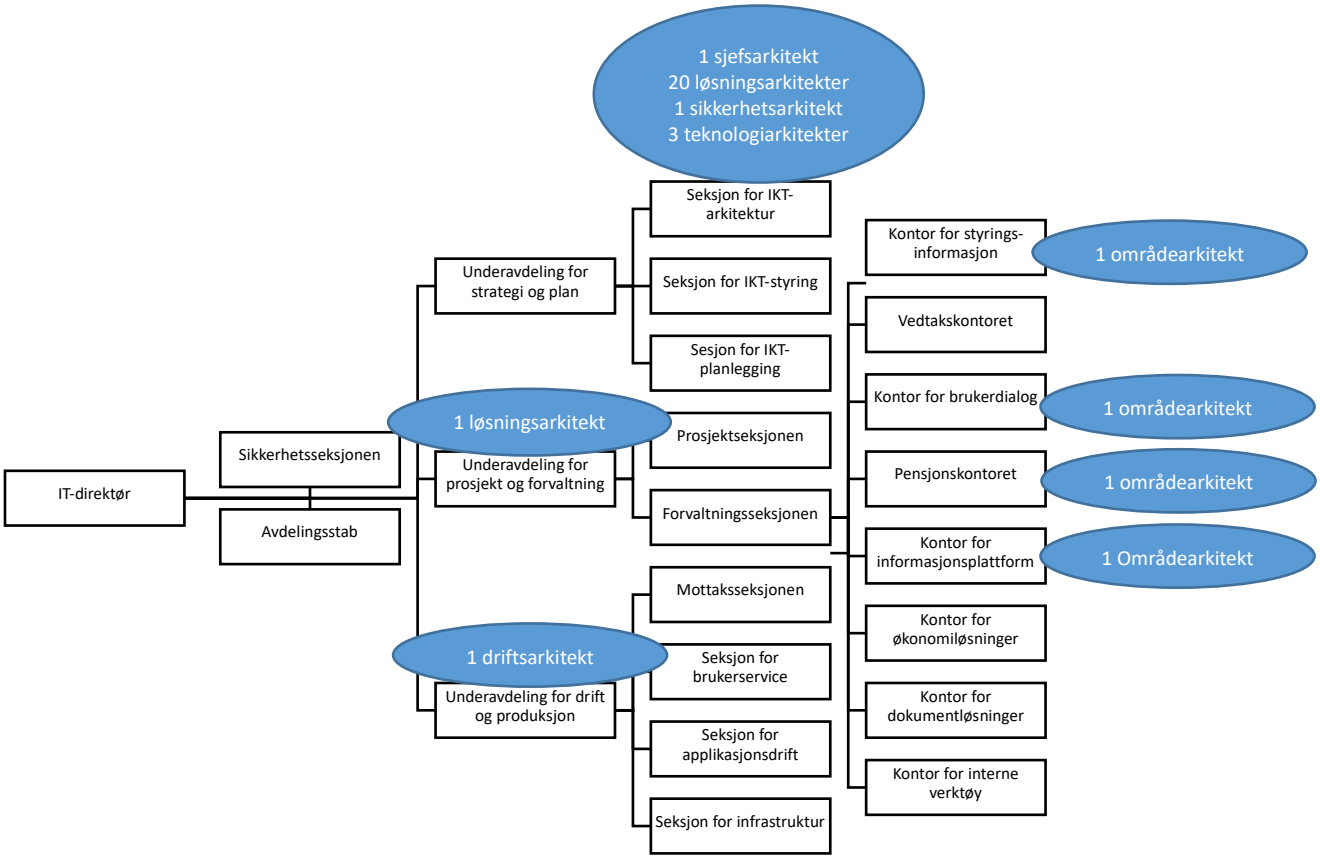


Figur 5 - Forenklet skisse av avhengigheter fra et sentralt fagsystem (fra intern wiki)

4.1 Organisering

Figur 6 viser organisasjonskartet som det så ut da undersøkelsene ble gjennomført. Ansvaret for IT-avdelingens oppgaver var fordelt på tre underavdelinger.

For å klare å levere i raskere og i større omfang enn før har det vært behov for endring av organisering og prosesser. I 2016 ble det gjennomført et OU-prosjekt i IT-avdelingen, med mål om å kunne levere IT-baserte tjenester raskere og billigere enn før. IT-avdelingen satte i gang en prosess for å rekruttere flere ansatte med teknisk kompetanse, og gradvis redusere bruken av innleide konsulenter. Kontinuerlige leveranser er et langsiktig mål.



Figur 6 - Organisasjonskart for IT-avdelingen i 2016

Ny organisasjon gjelder fra 1.1.2017. Den viktigste endringen som påvirker arkitekturprosessene er at utvikling skal utføres av tverrfaglige team, som har helhetlig ansvar for kvaliteten i de løsningene de leverer, og at alle arkitektene samles i samme seksjon. Arkitektene forventes å bidra aktivt inn i teamenes arbeid, gjennom en form for matriseorganisering. I kapittel 4.8 beskrives målsetninger som gjelder for den nye organisasjonen.

Omorganiseringen påvirker ikke ansvarsdelingen mellom IT-avdelingen og andre avdelinger i direktoratet.

4.2 Arkitekturroller

Seksjon for IKT-arkitektur har hovedansvaret for IT-arkitekturen. I gammel organisasjon var det i tillegg arkitekter tilknyttet noen av forvaltningskontorene («Områdearkitekter»), en løsningsarkitekt knyttet til Kontor for prosjektgjennomføring, og en driftsarkitekt i Underavdeling for Drift og produksjon. Figur 6 viser hvor de forskjellige arkitektene har vært plassert i organisasjonen.

Seksjon for IKT-arkitektur i IT-avdelingen har ansvaret for operasjonalisering av IT-delen av NAVs virksomhetsarkitektur. Øvrig virksomhetsarkitektur håndteres av Seksjon for Virksomhetsarkitektur, som ligger utenfor IT-avdelingen.

Prosjekter og større leveranser bemannes med en eller flere løsningsarkitekter. Disse arkitektene kan være lånt ut fra Seksjon for IKT-arkitektur, eller være innleide konsulenter.

Det meste av programvareutvikling i IT-avdelingen gjøres av innleide leverandører, som er tilknyttet forvaltningskontorene. De fleste av disse leverandørene har også arkitekter i tilknyttet sine utviklingsteam.

4.3 Arkitekturutvikling

Målbilder, arkitekturprinsipper og arkitekturkrav utvikles i regi av Seksjon for IKT-arkitektur, som har ansvar for å involvere relevante interessenter.

Dokumentasjon av eksisterende arkitektur, målbilder og krav og prinsipper er samlet i wiki`en «Arkitekturportalen».

Nye målbilder for arkitekturen utvikles etter en prosess basert på TOGAF ADM (se kapittel 2.1.1).

4.4 Arkitekturstyring

Seksjon for IKT-arkitektur kvalitetssikrer og godkjenner løsningsarkitekturen som utarbeides i prosjektene. Avvik fra målbilder og prinsipper må behandles i Arkitekturbeslutningsmøtet. Arkitekturbeslutningsmøtet behandler ofte også saker der det er stor uenighet om hva som er riktig løsning på et problem.

Beslutninger tas av Sjefsarkitekt, eller ved behov av ledere på høyere nivå i organisasjonene, og logges i en Arkitekturbeslutningslogg. Arkitekturvalg med en begrenset konsekvens kan tas i de enkelte forvaltningsmiljøene. Større prosjekter får normalt også delegert beslutningsmyndighet innenfor definerte rammer. Beslutninger som tas «lokalt» kan også dokumenteres på samme format og logges i Arkitekturbeslutningsloggen.

Beslutningene dokumenteres på et strukturert format som beskriver alternativer som er vurdert, vurderingskriterier som er benyttet, og hvem som har vært involvert.

I tillegg til dette finnes Teknologistyringsprosessen, som godkjenner nye teknologiprodukter før de kan tas i bruk. Denne prosessen er svært omfattende dersom teknologiendringen medfører anskaffelser eller endringer i lisensbruk, og mer lettvekt dersom det er mindre endringer. Teknologiendringer med begrenset konsekvens godkjennes lokalt av det miljøet som har behov for dem.

4.5 Prosjekter og leveranser

Nyutvikling og omfattende systemendringer organiseres som prosjekter. I tillegg til prosjektene kommer endringer, nyutvikling og feilrettinger av mindre omfang. Disse «pakkes sammen» og organiseres i felles leveranser, såkalte hovedleveranser, ofte forkortet til «HL».

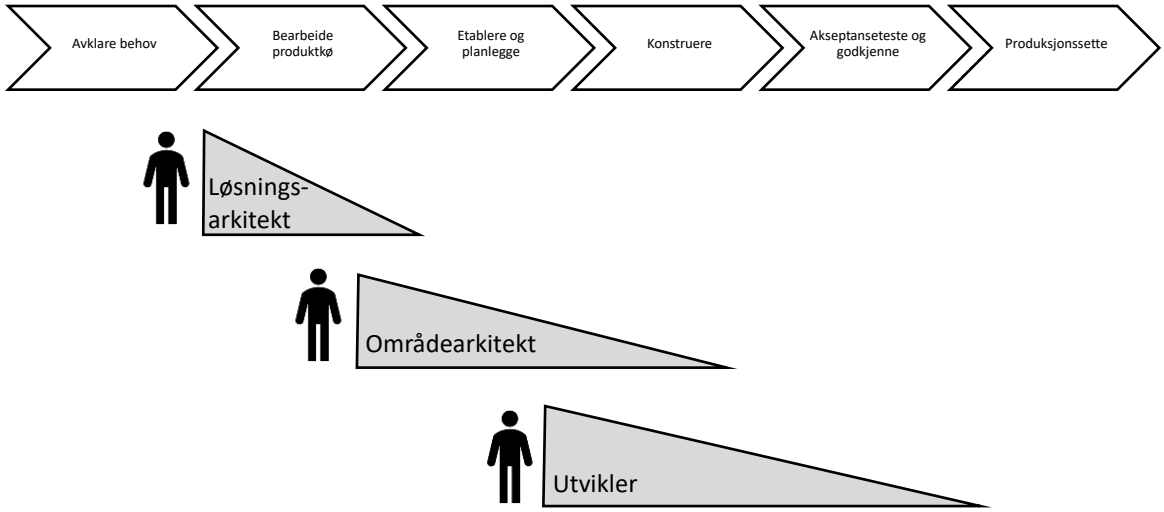
Både prosjekter og hovedleveranser baseres på leveransemetodeverket «SLEM», som er basert på prinsippene i Figur 7. SLEM er inndelt i seks delprosesser. Disse beskrives overordnet i Figur 8 og Tabell 7.

- *Forretningsverdi som viktigste kvalitetsmål*
- *Kontinuerlig prioritering av funksjonalitet ut fra kost/nytte*
- *Tett dialog mellom fagpersoner og utviklere*
- *Selvorganiserte tverrfaglige team*
- *Eliminere overflødig arbeid*
- *Korte iterasjoner med hyppige leveranser*
- *Beslutninger tas så sent som mulig (rullerende planlegging)*
- *Evaluering, læring og forbedring underveis*

Figur 7 - Førende prinsipper for SLEM

Prosjekter og andre bestillinger som fører til endringer i IT-systemene kvalitetssikres av løsningsarkitekter i Seksjon for IKT-arkitektur (i «Bearbeide produktkø»-fasen i SLEM), for å avdekke omfang og konsekvenser, og sikre at endringene gjøres i henhold til prinsipper og målbilder.

Fra «Konsekvensutredningsfasen» overtar forvaltningskontorer eller evt prosjektorganisasjon arbeidet med å definere løsning. Løsningsdesignet utarbeides i «Etablere og planlegge», i hovedsak av leverandørene. Det er disse løsningsbeskrivelsene som ligger til grunn for estimater og inngåtte avtaler med leverandørene. Arbeidet med konsekvensutredning og kvalitetssikring av løsningsbeskrivelser utføres av Områdearkitekter og Teknisk ansvarlige i



Figur 8 - Delprosesser i SLEM – når deltar forskjellige roller (forenklet)

forvaltningskontorene. De følger også opp problemstillinger som oppdages underveis i konstruksjon. Leverandørenes arkitekter og utviklere utfører løsningsbeskrivelsen, og er i varierende grad involvert i konsekvensutredningen.

Arkitekter deltar normalt ikke i akseptansetest/godkjenning eller produksjonssetting.

Tabell 7 - Beskrivelse av delprosessene i SLEM

Delprosess	**Beskrivelse av relevante aktiviteter**	**Arkitektenes bidrag**
Avklare behov	Produkteiere i fagavdelingene beskriver sine behov, i form av epos, som legges inn i en felles produktkø.	Deltar normalt ikke
Bearbeide produktkø	• Epos forbedres og tydeliggjøres i samarbeid mellom fagside og IT-avdeling. • Konsekvensutredning: Overordnet analyse av løsning og grovestimater av epos som skal inngå i en leveranse	Bidra til tydeliggjøring av behov. Delta i konsekvensutredning.
Etablere og planlegge (iterativt)	• Detaljering av brukerhistorier knyttet til prioriterte epos. • Leverandørene utarbeider løsningsbeskrivelser og estimater, som kvalitetssikres av kunden.	Kvalitetssikring av løsningsbeskrivelser (områdearkitekt).
Konstruere (iterativt)	• Utvikling og testing av kode, basert på løsningsbeskrivelser for et gitt antall brukerhistorier	Avklaringer etter behov
Akseptanseteste og godkjenne	• Kundesiden akseptansetester og godkjenner leveransen. Eventuelle feil legges tilbake i produktkøen, og tas inn i en konstruksjonssprint.	Deltar sjelden
Produksjonssette	• Løsningen settes i drift	Deltar sjelden

4.6 Utfordringer

Det tar ofte lang tid fra fagavdelingene i Arbeids- og velferdsdirektoratet identifiserer et behov som skal løses ved hjelp av IT, til løsningene er på plass. Hovedleveransene har en lang planleggingshorisont, der rammene for hva som skal leveres låses mange måneder før produksjonssetting. I store prosjekter kan tiden fra idé til leveranse være enda lengre.

Samtidig øker presset på at NAV skal løse sine oppgaver mer effektivt. Nye og forbedrede IT-løsninger er en viktig del av dette effektiviseringsarbeidet, og det er ventet at behovet for IT-utvikling vil øke i årene framover. I tillegg til dette stilles det krav om høyere grad av digitalisering av offentlige tjenester. Dagens arbeidsprosesser er presset til ytterpunktene, og det ansees som urealistisk å gjennomføre større hovedleveranser enn man gjør i dag. IT-avdelingen har gjennom ti år hatt «Stabil drift» som sin høyeste prioritet, og hatt lite fokus på hvordan endringer kan utføres raskt.

Dagens arkitekturprosesser er basert på virksomhetsarkitekturrammeverket TOGAF, som legger vekt på å jobbe grundig med analyse av forretningsbehov, før man utarbeider målbilder og migreringsplaner som kan gjennomføres i implementeringsprosjekter. For mange områder mangler disse målbildene og planene, og det er tidkrevende å få dem på plass.

Arkitekturfunksjonen, i hovedsak organisert i Seksjon for IKT-arkitektur har fokus på å etablere målbilder for prioriterte områder, og kvalitetssikring tidlig i planleggingsprosessen. På tross av at det er forholdsvis mange arkitekter ansatt i IT-avdelingen, oppleves det som at det er for lite kapasitet til å dekke dette behovet, bidra til kvalitetssikring av prosjektene og samtidig være tilgjengelige for prosjekter og forvaltningskontor som har behov for råd.

Som en del av omstillingen av IT-avdelingen er det derfor også behov for å se på hvordan man kan effektiviseres arkitekturarbeidet, og hvordan man skal prioritere hvilke oppgaver arkitektene skal utføre.

Det er også en utfordring at mange eksisterende systemer er preget av *teknisk gjeld* det vil si vedlikeholdsetterslep eller designsvakheter som gjør det krevende å gjøre endringer i systemet. IT-avdelingen har satt opp et løpende prosjekt «FKON», som finansierer nedbygging av teknisk gjeld. Det er arkitekter fra Seksjon for IT-arkitektur som prioriterer hvilke tiltak som skal få finansiering, basert på innspill fra de enkelte forvaltningskontorene.

4.7 Forsøk med mer smidig tilnærming

Parallelt med OU-prosjektet gjøres det også en del forsøk med mer smidige gjennomføringsmodeller. Prosjektet digiSYFO tester ut ny metodikk, basert på tilnærmingen *Lean Startup*[2]. Aura-teamet er et internt utviklingsteam i IT-avdelingen som er organisert som et *autonomt team*, som i større grad enn andre team har fått delegert ansvar for egne prioriteringer og gjennomføring. Flere andre prosjekter forsøker å tilpasse seg til en mer smidig tilnærming, innenfor dagens rammer.

4.8 Nye målsetninger for IT-avdelingen

I forbindelse med omorganiseringen har IT-avdelingen utviklet nye «grunnpilarer», som gjelder fra 1. januar 2017. Disse pilarene var ikke beskrevet da intervju-undersøkelsen ble gjennomført, men viser en retningsendring, som kan påvirke hindringer og muligheter for smidighet i arkitekturprosessen.

Tabell 8 - Grunnpilarer fra nye NAV IT (hentet fra NAVs intranett)

Fire grunnpilarer ligger til grunn for nye NAV IT

1. Tydeligere eierskap på strategisk viktige områder for NAV
    - Styrke og utvikle strategisk/faglig lederskap
    - Bygge intern spisskompetanse
    - Sikre at vi bruker markedet på en optimal måte
2. Innovative og digitale løsninger til det beste for brukerne
    - Økt spisskompetanse, sterkere digital forståelse
    - Partnerskap med forretning, brukerdrevet utvikling
    - Mer fleksibel organisering
3. Gjøre mer med mindre og få opp farta
    - Standardsystemer vs. egenutvikling
    - Bygge/bruke intern kompetanse og utviklingskapasitet
    - Smidig tilnærming, differensierte leveranser
4. Kontinuerlig søke etter å bli bedre
    - Flat og fleksibel organisasjon med bedre samhandling
    - Eksperimentere og risikere mer; enten så lykkes vi eller så lærer vi
    - Kompetent, ledet og brukerorientert

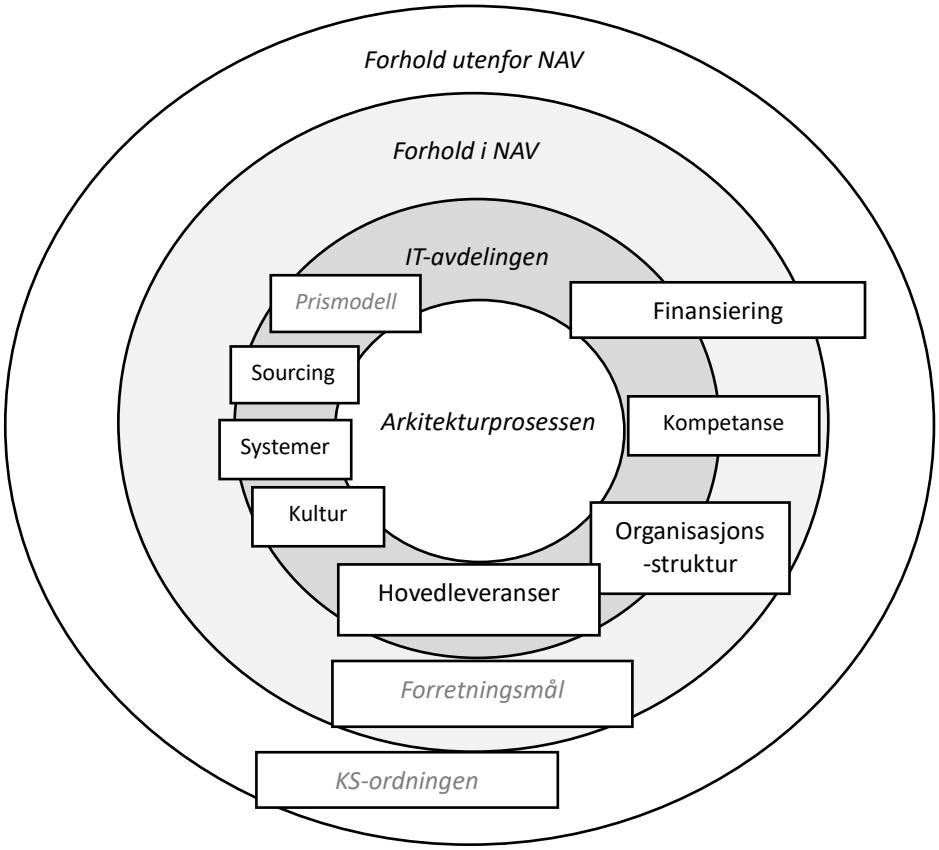
[2] [2] Iterativ metode som har som mål å få første versjon av et produkt realisert så raskt som mulig, og så hente inn læring for fortløpende forbedringer og videreutvikling (https://en.wikipedia.org/wiki/Lean_startup).

5 Funn

I dette kapittelet oppsummeres funn fra intervjuer og dokumentundersøkelse. Funnene er strukturert etter rammeverket som er beskrevet i kapittel 2.3.

5.1 Kontekst for prosessen

Informantene peker på flere forhold som er knyttet til arkitekturprosessens kontekst. I de neste avsnittene oppsummeres slike forhold, hovedsakelig internt i IT-avdelingen. Finansieringsprosesser ligger delvis utenfor IT-avdelingen. Dette behandles under temaet Organisasjon.



Figur 9 - Faktorer i arkitekturprosessens kontekst

Noen av informantene har også pekt på forhold i fagavdelingene i direktoratet (utydelige forretningsmål) og krav som stilles til Arbeids- og velferdsdirektoratet fra departementsnivå (KS-ordningen). Dette behandles ikke videre i denne oppgaven. Figur 9 illustrerer faktorer fra prosessens kontekst som er tatt opp av informantene. Tekst i kursiv er faktorer som ikke blir behandlet videre i oppgaven.

5.1.1 Organisasjon

Informantene peker på at det ofte er avstand mellom arkitektene, som er organisert i en egen seksjon, og utviklingsaktivitetene i prosjekter og forvaltning. Selv om kommunikasjonen oppleves som bedre enn tidligere, oppleves fortsatt arkitektene som for lite tilgjengelige.

- *Det er litt sånn «De der oppe i åttende, de sitter og tegner, og vet ikke hva som skjer.» Oppleves som litt for teoretisk, og ikke så «hands-on». (Informant 4)*
- *IKT-arkitektur jobber jo så tidlig i prosessen, før vi er involvert, så det er ingen som vet helt hva de driver med når vi får ballen. (Informant 6)*

Flere av informantene trekker også fram skillet mellom kunde og utviklingsleverandør som et hinder for smidighet. En arkitekt tilknyttet en utviklingsleverandør forteller at det er svært lite kontakt med Seksjon for IKT-arkitektur. Han opplever at kompetansen de sitter på ikke benyttes i arkitekturprosessen, eller at de kobles inn for sent. En annen av arkitektene er kritisk til prismodellen som benyttes, og mener den i seg selv er et hinder for smidighet.

En av arkitektene peker på at tydelig deling av ansvar mellom forskjellige forvaltningskontor er til støtte for smidighet, en annen ser på det som en utfordring at det er mange «småsjefer», og mener det gir økt behov for arkitekturstyring.

Informantene gir uttrykk for at det ønsker seg at arkitektene er mer tilstede, og involverer seg mer konkret i de enkelte områdene.

- *Jeg tenker på det at skillet mellom arkitektur i NAV og utvikling i NAV, det er et gigantisk hinder for smidig utvikling og gevinstrealisering og styring i forhold til målbilder og alt mulig. (Informant 8)*
- *Vi må være ute, ikke sitte i et hjørne og modellere. (Informant 1)*

Forhold knyttet til finansiering og budsjett nevnes som en hindring av flere av informantene. Finansiering er stort sett knyttet til prosjekter, som er drevet av faglige behov. Det kan være utfordrende å få prioritert tekniske egenskaper i løsningene.

- *Budsjettmodellen til NAV gjør at vi ikke kan jobbe smidig – vi har en pott penger i prosjektet, når det er over må vi bare forlate arkitekturen, vi har ikke penger til å forvalte den, endre forbedre, ingen veier for å få midler til det, utenom FKON. (Informant 5)*

5.1.2 Prosjektenes egenskaper

Organiseringen av programvareutvikling i såkalte «hovedleveranser» (se kapittel 4.54.3) blir nevnt som en utfordring i mange av intervjuene. Endringsprosessen i prosjektene, og spesielt i hovedleveransene, er svært omfattende, slik at hovedleveransene får en fossefallstilnærming.

Tidsfristene knyttet til hovedleveransene gjør at mange arkitekturoppgaver må løses i parallell, innenfor korte frister tidlig i prosessen. Det er også utfordringer knyttet til at endringer som i utgangspunktet ikke har avhengigheter til hverandre, får det gjennom at de organiseres inn i samme hovedleveranse.

En av arkitektene beskriver det som at man *«må til Kongen i statsråd og sende endringsanmodninger om den minste ting»,* og endringsprosessen tilpasses alvorlighetsgraden av forskjellige endringer. Samme prosess benyttes for mindre endringer underveis i prosjektet, som når man jobber med totalomfanget, noe som gir unødvendig mye arbeid.

En annen arkitekt tar opp at det er utfordrende at små prosjekter må forholde seg til samme krav planverk, rapporteringsregime, maler etc som store prosjekter, og etterlyser kontrollmekanismer som er bedre tilpasset små, smidige prosjekter.

IT-avdelingen har gjennom 2016 jobbet med å redusere bruken og omfanget av hovedleveranser. Målet er at endringer lettere kan settes i produksjon uavhengig av andre endringer.

5.1.3 Kompetanse og kultur

Informanter nevner en form for risikoaversjon i IT-avdelingen, i sammenheng med flere av temaene som er undersøkt. I deltakervalideringsmøtet ble også kontroll- og kvalitetssikringsfokus trukket tydeligere fram som en faktor som hindrer smidig gjennomføring. En påstand er at dette er en «kontrollkultur» som hindrer samarbeid og smidig tilnærming.

- *Fokus på å finne feil er en generell utfordring vi har, det er en sykdom vi har her, henger sammen med frykten for å gjøre feil. Istedenfor å tenke «hva er intensjonen bak?» tenker man «her skal jeg finne en feil, da har jeg gjort jobben min som kvalitetssikrer». (Informant 4)*

Noen av arkitektene uttrykker et ønske om å gå vekk fra en slik kontrollerende rolle, og heller bli en bidragsyter *«der det er viktig».* Flere peker også på at det er mange som jobber for å få til en mer smidig tilnærming, og som bidrar aktivt i den retning, selv om ikke alle rammebetingelsene er ideelle.

- *Det er mange som er støttende fordi man har et ønske om å bli mer smidig, og derfor prøver man å bidra til at ting skal fungere. (Informant 2)*

Noen av informantene etterlyser mer teknisk kompetanse og kunnskap om de enkelte systemene hos arkitektene og andre internt ansatte. En peker på at det finnes internt ansatte med teknisk kompetanse, men at disse ofte er opptatt med å analysere feilrettinger, ikke analysere endringer. I 2016 gjennomfører IT-avdelingen en omfattende rekrutteringskampanje for å ansette flere med teknisk profil.

To av informantene mener også at mange ansatte har en skepsis mot konsulenter, og at holdningen kan være et hinder for smidig arbeid.

Ingen av informantene i undersøkelsen hadde kjennskap til eller erfaring med lettvektsvarianter av arkitektursentriske metoder, men de fleste har erfaring fra smidige prosjekter, og noen har erfaring med å bruke smidige teknikker i arkitekturarbeidet.

5.1.4 Systemenes egenskaper

NAV har en forholdsvis omfattende systemportefølje, der det normalt er flere større prosjekter i gang samtidig, i tillegg til løpende vedlikehold og mindre endringer i systemer og infrastruktur. Dette gjør at man alltid må vurdere om endringer kan påvirke andre deler av porteføljen, og at prosjekter sjelden er *green field*[3].

[3] Utviklingsprosjekter som ikke behøver å ta hensyn til tidligere arbeid eller eksisterende systemer. (http://www.webopedia.com/TERM/G/greenfield.html)

Kompleksitet i avhengigheter mellom systemene kan være et hinder for smidig tilnærming. Det er vanskelig å isolere problemstillinger, og delegere ansvaret for løsning til mindre team. En av informantene peker på at disse avhengighetene også kan tvinge fram en «ovenfra-ned» tilnærming til hvordan man jobber med arkitektur.

- *Det er vel ikke så smidig med ovenfra ned styring, prøve å kontrollere alt – det er jo fordi vi ikke har tydelige grensesnitt mellom områdene, ikke godt nok modulært og innkapsla, da blir det orkestrering og koordinering og hierarkisk å jobbe med arkitektur. (Informant 6)*

Det er etablert et utviklingsteam, Aura-teamet, som jobber med automatisering av prosesser som oppsett av miljøer, produksjonsetting (deployment), inspirert av devops-tilnærming. Disse løsningene oppleves som en verdifull støtte for smidig arbeid i de systemene som kan nyttiggjøre seg av dem.

5.2 Arkitekturprosessen

Leveransemetoden SLEM beskriver blant annet hvilke vurderinger knyttet til arkitektur og design som skal gjøres i hvert steg av prosessen, og hvem som skal delta (se mer detaljer i kapittel 4.5). Metoden tar utgangspunkt i smidige prinsipper. Leveranseprosessen og arkitekturarbeidet som utføres i den oppleves likevel ikke som smidig av informantene.

5.2.1 Målsetninger for prosessen

- *God arkitektur forenkler. Man lager ikke arkitektur for at det skal bli vanskeligere. (Informant 1)*

En av informantene mener arkitektene fyller en rolle som «portvakt», som sikrer at de store problemstillingene er tatt ned før prosjektet starter, og at dette gir handlingsrom til smidighet i prosjektene. Men flere peker på at alle endringer blir utredet etter samme lest, også når de ikke har konsekvenser for arkitekturen.

Utredninger og kvalitetssikring som utføres av Seksjon for IKT-arkitektur oppleves noen ganger som forsinkende på framdriften for prosjekter og leveranser, og i en del tilfeller som for fokusert på detaljer og formaliteter. En utfordring er at man i gjennomføringen i for liten grad differensierer tilnærmingen basert på problemene som skal løses, f.eks. basert på kompleksitet eller risiko.

- *Skal det gjøres store endringer som kan føre til at brukerne ikke får penger, da må vi ha veldig god kontroll, men på andre områder kunne vi sluppet opp i større grad. (Informant 4)*

Det kan være utfordrende for arkitektene å balansere ønsket om kontroll og risikostyring på den ene siden, opp mot ønsket om smidighet:

- *Hvordan får man det til sammen, de gode målbildene og retningslinjene og prinsippene som gjør at man tør slippe litt kontroll, spesielt for arkitekturseksjonen. Det blir alltid «Om vi bare hadde kommet litt lenger, da kunne vi fått det perfekte systemet…». Man må tørre å gjøre det beste ut fra det man har her og nå, og tørre å ta beslutninger ut fra det man vet her og nå. (Informant 2)*

Noen av informantene peker på at langsiktige behov, som å legge til rette for vedlikeholdbarhet og endringsevne, prioriteres ned i prosjektene til fordel for funksjonalitet som er ønsket av fagsiden. Dette kan resulterer i systemer som er lite smidige. Det er også lite midler tilgjengelig for å implementere og vedlikeholde en god arkitektur utenfor prosjektene.

Flere av informantene mener at systemenes framtidige endringsevne i større grad må prioriteres inn som en del av de faglig drevne prosjektene. En informant peker også på at det kan være utfordrende å få tillatelse til å prøve ut nye tilnærminger:

- *Man baserer seg på gamle sannheter […], man er for engstelige for risiko, at man ikke tør å prøve ting som ikke er prøvd før. Det er et ganske alvorlig sykdomstegn. (Informant 8)*

5.2.2 Analyse, implementasjon og vedlikehold

Noen av arkitektene fra Seksjon for IKT-arkitektur forsøker å bli mer tilstedeværende i prosessen, og legge til rette for samarbeid mellom de forskjellige miljøene:

- *Jeg tenker at det at man er med og gjør workshoper med [forvaltningskontor], med leverandører og kontoret og KES[4] og vi, og diskuterer hvordan ting skal være, det tenker jeg er smidighet. Jeg tror på å samle forskjellige roller og funksjoner på tvers i en tidlig fase, for å få noe av det samme tankegodset, da blir arkitekturen en samarbeidssak. Der synes jeg vi er på vei til noe. (Informant 1)*

[4] Kontor for elektronisk samhandling, som forvalter infrastruktur for integrasjon og sikkerhet.

Våren 2016 ble det gjort justeringer i SLEM. De enkelte analyse-teamene er ansvarlige for å utrede sakene, og skal hente inn arkitekter når det er behov for deres kompetanse. Løsningsarkitekten fra Seksjon for IKT-arkitektur framhever at det er viktig å vise tillit til de enkelte kontorene som forvalter løsningene, og fokusere tidsbruken til seksjonen på «de store, vanskelige tingene». En av prosjektarkitektene trekker også fram denne endringen som positiv.

- *Noen forvaltningskontorer synes vel bare arkitektene lager støy, mener de har full kontroll selv, samtidig som mange av dem har et veldig snevert syn, ser sin egen silo, lite fokus på det som skjer utenfor. Da spiller jo arkitektur en viktig rolle. (Informant 4)*

Noen av informantene nevner metodeverket SLEM er til hinder for smidighet, andre mener det er en støtte. Flere av informantene beskriver en praksis som ikke er i tråd med metodeverket.

I følge SLEM skal konsekvensutredningen av et behov utføres av et tverrfaglig analyse-team. Produkteieren, en arkitekt og representant for utviklingsteam(ene) som skal utvikle løsningen skal alltid være med i analyse-teamet.

Flere av svarene tyder på at praksis varierer, og avviker fra den dokumenterte metoden i en del tilfeller. I en del tilfeller deler deltakerne i et analyse-team oppgavene i analysen mellom seg, og løser dem hver for seg. Analyseteamkoordinatoren forsøker så å pusle sammen det «store bildet» av de forskjellige delene. Det er da ikke synlig for de andre deltakerne hvorfor valgene er tatt, og hvordan de for eksempel har vektlagt kvalitetskrav.

En annen av løsningsarkitektene peker på at hvis oppgaver skal delegeres, er det bedre å sette seg sammen og bli enige om det store bildet først, så man har sikkerhet for at man jobber etter samme mål. En slik tilnærming samsvarer bedre med den dokumenterte prosessen.

Det skal etableres en løsningsskisse i konsekvensutredningen, men ikke mer detaljert enn det som skal til for å grovestimere omfanget av utviklingsarbeidet. Estimatet brukes som en del av grunnlaget for prioritering av produktkøen, men er ikke bindende.

I noen tilfeller behandles likevel den tidlige løsningsskissen og grovestimatet som bindende. Det kan føre til at omfanget av arbeidet i tidlig fase blir mer omfattende, og at beslutninger tvinges fram tidlig. Leverandørarkitekten peker på at dette gir en type ond sirkel, der man gjør

mer og mer arbeid med konsekvensutredning grovestimater, for å forhindre å «bli tatt» hvis det blir behov for endringer senere, eller hvis de detaljerte estimatene som gjøres senere i prosessen blir høyere enn grovestimatene.

En av løsningsarkitektene peker også på at han tidligere har fått «smekk på labben» for å ikke følge detaljene i metoden nøye nok. Det gjorde at prosjektet etterpå prøvde å bli «flinkest i klassen», også på områder der de selv ikke opplevde at metoden og rutinene tilførte verdi.

Arkitekturmetoden i tilknytning til SLEM åpner for at arkitekter kan delta inn i konstruksjon, og krever også at de trekkes inn dersom arkitekturspørsmål dukker opp underveis. Dette praktiseres i liten grad. Det blir bare unntaksvis undersøkt om den implementerte løsningen er i henhold til løsningsskissene fra tidligere faser. Arkitektene oppgir tidspress som årsaken til at de ikke klarer å følge sakene gjennom utvikling og til produksjon, og flere av dem uttrykker et ønske om å kunne følge opp bedre i senere faser.

Selv om det er en del hindringer for smidig tilnærming i dagens praksis, oppleves det også at den gir støtte på noen områder. En av arkitektene trekker fram at det er en fordel at et felles metoderammeverk gir felles forståelse av hvordan ting skal løses, og at dette sammen med definerte ansvarsområder skaper forutsigbarhet.

I oppsummeringen av funn sammen med informantene (Vedlegg 4), ble det diskutert om metoden var godt nok kommunisert og forstått i IT-avdelingen. En utfordring kan være at metodekursene fokuserer på detaljer, istedenfor å formidle intensjon om mål bak metoden, og at de smidige prinsippene i SLEM derfor ikke er tilstrekkelig forstått.

Det gjøres også forsøk med andre metodetilnærminger i IT-avdelingen. Et prosjekt baserer seg på *Lean Startup*. I dette prosjektet er arkitekten en integrert del av det tverrfaglige teamet, og kan følge opp designet gjennom å delta i de daglige standup-møtene.

- *Her er det hele tiden å gå opp, hva var det du mente, hvis vi endrer dette, blir det bedre eller dårligere, hvordan skal det bli, hvordan skal det virke […] Det er kjempegøy å få lov til å jobbe i denne typen prosjekt, så jeg håper jo uansett om vi lykkes at vi kan ta med de gode erfaringene og dra de med videre. (Informant 2)*

Aura-teamet (som jobber med automatisering og devops) jobber etter mønster av «autonome team», og prioriterer fortløpende oppgaver fra produktkøen, basert på ønsker fra andre utviklingsteam. De jobber kontinuerlig med å forbedre sin egen metode.

- *Det er veldig gøy i Aura-teamet nå, vi er smidige med egen metode, hele tiden i bevegelse, under stadig optimalisering. Ingen artefakter eller seremonier som er hellige, alt er gjenstand for diskusjon, hvis det ikke anses å gi verdi, dropper man det. […] Man må være smidig med egen prosess. Det er samme hva prosessen er, bare du har en prosess for å endre den. (Informant 8)*

Utvikleren trekker også fram at autonome team som får ansvar for sin egen applikasjonsarkitektur, også tvinges til å ta mer ansvar, gjennom at de «kjenner smerten» selv når applikasjonsarkitekturen er feil.

I tillegg til de smidige forsøksprosjektene, er det andre som omtaler at de har tatt i bruk smidige teknikker som en del av arkitekturarbeidet. Et eksempel er en områdearkitekt som forteller at de har tatt i bruk en kanban-tavle for tekniske problemstillinger, der man fortløpende prioriterer hvilke oppgaver som påbegynnes.

En annen løsningsarkitekt har erfaring med at tidspress i noen situasjoner har presset fram en mer smidig tilnærming, gjennom at man ikke har noe annet valg enn å sette de formelle prosessene og overleveringene til side. I slike «kriser» setter man ned tverrfaglige team som jobber sammen med å løse problemstillingene, setter skillet mellom kunde og leverandør til side, og tar i bruk praksiser som daglig deployment.

Seksjon for IKT-arkitektur baserer seg i dag på «tungvekts»-metoder som TOGAF ADM og ATAM. Noen av informantene peker på at TOGAF er tungvint å anvende i praksis, og en at det hadde vært nyttig å finne fram til enklere tilnærming. En av arkitektene peker også på at modelleringsverktøy som i utgangspunktet er beregnet på virksomhetsarkitektur er tungvint i bruk i designprosessen.

5.3 Arkitekturbeslutninger

En del av informantene opplever at de har innflytelse på beslutninger, og at prosessen fungerer godt. De mener det er nyttig at man dokumenterer hvilke alternativer som har vært

vurdert, og hva man ble enige om, særlig i tilfeller der det har vært uenighet om hva som er riktig løsning.

Men flere peker også på at for mange beslutninger tas på for høyt nivå. Utviklingsteam involveres sjelden i beslutninger tidlig i arkitekturprosessen. Flere gir uttrykk for at de er i stand til å ta flere beslutninger selv, enn det gis rom for.

- *Som leder kan man si at man vil være smidig, men når det kommer til å måtte delegere beslutningsmyndighet er det vanskeligere […] Det er lett å mene noe om prinsipper, men når det treffer den enkelte blir det brått mye vanskeligere. (Informant 2)*

Det kan være utfordrende for et smidig prosjekt å få beslutninger raskt nok. Det er mange interessenter som skal involveres, og noen ganger må beslutningen løftes til et høyere nivå i organisasjonen. Behandling i arkitekturbeslutningsmøtet gjør at dokumentasjonen og fremstillingen av beslutningsgrunnlaget også må ta høyde for flere mottakere med varierende grad av innsikt i problemstillingen. Dette gjør at behovet for dokumentasjon og forberedelser blir mer omfattende enn ved en mer «lokal» beslutning.

En av prosjektarkitektene trekker fram at det oppleves som at «prosessen føles langt unna», og at det virker for omfattende når det man ønsker seg er en rask bekreftelse på et valg som er tatt, eller bare å kunne logge en beslutning.

Den dokumenterte metoden som benyttes for arkitekturbeslutninger setter ikke krav til hvilket nivå i organisasjonen beslutninger skal tas på, men peker på at dette må avgjøres i hvert enkelt tilfelle. Arkitektene kan også innta en rolle som tilrettelegger, istedenfor å gjennom føre analysen av beslutningsgrunnlaget alene:

- *Jeg tenker at det at man er med og gjør workshoper med [forvaltningskontor], med leverandører og kontoret og [integrasjonsarkitekter] og vi, og diskuterer hvordan ting skal være, det tenker jeg er smidighet. Jeg tror på å samle forskjellige roller og funksjoner på tvers i en tidlig fase, for å få det samme tankegodset, da blir arkitekturen en samarbeidssak. Der synes jeg vi er på vei til noe. (Informant 1)*

5.4 Arkitekturbeskrivelse

- *Mye arkitekturdokumentasjon gjør man jo for å kommunisere et mål eller en beslutning, så det må være egnet til å kommunisere, og det har man glemt litt. (Informant 8)*

Det finnes store mengder dokumentasjon av arkitektur og systemer i IT-avdelingens wiki, men intervjuene viser at dokumentasjonen som finnes ofte ikke dekker informasjonsbehovene informantene faktisk har. Mengden med informasjon kan i seg selv være en utfordring, noen peker også på at man ikke kan være helt sikker på om informasjonen finnes eller ikke, siden det er vanskelig å finne fram.

Dokumentasjon finnes både i form av prosa, formelle modeller basert på standarder som UML og Archimate, og som skisser eller figurer. En av informantene peker på at utviklerne ikke alltid er kjent med notasjonen arkitektene bruker i sine modeller.

For flere av områdene er det også en utfordring at målbildene og strategier for systemene ikke er dokumentert, slik at diskusjoner om retningsvalg må tas ad-hoc. Arkitekten fra Seksjon for IKT-styring trekker fram at det er utfordrende å få godkjent målbilder og andre førende dokumenter, noe som igjen gir arkitektene færre verktøy som kan brukes i arbeidet.

Design som tas fram som en del av konsekvensanalysen oppleves i en del tilfeller som for abstrakt, og en av arkitektene peker på utfordringer med å «oversette» beskrivelser på et logisk, abstrakt nivå til det praktiske.

I andre tilfeller blir arkitekturskissene for detaljerte, og gjør det vanskelig å de store sammenhengene. Når deltakere i analyseteam jobber hver for seg, blir det også en utfordring at man ikke har en felles tilnærming til hvordan man dokumenterer, slik at hver skisse må tolkes på forskjellig måte. I noen tilfeller mangler også relevant informasjon.

Noen av løsningsarkitektene opplever kravene til hva som skal etableres av arkitekturdokumentasjon i et prosjekt er uklare. På noen områder er det etablert maler eller retningslinjer, men de er kan være for komplekse for oppgaven som skal løses.

- *Vi lager en komplisert mal, som har med seg alt mulig kult, skremmer vettet av de som skal gjøre noe på et lite område, når malen bare egner seg for det mest avanserte vi skal gjøre. (Informant 1)*

Flere peker også på at det ikke er tydelig skille mellom målbilder, system dokumentasjon, og modeller og skisser som lages som en del av designprosessen. Det er i en del tilfeller også uklart hvem som er målgruppen for dokumentasjonen.

- *Vi bør ha litt mer fokus på hvordan dokumentasjonen inngår i samhandling – hvordan brukes den i forskjellig kontekst. Det man diskuterer rundt det kaster man, dokumentasjonen den må vi ta vare på, siden det kommer folk seinere som ikke kan spørre. (Informant 3)*

Informantene trekker også fram noen eksempler på arkitekturdokumentasjon som er nyttige i arbeidet, og som de bruker aktivt.

- *Jeg bruker IKT-prinsipper, kvalitetskrav, retningslinjer og målbilder – ved at man har noe som er en metode eller en retningslinje eller målbildet, det blir faktisk godt mottatt, og man slipper en del diskusjoner rundt det man skal jobbe med, i stedet for at man diskuterer alt. Det gjør det enklere å være arkitekt, og gjør ting mer helhetlig. (Informant 1)*

Utvikleren forteller om erfaringer med å bruke prototyping i form av kode som en form for designdokumentasjon.

- *[…] veldig for å prototype ting, skisser er også kode, å skrive kode, så fort som mulig egentlig. Den koden man skriver da er bare en slags dokumentasjon, braindump av det man tenker der og da, ikke nødvendigvis noe som skal bevares videre, bare en annen måte å tenke på, som kan fungere bra i å kommunisere hensikt. En konkret og enklere måte å få feedback på. (Informant 8)*

5.5 Arkitekturforståelse

- *Jeg har lite sans for overleveringer, «her er det jeg har gjort, nå er det din tur». Det er vannfall, hviskeleken. Vi kan ikke leke hviskeleken, vi må jobbe med ting sammen. (Informant 1)*

Arkitektene deltar sjelden i arbeidet etter at bestillingene er konsekvensvurdert, og oppleves ofte som lite tilgjengelige av andre. De har sjelden kontakt med utviklere. Det er ofte lite muntlig kommunikasjon i forbindelse med arkitekturprosessen. Konsekvensen kan være både informasjonstap og misforståelser.

- *Det er en forståelse av målbildet som blir borte – intensjonen er veldig ofte god, men blir ikke med videre […]. En ser bare på designet, og glemmer hvilke egenskaper man forsøker å oppnå gjennom arkitekturen, det blir borte på veien. (Informant 5)*

- *God arkitektur innebærer god kommunikasjon. Det handler veldig mye om formidling, formidle mellom ulike grupper mennesker, forstå, omsette, hvis man ikke greier å håndtere det, da mister man mye på veien. Det hjelper ikke med perfekte modeller hvis de som skal bruke dem ikke forstår dem. (Informant 2)*

Utvikleren peker på at slik dokumentasjon ofte utarbeides i en notasjon (Archimate) som utviklerne, som er en viktig målgruppe, ikke er kjent med.

Løsningsarkitekten fra Seksjon fra IKT-arkitektur ønsker seg også at arkitekturvalg i større grad blir et resultat av samarbeid og kommunikasjon mellom flere interessenter

- *Arkitekturvalg må eies av de som skal utvikle, ikke måtte lese seg til det. Det hjelper ikke bare å komme inn i et møte og si hvordan ting skal være, de må være med. Teamet må ha eierskap. (Informant 1)*

Det er et ønske om å være mer tilgjengelige og deltakende i praktisk arbeid hos samtlige arkitekter som ble intervjuet. Flere peker også på at rollen til arkitektene i Seksjon for IKT-arkitektur bør være å ha overblikk, ikke detaljkunnskap, og at mer ansvar bør delegeres.

I noen tilfeller inntar arkitektene en mer fasiliterende og mindre kontrollerende rolle.

- *Viktig å være ute, og være ydmyk for den kompetansen som sitter på detaljnivå i kontor og prosjekter. Samtidig som man vet at man har med seg en del kunnskap selv, om helheten. Men i møtepunktet, det er der mulighetene skjer. Jeg lærer noe hver gang, samtidig som jeg lærer dem noe, så kommer vi fram til nye muligheter. (Informant 1)*

5.6 Andre arkitekturaktiviteter

5.6.1 Analyse av påvirkning

De fleste utviklingsjobber i NAV innebærer endringer i eksisterende systemer. *Analyse av påvirkning* utgjør vesentlig del av konsekvensutredningene som gjøres som en del av SLEM-prosessen. Noen systemer har mangelfull systemdokumentasjonen, med den følge at hver analyse av en endring begynner med å tegne opp hva som finnes fra før. I praksis er det ofte bare leverandørene som har god nok oversikt til å beskrive systemene, og mangelen på dokumentasjon gir derfor en avhengighet til leverandør hver gang påvirkning skal analyseres.

- *Det flyter litt på leverandørens premisser, uten at vi har kontroll selv. (Informant 6)*

5.6.2 Arkitekturgjenbruk

Leverandørarkitekten mener at overdrevet fokus på standardisering og rammeverk gir usmidige systemer og økte kostnader.

- *[…] noen ganger legger man for harde føringer, da blir det suboptimale løsninger – man tror man reduserer risiko, f.eks. ved å standardisere rammeverk […] Og det er så stort spenn i løsninger i huset – forutsetninger – brukskvalitet, oppetid, interne, eksterne, det gjøres ikke forskjell på om det er mot bruker, inn mot fagsystem – gir for stramme rammer. (Informant 5)*

Samtidig kan standardisering av repeterende utfordringer gjøre at det frigjøres mer tid til å jobbe med å finne løsninger på forretningsfunksjonalitet:

- *For eksempel på tjenesteintegrasjon, da løser du mange krav når du bare bruker det som er satt opp. Det å kunne peke på en metode eller et prinsipp for hvordan du løser ting. Da får man smidig utvikling. Tar involveringen på forhånd. Standardisere det som skal gjentas, jobbe smidig med det funksjonelle. (Informant 3)*

5.6.3 Arkitekturgjenvinning

- *Det skjer jo fort at man etablerer en applikasjonsarkitektur, men hvordan den faktisk implementeres, det er sjelden en-til-en, og sjelden med feedback loop for å oppdatere arkitekturen. Da er det bare leverandørene som vet hvordan det egentlig ser ut. (Informant 8)*

Design som utarbeides i forkant av utvikling blir ofte ikke oppdatert basert på den faktiske løsningen. Dette gir en usikkerhet knyttet til om modeller i dokumentasjonen avspeiler det faktiske systemet eller ikke. Det er tidkrevende å forsøke å holde dokumentasjon av den faktiske nå-situasjonen oppdatert, og flere av informantene på at man bør vurdere å ta i bruk verktøy som genererer dokumentasjon basert på kode-repositories og andre produksjonsnære kilder som «beskriver sannheten».

5.6.4 Arkitektur-refaktorering

Det jobber ofte lenge med analyser på teoretisk nivå før man setter i gang med koding. Hensikten med analysene er gjerne å redusere risiko. Noen av informantene mener en mer

iterativ tilnærming gjennom koding og løpende refaktorering ville være er egnet til å redusere risiko.

- *NAV er ekstremt opptatt av å redusere risiko. Alt handler om å ha lite risiko i det man gjør. […] man driver jo med så mye i forkant for å ta ned risiko, men man hadde tatt ned vel så mye ved å sette i gang utvikling og vist og korrigert, jobbet iterativt, fem-seks stykker, kunne landa mange problemstillinger som hadde vært konkrete i stedet for abstrakte. Tror intensjonen er å ta den [risiko] ned, men ikke at man klarer det (Informant 8)*
- *Skal man jobbe smidig arkitekturmessig, må man jo ta risiko, produserer kanskje litt for lite [arkitektur] noen ganger, men det må man rette opp, det er smidighet. (Informant 1)*

5.7 Oppsummering av funn

Funnene fra intervjuer og dokumentundersøkelser er oppsummert i Tabell 9. En oversikt over hindringer og muligheter knyttet til hvert tema finnes i Vedlegg 5.

Dagens arkitekturprosess oppleves ikke som smidig i praksis, selv om de dokumenterte prosessene er basert på smidige prinsipper. Prosessen sikrer heller ikke i stor nok grad framtidig smidighet i systemene.

En utfordring som nevnes av de fleste informantene er hovedleveransene, som samordner produksjonssetting fra flere prosjekter og forvaltningsmiljøer. De forskjellige delene av leveransen er ikke nødvendigvis avhengige av hverandre i utgangspunktet, men organiseringen gir avhengigheter, og standardiserte krav og tidsfrister gir vanskelige rammebetingelser for smidig tilnærming.

Flere av informantene peker også på at arkitektene kvalitetssikrer alle endringer som skal utføres etter samme lest. Det gjør at arkitektene noen ganger oppleves som en flaskehals. Noen bruker ordet «kontrollkultur», at det er en risikoaversjon i IT-avdelingen som fører til mye bruk av sjekklister og standardiserte kvalitetskontroll, som ikke nødvendigvis tilfører verdi.

Det etterlyses også mer direkte kommunikasjon med arkitektene. Forvaltningskontor og leverandører opplever i mange tilfeller at deres kompetanse om systemene ikke blir brukt, og at beslutninger tas over hodet på dem. Det er også vanskelig å sette seg inn i tanken bak målbilder eller beslutninger når de utelukkende kommuniseres skriftlig.

Tabell 9 - Oppsummering av funn

Tema	Funn
Kontekst	**Konteksten inneholder både muligheter og hindringer for en mer smidig tilnærming**
Organisasjon	• IT-avdelingens organisering gir uønskede skiller mellom grupper som bør samarbeide. • IT-avdelingen er avhengig av finansiering fra fagavdelingene, for å kunne ivareta viktige ikke-funksjonelle krav, som framtidig smidighet i løsningene. • Organisasjonsstrukturen endres fra 1.1.2017
Prosjektenes egenskaper	• Når uavhengige endringer organiseres inn i samme leveranse, reduserer det muligheten til å jobbe smidig med de enkelte endringene. • Samme formelle krav stilles til små som til store prosjekter.
Kultur og kompetanse	• Risikoaversjon og «kontrollkultur» i IT-avdelingen påvirker arkitekturprosessen. Arkitektene ønsker å være mindre «politi». • Det etterlyses mer teknisk kompetanse hos arkitektene og andre ansatte. • Mange av informantene positive erfaringer med mer smidig tilnærming og bruk av smidige teknikker. • Ingen av informantene har kjennskap til arkitektursentriske lettvektsprosesser.
Systemer og infrastruktur	• Avhengigheter mellom systemene hindrer smidig tilnærming. • Det jobbes med å redusere teknisk gjeld, men ikke nok. • Automatisering av test, utrulling og miljøoppsett støtter smidig tilnærming.
Arkitekturprosessen	**Informantene opplever i hovedsak ikke arkitekturarbeidet som smidig.** • Framtidig smidighet i systemene vektlegges ikke nok i prosjektene. • Innsats differensieres ikke basert på risiko, endringer er tidkrevende. • Smidige prinsipper og mål bak metoden ikke er godt nok forstått i organisasjonen.
Arkitektur-beslutninger	**Beslutningsprosessen oppleves som nyttig i en del sammenhenger, men utviklere blir sjelden involvert.** • Beslutningsprosessen oppleves som nyttig, men kan være tidkrevende. • Mange beslutninger tas på høyt nivå, uten involvering av utviklingsteam. • Arkitekturbeslutningsprosessen legger til rette for delegerte og lokale beslutninger, men dette er ikke godt kjent.
Arkitektur-beskrivelse	**Det mangler rammer i form av målbilder, samtidig som designdokumentasjon oppleves som unødvendig detaljert.** • Mange steder mangler målbilder og tilsvarende førende dokumenter. • I mange tilfeller oppleves designdokumentasjonen som tas fram i konsekvensutrednings-aktiviteten som for detaljert. • Det er uklare krav til hvordan man dokumenterer arkitektur og design. • Arkitekturdokumentasjonen har noen ganger en form som ikke forstås av utviklerne. • Det er utfordrende å vedlikeholde dokumentasjon av den realiserte arkitekturen.
Arkitekturforståelse	**Arkitekturen formidles ofte skriftlig, med lite muntlig kommunikasjon.** • Overleveringer av oppgaver og dokumentasjon skjer ofte uten muntlig kommunikasjon. • Intensjonen bak arkitekturen går ofte tapt på veien fram til utvikler. • Informantene ønsker seg mer direkte kommunikasjon med arkitektene.
Andre aktiviteter	**Informantene ser både hindringer og muligheter knyttet til andre arkitekturaktiviteter.** • Manglende systemdokumentasjon gjør *Analyse av påvirkning* krevende. • Noen av informantene peker på at dokumentasjon kan genereres ved hjelp av verktøy • Noen av informantene mener mer *Arkitektur-refaktorering* vil kunne bidra til å redusere risiko • *Arkitekturgjenbruk* av IT-infrastruktur som integrasjon og sikkerhet oppleves som en støtte for smidig tilnærming. Krav om gjenbruk av uegnet funksjonalitet oppleves som et hinder

6 Diskusjon

Oppgaven skal besvare to forskningsspørsmål:

1. Kan forskningen peke på smidige arkitekturprosesser og -teknikker som understøtter hurtige leveranser av IT-løsninger i store organisasjoner?
2. Hvilke faktorer i IT-avdelingen er til hinder eller gir muligheter for en mer smidig arkitekturprosess?

Den empiriske forskningen om temaet er stort sett orientert rundt arkitekturprosessene i enkeltprosjekter eller knyttet til enkeltprodukter, og tilbyr *lessons learned*, men få empiriske bevis.

Det finnes flere modeller for arkitekturarbeid i den normative litteraturen. TOGAF ADM (2009) som er i bruk i NAV i dag beskriver en modell for arkitekturarbeid i store organisasjoner, men er lite smidig i sin tilnærming. Bloomberg (2013) beskriver hvilke egenskaper metode og prinsipper må ha, men er lite konkret i hvordan en organisasjon kan konkretisere dette. Leffingwell (2011) beskriver en normativ modell for smidig arkitekturarbeid i store organisasjoner, men forutsetter et handlingsrom for prioritering av arkitekturutvikling som er mer omfattende enn det IT-avdelingen har.

Litteraturen gir altså ingen verifisert «blueprint» som dekker summen av problemstillinger som må løses i en stor organisasjon som skal bevege seg i en mer smidig retning. Men det er stort sammenfall mellom utfordringer beskrevet og søkes løst i litteraturen, og de som beskrives av informantene i undersøkelsen. Det er derfor grunn til å tro at normativ litteratur og «lessons learned» fra empirisk forskning kan brukes som utgangspunkt for forbedringer av arkitekturprosessen i IT-avdelingen.

Tabell 10 viser en forenklet oppsummering av forventet og empirisk mønster, som er beskrevet i henholdsvis kapittel 2.3 og kapittel 5.7, og i hvilket av de følgende delkapitlene hvert enkelt tema drøftes. Diskusjonen oppsummeres i form av anbefalinger i kapittel 6.7. Flere detaljer finnes i en utvidet tabell i Vedlegg 6.

Tabell 10 - Forenklet oppsummering av forventet og empirisk mønster

Dimensjon	Mønster fra litteratur	Funn	Kap
Kontekst	Prosessens kontekst er avgjørende for smidighet.	Konteksten gir både muligheter og hindringer for en mer smidig tilnærming	6.1
Arkitekturprosess	Arkitekturarbeidet bør konsentreres om risiko-områder. En smidig prosess må resultere i et smidig system.	Innsatsen differensieres ikke basert på risiko. Informantene opplever ikke arkitekturprosessen som smidig, og den ivaretar ikke godt nok framtidig smidighet i systemene.	6.2
Arkitektur-beslutninger	Arkitekturbeslutninger bør tas trinnvis, og involvere viktige interessenter.	Beslutningsprosessen oppleves som nyttig i en del sammenhenger, men utviklere blir sjelden involvert.	6.3
Arkitektur-beskrivelse	Arkitekter skal ikke detaljstyre design, men definere rammer og handlingsrom for utviklingsteamene.	Det mangler rammer i form av målbilder, samtidig som designdokumentasjon oppleves som unødvendig detaljert.	6.4
Arkitektur-forståelse	Direkte muntlig kommunikasjon mellom interessenter er vesentlig for arkitekturforståelse.	Arkitekturen formidles ofte skriftlig, med lite muntlig kommunikasjon.	6.5
Andre aktiviteter	Det er lite empiri knyttet til andre arkitekturaktiviteter.	Funnene viser både hindringer og muligheter knyttet til andre arkitekturaktivteter.	6.6

6.1 Kontekst for prosessen

Kruchten (2013) peker på at kontekst er vesentlig for å kunne vurdere hvorvidt en smidig tilnærming er egnet for et prosjekt, med eller uten tilpasning. Funnene viser at forhold i omgivelsene oppleves som hindringer for en mer smidig praksis i arkitekturprosessen. Informantene har pekt på forhold både i og utenfor IT-avdelingen, men drøftingen avgrenses til forhold internt i IT-avdelingen.

Flere av rammebetingelsene i IT-avdelingen er i endring. Endringene er tilsynelatende til fordel for en mer smidig tilnærming, og kan gi handlingsrom for å gjennomføre endringer i arkitekturprosessen. I de neste delkapitlene drøftes disse rammebetingelsene nærmere.

6.1.1 Organisasjon

Flere av informantene peker på at forhold knyttet til organisering har ført til skiller mellom arkitekter og utviklingsteam, som er til hinder for dialog og samarbeid.

I ny organisering gjeldende fra 1. januar 2017 organiseres utviklingen rundt team, og det forutsettes at arkitektene skal samarbeide tett med disse teamene. Selvorganiserende team kan innebære mindre behov for godkjenning og kvalitetssikring fra arkitekter «utenfra». Det kan føre til at arkitektenes rolle tydeligere avgrenses til å definere rammer, og se sammenhenger som går på tvers av de forskjellige produktområdene.

Ideen om produktteam som har ansvar og myndighet for å løse oppgaver knyttet til en applikasjon er sentral i smidig metodikk, og organiseringen kan gi bedre forutsetninger for en smidig prosess.

En utfordring kan være om arkitektene har riktig kompetanse og vilje til å gå inn i den mer deltakende rollen som er forventet av ledelsen. Dette er drøftet nærmere under 6.1.3 Kompetanse og kultur.

Organisasjonsendringene medfører ingen endring av ansvarsforholdet mellom IT-avdelingen og fagavdelingene som eier systemene, eller i finansieringsmodeller. Utfordringene med at det kan være vanskelig å få finansiert systemforbedringer og å forebygge teknisk gjeld i prosjektene er dermed heller ikke løst.

Funnene viser også at det ofte er avstand mellom internt ansatte arkitekter og utviklingsleverandørene. *«Samarbeid med kunden fremfor kontraktsforhandlinger»* er et viktig punkt i «The agile manifesto» (Beck et al., 2001). NAV er avhengig av å ha et profesjonelt forhold til leverandører, men overdrevet fokus på kunde/leverandørforholdet i det daglige samarbeidet kan føre til og at leverandørenes kompetanse ikke nyttiggjøres i arkitektur og designprosessen. Det er imidlertid usikkert om avstanden som beskrives er forårsaket av kunde/leverandør-forholdet i seg selv, eller om det er et utslag av at det jevnt over er for lite dialog mellom arkitekter og utviklingsprosjekter.

6.1.2 Prosjektenes egenskaper

IT-avdelingens praksis med å organisere flere mindre, og i utgangspunktet uavhengige, endringer inn i samme leveranse, strider med empiri fra forskning som tilsier at man bør

redusere omfanget av prosjekter (Jørgensen, 2015), og med det omfanget av tidlig plan- og arkitekturarbeid (Boehm, 2011). Store prosjekter er også mindre egnet for smidig gjennomføring (Dybå & Dingsøyr, 2009).

IT-avdelingen jobber med å redusere omfanget av hovedleveransene, og legge til rette for flere frittstående produksjonssettinger og en mer smidig tilnærming. Det er også prosjekter og team som eksperimenterer med mer smidig organisering. I videre drøfting legges det like vel til grunn at det fortsatt vil det være utviklingsoppgaver og prosjekter som er omfattende nok til at det utløser behov for arkitekturarbeid i forkant av konstruksjon (Boehm, 2011).

6.1.3 Kompetanse og kultur

Funnene viser to hovedutfordringer knyttet til kompetanse og kultur. Manglende tilgang til teknisk kompetanse, og «kontrollkulturen».

Madison (2010) peker på at det er sentralt at arkitektene har tillit til at utviklingsteamene er i stand til å ta gode beslutninger om design. «Kontrollkulturen», med mange sjekklister og standardiserte kontroller, står i motsetning til dette. En mulig medvirkende årsak til standardisert kvalitetskontroll kan være manglende tilgang til intern kompetanse. Kontroll ved hjelp av sjekklister forutsetter mindre faglig forståelse av innholdet som skal kontrolleres, enn mer kvalitative tilnærminger.

I IT-avdelingens nye grunnpilarer (se kapittel 4.8) gis det signaler om at ledelsen ønsker seg en mer risikovillig kultur: *«Eksperimentere og risikere mer; enten så lykkes vi eller så lærer vi».* Kombinert med at IT-avdelingen har begynt å ansette egne utviklere, og at arkitektene skal gå inn i en mer deltakende rolle, kan gi mulighet for å følge opp utviklingsleverandører på en mer kvalitativ måte. Flere egne ansatte med teknisk kompetanse kan gjøre det enklere å gjøre kvalitative vurderinger av hva konsulenter foreslår, og legge til rette for mer dialog og mindre kontroll.

Mer aktiv deltakelse vil kunne bidra med verdi til utviklingsteamene ved å bygge *Arkitekturforståelse* hos utviklingsteam og interessenter, samtidig som arkitektene får mer praktisk kunnskap om områdene de skal bidra til å utvikle. Over tid kan dette redusere påvirkningen fra «kontrollkulturen» og avstanden mellom arkitektene og andre deltakere i prosessen som beskrives av informantene.

Arkitektenes rolle bør være å forhindre grep som er til hinder for videre endringsevne. Det forutsetter systemkunnskap og «hands on» deltakelse. Rendell (2009) peker på at arkitekter ofte avskjæres fra å benytte den kompetansen og erfaringen som gjorde dem i kvalifiserte til å bli arkitekter. En mer deltakende rolle, som «tjenesteyteren» beskrevet av Faber (2010). Dette vil kunne gi en bedre utnyttelse av den tekniske og praktiske kompetansen arkitektene sitter på, til bidrag inn i problemløsning i design- og utviklingsprosessen.

Selv om arkitektene som er intervjuet i undersøkelsen har teknisk bakgrunn, og de fleste av dem har erfaringer med smidig tilnærming, er det ikke gitt at alle arkitekter i IT-avdelingen har riktig kompetanseprofil eller personlige egenskaper til å gå inn i en tjenesteyter-rolle. Faber peker på at medarbeidere i et arkitekturteam kan utfylle hverandres kompetanseprofil. Det er altså ikke nødvendig at alle arkitekter bidrar på kodenivå, selv om noen bør være i stand til det.

6.1.4 Systemenes egenskaper

Systemenes størrelse, alder og kritikalitet er noen av faktorene (Kruchten, 2013) nevner som mulige hindringer for en smidig tilnærming. De fleste prosjekter i NAV gjør endringer eller utvidelser av eksisterende systemer. Nye løsninger må som regel integreres med eksisterende systemer. Harde avhengigheter mellom systemer, eller intern kompleksitet i den enkelte systemene kan være et hinder for å kunne jobbe smidig med endringer.

Det er behov for forbedringer i eksisterende systemer, slik at de på sikt blir mindre til hinder for en smidig tilnærming. Informantene peker imidlertid på at det er vanskelig å få finansiert slike tilpasninger. Dette gjør at det sannsynligvis i lang tid framover vil være en variasjon i hvor smidige de enkelte systemene er, og om de enkelt lar seg inkludere i en smidig prosess.

Legacy-systemer med høy intern kompleksitet, og løsninger med mange harde avhengigheter kan være avhengige av mer planlegging på forhånd, siden endringsevnen er lav. Dagens prosesser er først og fremst tilpasset disse systemene. Dersom de ikke skal være til hinder for å innføre smidigere prosesser på andre områder, kan derfor være behov for å differensiere prosessene ut fra hvilke systemer som inngår i endringene.

(Bloomberg, 2013) peker på DevOps-egenskaper som må være på plass for å kunne jobbe smidig med arkitektur. IT-avdelingen har mye av dette på plass gjennom funksjonene som

leveres av Aura-teamet. Dette oppleves som en god støtte for smidighet av flere av informantene, men funksjonene først og fremst til nytte for nyere løsninger på Java-plattform.

6.2 Arkitekturprosessen

Funnene viser en praksis som er ikke i tråd med de målsetningene for en smidig arkitekturprosess som er beskrevet i litteraturen. Arkitekturprosessen differensieres ikke basert på risiko, og prosessen oppleves heller ikke som smidig av informantene. Prosessen ivaretar heller ikke systemenes framtidige smidighet på en god nok måte.

Deler av disse avvikene skyldes at de dokumenterte prosessene og metodene ikke følges, mens andre kan skyldes svakheter med dagens prosesser og metoder.

6.2.1 Målsetninger for arkitekturprosessen

Våren 2016 er det gjort en endring i leveranseprosessen, slik at arkitektene ikke lenger kvalitetssikrer alle endringer, og analyseteamene har fått ansvar for å trekke inn arkitektene etter behov. Dette gir en mer tillitsbasert prosess, men den forutsetter at teamene er i stand til å identifisere risiko som treffer utenfor eget område, og har bevisst forhold til arkitekturkrav, målbilder og lignende overordnede føringer for løsning. I motsatt fall kan det hende at arkitektene ikke kobles inn når de bør, og at arkitektenes kunnskap om sammenhenger på tvers av systemer ikke benyttes på en god nok måte.

Dersom mer ansvar skal delegeres til utviklingsteamene, er det nødvendig med felles forståelse av målsetninger og rammer. Felles forståelse av mål og krav forutsetter mer samarbeid og direkte kommunikasjon mellom arkitekter, prosjekter og utviklingsteam. Det innebærer både at utviklere må involveres tidligere i prosessen, og at arkitektene være tilgjengelige for utviklingsteamene senere i prosessen, hvis det er behov for det.

Responsibility Driven Architecture (RDA) (Blair et al., 2010) kan være en god tilnærming for å balansere behovet for kontroll på tvers av prosjekter og systemer, og smidighet i utvikling. Ved å etablere en plan for beslutningsbehov og designvalg tidlig i prosessen, kan man identifisere hvilke aktiviteter arkitektene er reelle interessenter i. Dette kan bidra til at arkitektene deltar der det er behov, istedenfor å være «bundet» til de tidlige fasene i prosessen. Den planlagte matriseorganiseringen av arkitektene kan bidra til å støtte opp under dette under en slik tilnærming, og også gjøre det lettere å hente inn arkitektene når uventede problemstillinger dukker opp.

En viktig målsetning for en smidig arkitekturprosess at nye systemer eller komponenter understøtter smidig endring av forretningsbehov, og at hindringer i eksisterende systemer reduseres. Bloombergs *meta-krav* om smidige krav påvirker utformingen av ikke-funksjonelle krav, og ivaretakelse av smidigheten til systemet blir dermed også en viktig del av arkitektens rolle.

Funnene tyder på at dagens prosesser ikke i tilstrekkelig grad sikrer framtidig smidighet i systemene. Flere peker på at langsiktige behov som endringsevne og fleksibilitet ofte nedprioriteres til fordel for funksjonalitet, og etterlyser mer fokus på dette i prosjektene. Selv om FKON-prosjektet finansierer noen arkitekturforbedringer, peker flere av informantene på at det kan være vanskelig å få dette prioritert slike tiltak i vanlige prosjekter.

Dette er en hindring som ikke kan løses internt i arkitekturfagmiljøet. IT-avdelingen kan heller ikke løse denne problemstillingen alene, siden de økonomiske prioriteringene i prosjektene gjøres utenfor IT-avdelingen. Ledelsen i IT-avdelingen bør formidle at denne typen problemstillinger går ut over evnen til å levere ønsket funksjonalitet til fagavdelingene, og ikke er et isolert IT-problem.

6.2.2 Analyse, implementasjon og vedlikehold

Metodeverket SLEM vektlegger samarbeid tverrfaglige team, og tett dialog mellom fagpersoner og utviklere (se kapittel 4.3). Analysen skal være iterativ, med lav grad av detaljering i tidlige faser. Funnene viser at praksis ofte avviker fra det som er beskrevet i metodeverket.

En praksis som behandler tidlige vurderinger som bindende, kan føre til en «ond sirkel» der enda mer av analysearbeidet gjøres tidlig i prosessen, fordi utviklingsteamene vil «dekke ryggen sin». Dette kan også komme i konflikt med prinsippet fra SLEM om å eliminere overflødig arbeid. Manglende samarbeid mellom forskjellige miljøer som skal utføre en analyse, kan føre til manglende *Arkitekturforståelse*, og fragmenterte løsninger.

På tross av at det også er elementer i SLEM som ikke er helt i tråd med smidige prinsipper, vil det trolig være en del å hente på å etterleve intensjonen bak SLEM i større grad. I gjennomgangen av funn pekte informantene på at det kunne være en utfordring at intensjonen bak metoden ikke var godt nok kommunisert, og at opplæring har mye fokus på detaljer.

En bedre formidling av de smidige verdiene som ligger til grunn for SLEM kan bidra til en mer smidig praksis. Det er viktig at *meta-metoden* (Bloomberg, 2013), hensikten bak metoden og hvordan den kan utvikles formidles, ikke bare detaljerte rutiner. Aura-teamet viser en form for slik «meta-tilnærming», der de er *«smidige med egen metode»* og *«ingen artefakter eller seremonier er hellige».*

Arkitektene bør bidra til en tydeligere formidling av de smidige prinsippene som ligger til grunn for dagens metode. Metoden bør forbedres kontinuerlig, basert på smidige prinsipper, og interne erfaringer. (Bloomberg, 2013). Flere miljøer i IT-avdelingen eksperimenterer med smidige metoder. Positive erfaringer fra disse forsøkene bør bearbeides inn i det felles metodeverket, så flere kan dra nytte av dem.

Det kan det være behov for å differensiere hvordan prosessen implementeres, ut fra hvilke hindringer som finnes i konteksten for det enkelte prosjektet. Tilfeller der konteksten (f.eks. begrensninger i systemene) utgjør en hindring for smidig framgangsmåte bør dette håndteres som unntak, ikke være definerende for standardprosessen.

Ny organisering i produktteam, og føringene for matriseorganisering av arkitektene kan potensielt gi bedre rammebetingelser for at arkitektene deltar mer aktivt inn i analyseteamene. Arkitektene bør være «ambassadører» for den smidige tilnærmingen, og legge til rette for at beslutninger kan delegeres og utsettes til «siste forsvarlige tidspunkt», i tråd med anbefalingene fra Blair (2010) og prinsippene fra SLEM om at beslutninger skal tas «så sent som mulig (kapittel 4.5).

I noen tilfeller vil det være behov for å gjøre «ad hoc» arkitekturanalyser «i fart». Nord og Tomayko (2006) peker på at tradisjonelle arkitektursentriske metoder er for omfattende og tidkrevende til at de enkelt kan tilpasses smidige prosesser. Ingen av informantene oppga erfaring med slike metoder. Seksjon for IKT-arkitektur kan vurdere å utforske mulighetsrommet i arkitektursentriske lettvektsmetoder, som beskrevet av Yang et al. (2016), og bruk av smidige teknikker for å lettere kunne integrere arkitekturarbeidet i smidige prosjekter.

6.3 Arkitekturbeslutninger

Abrahamsson et al. (2010) peker på at det er en fare for at et hvert designspørsmål behandles som et arkitekturspørsmål, og at det fører til unødvendig omfattende prosesser. En sentralisert

beslutningsprosess kan og bør heller ikke skaleres til å ta «alle» beslutninger. Funnene viser at det kan være utfordrende å få tatt beslutninger raskt nok til å støtte en smidig prosess. Flere opplever at beslutninger tas på for høyt nivå.

Arkitektene bør jobbe aktivt for at flere beslutninger kan tas distribuert, og at beslutningsmyndighet kan delegeres til de som har praktisk kunnskap om de enkelte problemstillingene. Dette vil være i tråd med tilnærmingen som er beskrevet i RDA (Blair et al., 2010), og også med intensjonene i omorganiseringen i IT-avdelingen, der de tverrfaglige produktteamene skal ha ansvar for løsningene de forvalter.

Mange beslutninger kan tas av produktteam, eller som følge av samarbeid mellom produktteam. Dersom team og prosjektene forplikter seg til å etablere beslutningsplaner etter mønster fra RDA vil den sentrale arkitekturfunksjonen likevel kunne følge beslutninger der det er behov for det. Beslutningsplaner med trinnvise beslutninger kan også bidra til at færre interessenter må involveres i hver enkelt beslutning.

Manglende mulighet til å delta i de senere fasene i utviklingsprosessen, kan være årsak til at arkitekter og andre interessenter ønsker å ta beslutninger så tidlig som mulig. Det kan føre til at beslutninger tas på for tynt informasjonsgrunnlag (Poort, 2012), og dermed skape en falsk trygghet. Mer samarbeid gjennom hele prosessen kan øke tilliten hos arkitektene til at utviklerne har forstått rammene tilstrekkelig til å ta gode beslutninger, og gi utviklerne større trygghet for at *«de der oppe i åttende»* forstår hva som skjer.

Å støtte opp om prosessen i de distribuerte beslutningene, kan gi mer kontakt mellom arkitekter og utviklere, og bidra til å forsterke tilliten til at utviklere kan ta beslutninger (se kapittel 6.1.3). Deltakelse i eller observasjon av utviklingsnære beslutninger kan også hjelpe arkitektene i å bygge bedre kompetanse om «den realiserte arkitekturen». Arkitektene kan bidra som prosessfasilitator ved behov, og ellers delta i beslutninger og tilføre egen kunnskap på lik linje med andre interessenter, og innta «tjenesteyter»-rollen som beskrevet av Faber (2010).

6.4 Arkitekturbeskrivelse

Funnene viser at selv om det er mye dokumentasjon tilgjengelig, er den og ofte ikke egnet til å dekke behovene i arkitekturprosessen. Samtidig er det investert mye tid i å etablere felles maler og standarder for dokumentasjon, og i å utarbeide dokumentasjon. Det er et høyt

ambisjonsnivå for hvor mye dokumentasjon som skal være på plass for hvert prosjekt eller system. Når dokumentasjonen ikke kommer på plass kan det være på grunn av manglende ressurser, eller fordi andre oppgaver i realiteten prioriteres høyere.

Arbeidet med design- og arkitekturdokumentasjon kan framstå som lite målrettet, ved at så mange interessenter oppgir at deres behov ikke dekkes. En av informantene trekker fram at man bør skille mellom modeller som brukes som utgangspunkt for diskusjon, der man har mulighet for å få avklaringer av detaljer muntlig (designdokumentasjon), og dokumentasjon som lages for ettertiden der det *«kommer folk seinere som ikke kan spørre»* (systemdokumentasjon). Hvis man fokuserer på hva dokumentasjonen skal brukes til, kan det brukes til å bedre definere krav til form og innhold.

Leffingwell (2011) framhever verdien av visjon dersom man skal oppnå en smidig arkitekturprosess. Dersom man delegere og utsette beslutninger, som beskrevet i kapittel 6.3, er det sentralt at de involverte i utviklingsprosessen har samme bilde av hva målsetninger og rammer er. Funnene viser at det mangler målbilder på mange områder, samtidig som prinsipper og målbilder oppleves som nyttige der de er på plass. Dersom målbilder kan utarbeides i samarbeid mellom produktteam og arkitekter, vil det kunne bidra til bedre *Arkitekturforståelse*. Målbilder kan hjelpe til å definere det handlingsrommet teamene kan operere innenfor, og redusere behovet for at arkitekter kontrollerer og godkjenner detaljer i designet.

Designdokumentasjon er av midlertidig karakter, og kan normalt ledsages av muntlig kommunikasjon. Den må ikke nødvendigvis være underlagt strenge krav til form og innhold. Selic (2009) argumenterer for at detaljert design ofte er bortkastet, og er så godt som umulig å holde oppdatert opp mot den detaljerte koden.

Funnene viser eksempler på positive erfaringer med bruk av kode i designprosessen, og eksempler på at modeller som utvikles av arkitekter ikke alltid er forståelige for utviklere. Kode kan supplere modeller og tekst som virkemidler for å formidle arkitektur, og bidra til bedre *Arkitekturforståelse* blant de mer teknisk orienterte interessentene. Faber (2010) peker på at kode er *«utviklernes eget språk»,* og også kan brukes for å formidle arkitektur og designvalg.

Omfanget av dokumentasjon som utarbeides i forbindelse med konsekvensutredning og løsningsbeskrivelse bør også sees i sammenheng med graden av muntlig kommunikasjon. (Faber, 2010) legger vekt på at skriftlig designdokumentasjon aldri skal overleveres uten muntlig kommunikasjon om innholdet. Prinsippet om *«akkurat nok design»* som beskrives av Yang et al. (2016) kan være en god rettesnor for utforming av designdokumentasjon.

Systemdokumentasjon har derimot lang levetid, og det er ikke gitt at mottakeren av dokumentasjonen har tilgang til å stille spørsmål til de som har utformet systemene. Det er viktigere at den skriftlige dokumentasjonen kan stå på egne bein. Samtidig er det viktig at denne dokumentasjonen avspeiler det faktisk realiserte systemet. Rendell (2009) mener utviklingsteamet sammen med arkitekten bør avgjøre hva som er passende nivå for dokumentasjonen i hvert enkelt tilfelle.

En mulig løsning kan være å generere diagrammer som kan inngå i systemdokumentasjonen basert på den «realiserte arkitekturen», og på den måten *Gjenvinne* arkitekturdesignet (se kapittel 6.6.3). Dette vil kunne forenkle *Analyse av påvirkning*. (se kapittel 6.6.1). I tillegg til beskrivelser av strukturen er det viktig å formidle sentrale krav som påvirker arkitekturen. Dokumentasjon av viktige *Arkitekturbeslutninger* må også være tilgjengelig som en del av arkitekturbeskrivelsen.

6.5 Arkitekturforståelse

Innsiktene fra forskningen viser at tilstedeværelse og direkte dialog er sentrale for å lykkes i en smidig arkitekturprosess. Manglende arkitekturforståelse kan være et resultat av utfordringer som er diskutert under flere av de andre temaene, som

- kontekst for prosessen (organisering, kompetanse og kultur)
- implementasjon av selve prosessen (arkitektene deltar tidlig i prosessen, utviklerne sent)
- utforming av arkitekturbeskrivelser (arkitektene bruker språk eller modeller som ikke forstås av utviklingsteam).

Dersom analyseteamene jobber mer i tråd med de smidige prinsippene som ligger til grunn for SLEM, vil det innebære mer direkte kommunikasjon. Metoden tilsier at utviklingsteamet alltid skal delta i analysen, og at arkitektene også deltar der det er behov. Blant prinsippene

som ligger til grunn for SLEM er *«Tett dialog mellom fagpersoner og utviklere*» og *«Selvorganiserte tverrfaglige team»*.

Funnene viser at flere av arkitektene ønsker å være mer tilgjengelige for analyseteam og utviklingsteam, men at mangel på tid er en hindring. Flere tiltak som er skissert tidligere i diskusjonen, f.eks. å differensiere innsatsen basert på risiko, kan bidra til å frigjøre tid til mer aktiv deltakelse i prosessen.

Dersom arkitektene ikke «låses» til å delta tidlig i prosessen, men kan prioritere hva som følges opp ut fra risiko og ut fra hvor det er behov for bistand, gir det bedre muligheter for fysisk tilstedeværelse og direkte dialog.

I den planlagte organisasjonsendringen forutsettes det også at arkitektene må «ut av kontoret» og delta aktivt inn i prosjekter og produktteam. Det kan gjøre det enklere å organisere seg rundt konkrete oppgaver som skal løses, og at organisasjonskartet i mindre grad blir et hinder for samarbeid.

Funnene inneholder også eksempler på positive erfaringer med mer aktiv deltakelse i prosessen, f.eks. ved å delta i utviklernes standup for å bidra med avklaringer.

6.6 Andre arkitekturaktiviteter

Funnene omfatter også de temaene i modellen som i mindre grad er beskrevet i litteraturen som er gjennomgått.

6.6.1 Analyse av påvirkning

Funnene viser at *analyse av påvirkning* utgjør en stor andel av analysene som gjøres i arkitekturprosessen. Dette er utfordrende blant annet fordi det ofte mangler tilstrekkelig dokumentasjon av systemene, og at man mangler kompetanse om systemene blant egne ansatte. Litteraturen sier lite om hvordan man kan utføre analyse av påvirkning på arkitekturen i en smidig prosess.

En mulighet er å se på denne problemstillingen i sammenheng med aktiviteten *Arkitekturgjenvinning*. Dokumentasjon av den realiserte arkitekturen, basert på kode eller metadata om de faktiske applikasjonene, kan bidra til å forenkle analyse av påvirkning av endringer. Dette beskrives nærmere i kapittel 6.6.3.

6.6.2 Arkitekturgjenbruk

Funnene inneholder eksempler på at der infrastruktur som sikkerhet og integrasjonsmekanismer er på plass, er det enklere å jobbe smidig. Dette er områder som også inngår i IT-infrastrukturen som beskrives av Weill et al. (2002) som en forutsetning for smidig virksomhetsutvikling. Med slike grunnleggende elementer på pass kan teamene konsentrere seg om å løse de funksjonelle behovene som er spesifikke for leveransen. Dette samsvarer med det Waterman et al (2015) beskriver i sin *Bruke rammeverk og standardarkitektur*-strategi. Men funnene viser også eksempler på at standardisering og gjenbruk kan være til hinder når prosjektene påtvinges rammeverk som ikke understøtter behovene på en god måte.

Standardisering og gjenbruk av arkitekturkomponenter må ha som mål å redusere kostnader og øke fleksibilitet, men det kan være vanskelig å forutse hvilke grep som faktisk vil gi nytte. Standardiseringsgrep som er fordelaktige for organisasjonen som helhet kan også likevel oppleves som en hindring av enkeltprosjekter. IT-avdelingen bør vurdere hvilke standard-krav som stilles til utvikling, ut fra om det bidrar til forenkling og effektivisering. Man bør være forsiktig med å stille krav man ikke er sikre på effekten av. Utviklingsteam har ofte de beste forutsetningene for å finne effektive løsninger.

Områder som er egnet for standardisering bør identifiseres i samarbeid med produktteam og utviklingsleverandører. Tiltak kan legges inn i en arkitekturkø. Ideelt sett bør slike fellesløsninger være på plass «just in time», etter mønster fra Leffingwells (2011) *architecture runway* (Leffingwell, 2011). Denne tilnærmingen kan imidlertid være vanskelig å få til innenfor dagens rammer for finansiering i NAV, der midlene som hovedregel følger etter forretningsmessige behov.

6.6.3 Arkitekturgjenvinning

Flere peker på at det er krevende å få oversikt over sammenhengene mellom systemer som inngår i et prosjekt. Det er også krevende å holde slike oversikter oppdatert sentralt, og utviklerne kjenner seg ikke alltid igjen i notasjon og modeller som brukes i kartleggingen. I den andre enden av prosessen forholder arkitektene seg sjelden til kode. En av informantene peker på at kodebasene inneholder informasjon om avhengigheter, og oversikter kan genereres automatisk.

Rendell (2009) benyttet diagrammer generert på basis av kodebasen som grunnlag for kvalitetssikring av løsningen. Det er tilsvarende mulig å generere diagrammer basert på det Erder og Pureur (2016) omtaler som den *realiserte arkitekturen*, og på den måten *gjenvinne* arkitekturdesignet.

Dokumentasjon av systemene og sammenhenger mellom dem («nå-situasjon») er en forutsetning for å kunne *Analyse av påvirkning*. Generert dokumentasjon vil kunne gi et format både utviklere og arkitekter forstår, forbedre *Arkitekturforståelse* og muliggjøre raskere analyse av endringer.

6.6.4 Arkitektur-refaktorering

Noen av informantene etterlyser mer aktiv bruk av refaktorering underveis, framfor omfattende analyser før oppstart. Refaktorering er et viktig element i smidige metoder, for å understøtte endringshåndtering.

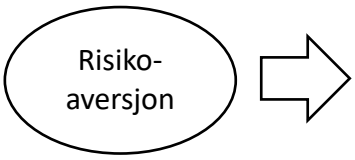
(Sharifloo et al., 2008) beskriver en tilnærming der man inspiserer arkitekturen ved slutten av hver iterasjon, og legger prioriterte refaktoreringsoppgaver inn i produktkøen for prioritering. En slik tilnærming kan benyttes innenfor rammene av SLEM, men forutsetter at produkteiere har forståelse for disse behovene, og velger å prioritere dem.

6.7 Mulige forklarende mekanismer

Noen mulige forklarende mekanismer for utfordringene i arkitekturprosessen kan finnes i konteksten for prosessen, som kultur, kompetanse og organisering

Funnene viser at det er en form for ***risikoaversjon*** i IT-avdelingen, som blant annet gir seg utslag i «kontrollkultur».

IT-avdelingen har lenge hatt «Stabil IKT-drift» som sitt fremste mål. Det er svært viktig at brukere får utbetalt det de har krav på til rett tid. IT-avdelingen har lagt lista for kvalitetssikring svært høyt, også for systemer der feil ikke har så alvorlige konsekvenser. Omfattende krav til test og kvalitetssikring er en av årsakene til organisering i hovedleveranser. Dette medfører til tider unødvendig mye testing sett i forhold til risiko for den enkelte endringen. Noen mulige direkte og indirekte effekter av denne mekanismen er oppsummert i Figur 10



Tema	Effekt
Systemer	• Uegnet standardisering og rammeverk • Vanskelig å få godkjent ny teknologi
Prosjekter	• Store og sjeldne leveranser
Kompetanse og kultur	• Intern kompetanse brukes på feil sted • «Kontrollkultur» • Involverer ikke leverandører godt nok
Arkitekturprosess	• Kvalitetssikrer «alt» tidlig i prosessen • Sjekklister og «revisor-tilnærming»
Arkitekturbeslutninger	• Delegerer ikke myndighet
Arkitekturbeskrivelse	• Omfattende designdokumenter
Arkitekturforståelse	• Ikke nok tid tilgjengelig direkte dialog, på grunn av dokumentorientert arbeidsform.

Figur 10 - Forklarende mekanismer

Noen av effektene av risikoaversjon, kan i seg selv føre til økt risiko. Det er for eksempel et paradoks at man øker omfanget på leveransene for å få kontroll, når forskning viser at mindre og hyppigere leveranser reduserer risiko (Jørgensen, 2015). En annen mulig bivirkning er at omfattende standardkontroller og sjekklister tar tid og fokus vekk fra samarbeid og direkte dialog mellom arkitekter og andre interessenter, og dermed fra en mer kvalitativ orientert kvalitetssikring av arkitektur og design.

Kontrollmekanismene som benyttes bør være tilpasset prosjektets risikoprofil. «Standardprosessen» bør ikke ta høyde for hindringer som bare påvirker noen av prosjektene, eller risikoprofilen til de mest krevende prosjektene, og bør legge til rette for at utviklingsteamene kan ta beslutninger selv. IT-avdelingens nye grunnpilarer, som søker etter kontinuerlig forbedring og læring, og mer risikovillighet kan gi bedre rammer for en slik tilnærming enn det har vært tidligere.

6.8 Oppsummering og anbefalte tiltak

Diskusjonen av funn knyttet til arkitekturprosessen i IT-avdelingen opp mot innsikter fra forskning og faglitteratur viser flere avvik. Det er også avvik mellom den dokumenterte metoden i IT-avdelingen og praksis.

I ny organisering vil alle arkitekter være under samme faglige ledelse. Dette gir stort handlingsrom for sjefsarkitekten til å redefinere praksis. En del av rammebetingelsene som oppleves som hindringer for smidighet i prosessen ligger imidlertid utenfor arkitekturmiljøets kontroll. Dette gjelder for eksempel sourcing, prosjektorganisering og finansiering.

Det er mulig å se på disse faktorene som rammebetingelser «man må leve med» i en periode, og jobbe med forbedringstiltak som kan gjennomføres av arkitekturfagmiljøet. Funnene viser mange slike mulige forbedringer, og mange av forslagene informantene kommer med til forbedringer har også støtte i litteraturen.

I Tabell 11 oppsummeres noen mulige tiltak for å gjøre arkitekturprosessen mer smidig, og lette til rette for skifte av strategi fra *Big Design Up Front* til en *Respondere på endring*-strategi som bedre understøtter IT-avdelingens nye mål.

Tabell 11 - Anbefalte tiltak for forbedringer av arkitekturprosessen.

Tema	**Prioriterte tiltak**
Overordnet tiltak	• Prioritere omfang og form på arkitekturarbeidet etter risiko. Områdene med høy risiko må ikke være definerende for arbeid med områder med lavere risiko.
Arkitekturprosessen	• Arkitekturarbeid utføres samarbeid med utviklingsteamene som påvirkes. • Metoden bør forbedres kontinuerlig, basert på smidige prinsipper og erfaringer i organisasjonen.
Arkitekturbeslutninger	• Arkitektur- og designbeslutninger bør i større grad delegeres, og utsettes til «siste ansvarlige tidspunkt».
Arkitekturbeskrivelse	• Det må utarbeides visjoner og målbilder som understøtter virksomhetens mål. • La teamene vurdere hva som er passende form og omfang for den skriftlige designdokumentasjonen.
Arkitekturforståelse	• Bruk muntlig kommunikasjon til å formidle og underbygge innholdet i skriftlig dokumentasjon, der det er mulig.

Slike forbedringer vil kunne gi et løft for effektivitet og kvalitet i arkitekturarbeidet, men løser ikke alene utfordringen med at prosessen ikke godt nok ivaretar framtidig smidighet. Dersom smidighet i systemene ikke prioriteres og finansieres i framtidige utviklingsprosjekter, og eksisterende teknisk gjeld ikke reduseres, vil ikke forbedringer av arkitekturprosessene alene være nok til å ivareta denne viktige målsetningen. Manglende smidighet i IT-systemene er ikke et isolert IT-problem, men er til hinder for at ønsket funksjonalitet kan leveres. Arkitekturfagmiljøet bør jobbe aktivt for å øke forståelsen av disse problemstillingene i ledelsen, slik at egnede finansieringsmekanismer kan komme på plass.

7 Konklusjon

Den nye organiseringen der produktteam får helhetlig ansvar for løsningen de bygger og vedlikeholder, gir behov for endring av arkitektenes praksis, og hvordan arkitekturprosessen implementeres.

7.1 Hva sier forskningen om smidige arkitekturprosesser i store organisasjoner?

Forskningsspørsmål 1: Kan forskningen peke på smidige arkitekturprosesser og -teknikker som understøtter hurtige leveranser av IT-løsninger i store organisasjoner?

Det er skrevet mye om kombinasjonen smidig og arkitektur de siste årene, men litteraturen som er gjennomgått sier lite om hvordan man kan jobbe smidig med arkitektur i organisasjoner med store systemporteføljer.

Men mange av utfordringene som beskrives av informantene er sammenfallende med utfordringer som beskrives og forsøkes løst i normativ og empirisk teori. Det er derfor grunn til å tro at innsikter og «lessons learned» fra teorien kan brukes som utgangspunkt for forbedringer av arkitekturprosessen.

Viktige grep for endring kan være:

- Å tilpasse innsats og omfang av arkitekturarbeidet etter risiko-nivå.
- Mer direkte, muntlig kommunikasjon mellom sentrale interessenter, spesielt mellom arkitekter og utviklere.
- At arkitektene i større grad delegerer beslutningsmyndighet til utviklingsteamene, og konsenterer seg om å identifisere og definerere handlingsrom og rammer teamene må forholde seg til

Disse grepene kan understøtte en bevegelse fra *Big design up front*-strategi *til en Respondere på endring*-strategi, som er mer i tråd med smidige prinsipper men som samtidig tar høyde for håndtering av risiko.

7.2 Hindringer og muligheter i IT-avdelingen

Forskningsspørsmål 2: Hvilke faktorer i IT-avdelingen er til hinder eller gir muligheter for en mer smidig arkitekturprosess?

I undersøkelsen av arkitekturprosessen skiller vi selve prosessen fra konteksten den inngår i. Det er både hindringer og muligheter knyttet til selve prosessen, og knyttet til konteksten.

Viktige hindringer i konteksten for prosessen er:

- Det er utfordrende å finansiere tekniske forbedringer, som kan muliggjøre mer smidige systemer, og å ta høyde for ikke-funksjonelle krav underveis i prosjekter.
- Legacy-systemer med mange avhengigheter kan gjøre det vanskelig å gjennomføre endringer på en smidig måte.
- «Kontrollkultur» bidrar til arkitektene bruker tid på å kvalitetssikre hva andre gjør, istedenfor å bidra som deltakere i prosessen.

Viktige hindringer knyttet til selve prosessen er:

- Form og omfang på arkitekturarbeidet differensieres ikke tilstrekkelig – alle skjæres over en kam.
- Arkitektene jobber i stor grad adskilt fra utviklingsteamene.
- Det er mye skriftlig designdokumentasjon, og lite direkte dialog.
- Arkitekter og utviklere mangler en felles arkitekturforståelse.

De viktigste mulighetene for endringer i prosessens kontekst er knyttet til IT-avdelingens nye målsetninger om å levere IT-løsninger mer effektivt enn tidligere, og økt vilje til å ta risiko for å søke etter læring og forbedringer. Endringene i IT-avdelingen påvirker flere av faktorene i arkitekturprosessens kontekst som i dag oppleves som et hinder for smidighet, som organisasjonsstruktur, «kontrollkulturen», størrelsen på leveranser og kompetanseprofilen i IT-avdelingen.

Det er også muligheter knyttet til å bedre formidle intensjonen og målene bak prosesser og metodikk som benyttes. Metodikken er basert på smidige prinsipper. Funnene viser at praksis ofte avviker fra denne intensjonen.

I de neste avsnittene oppsummeres viktige grep for å håndtere sentrale hindringer for en smidigere arkitekturprosess.

7.2.1 Felles retning og prinsipper

Funnene viser at der målbilder og prinsipper er definert, oppleves det som nyttig, men det mangler målbilder og prinsipper for mange områder. Der målbilder er etablert, er de ofte ikke formidlet på en god måte til utviklingsteamene.

Arkitektmiljøet bør etablere nødvendige målbilder og prinsipper for områdene der det forventes utvikling, i samarbeid med utviklingsteamene. Samarbeid gir mulighet til å utnytte praktiske erfaringer, og skape en felles *Arkitekturforståelse*.

7.2.2 Sikre framtidig smidighet i systemene

En prosess som ikke er i stand til å produsere et smidig system undergraver framtidig smidighet. I tillegg til å tiltak med å gjøre arkitekturprosessen mer smidig, bør arkitekturfagmiljøet jobbe for å synliggjøre behovet for finansiering av grep som øker framtidig smidighet i systemene. Dette kan omfatte teknisk gjeld i eksisterende systemer, etablering av *architecture runway* og krav som stilles til nye systemer som utvikles.

7.2.3 Prioritering av innsats og delegering av ansvar

Arkitektene bør prioritere å bruke tid på prosjekter og initiativ med stor risiko, eller som berører nye områder. På de områdene det er behov for arkitekturarbeid, bør oppgavene løses i tett samarbeid med utviklingsteamene.

Gjennom å etablere beslutningsplaner tidlig i prosjektene, kan man balansere behovene for risiko-håndtering og sentral oversikt på den ene side, mot behovene for selvstendige beslutninger i teamene og å ha best mulig informasjonsgrunnlag før beslutninger tas.

Arkitektene bør gi teamene tillit til å løse oppgavene som er definert, og stole på at de tar kontakt når oppgaven ikke kan løses på en god måte innenfor teamets rammer.

7.2.4 Dokumentasjon og kommunikasjon

Dokumentasjonen bør i større grad enn i dag avspeile behovene til interessentene som bruker dokumentasjonen. I designprosessen bør brukerne av designdokumentasjonen avgjøre hva som er egnet form og innhold. Formidling av arkitektur bør i større grad baseres på dialog og

deltakelse fra arkitektenes side, og i mindre grad kvalitetssikring gjennom inspeksjon av dokumenter.

7.3 Forslag til videre undersøkelser

Flere av temaene i denne oppgaven er lite undersøkt i forskning. Her oppsummeres noen av disse områdene. Forslag til problemstillinger for andre oppgaver eller undersøkelser er beskrevet i Tabell 12.

På tross av at refaktorering er en sentral teknikk i smidig utviklingsmetodikk, er det nesten ikke gjort forskning på hvordan *Arkitekturrefaktorering* kan brukes som virkemiddel i en smidig arkitekturprosess. (Yang et al., 2016) peker på at refaktorering av arkitektur kan være krevende og kostbart. (Sharifloo et al., 2008) beskriver en tilnærming der arkitektur-refaktorering gjøres løpende under hele utviklingsprosessen.

Det er også lite forskning knyttet til om *Arkitekturgjenbruk* og standardisering støtter smidige prinsipper. Informantene peker på eksempler på at forsøk på gjenbruk og standardisering kan være både til støtte og til hinder. Weill et al (2002) beskriver IT infrastruktur som bør være på plass for å understøtte smidig virksomhetsutvikling. Leffingwell (2011) argumenterer også for å etablere gjenbrukbare komponenter for å understøtte mer kostnadseffektiv utvikling, og at disse ofte kan etableres ved hjelp av *Arkitektur-refaktorering*. Waterman et al. (2015) beskriver hvordan standardrammeverk kan øke smidigheten i et prosjekt, men er lite konkret på hvilke områder eller egenskaper som er best egnet for denne strategien. (Durdik, 2011) peker på at smidige metoder i utgangspunktet ikke legger godt til rette for *Arkitekturgjenbruk*.

Arkitekturgjenvinning i en smidig kontekst er heller ikke beskrevet i noen av studiene som er undersøkt av Yang et al. (2016). Rendell (2009) beskriver bruk av verktøy for å gjenvinne design basert på kode, til bruk i kvalitetssikring av design, men ikke som en permanent løsning for dokumentasjon. Å undersøke om denne tilnærmingen kan understøtte en smidig arkitekturprosess i praksis kunne vært et interessant tema for videre undersøkelser.

Flere av informantene i undersøkelsen nevner sourcing og prismodeller som en faktor som påvirker evnen til smidig arbeid. Dette er en faktor som hverken beskrives av Yang et al. (2016) eller Kruchten (2013). Det er ikke mulig å konkludere ut fra data som er samlet inn i forbindelse med denne oppgaven på om det at utviklingsoppgaver er outsourcet i seg selv

påvirker evnen til smidig, eller om dette isolert sett kan fungere bra dersom andre identifiserte utfordringer løses.

Tabell 12 - Mulige tema for videre undersøkelser

Tema	Problemstillinger
Prosessens kontekst	Påvirker outsourcing av systemutvikling evnen til smidighet i en organisasjon?
Arkitekturrefaktoring	Hvordan kan *Arkitekturrefaktoring* benyttes som en del av en smidig arkitekturprosess?
Arkitekturgjenbruk	Hvordan og i hvilke former kan *Arkitekturgjenbruk* brukes som virkemiddel i en smidig prosess?
Arkitekturgjenvinning	Kan *Arkitekturgjenvinning* benyttes som virkemiddel for å understøtte en smidig prosess?

8 Litteratur

Abrahamsson, P., Babar, M. A., & Kruchten, P. (2010, March). Agility and Architecture: Can they Coexist. *IEEE Software*, 16–22.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). Manifesto for Agile Software Development. Retrieved July 2, 2016, from http://agilemanifesto.org/iso/no/manifesto.html

Blair, S., Watt, R., & Cull, T. (2010). Responsibility-driven architecture. *IEEE Software*, *27*(2), 26.

Bloomberg, J. (2013). *The Agile Architecture Revolution*. Hoboken, New Jersey: John Wiley & Sons.

Boehm, B. (2011). Architecting: How Much and When. In A. O. & G. Wilson (Ed.), *Making Software - What Really Works, and Why We Believe IT* (pp. 161–185). Sebastapol, California, USA: O'Reilly Media.

Breivold, H. P., Sundmark, D., Wallin, P., & Larsson, S. (2010). What Does Research Say About Agile and Architecture (pp. 32–37). Presented at the 2010 Fifth International Conference on Software Engineering Advances, IEEE Computer Society.

Brown, N., Yuanfang, C., Guo, Y., Kazman, R., Kim, M., Krutchen, P., … Zazworka, N. (2010). Managing Technical Debt in Software-Reliant Systems. *FoSER '10 Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, 47–51.

Buschmann, F., & Henney, K. (2013). Architecture and agility: Married, divorced, or just good friends? *IEEE Software*, *30*(2), 80–82.

Bygstad, B., & Munkvold, B. E. (2007). The significance of member validation in qualitative analysis: experiences from a longitudinal case study. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on* (p. 243b–243b). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4076875

Bygstad, B., & Pedersen, N. (2012). "Arkitektur handler om praktisk arbeid i organisasjonen, ikke en tegning". En forskningsagenda om IT-arkitekters utfordringer. Presented at the Norsk konferanse for organisasjoners bruk av informasjonsteknologi - NOKOBIT.

Cleland-Huang, J., Czauderna, A., & Keenan, E. (2013). A persona-based approach for exploring architecturally significant requirements in agile projects. In *International Working Conference on Requirements Engineering: Foundation for Software Quality* (pp. 18–33). Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-37422-7_2

Diaz, J., Perez, J., Garbajosa, J., & Yague, A. (2013). Change-Impact Driven Agile Architecting (pp. 4780–4789). IEEE. https://doi.org/10.1109/HICSS.2013.127

DIFI. (2014). Virksomhetsarkitektur | Prosjektveiviseren - Difi. Retrieved November 18, 2016, from http://www.prosjektveiviseren.no/_library/node/tema/virksomhetsarkitektur

Durdik, Z. (2011). Towards a Process for Architectural Modelling in Agile Software Development. Presented at the Proceedings of the Joint 7th International Conference on the Quality in Software Architecture & 6th European Conference on Software Archtecture (QoSA/ISARCS), Boulder, Colorado, USA.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, *50*(9–10), 833–859. https://doi.org/10.1016/j.infsof.2008.01.006

Dybå, T., & Dingsøyr, T. (2009). What do we know about agile software development? *IEEE Software*, *26*(5), 6–9.

Erder, M., & Pureur, P. (2016). What's the Architect's Role in an Agile, Cloud-Centric World? *IEEE Software*, *33*(5), 30–33.

Faber, R. (2010). Architects as service providers. *IEEE Software*, *27*(2), 33.

Hadar, I., Sherman, S., Hadar, E., & Harrison, J. J. (2013). Less is more: Architecture documentation for agile development. In *Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on* (pp. 121–124). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6614746

ISO. (2011). International standard ISO/IEC/IEEE 42010: Systems and software engineering - architecture description.

Jørgensen, M. (2015). *Suksess og fiasko i offentlige IT-prosjekter: En oppsummering av forskningsbasert kunnskap og evidensbaserte tiltak*. Oslo: Simula Research Laboratory/Universitetet i Oslo/Scienta.

Kruchten, P. (2013). Contextualizing agile software development: CONTEXTUALIZING AGILE SOFTWARE DEVELOPMENT. *Journal of Software: Evolution and Process*, *25*(4), 351–361. https://doi.org/10.1002/smr.572

Kvale, S., & Brinkmann, S. (2015). *Det kvalitative forskningsintervju. 3. utgave.* Oslo: Gyldendal Akademisk.

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise* (1st ed.). Addison-Wesley Professional.

Li, Z., Liang, P., & Avgeriou, P. (2013). Application of knowledge-based approaches in software architecture: A systematic mapping study. *Information and Software Technology*, *55*(5), 777–794. https://doi.org/10.1016/j.infsof.2012.11.005

Madison, J. (2010, March). Agile-Architecture Interactions. *IEEE Software*, 41–48.

Myers, M. D. (2013). *Qualitative Research in Business and Management, 2nd Edition* (2nd Revised edition edition). London: Sage Publications Ltd.

Nord, R. L., & Tomayko, J. E. (2006). Software architecture-centric methods and agile development. *IEEE Software*, *23*(2), 47–53.

Poort, E. R., van Vliet, Hans. (2012). RCDA: Architecting as a risk- and cost management dicipline. *The Journal of Systems and Software*, *85*, 1995–2013.

Rendell, A. (2009). Descending from the Architect's Ivory Tower (pp. 180–185). IEEE. https://doi.org/10.1109/AGILE.2009.17

Selic, B. (2009). Agile documentation, anyone? *IEEE Software*, *26*(6), 11–12.

Sharifloo, A. A., Saffarian, A. S., & Shams, F. (2008). Embedding architectural practices into extreme programming. In *19th Australian Conference on Software Engineering (aswec 2008)* (pp. 310–319). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4483219

Tang, A., Avgeriou, P., Jansen, A., Capilla, R., & Ali Babar, M. (2010). A comparative study of architecture knowledge management tools. *Journal of Systems and Software*, *83*(3), 352–370. https://doi.org/10.1016/j.jss.2009.08.032

The Open Group. (2009). *TOGAF 9*. Zaltbommel: Van Haren Publishing.

van Heesch, U., Eloranta, V.-P., Avgeriou, P., Koskimies, K., & Harrison, N. (2014). Decision-Centric Architecture Reviews. *IEEE Software*, (January/February 2014), 69–76.

Waterman, M., Noble, J., & Allan, G. (2015). *How Much Up-Front? A Grounded Theory of Agile Architecture*. Victoria University of Wellington.

Yang, C., Liang, P., & Avgeriou, P. (2016). A systematic mapping study on the combination of software architecture and agile development. *Journal of Systems and Software*, *111*, 157–184. https://doi.org/10.1016/j.jss.2015.09.028

Yin, R. K. (2013). *Case Study Research: Design and Methods, 5th Edition*. SAGE Publications.

8.1 Interne dokumenter fra NAV

- Teknofyr – intern wiki
  - Dokumentasjon av organisasjonsutviklingsprosjekt
- Navet – intranett
  - Organisasjonskart
  - «Transformasjonen av NAV IT er i gang» - nye grunnpilarer.
- Sysdok – intern wiki
  - Systemdokumentasjon
- «Arkitekturportalen» - intern wiki
  - Referat fra Områdearkitekt-møter
  - Referat fra IKT-arkitekturforum
  - Dokumentasjon av arkitekturbeslutningsprosess
  - Arkitekturdokumentasjon
- «Metodeportalen» - intern wiki
  - Dokumentasjon av SLEM-prosessen og tilhørende metodeverk for arkitektur.
  - Rutine – Mindre Funksjonelle Applikasjonsendringer (MFA)

9 Vedlegg

Vedlegg 1: Intervjuguide

Vedlegg 2: Innsikter og svar fra informanter

Vedlegg 3: Presentasjon fra gjennomgang av funn for informanter

Vedlegg 4: Referat fra gjennomgang av funn

Vedlegg 5: Oversikt over hindringer og muligheter

Vedlegg 6: Sammenligning av forventet og empirisk mønster

Vedlegg 7: Informasjonsskriv til informanter

Vedlegg 8: Godkjenning fra NSD

Vedlegg 1: Intervjuguide

Intervjuguiden ble justert to ganger, til henholdvis versjon 1.1 etter første intervju, og 1.2 etter andre intervju. Spørsmålene som ble lagt til i disse versjonene er markert under.

Introduksjon (stikkord)

- Dette er et semi-strukturert intervju, det vil si at jeg har en liste med spørsmål som er forberedt, men også kan stille andre spørsmål basert på det som kommer fram i intervjuet, eventuelt droppe spørsmål dersom de besvares direkte eller indirekte som en del av samtalen. Det blir gjort lydopptak
- Du er en av flere som intervjues, de som intervjues er arkitekter i IT-avdelingen eller interessenter i arkitekturarbeidet. Informasjonen brukes i arbeidet med min masteroppgave.
- Hvis det er aktuelt å bruke ideer som kommer fram eller lignende på arbeidsplassen, vil jeg ikke gjøre det uten å avklare det med deg først.
- Forklare overordnet hva begreper som brukes i spørsmålene, som «smidig», «arkitekturarbeid» og «arkitekturprosess».

Spørsmål

1. Rolle og bakgrunn

1.1 Hva er din utdanning og faglige bakgrunn?

1.2 Hvor lenge har du jobbet i/for NAV?

1.3 Hvor i NAV/IT-avdelingen er du ansatt/tilknyttet? (fra versjon 1.1)

1.4 Hva er (vanligvis) din rolle i tilknytning til arkitekturprosessen?

1.5 Hvor lang erfaring har du med rollen?

1.6 Hvilke prosjekter har du deltatt i/jobbet med de siste 1-2 årene?

2 Generelle erfaringer

2.1 Hvilke smidige metoder/praksiser har du erfaring med (fra NAV eller andre steder) (ikke hjulpet)

2.2 Hvilke arkitekturmetoder/teknikker har du erfaring med (ikke hjulpet)

2.3 Har du erfaring med å benytte smidige teknikker i arkitekturarbeid?

2.3.1 Hvilke? (Fra versjon 1.1)

2.4 Hvordan fungerte det?

3 Erfaringer med dagens arkitekturprosess i IT-avdelingen

Fra versjon 1.2 Ble spørsmål 3.7 og 3.5 stilt før spørsmål 3.4. Nummereringen av spørsmålene ble ikke endret, for å ikke forvanske analyse av data fra intervjuene.

3.1 Hvordan opplever du dagens arkitekturprosess?

3.2 Hvordan opplever du kommunikasjonen og samhandlingen mellom Seksjon for IKT-arkitektur og arkitekter i prosjekter eller forvaltningskontor?

3.3 Føler du at du har medbestemmelse over arkitekturspørsmål i din rolle?

3.4 Kjenner du til situasjoner der det var «for mye arkitekturarbeid»? Hva skjedde?

3.4.1 Oppfølgingsspørsmål, hjulpet:

- Eksempler på overflødige analyser (fra versjon 1.2) eller arkitekturdokumentasjon?
- Eksempler på tidkrevende beslutningsprosesser?

3.5 Kjenner du til situasjoner der det var «for lite arkitekturarbeid»? Hva skjedde?

3.5.1 Oppfølgingsspørsmål

- Eksempler på dårlige analyser (fra versjon 1.2) og dokumentasjon?
- Eksempler på beslutninger som burde vært tatt, burde vært tydeligere etc?

3.6 Opplever du at arkitekturarbeidet som utføres bidrar til å redusere risiko? Hvorfor/hvorfor ikke?

3.7 Hva erfarer du at er viktigst å avklare tidlig i leveranseprosessen? (fra versjon 1.2)

4 Hindringer og muligheter

4.1 Er det elementer av arkitekturprosessen som hindrer hurtige/smidige utviklingsløp

4.1.1 Hvordan?

4.2 Er det elementer av arkitekturprosessen som bidrar/kan bidra til hurtige/smidige utviklingsløp?

4.2.1 Hvordan?

4.3 Har du erfaringer med at andre prosesser, rammebetingelser osv har vært en hindring for å jobbe effektivt/smidig med arkitektur? (Fra versjon 1.1)

4.4 Har du erfaringer med at andre prosesser/rammebetingelser etc har hjulpet til med å jobbe mer effektivt/smidig med arkitektur? (Fra versjon 1.1)

5 Annet

5.1 Er det noe annet du ønsker å tilføye eller spørre om?

5.2 Har du tips om andre jeg burde snakke med om dette temaet? (prosjektledere, leverandører...)

5.3 Hvordan opplevde du dette intervjuet? Har du innspill til forbedringer? (Fra versjon 1.1)

Vedlegg 2: Innsikter og svar fra informanter

Utsagn fra intervjuene, sortert etter kategoriene i modellen som er beskrevet i kapittel 2.3. Noen utsagn passer inn i flere kategorier, og er derfor plassert inn flere steder i modellen.

Kolonne A: Informantnummer

Kolonne B: Referanse til spørsmål i intervjuguide, eller DV for deltakervalideringsmøte.

Kontekst - Organisering		
A	**B**	**Svar**
1	3.1	Vi må være ute, ikke sitte i et hjørne og modellere. Man må noen ganger trekke seg litt tilbake og tegne, men så gå ut igjen og sjekke: «Er det sånn vi forstår dette?».
3	3.3	Blir spredd ut over så mange oppgaver? Ja, det er noe med organiseringa. Jeg sitter på forvaltningskontoret, prosjektet er organisasjoner i seg selv, egen økonomi og styring, på organisasjonskartet er det IKT-arkitektur som har ansvaret, Plan-Build-Run-prosess, vi på forvaltning kjenne i prinsippet ikke til prosjektene før de er i produksjon. Det er mer organisering enn arkitekturprosessen – håper omorganisering gjør dette til saga blott.
3	3.2	Det har blitt bedre over tid. Før var det veldig fraværende, da jeg begynte her, (SPA), lå utafor, på sidelinja. Mens tida etter når det er utviklet metoder, arkitekturstyring, møtepunkter i forbindelse med systemutvikling er det bra, kan bli bedre, men vi har møtepunktene vi trenger. Det har gått framover. Kan bli bedre? Proaktivitet på hvor vi vil gå, men tror det henger delvis på manglende forretningsarkitektur. Kanskje vi heller begynne å tegne opp transisjonsarkitekturen, mellomting mellom å sette knagger, mål og ha en iterativ utvikling – der er det litt igjen.

3	4.2	Vi har delt opp ansvarsområder, det bidrar. Det skal finnes kapasitet på områdene, det gjør at vi kan sette ut oppgaver og få svar på dem. Tydelig ansvar og kapasitet på de enkelte områdene – litt opp og ned, men når vi har en problemstilling kan vi dele den opp. Dokumenter, samhandling osv.
3	4.2	Mange miljøer å spille på, som gir en smidig arkitekturprosess.
3	4.3	Finansieringsmodellen! Og hvordan vi er organisert rundt den. Vi er for reaktive på når vi begynner å gjøre noe. Budsjettene er ikke satt før i mars, men vi vet jo at de kommer, kan jobbe mer proaktivt. Får ikke gjort noe tidlig på året, så begynner det å rulle, da har man litt kort tid til å gjøre ting.
4	4.3	Finansiering er en hindring. Det har jeg jo nevnt. - Kan du utdype det med finansiering? - Det kombinert med at ting er overlevert til leveranser, da er toget starta, da skal det leveres, sånn som noen trodde det var, da er det ingen smidighet, det går i 80 km/t uansett. Noen eksempler der det hadde vært hensiktsmessig for å redusere risiko eller heve kvalitet, å utsette noe til neste år, ikke mulig pga finansiering, da kan man ikke disponere pengene som er gitt for det ene året neste år. Sånt tar veldig mye tid, finne workarounds, diskutere, følge opp, der vi egentlig har en enkelt løsning, men har ikke finansiering etter årsskiftet.
5	3.2	Lite samhandling med dem [IKT-arkitektur] – noen møter innimellom, begrensa, får det overlevert hvordan det skal gjøres. Usikker på når de er inne – når jeg kommer inn er målbildet satt. - Hadde det vært nyttig med mer kommunikasjon med IKT-arkitektur?

		– ja, det handler om det samme som når vi jobber med fag – det er jo en faginstans, de har sine krav som skal løses gjennom design – og det å forstå det man faktisk prøver å oppnå, det er vanskelig gjennom en figur eller et dokument. Det ligger tanker bak, men det kommer ikke nødvendigvis fram på papir. Skulle gjerne hatt mer samarbeid – må ha god arkitektur på NAV, så mange systemer, vi leverandører må skjønne hva dere prøver å oppnå, og hva dere ønsker å involveres i – jeg vet ikke nå hva jeg trenger å løfte som en arkitekturbeslutning – hva er overordnet mål for Ditt NAV osv. Overordnede bilder på det vi lager, hvor passer det inn.
5	3.4	Budsjettmodellen til NAV gjør at vi ikke kan jobbe smidig – vi har en pott penger i prosjektet, når det er over må vi bare forlate arkitekturen, vi har ikke penger til å forvalte den, endre forbedre, ingen veier for å få midler til det, utenom FKON.
5	4.3	Kontrakter og arbeidsform knyttet til målpris og budsjettprosessen fordi den hindrer løpende vedlikehold på arkitektur- Det å innimellom kunne ha litt tid til å se på sånt uten at det er koblet til prosjekt. På samme måte som dere jobber med overordnet arkitektur trenger vi å kunne se på om vi går i riktig retning osv. Det tror jeg er til hinder, at man ikke ser at det er behov for videreutvikling av løsningsarkitektur – nå skjer det bare noe hvis det er et prosjekt, og da er det pressa på midler. Noe annet er at det er for lite kommunikasjon – vet ikke om det kommer av prosesser eller dårlig tid.
5	5.3	Det er vanskelig å jobbe med dette på NAV, det er så stort, så mange kontorer, så mange forutsetninger som må brytes ned, så mange småsjefer som skal ha det på sin måte, stikker kjepper i hjulene for å lage brukervennlige løsninger – trenger en arkitektur for å presse det gjennom.
6	3.2	Det har blitt bedre med OAM-møtene, men IKT-arkitektur jobber jo så tidlig i prosessen før vi er involvert, så det er ingen som vet helt hva de driver med når vi får ballen. Så kommunikasjonen kunne vært mye bedre, og med mer samarbeid gjennom hele prosessen, og at arktiktene var knyttet til et fagområde og var eksperter på det, ikke så generelle.

6	4.2	(tenker lenge) Man har jo prøvd å dedikere arkitekter til kontorer, det hjelper litt, men fortsatt ikke nok «neppå» og konkrete. Men litt med organisering, kontorer er veldig store, dekker mange domener. Jeg tror det å jobbe på samme dokumentasjon og tettere på utviklingsteamet – vi må komme dit før vi kan jobbe smidigere og levere hyppigere med kvalitet, uten å koordinere oss i hjel.
8	5.1	Jeg tenker på det at skille mellom arkitektur i NAV og utvikling i NAV det et gigantisk hinder for smidig utvikling og gevinstrealisering og styring i forhold til målbilder og alt mulig, og det at man ikke fram til nå ikke har erkjent at man driver med it og utvikling, det ... jeg er så glad for ... hovedgrunnen til at jeg begynte i NAV, man trenger å eie løsningene sine helt ned på kodenivå. Jeg tror det å skille ut arkitektur som en egen funksjon med en egen verdi uten utvikling ikke er en god måte å gjøre det på, når slutter utvikling, når begynner applikasjonsarkitektur, når begynner løsningsarkitektur.... Virksomhetsarkitektur derimot er et eget fagfelt, det trenger ikke være så tett knytta til kode, men de andre nivåene, har mye å spille på hverandre, for å utvikle gode løsninger må løsningsarkitekt og applikasjonsarkitekt spille sammen, det handler om de samme tingene, man bare jobber på forskjellig abstraksjonslag. Jeg tror veldig på det.
9	3.3	Det var et eksempel på at fagsiden bruker penger de ikke burde brukt, hatt is i magen, tåle at det er ikke er komplett nå, få opp finansiering av det ferdigstilles neste år. Usikkert om det som ble laget noen gang vil bli kjørt i produksjon. Det går tilbake til hvordan du finansierer utvikling i NAV, årlig finansiering, vet ikke hva du får neste år, det som ikke brukes i år eller i prosjektet, det blir borte igjen. Burde kunne beslutte at man ikke bruker pengene nå, men neste år. Da gjør man ikke-optimale arkitekturvalg
9	DV	(Om finansiering) Det må jo ikke nødvendigvis være sånn [at man får ketchupeffekt]– det som mangler er en produktkø man kan ta av, man skal «fore neste sprint» - du kan velge ut fra sannsynlighet, dette kan vi jobbe med, uten å sende det til konstruksjon, men ha en produktkø som er bedre enn i dag. Fordrer styring av produktkøen, ikke umulig.
Kontekst - Prosjektenes egenskaper		

A	B	Svar
2	3.1	Tilsynelatende stilles de samme kravene til digiSYFO som til FO, men så er forskjellen på prosjektene enorm (Størrelse, budsjett), i en mer smidig metodikk må man se på hvilke kontrollmekanismer man må ha, følge opp, hva kan man overlate til prosjektet. Der har vi en vei å gå. Samme planverk, milepæler, månedsrapporter som andre prosjekter.
2	4.3	Det går jo på hele apparatet, hvordan vi er skrudd sammen, hvordan prosjektet godkjennes, leveranseorganisasjonen. Det er jo ikke tilrettelagt, men fordi det er mange gode krefter kan vi få til noe likevel, da håper jeg vi kan lage en prosess for smidig, som kan leve side ved side. Som første [smidige] prosjekt er det mye å gå opp.
3	4.2	Ulempen nå, det er det med organisering, at vi legger alt i HL, prosessen kan brukes om vi har flere leveranser også.
4	3.7	Da hadde vi hatt enklere produksjonssettinger, nå putter vi inn masse, ser ikke avhengigheter før det nærmer seg, avhengighetskart med piler på kryss og tvers. Ikke «for lite arkitekturarbeid», men for isolert, og ikke fokus på konsekvenser for produksjonssetting, rekkefølge, det ser vi først på i konstruksjon, men med de dreiningen vi nå skal gjøre, så må vi ha bedre kontroll på det! Så vi kan produksjonssette i riktig rekkefølge. Og få til løsere koblinger selvfølgelig, men det er ikke gjort over natta, vi har de systemene vi har i mange år framover.
4	4.3	Og så tenker jeg at hele leveranseprosessen knytta til HL er en hindring for smidighet, når ting overleveres skal alt være klart, arkitekturarbeid skal være gjort i forkant, hvilke komponenter og avhengigheter, men så ser vi at vi må ta en runde til, ta inn endringer, da har vi ikke noen god prosess for det, må til kongen i statsråd, endringsanmodninger om den minste ting, inn til koordineringsgruppe med ledere som ikke forstår hva det dreier seg om, den prosessen skiller ikke på hvilken type endring, uansett skal det til koordineringsgruppa, det er nok den største hindringen for smidighet.

4	4.4	Noen mener jo at i en helt smidig verden trenger man ikke arkitekter i det hele tatt, jeg er ikke enig, men om det bidrar til smidighet? Det vet jeg ikke, det hadde vel ikke vært noe bedre uten. Det blir vel «smidigere», men kvaliteten blir jo ikke det samme.
7	4.4	SLEM-prosessen gjør det, hvor smidig den er... Det er satt inn i iterasjoner, kontrollpunkter, får inn den ordbruken, på godsida gjør jo SLEM smidig, men svakhetene er jo når det blir omstendelig, hovedleveransene, da blir det lange løp, man får ikke sånne minimumsversjoner. SLEM støtter smidig, men det blir lange prosesser.
7	4.4	Hovedleveransen skal ha 4 kode-iterasjoner, 6 testiterasjoner, kodefrys... SLEM støtter opp om smidig arbeid, men det blir for langt og omstendelig. HL får veldig IT-fokus, i FO, vi prøver å jobbe mer tverrfaglig, smidig, men gjennom hovedleveransen blir vi dratt inn i sånne IT-rutiner og IT-verktøy, fokus på Jira, det tar litt av, vi blir veldig dratt inn i de IT-rutinene i stedet for å jobbe tett på forretning. Være med på å se på forbedringer i løsningene og jobbe bedre sammen. Det er to verdener. Forretning ønsker nå å jobbe veldig smidig, hovedleveranse og SLEM har veldig IT-fokus. I HL2, statusmøtene, lurer noen ganger på hvorfor sitter jeg her, hører på utfordringer i PESYS, dyrking av SLEM, verktøyet, «du har ikke oppdatert risikomatrisa i Jira», selv om vi har risikoarbeid, jobber med forretning. Som prosjektleder ønsker jeg å jobbe mer mot prosjektleder på forretning, og tone ned leveranseledelsen.
9	3.1	Slippe hovedleveranser der alle påvirker alle og man trenger felles testing. Mange av hl-aktivitetene er jo små ting, likevel kjører man felles ytelsestester og driftstester man ikke hadde trengt for de mindre tingene, for de større kunne man overlevert det til prosjektene å gjennomføre omfattende test.
Kontekst – Kompetanse og kultur		
A	**B**	**Svar**

2	2.1	Prøver å gå fra å være «politi», til å koble meg inn der det er viktig, være inne underveis, så kravene blir oppfylt, og kvalitetssikringen ikke blir noen stor jobb.
2	2.1	Lang erfaring med militære arkitekturrammeverk, som jo gir mye mer retningslinjer på hva som skal uttrykkes, hvilke views når, mye sterkere på språk, metamodeller for hvordan ting uttrykkes.
3	2.1	Har erfaring NAVs metoder, basert på TOGAF. Ikke så mye formell modellering, mest enkle skisser, diskuterer med interessenter i løsningen, litt avhengig av hvem du snakker med har du forskjellig presentasjon av innholdet. For de funksjonelle må man tenke på de egenskapene de er opptatt av og hvordan det fungerer, hvis det er mer mot teknisk eller drift er det mer på det, hvis det er helhet opp mot arkitektur er det innkapsling av ansvar, hvordan brikke i et større landskap Tilpasse kommunikasjon til aktørene. Ikke så mye formelle metoder, annet enn kommunikasjon.
4	2.1	Jeg har erfaring med TOGAF, har sertifisering på det. Ellers er det mye «best practice», finne ut hva som funker. Tidligere har jeg ikke jobba med TOGAF, bare deler av det, ofte ikke noe strukturert arkitekturrammeverk og metodikker. - Hva har dere gjort da? - Har brukt mye modellering, EA og den type verktøy, ellers mye prosjektmetodikk som har vært brukt, som konsulent bruker man da kundenes metodikker, [tidligere oppdragsgiver] brukte mye Scrum, en del sånn type fossefallsprosjekter, litt sånn som her, overordnet prosjektmetodikk som er veldig faseinndelt, men noen smidig inndeling i konstruksjon. De fleste store prosjekter har jo en deadline, da skal vi i produksjon. Litt blanding av arkitektur der vi har hatt bevisst forhold, definert et målbilde, forsøkt å jobbe etter det, andre tilfeller der vi bare har prøvd å få ting til å funke, sånn er det der jeg jobber nå, bare kvalitetssikre at ting er i henhold til målbildet, ta tak i det som må ta tak i. Det er mer en operativ arkitekturrolle, ikke en strategisk arkitektur-rolle.

5	2.1	Ikke erfaring med spesifikke teknikker, ikke kursing i spesifikke tekniker. Veldig for smidig tilnærming, viktig med overordnede rammer men – fan av å sitte sammen og diskutere løsninger – sitter aldri alene og designer løsning, det gjør jeg sammen med utviklere, flere arkitekter fra oss, diskutere, begynne å tegne, få det ned på papiret – det er den kjedelige jobben. Ikke fan av rigid prosess på det – det er ikke jeg som vet hvordan dette funker i virkeligheten, men utviklere som jobber med i hverdagen – jeg kan legge til ett «ørnesyn» på det. Bruker tavle som verktøy i diskusjoner. Noe basert på UML når det skal ned på papiret, om det er helt riktig UML... men det er det jeg kan.
6	2.1	Nei, det eneste er modellering og enkel UML, jeg har Archimate-sertifisering, og nå skal vi være med på ATAM-prosessene, men ikke mer enn det.
7	2.1	Ikke operativt, men har jo vært sentralt i alle prosjekter jeg har drevet med. Prosjektleder i 15-20 år i store organisasjoner, og de har sterke arkitekturmiljøer og –enheter. Har jobbet mye i [annen virksomhet], det er liksom min hjemmearena. Arkitektursertifisert i [annen virksomhet]s arkitektur. - Har du noen erfaring med rammeverk eller metoder? - Bare selvstudium på TOGAF, tenkte på å ta sertfisering, men ikke kommet så langt. Men i [annen virksomhet] var det kurs med eksamen, for å bli godkjent som prosjektleder der, måtte kjenne, arkitekturprinsipper, -prosesser og slikt, hvordan prosjektet må forholde seg til arkitektur.
1	2.1	[Smidig erfaring fra tidligere arbeidsgiver] Ikke smidig leveransemessig, men i prioriteringer. Tilsvarer litt hvordan det har vært jobbet på NAV, hvor smidige er vi?
3	2.2	Scrum i mange varianter. Har vel ikke utført scrum på det «idealnivået», man kommer vel aldri dit. Drevet littegrann med lean-tilnærming på oppgaver vi gjør nå. -> Litt arkitekturarbeid og operativt arbeid, eksempelvis jobbe med estimeringsmodeller mens du

		har arkitekturarbeid å gjøre. I linja blir man preget av mange oppgaver lite tid på hver oppgave, dybde er fraværende. Vi har satt opp tavle for de tekniske, få oversikt over hva som skjer, plukke dagens oppgaver. Planlegger ikke sprinter, mer å ta en oppgave av gangen, ikke for mange oppgaver av gangen (kanban-aktig)
4	2.2	All min smidig-erfaring er som deltaker i smidig-prosjekter. Noe med kanban i et delprosjekt i [tidligere arbeidsgiver]. Men jeg vil si det stort sett har vært «smidig tilnærming», ikke veldig klart «Vi kjører scrum». Det er mye fossefall. Her i NAV er det jo lite smidig, i alle fall det jeg er involvert i, selv om vi kjører litt iterasjoner.
5	2.2	begynte med Scrum i forrige prosjekt – beveget oss over til mer kanban/lean-ish, nesten droppe iterasjoner til slutt, fokus på å få brukerhistorier gjennom, løpende prioritering, reprioritering av kunden. Til den mer smidige varianten her på NAV, hvis den kan kalles smidig, SLEM-prosessen. Jobber smidig internt i team, begrensa hva som faktisk er smidig der. Der jeg kom fra anser jeg til å være lean, til å være vannfall igjen her. Det er i alle fall mitt syn på hvordan vi jobber her.
6	2.2	Jobbet mest med Scrum, og teknikker fra XP, f.eks. TDD og parprogrammering. Kan en del om Lean, kjenner til metodikkene, og devops interesserer meg. Ta prinsippene og anvende dem, tilpasse dem kontekst, ikke nødvendigvis følge boka, når det passer med hvilke metoder. Ikke innføre fabrikkprosesser ovenfra ned, utviklere må organisere seg selv, de er smidige av natur, det er ofte organisasjonen rundt som ikke er det, vi må legge til rette for at de kan få ansvar og faktisk jobbe smidig. Krever noe annet av kunden enn det vi er vant med.
7	2.2	Ja, mitt første smidigprosjekt var i [annen virksomhet] i 2007 eller 2008. Prosjektleder for forretningssiden, it-prosjektleder hadde vært på kurs, kom til meg og ba om støtte, det syntes jeg var fornuftig, det var en god erfaring, prosjektet fikk seg et løft, ikke en revolusjon men felles tilnærming og metodikk, det var fornuftig. - Scrum, har også tatt scrum-kurs i ettertid, CSM. Scrum-master sertifisering.

8	2.2	Scrum er smidig på støttehjul, det kan hjelpe til med transformasjon fra vannfall til smidig, men det blir også fort rigid, det har så mange seremonier, du skal ha scrum master, og retrospekt og produkteier osv. I [konsulentselskap, tidligere arbeidsgiver] kjørte vi jo sånn, i [tidligere prosjekt i NAV] var det scrumteam og scrummastere og scrum of scrum, men i en ramme av vannfall, så «barna kunne leke smidig» inne i vannfallet. Det funker ikke så veldig bra, man har ikke det ansvaret da... Da jeg kom tilbake i jobb etter pappaperm, det var på slutten i [konsulentselskap, tidligere arbeidsgiver], fikk i oppgave å ta med meg fem nyutdanna inn i et team for å løse en oppgave for [annen virksomhet], lest masse om smidig i pappapermen, i stedet for å låse metode og leke med teknologi, gjorde jeg det motsatte. Løste teknologien, lot dem prøve seg fram sjøl. De satte jo bare i gang, jobba hver for seg, spurte etter en uke, hvordan går det, så ble vi enige om å snakke sammen, de lagde en tavle, som kom opp helt av seg selv, men det var mange andre ting fra Scrum de ikke gjorde. Parprogrammering dukket opp helt naturlig.
1	3.5	Det som av og til skjer er at forståelsen hos prosjektledere spesielt, som synes arkitektur er et hinder for fremdrift, de vil bare produsere kode for å svare på funksjonelle behov. Noe av det som vi traff – utbetalingstjenesten – det var viktigere å utvikle enn å lage arkitektur, den har litt karakter av det. [Forvaltningsområde] er kjennetegnet av det – lager store klumper av funksjonalitet, ingen [utenforstående?] forstår noe før det er for seint, og da er veien lagt, funksjonene beskrives som store klumper, for lite innsyn, valgene blir tatt før man får innsyn eller forstår.
1	4.1	Vi må gå til å være bidragsyter, ikke være «fy, fy, huske det og det». Det det hindrer er at de som skal få til noe opplever oss som «hassle», ikke som en bidragsyter.
2	2.1	Prøver å gå fra å være «politi», til å koble meg inn der det er viktig, være inne underveis, så kravene blir oppfylt, og kvalitetssikringen ikke blir noen stor jobb.
2	4.2	Er det prosessen eller menneskene... Det er veldig mange som er støttende fordi man har et ønske om å bli mer smidig, og derfor prøver man å bidra til at ting skal fungere, men om det er så mye prosessen, nå tror jeg det er mer enkeltmenneskene enn prosessen i

		seg selv. Tror det også handler om evne til delegering, tillit, hva kan man få lov til innenfor et område, at man ikke må løfte alle ting til et annet nivå.
2	5.1	Og så ser jeg jo at siden jeg i mange år har sittet og vært i en kontrollfunksjon, verifikasjonspunkter, kontra det å få til ting der og da, det er en utfordring, hvordan får man det til sammen, de gode målbildene og retningslinjene og prinsippene som gjør at man tør å slippe litt kontroll, spesielt for arkitekturseksjonene. «Alltid om vi bare hadde vært litt lenger, da kunne vi fått det perfekte systemet, man må tørre å gjøre det beste ut fra det man har her og nå, og tørre å ta beslutninger ut fra det man vet her og nå, og skape mye muligheter.
4	2.1	Det er jo en utfordring som arkitekt, at man kan bli litt teoretisk, for lite praktisk. Leverandørene er veldig praktiske, de skal faktisk levere kode som fungerer, ikke bare målbilder.
4	3.2	Det er jo også en greie: Der vi er veldig avhengig av leverandøren [i konsekvensutredningen] og ikke har kompetansen internt, og de tekniske ressursene er opptatt av produksjonssettinger og rette feil, gjør ikke den typen vurderinger.
4	DV	Fokus på å finne feil er generelt, er en sykdom vi har her, henger sammen med frykten for å gjøre feil. Istedenfor å tenke «hva er intensjonen bak?» tenker man «her skal jeg finne en feil, da har jeg gjort jobben min som kvalitetssikrer».
4	DV	Jeg tenker jeg blir glad når jeg leser det her [foil om arkitektens rolle], men skal man få til det, gjennom OU kommer samme tilbakemelding, må gjøre mindre av noe annet da, mindre politi og tåkefyrste. Henger sammen med tillit og ansvar.
6	3.1	Vi mangler systemnær kompetanse på teknisk arkitektur, det blir overordnet i tidlig fase. I mitt kontor har vi ingen arkitekturkompetanse, bemerkelsesverdig. […] Det er ikke tydelige målbilder og vi her ikke med på å forme utviklingen av systemene, vi bare bygger på den tekniske gjelden. Det flyter litt på leverandørens premisser uten at vi har kontroll selv

6	3.2	Så kommunikasjonen kunne vært mye bedre, og med mer samarbeid gjennom hele prosessen, og at arkitektene var knyttet til et fagområde og var eksperter på det, ikke så generelle.
6	4.2	(tenker lenge) Man har jo prøvd å dedikere arkitekter til kontorer, det hjelper litt, men fortsatt ikke nok «neppå» og konkrete. Men litt med organisering, kontorer er veldig store, dekker mange domener. Jeg tror det å jobbe på samme dokumentasjon og tettere på utviklingsteamet – vi må komme dit før vi kan jobbe smidigere og levere hyppigere med kvalitet, uten å koordinere oss i hjel.
7	3.4	Andre roller i prosjektet, forretningssiden, vil at ting skal gå framover, «er arkitektur så viktig da?».
8	3.1	[…] det er en slags mistillit til konsulenter, her kan det komme noen som har en annen agenda enn å gjøre NAVs beste og det ligger ganske langt framme i bevisstheten da, og det er ikke umiddelbart sant at det er sånn, skal ikke si at det ikke finnes andre agendaer her, det har i alle fall ikke vært sånn for min del, og jeg vet om mange andre konsulenter her også, de har egne meninger og egen integritet og er her for å gjøre NAVs beste. Jeg ønsker jo tillitsbasert organisasjon, ikke mistillitsbasert, det er vanskelig, men handler jo bare om å bestemme seg for å stole på noen eller ikke. Tror vi er litt uenige om det der da... ser dine argumenter også, og har sett det selv i noen porteføljer også, her burde det vært mer styring... det er ikke helt binært det der. Ville likevel holde tillit som en fanesak. Veldig få som er her for å ikke gjøre NAVs beste. Da er det heller misinformasjon enn malice..
	DV	• INFORMANT 2: Vi har en kultur der kontroll er svært viktig, det er en historikk, prosjekter der man ikke har fått det til. Mange som kontrollerer andre, det får fokus, overlevering og kontroll, jobber ikke etter metoden og sammen. • INFORMANT 1: Kontrollfokuset er alt for sterkt. • (Flere nikker) • INFORMANT 8: Enig, ikke ekspert på SLEM, men forstår det som om kontrollbehovet er implementert i SLEM, eller tolket inn i SLEM. Det er jo litt som organisasjoner som tar i bruk Scrum, det blir rigid, Scrum i seg selv blir viktig, SLEM i seg selv blir viktig. Men det er jo mindset som er viktig. Følge metoder slavisk tjener vi ikke på lang sikt.

		• INFORMANT 4: Differensiering er jo viktig da – det henger sammen med kontroll • INFORMANT 2: Det er jo vanskelig, du sitter i et mønster, det du følger, vanskelig å avvike. Mindsettet må jo endres for at man skal kunne differensiere. • INFORMANT 1: Vi må begynne å stole på at folk gjør jobben sin. Det er fundamentet. Da må måten man kontrollerer være annerledes • INFORMANT 3: Risikobasert • INFORMANT 1: Noen steder er det risiko, man skal etablere fundament for framtiden • INFORMANT 4: Re-etablere tillit, noen steder er det ikke tillit. Ikke sånn at alt vi gjør kan føre til at folk ikke får penga sine. Det argumentet brukes «over alt»
Kontekst – Systemenes egenskaper		
A	**B**	**Svar**
5	4.4	Aura! Fantastisk. De er alltid kjempepositive, legger til rette og hjelper oss- det er jo ikke bare arkitektur, men at de legger til rette for overvåking og utrulling osv. Veldig mye av dette er jo forutsetning for et mer smidig testmiljø også. Nyter mer og mer av det at det er mulig å automatisere.
6	4.3	Teknologien – systemlandskapet vårt, hvordan områdene er inndelt. Store saksbehandlingssystemer som går på tvers av hverandre og inneholder samme type informasjon – mye teknisk gjeld og mangel på generell teknisk kompetanse i NAV, arkitektene har kanskje ikke så mange sparringspartnere i kontorene? Blir da direkte mot leverandør. Vi har disse mellomleddene at det skal gå gjennom TA og OA osv. Det har jo med hvordan du deler inn systemene også – hva de har ansvar for og den arkitekturen innafor de forskjellige fagområdene de burde kanskje være mer ulike enn de er i dag – det er vel ikke så smidig med ovenfra ned styring, prøve å kontrollere

		alt – det er jo fordi vi ikke har tydelige grensesnitt mellom områdene, ikke godt nok modulært og innkapsla, da blir det orkestrering og koordinering og hierarkisk å jobbe med arkitektur.
9	3.1	Testmiljøene våre ligner jo ingen ting, det tar ukesvis å sette opp et testmiljø i NAV, det er jo et hinder for å få ting ut kjapt. Du kommer deg ikke gjennom testmiljøene, det er krevende å få ting ut i test. Selv om du hadde klart å sette opp automatisk test, får du det ikke gjennom testmiljøene. Men hvis man kom seg gjennom det, da...
9	5.1	En annen ting jeg ser er at man definerer en del teknisk gjeld man «sitter på», man burde kanskje finansiert dette separat, noe blir jo tatt av FKON, men det blir stort sett liggende til man toucher innom det området med funksjonelle endringer og fagsiden til å ta litt av kosten. Litt samme sfære som systemdokumentasjon, det er ikke faglig relatert, du får ikke finansiering til det, det «lønner seg» ikke på kort sikt. Det er alltid noen lovendringer eller noe som er nærest nesa di.

Arkitekturprosessen - Målsetninger

A	**B**	**Svar**
1	2.2	Arkitekturkvalitetssikring – alt fra «bittelite», kanskje bruker litt for mye tid på det, til det det store.
1	3.4	ofte er det summen av omfanget – hver for seg ser bitene kanskje fornuftige ut – det er jo krevende å finne ut hva som er nok. Skal man jobbe smidig arkitekturmessig, må man jo ta risiko, produserer kanskje litt for lite noen ganger, men det må man rette opp det er smidighet. Det kan være lett å gjøre enda litt mer for å «føle seg trygg nok».
1	3.4	I Prosjekt 2 også, kanskje tørre å lage den første piloten på saksbehandling, lage en prototype, fem utviklere som kunne utfordre arkitektene. Det hadde vært mer produktivt, noen avklaringer kunne kommet av seg selv. Kunne vært en del av en pilot. Det er noe av det dummeste vi holder på med her, ikke så mye at man gjør ting feil, men jobber for mye med arkitektur uten å pilotere, ikke

		tenke poc, så kaste, hvis det fungerer skal vi gå videre med det, noe av det viktigste med DDD er å utvikle, lære og så gjøre om på ting. Ha med arkitekter så man ikke går feil, men så tune underveis. Det er vanskelig å si at man har gjort «for mye» mer at man bruker så j'la mye tid, en tidstyv mer enn feil, en kostnad som man ikke får tilbakebetalt. (...) man tenker uten å teste ut det man gjør. Kan heller sette i gang med et lite utviklingsteam.
1	4.1	Ja. Det som hindrer er den kvalitetssikringsfunksjonene vi har på behov og konsekvens – vi må gå til å være bidragsyter, ikke være «fy, fy, huske det og det». Det det hindrer er at de som skal få til noe opplever oss som «hassle», ikke som en bidragsyter.
1	4.1	Og vi er til hinder for småsaker – kontorene må kunne ta ansvar selv, og vi må kunne stole på dem. Vi som jobber sentralt må passe på de store, vanskelige tingene, nå jobber vi for mye med smått. Jeg er litt bekymra for det.
1	4.1	Nå har vi noe som heter MFA, kan merke noe tidlig, så vi slipper å se på det senere. En del saker kan områdearkitekter og TA [Teknisk ansvarlig i forvaltningsenheter] ta kvalitetssikring på. Kvalitetssikre hvordan man endrer linjeavstanden på en utskrift. Det er dårlig økonomi at jeg bruker tid på det – hadde vi heller vært mer inne i [system] kunne vi gjort en forskjell.
2	4.1	[…] men skal man gå over til å være mer smidig, se på beslutningssykluser, hvem er involvert i hva, annerledes enn vannfallsprosjekter, hovedleveranser osv. Man kan ha ulike løp for de ulike modusene, modus 1 og modus 2 prosjekter. Annen vei gjennom for smidigprosjektene. Og å tenke jeg at når man skal kjøre smidigprosjekter, de bør være mindre, litt mer tydelig hva man skal gjøre, hvilken vei man skal kjøre, ta avgjørelser underveis som ikke har store konsekvenser – hvis det er større konsekvenser må det ta den tiden det tar.
3	3.5	En ting vi mangler – får aldri tid til å gjøre konsolidering av arkitektur på egne bein, bare i prosjekter som har penger, evig kamp om de skal ta det eller ikke, det som faller mellom blir ikke løst, og vi må kompensere i stedet for å løse ordentlig.

4	3.1	Men generelt fungerer prosessen godt. På enkelte områder er det kanskje for mye detaljstyring? - Har du noen eksempler på det? - Ja, det går litt på tillit, da tenker jeg ikke på et som skjer i beslutningsmøter, men i leveransene, er vi oppsatt på å kvalitetssikre det som allerede er kvalitetssikra, er veldig redd for risiko, ny runde med kvalitetssikring av det som allerede er gjort, f.eks. av av prosjektet. Da er vi borte i smidigheten: Vi er skrudd sammen for Plan-Build-Run, store overleveringer, når vi bryter med den prosessen, da er vi veldig risiko-averse, redde for endringer underveis. Men i de store overleveringene tar vi alt for god fisk, men så går det litt tid, det kommer noe nytt, da stoler vi ikke på det arbeidet som er gjort. Litt i forhold til forvaltningskontorene og hvilke mandat og rammer har de til å ta beslutninger på egen hånd, burde de fått større ansvar også for arkitekturen. Det praktiseres i alle fall ikke.
4	3.3	Men jeg føler jo noen ganger så brukes det litt, siden vi har et kvalitetssikringsregime, litt for mye fokus på å kvalitetssikre ting som egentlig allerede er kvalitetssikret, stole på det, eller fordele ansvaret. Man blir litt prega av kontrollregimet som er rundt, at alle endringer skal opp til en koordineringsgruppe som består av ressurser på fagsida. En tilleggsgreie: Når ting skal avvike fra det planlagte løpet, av arkitekturmessige grunner eller annet, når noe skal tas ut, da føler jeg ikke det er medbestemmelse. Da er det en mye tyngre prosess å stoppe noe, enn å få noe inn. Noen få tilfeller der dette har skjedd, fordi det er for dårlig kvalitet. Type «dette kan ta ned stormaskinen», da er det en veldig tung prosess å gå tilbake til tegnebrettet, dette kan vi ikke produksjonsette, det henger sammen med finansiering og at vi har tatt på oss et leveranseansvar ovenfor fag om at det skal leveres. Da må det stoppes i porten til drift.
4	4.3	Og dette kontrollregimet som vi har, at man tror at man har behov for å rapportere og ha kontroll, når man måler hvor mange testcaser som er levert osv, men differensierer i liten grad i forhold til risiko, uansett hva du skal gjøre skal du gjennom samme løypa,

		også ift produksjonssetting. Skal det gjøres store endringer som kan føre til at brukerne ikke får penger, da må vi ha veldig god kontroll, men på andre områder kunne vi sluppet opp i større grad. Det er jo en arkitekturmessig risiko/konsekvensvurdering.
7	3.1	Det må være noen portvakter, med ordentlig mandat, har vært borte i det motsatte også, der manglet man det [mandatet] og portvaktene, og det så vi i prosjektene og leveransene at det var mye det ikke var tenkt på.
7	3.2	Den er fungerer bra, det tror jeg arkitektene i prosjektet også synes. De føler jo sikkert også at IKT-arkitektur er litt tungt, men jeg mener det må jo være en portvakt. Det å ha orden i prosjektfundamentet, det har fungert bra i NAV, det er orden i eget hus. Man kan si det er omstendelig, men det gir en trygghet.
7	4.2	Sterk portvakt gir mer sikkerhet for å kjøre smidig når du er i gang. Diskusjon rundt ATK-rollen, hører fra noen at de skal ha en sterkere rolle. Da føler ikke jeg meg så trygg som prosjektleder.
7	3.7	(i tillegg til scoping) hvor er risiko i arkitekturen, områder som er umodne, slipper vel gjennom nåløyet, men dere må jobbe mer sammen med å ta ned avklaringene som gjenstår når det går videre gjennom 240. At dere ikke slipper ballen når det går gjennom, mer smidig overlevering, ikke alt er modent, men vi i kommer i gang, jobber med tiltakene sammen. Dokumentinnsyn er et eksempel, kunne avgrensa det, dette er fast, det kan dere jobbe med, for det andre få opp det som måtte jobbe videre med, uten at vi har godkjentstempen på alt i 240. Noe er klart nok men ikke alt.
7	3.6	Ja, når det er et godt arbeid og strenge portvakter tar det ned risiko
7	4.2	Tryggheten på at det er modent, det gir navigeringsrom til å kjøre på, så lenge ting er modent er det enklere å kjøre mer smidig, hvis det gjenstår avklaringer er det risiko for planen. Sterk portvakt gir mer sikkerhet for å kjøre smidig når du er i gang.

8	3.6	Det er i alle fall målet... NAV er ekstremt opptatt av å redusere risiko. Alt handler om å ha lite risiko i det man gjør. Intensjonen er det veldig, men jeg personlig , tilbake til FP, man driver jo med så mye i forkant for å ta ned risiko, men man hadde tatt ned vel så mye ved å sette i gang utvikling og vist og korrigert, jobbet iterativt, fem-seks stykker, kunne landa mange problemstillinger som da hadde vært konkrete i steden for abstrakte. Tror intensjonen er å ta den ned, men ikke at man klarer det.
8	4.1	(Etter intervju) At man ikke vil prøve ut ny teknologi, men baserer seg på gamle sannheter, også integrasjonsmønstre... Man er for engstelige for risiko, at man ikke tør å prøve ting som ikke er prøvd før. Det er ganske alvorlig sykdomstegn at man ikke tør å prøve.
Arkitekturprosessen – Analyse, syntese, evaluering, implementering, forvaltning		
A	**B**	**Svar**
1	2.1	Det viktige er å finne ut av ting i samarbeid tidlig. Ikke sitte på sidelinje og tegne arkitektur, men få tilbakemeldinger fra de som eier applikasjonen og forstår hvordan den fungerer. Søke etter det enkle og gode, ikke komplisere. Viktig å tenke på endringsevne og forvaltbarhet, at det er enkelt og forståelig. En viktig rolle som arkitekt da er det med formidlingsevne og kommunikasjonsevne, forklare så folk forstå hvor du vil, så de de griper det og «gjør det til sitt». Det er utrolig viktig i de fasene.
1	2.1	Har ikke fulgt opp sakene gjennom utvikling og produksjon. Det går på tid tilgjengelig.
1	2.1	Det er noe fornuftig i TOGAF, men av og til lurer jeg på om det blir for vanskelig, at vi burde ha noe enklere.
1	DV	Det er andre eksempler på smidig arkitekturarbeid – noen caser i FKON, vi er tett på men delegerer mye, det gjøres i kontorene – der er det ikke arkitekturarbeidet som hindrer framdrift, men andre ting.
2	DV	Vi bruker jo ikke SLEM i det hele tatt i digiSyfo, men vi bruker jo elementer av det, produktkø og analyse og design. Mye tettere vevd sammen, tar med mer input fra utvikling inn i hvordan man skal jobbe videre. <…> Prøver å definere sånne MVPer (Minimum Viable Product)– når vi er klare nok til å si at vi vet hva vi kan gjøre med det, så det blir en leveranse. Vanskelig å si at alt er helt klart, andre

		ganger må vi gå tilbake fra utvikling å finne ut av ting, ikke lett å vite når det er passe klart.
2	2.2	[Leveransemetoden i NAV kalles av noen] «Waterfall Scrum», skjønner det begrepet bedre nå som jeg sitter i et prosjekt som er «ekte smidig».
2	2.2	[Om metode som brukes i smidig forsøksprosjekt] Det går på at man ikke har helt klart for seg hva man skal gjøre, man har ikke gjort alle analyser alt design, før man går i gang med utvikling, interaktiv prosess begge veier, ideer fra utvikling til design, tilbakemelding fra brukertesting og får tilbakemelding, jobber tett sammen med fagsiden. Her er det hele tiden er det å gå opp [med de som har behovene], hva var det du mente, hvis vi endrer blir det bedre eller dårligere, hypotesetesting.
3	3.1	I SLEM synes jeg det er forutsigbart, det går an å etterse løsninger og få oversikt, hvilke beslutninger som er gjort. Ymse kvalitet på forskjellige historier avhengig av team, men da vet man i alle fall hvor det mangler dokumentasjon.
4	3.2	Kanskje ikke så mye med samhandlingen med arkitektene, ikke vært så mye med på det, men kommunikasjonen mellom arkitektur og prosjekter og kontorer og leveranser, men den har blitt litt bedre, men det er litt sånn «de der oppe i åttende, de sitter og tegner og vet ikke hva som skjer», oppfattes som litt for teoretisk og ikke så hands-on. Noen forvaltningskontorer synes vel bare arkitektene lager støy, mener de har full kontroll selv, samtidig som mange av dem har et veldig snevert syn, ser sin egen silo, liten fokus på det som skjer utenfor. Der spiller jo arkitektur en viktig rolle.
4	3.2	Man kunne da vurdert å ta ut dette som en egen leveranse i januar, eller bygge det så det gikk an å ta det ut, hvis man hadde tatt dialogen med leverandør før. Konsekvensutredningen var nok grei nok, men ikke sett opp mot utfordringer med gjennomføring. Det er jo også en greie: Der vi er veldig avhengig av leverandøren og ikke har kompetansen internt, og de tekniske ressursene er opptatt av produksjonssettinger og rette feil, gjør ikke den typen vurderinger.

4	4.1	Gjøres det en vurdering av om noe må inn i arkitekturprosessen, eller om endringen «bare kan gjøres». Det gjøres jo en vurdering, feilrettinger går jo ikke inn, men i større grad, skal vi kjøre arkitekturprosess i det hele tatt, om vi skal ha smidigere løp med mer ansvar til produkteamene, forvaltningskontorene, gi dem mer ansvar. Det som oppleves som et hinder i dag er vel tilgjengelighet og responstid fra arkitektene, man venter på konsekvensutredninger, så får man spørsmål tilbake, ting tar lang tid. Mye fokus på å kutte ned tiden fra behov til prod, der kan man kutte flere steder, men så har det vel vist seg at mye av tiden går med før ting legges inn i en leveranse, om det er arkitekturprosessen som er flaskehalsen vet jeg ikke, men det er nok mye å hente der. Skal det bli smidigere tenker jeg vi må være flinkere til å gi produkteamene tillit til å gjøre definerte endringer.
4	4.4	Ja, godt eksempel er lovendringer, når noe skal iverksettes 1.januar da blir vi veldig smidige, fordi da er det ikke mulig å følge prosessene, lovendringer før 1.1.2016, da vet jeg ikke om det var innom arkitektur en gang, smidig men kaotisk, men det positive var at man fant nye måter å jobbe sammen med leverandør, daglig deploy, testa sammen med leverandør osv, vi ønsker ikke å havne helt der, men mye vi gjorde der kunne vi gjort i større grad. Var mye endring er på Arena, kombinert med at vi ikke hadde helt kontroll, det gikk bra, men var redd for at meldekort ikke skulle fungere, pga endringer på feriepenger.
5	2.1	Veldig for smidig tilnærming, viktig med overordnede rammer men – fan av å sitte sammen og diskutere løsninger – sitter aldri alene og designer løsning, det gjør jeg sammen med utviklere, flere arkitekter fra oss, diskutere, begynne å tegne, få det ned på papiret – det er den kjedelige jobben. Ikke fan av rigid prosess på det – det er ikke jeg som vet hvordan dette funker i virkeligheten, men utviklere som jobber med i hverdagen – jeg kan legge til ett «ørnesyn» på det. Bruker tavle som verktøy i diskusjoner. Noe basert på UML når det skal ned på papiret, om det er helt riktig UML... men det er det jeg kan.
5	2.2	Veldig fan av det – jeg ser at du trenger – jo større bildet er, jo tidligere ute må du være, legge retningslinjer, mene noe om om komponenter og områder – men når man komme til løsningsdesign, da må man kunne endre underveis, kan ikke vite alle løsninger, tekniske begrensninger underveis – noen ting er satt, må forholde seg til – vite hva som kan endres og som ikke kan endres. Være

		bevisst på det når man får oversikt på starten – fagområder, nettverkss-soner osv – tegne opp det – finne ut seinere hvordan det skal samhandles og utformes. Det er sånn jeg har jobba – jeg var nok mer smidig på det forrige prosjektet – der revurderte vi hele tiden arkitekturen vår, gjorde forholdsvis radikale endringer – men her er det mer begrensninger.
5	3.1	Løsningsdesign er ikke-smidig. Satt på forhånd, sånn blir det, vanskelig å gjøre om på noe, hvis vi ønsker å gjøre noe med det seinere, blir det kritisk, anklagende, «hvorfor gjorde dere ikke dette før» - da kvier man seg kanskje for det. Man «klatter», i stedet for å fikse arkitekturen.
5	3.1	Noen ganger legger vi kunstige forutsetninger på oss selv [leverandøren] – for å sikre oss selv - vi ønsker ikke å bli «tatt i etterkant».
5	3.7	vi konsekvensutreder med grovestimat, hvis det kommer flere timer på finestimatet så må vi forsvare det, det er litt merkelig, det gjør at man må gjøre mye mer jobb på grovestimatet. Jeg tror ikke man bør gjøre mye løsningsdesign i tidlig fase, mer hva berøres og interaksjonsmønster. Hvilken informasjon flyter og samhandlingsmønster.
5	4.3	Og SLEM i seg sjøl – er det riktig måte å gjennomføre prosjekter på – eller om det er implementasjonen av det som er feil?
5		Har det [medbestemmelse] i forbindelse med løsningsdesign. Men det er mange nivåer. Jeg har innvirkning på det interne løsningsområdet, det som vi har ansvar for – ikke mappe direkte mellom virksomhetsarkitektur og løsningsarkitektur. Føler at jeg har en viss grad av innflytelse, men ikke så mye som jeg ønsker, men det får man jo aldri. Jeg går ikke rundt og er frustrert, men det er jo alltid noe man er uenig med, men andre kan jo ha bedre løsninger enn meg. Andre ting jeg tenker er feil… tilbake til problemet med at noen på kontoret sitter veldig hardt på forslag de har uten å høre på innspill. Noen ganger litt dumt. Noen ganger blir vi tatt med litt sent. Noe vi har tenkt på lenge, og vi har en forståelse, kunne vært løst enkelt, så kommer rolls royce løsninger som koster 4-5 ganger mer enn det kunne kostet. Det er litt Rolls Royce tankegang på mye som skjer, hadde holdt med en Volkswagen.

7	2.3	Ja, har for så vidt det i [annen organisasjon], var det et prosjekt, jeg var prosjektleder, digitale kanaler, vi prøvde jobbe med arkitekturen løpende underveis i leveransen, hadde kjente mål og forbedringer vi skulle ta tak i, vi visste hva målene var, og så jobbet arkitekten løpende med å utvikle ark ut fra det. - Erfarte at vi måtte sette grensa for vårt arkitekturdomene, begynte å jobbe for bredt først, inn mot mellomvare og baksystemer, fikk ikke framdrift, bare diskusjon og krangling, vi må snevre inn, hva er vårt arkitekturdomene, da ble det veldig snevert, men vi fikk veldig godt handlingsrom der, men ikke store grep. Verdikjedeanalyser i de store verdikjedene, hvordan kundesentre opplevde at IT-løsningene fungerte, så på incident management etc, valgte ut problemområder som vi skulle jobbe videre med. Det er vel ikke fasiten, men man må snevre det inn, hva er scopet på prosjektet. Jobbe gradvis med forbedringer av verdikjeder. Vi gjennomførte «lett-på-tå-grep», som ikke trengte stor oppmerksomhet av organisasjonen. Når vi avgrenset fungerte det bra, skulle det vært større hadde det blitt et stort moderniseringsprosjekt og det er noe annet.
7	3.2	Den [kommunikasjon mellom prosjektet og IKT-arkitektur] fungerer bra, det tror jeg arkitektene i prosjektet også synes. De føler jo sikkert også at IKT-arkitektur er litt tungt, men jeg mener det må jo være en portvakt. Det å ha orden i prosjektfundamentet, det har fungert bra i NAV, det er orden i eget hus. Man kan si det er omstendelig, men det gir en trygghet.
7	4.4	SLEM-prosessen gjør det, hvor smidig den er... Det er satt inn i iterasjoner, kontrollpunkter, får inn den ordbruken, på godsida gjør jo SLEM smidig, men svakhetene er jo når det blir omstendelig, hovedleveransene, da blir det lange løp, man får ikke sånne minimumsversjoner. SLEM støtter smidig, men det blir lange prosesser.
7	4.4	Hovedleveransen skal ha 4 kode-iterasjoner, 6 testiterasjoner, kodefrys... SLEM støtter opp om smidig arbeid, men det blir for langt og omstendelig. HL får veldig IT-fokus, i [prosjektet], vi prøver å jobbe mer tverrfaglig, smidig, men gjennom hovedleveransen blir vi dratt inn i sånne IT-rutiner og IT-verktøy, fokus på Jira, det tar litt av, vi blir veldig dratt inn i de IT-rutinene i stedet for å jobbe tett på

		forretning. Være med på å se på forbedringer i løsningene og jobbe bedre sammen. Det er to verdener. Forretning ønsker nå å jobbe veldig smidig, hovedleveranse og SLEM har veldig IT-fokus. I HL2, statusmøtene, lurer noen ganger på hvorfor sitter jeg her, hører på utfordringer i PESYS, dyrking av SLEM, verktøyet, «du har ikke oppdatert risikomatrisa i Jira», selv om vi har risikoarbeid, jobber med forretning. Som prosjektleder ønsker jeg å jobbe mer mot prosjektleder på forretning, og tone ned leveranseledelsen.
8	1.4	Det er konsekvensen av at man har et autonomt team som eier det, kjenner på smerten, da blir applikasjonsarkitektur noe man er ekte opptatt av, ønsker ikke applikasjoner med for stort spenn.
8		Og så det siste halvåret i 2015 i [prosjekt i NAV]. Så på arkitekturprinsipper for modularisering av vedtaksløsningen, så på arkitekturprosesser som de er i NAV. Det som funka best for meg da var å sitte sammen med [annen arkitekt i NAV] og snakke og «rubber ducke» hverandre, teste hypoteser, prøve aktivt å felle hypotesene. Før vi endte opp med skisser vi tok videre til andre interessenter og snakka med dem, gikk tilbake, tok de innspillene jobbet videre med de forbedringspunktene og forslagene vi fikk. Passer ikke inn i noen faser i SLEM eller noe, men en måte å jobbe med løsningsarkitektur på som funka ganske bra, og egentlig ganske likt som vi har jobba med applikasjonsarkitektur i Aura. Handler om å tenke ut løsninger, lage skisser, kommunisere med andre, gjøre iterative forbedringer av dem.
8	2.1	Har aldri jobba med spesielle arkitekturmetoder. Har sett sånne rigide prosesser, men har aldri jobbet sånn selv Har aldri jobba ikke-smidig. Jeg klarer ikke å tenke ut bra løsninger i et vakum. Kanskje Elon Musk klarer det... Jeg trenger feedback og noen å teste hypoteser med. […]

		Men skrive eller tegne ned tankene sine på et vis og få dem iterativt raffinert. Jeg er jo ikke erfaren arkitekt, men mye av det jeg har sett ellers, jeg er skeptisk til for mye prosess, at det kommer i veien for de beste løsningene. Fordi jeg tenker på meg selv som en utvikler, er jeg veldig for å prototype ting, skisser er også kode, å skrive kode, så fort som mulig egentlig. Den koden man skriver da er bare en slags dokumentasjon, braindump av det man tenker der og da, ikke nødvendigvis noe som skal bevares videre, bare en annen måte å tenke på, som kan fungere bra i å kommunisere hensikt. Er konkret og enklere å få feedback på. Selv om man ikke ender opp der, ender kanskje opp med en annen teknisk løsning, så vil første gangen man ser en webapp man har pælma sammen sette i gang gode diskusjoner.
8	3.1	Vet ikke om jeg kan svare på det, sitter ikke så mye i dag. Fra utsiden virker det tungrodd. SLEM. Kanskje ikke SLEM, men hvordan organisasjonen bruker den, veldig mange handovere, og at arkitektur sitter frikobla fra resten av organisasjonen. Det tenker jeg ikke er optimalt. Når jeg sier SLEM så mener jeg disse fasene, implisitt vannfall, med mye handover mellom hver fase, som jeg ikke har tro på. Veldig mye som skal innom arkitektur alltid, tenker det hadde vært bedre om de som hadde behovet involverte arkitektur når det var behov for det.
8	4.1	Mye handover, kvalitetssikring av teoretiske problemstillinger, det er jo at man skal sikre her er det muligheter for at noen kan bruke et API feil, eller se ting som kanskje ikke skulle sett, og da tar man heller og låser det hindra at det oppstår i stedet for å etablere mekanismer som fanger opp at det skjer, pessimistisk vs optimistisk låsing...
8	2.2	Det er veldig gøy i Aura-teamet nå, vi er smidige med egen metode, hele tiden i bevegelse, under stadig optimalisering. Ingen artefakter eller seremonier som er hellige, alt er gjenstand for diskusjon hvis det ikke anses å gi verdi dropper man det. Har vært på en slags reise der sjøl fra mye seremoni til nå å være ganske radikal på smidig.

8	2.4	Største take away fra smidig er å tørre å være smidig med egen prosess, være hard på at selv om dette funka før behøver det ikke funke nå. Finne ut av om vi fortsatt trenger å gjøre dette, har vi tid, hvordan løser vi dette. Hvis alle i teamet mener det er viktig, men ingen kan prioritere det, da er det ikke viktig nok. Smidig med egen prosess. Samme hva prosessen er bare du er en prosess for å endre den.
8	2.4	Det som scrum kaller retrospekt... en eller annen mekanisme for å ta inn det man har lært siden sist. Og demoer. Demoer i arkitekturarbeid må ikke være working software, kan være «her er det vi tror nå», dele litt tanker hele tiden, og ønske mer å få feedback. I Aura gjør vi jo det. Jeg tror det ville vært nyttig i P2 også, på et sånt nivå, at man snakket sammen litt mer, i dag har jeg tenkt å sitte sammen med [annen løsningsarkitekt] for å snakke om det og det, er det noen som lurer på noe. Prøver å dele sine hindringer og framdrifter i større grad. Det tror jeg på, og det å lage til demoer uten kjempeseremoni.
9	3.2	jeg som LA trenger å få et epos gjennom til 300, det skal gjennom noen QAer, fra 100 til 200, 200 til 300, vært borte i litt forskjellig – forskjellige personligheter som gjør ting veldig forskjellig. Noen skal ha detaljert utrolig langt ned i pensjonssfæren for å konsekvensvurdere, mange runder for å få godkjent, noen andre har vært mer «har du fylt ut riktige felter i epos», ikke brydd seg om innholdet. Opplevd ytterpunktene, hvor mye detaljeringsgrad blir egentlig kvalitetssikra i den prosessen, synes vel egentlig at så lenge du ikke påvirker arkitekturen, burde sjekken vært mer har du finansiering, har du fylt ut riktig, ikke dypdykk av de som ikke egentlig har nok detaljkompetanse. Nå i det siste har en del QA-arbeid blitt overlatt til områdearkitekter, det her kan områdearkitekten faktisk ta, så lenge det overordnede ser greit ut, ikke påvirker målbilder, la de som har domenekompetanse gjøre qa. Fra IKT-arkitektur skal ha kontroll på alt og forstå alt. Det synes jeg har vært bra
9	3.4	Noen i gikk alt for langt ned, hva skal arkitektene faktisk gjøre QA av, noen går alt for dypt ned, ikke stoler på at de som har jobbet med epos og løsningsskisser og sånt har nok domenekunnskap. Burde egentlig bare sett at dette er jo bare videreføring av eksisterende løsning. Tror det er veldig personavhengig, noen er ikke sånn, noen er på en middelvei..

9	3.5	Jeg synes kanskje noe av det jo jobbe med nå, når jeg jobber innenfor [domene], vanskelig område med kanskje litt for lite arkitekturarbeid, lukka prosesser internt i kontoret, hvor egentlig prosessen er at man prøver å sette opp en skisse, så forsvinner kontoret inn til seg selv, dukker opp igjen med hvordan de mener det kan løses, lite synlig for andre hva de har gjort, hvordan de har vektlagt kvalitetskrav. [Et forvaltningskontor] synes jeg er flinke på det, de har en områdearkitekt, usikker på om [et annet forvaltningskontor] har det... Det er kanskje en viktig rolle, forankringspunkt, en rolle en arkitekt som har hovedansvaret for et område, slipper det bindeleddet mellom FA/TA og en ekstern arkitekt i IKT-arkitektur. Det blir fort at kontoret bare «skal løse det sånn og sånn» uten at det er forståelig hvilke vurderinger de har gjort.
9	4.1	SLEM og hvordan du skal dra gjennom epos passer dårlig sammen med endringshåndtering. Fordi i et prosjekt i en fase der det kommer små endringer, tvinges vi til å at en liten sak, lage et eget epos, qa-prosess med ikt-arkitektur for småendringer, vi kan gjøre noen avtaler om å lirke det til, men SLEM-løpet med alle fasene medfører unødvendig merarbeid for mindre endringer i et prosjekt. Det oppleves som en hindring i prosjektene. Alle prosjekter må jo gjennom det der, det løses ulikt, avtaler om snarveier og sånt, men ikke understøtta i prosessen. Det oppfattes at du må skrive masse dokumentasjon, epos med alt mulig rart. I Uføre fikk vi først blanko, så fikk vi kortere lenke. Nå prøver vi liksom å være «best i klassen», fordi vi har fått skikkelig på pukkelen tidligere. Man har jo lyst til å gjøre det «riktig». - Som i etter boka? - Ja, og at epos brukerhistorier skal være i samme fase og i takt med hverandre. Skjønner jo at det ikke kan være vill vest og alle gjør som de har lyst til, må jo ha noe felles, men jeg synes det er ikke veldig smidig den måten man tvinges til å bruke epos og brukerhistorier som verktøy for småendringer, måtte dra det gjennom alle SLEM-fasene.
9	4.2	Jeg synes jo det er en god og veldefinert prosess egentlig, som kunne gått fortere med de justeringene jeg har nevnt. Jeg synes jo man greier å levere til ... holde seg innafor 3-ukers-sprinter da, man greier jo å definere behov som lar seg bygge behov i løpet av kortere

		tidsrom. Man har definert en metode som gjør at man stort sett klarer å konstruere ting innafor en treukersperiode. For en stund siden var det jo overhode ikke sånn, da var det lange løp med stor risiko, men nå synes jeg jo SLEM er en ganske god metodikk i bunn, selv om det er forbedringspotensiale.
	DV	(Om SLEM) • INFORMANT 1: SLEM muliggjør smidighet, men organiseringen rundt ødelegger, ikke SLEM som er problemet • INFORMANT 3: Det er mest organiseringen rundt [som er problemet], metoden er kanskje litt detaljert, bør tunes • INFORMANT 1: Vi har en verktøykasse vi kunne tune, men den hindrer ikke • INFORMANT 8: Er SLEM ikke forstått? • INFORMANT 1: Tror ikke det er forstått hvordan man jobber smidig • INFORMANT 8: Filosofere mer: Hvis det i stor grad ikke forstått, er det for komplisert eller er vi dumme? • INFORMANT 1: Jeg tror kommunikasjonen er mer rundt detaljer i metoden, mer enn om hva vi prøver å oppnå • INFORMANT 2: Vi har en kultur der kontroll er svært viktig, det er en historikk, prosjekter der man ikke har fått det til. Mange som kontrollerer andre, det får fokus, overlevering og kontroll, jobber ikke etter metoden og sammen. • INFORMANT 1: Kontrollfokuset er alt for sterkt. • (Flere nikker) • INFORMANT 8: Enig, ikke ekspert på SLEM, men forstår det som om kontrollbehovet er implementert i SLEM, eller tolket inn i SLEM. Det er jo litt som organisasjoner som tar i bruk Scrum, det blir rigid, Scrum i seg selv blir viktig, SLEM i seg selv blir viktig. Men det er jo mindset som er viktig. Følge metoder slavisk tjener vi ikke på lang sikt. • INFORMANT 4: Differensiering er jo viktig da – det henger sammen med kontroll

		• INFORMANT 2: Det er jo vanskelig, du sitter i et mønster, det du følger, vanskelig å avvike. Mindsettet må jo endres for at man skal kunne differensiere. • INFORMANT 1: Vi må begynne å stole på at folk gjør jobben sin. Det er fundamentet. Da må måten man kontrollerer være annerledes • INFORMANT 3: Risikobasert • INFORMANT 1: Noen steder er det risiko, man skal etablere fundament for framtiden • INFORMANT 4: Re-etablere tillit, noen steder er det ikke tillit. Ikke sånn at alt vi gjør kan føre til at folk ikke får penga sine. Det argumentet brukes «over alt»
	DV	• INFORMANT 8: Det er jo relevant hvis vi søker en org basert på tillit, er det da behov for SLEM? • INFORMANT 1: Kanskje ikke i den formen, så detaljert • INFORMANT 8: Hvis vi har autonome team som eier sin bit av NAV, så … • INFORMANT 1: Men man ønsker en enhetlig måte å håndtere produktkø på, kunne bytte mellom team osv. Alle kan ikke gjøre alt forskjellig • INFORMANT 8: NAV IT må ha noen krav til grensesnitt. Produktkø inn til et team er et sånt grensesnitt. • INFORMANT 4: Vi skal jo jobbe mer forskjellig, det gjør vi ikke i dag. Mye kontroll og sjekklister og telle det ene og andre, overlevering til drift er ikke kvalitetssikring, bare kontroll.
Arkitekturbeslutninger		

A	B	Svar
1	2.2	Erfaring med metodeprosess, arkitekturbeslutningsmetoden (i NAV), hvordan forankre arkitekturbeslutninger, for å sikre at man har noe å peke på om noen vil ha omkamper osv. Det er et verktøy jeg synes har fungert veldig bra. Å dokumentere valg man tar.
2	3.2	Det er mye fokus på når man gjør noe som går mot, arkitekturbeslutninger, det er viktig, men kunne vært mer fokus på hvor går veien, hvordan kan vi buke dette på en god måte, hvor ha dialog, hvor lærer vi av hverandre.
2	3.3	Ja, det har man jo absolutt [påvirkning på beslutninger], fordi man er jo med i prosessen, og selv om det er andre som tar beslutningene, så har man jo stor påvirkning. Det føler jeg veldig at jeg har.
2	3.4	Som arkitekter er vi jo opptatt av de samme spørsmålene, se på hverandre som gode krefter, nå mål, ikke bruke så mye energi på hvem som har hvor mye beslutningsmyndighet.
2	4.1	Opplever at man er vant til at ting går ganske sakte fart, man har beslutninger basert på at ting tar tid, det er greit, kan levere i neste leveranse, det går fint, når man kommer som smidig prosjekt og trenger beslutning nå, fordi sprinten starter neste uke, det å få organisasjonen rundt seg som er vant til at man kan jo heller møtes neste uke, det kan man si er et hinder.
2	4.4	Som leder kan man si at man vil være smidig, men når det kommer til å måtte delegere beslutningsmyndighet er det vanskeligere – hvilket nivå det treffer den enkelte. Det er lett å mene noe om prinsipper, men når det treffer den enkelte er det brått mye vanskeligere.
4	3.1	Litt i forhold til forvaltningskontorene og hvilke mandat og rammer har de til å ta beslutninger på egen hånd, burde de fått større ansvar også for arkitekturen. Det praktiseres i alle fall ikke.

6	3.3	Ja, jeg føler at jeg kan stille spørsmål, jeg jobber jo med kvalitetskravene, de hjelper jo, ved at jeg kan si at hvis vi gjør det og det, så bryter vi kravene. Jeg er jo ikke arkitekt men jeg vil at folk skal tenke fornuftig, og at de riktige menneskene tar avgjørelsene, ikke at det blir tilfeldig.
7	3.7	den største forskjellen i [tidligere arbeidsgiver] var at der tok folk beslutninger, i NAV blir det mye mer fram og tilbake og omkamper.
8	3.1	Noen er jo tredd nedover ørene, enkelte beslutninger, type teknologikatalog, dette har jeg bestemt, ikke lek med Docker uten at vi har sjekka det... Skulle ønske vi hadde mer medbestemmelsesrett på det. Tror det er noen prosesser her som er laget for et litt annet regime enn satsningen på egne utviklere innebærer. Hvis man har hele den riggingen med at dette er teknologi som er ok og ikke ok, det er jo for å styre leverandører og sikre fornuftig governance av teknologo, og det er jo gitt at man da ikke gjøre det sjøl, men at leverandører gjør det for seg. Men hvis det er NAV sitt hele veien og NAV gjør governance, er det mindre viktig med «divine» føringer for teknologi da. Det er et område der jeg føler vi (utviklere) ikke har tilstrekkelig medbestemmelsesrett. For meg selv, særlig etter at jeg ble ansatt, blir jeg hørt på i langt større grad over alt, og det er jo fint for meg, men det er en greie som er litt skummel også, jeg er jo den samme fyren, samme meninger som før, men nå har jeg ikke stripe på kortet, så da blir jeg ikke mistenkt for å ha en annen agenda. Det merker jeg, nå blir jeg hørt mye mer på.
8	4.1	i alle fall i starten av NAV, alle kunne si nei, veldig få kunne si ja...
8	4.1	Hvis man kan ta beslutninger på lavest mulig nivå er det bra, det fordrer tillit, men hvis man har det som et slags grunnfjell så vil mange av prosessene også kunne bli enklere, da er det ikke så nøye om det har vært innom alle tenkelige interessenter først.
8	4.2	Det at de som sitter ute i prosjekter, de føler jeg har mandat til å gjøre ting, det er viktig, og sånn er det vel i dag, at man er gitt et visst ansvar til å fatte beslutninger. Det er en god ting. Det vil jeg ha mye mer av. At man kan kjøre prosessene der det skjer.

9	3.2	Den andre biten er når du skal ta valg som faktisk berører arkitektur, påvirker forvaltbarhet, løsning framover, lat fra «vi er usikre på hva som er lurt» til «vi er ikke enige», da har det vært en ryddig prosess det å ta det opp til arkitekturbeslutning, ta det til [sjefsarkitekt], kan vi få en avklaring her. Vi må lage grunnlag da, det er jo en kostnad, men det syns jeg har vært en ryddig og bra prosess de gangene jeg har vært borte i det. I etterkant ser jeg at noen av valgene burde vært anderledelse, men i et prosjekt må man ta et valg, det å ikke ta et valg er verre, noen ganger blir det riktig, andre ganger ikke.
9	3.3	Til en viss grad, men egentlig ikke så mye. Føler at arkitekturen og målbildene i stor grad er gitt, har jobbet som LA innenfor et prosjekt, de rammene rundt i den settingen er at du forholder deg mer til arkitekturen som den er, finner løsninger innenfor den, og at det å i et prosjekt skulle si «på sikt hadde det vært lurt å...», men innenfor rammene av prosjektet og budsjettet kan vi ikke gjøre det. Handler mer om rammene og scope og finansiering av prosjektet, ikke så mye om rollen LA i prosjekt. Det er finansiert av fagsiden, og mye fokus på faglige behov som skal oppnås går mye på kompromiss på et langsiktige. Veldig ofte tar man ikke merkostnaden nå, siden de som finansierer det ikke ser gevinsten umiddelbart. I forhold til å ha påvirkning på arkitekturen, i den settingen har jeg ikke så stor påvirkningskraft, men det er ikke nødvendigvis IKT-arkitektur som setter de begrensningene, men finansieringen og fagsiden.
		INFORMANT 8: Er det riktig å si at det fungerer greit når ikke utviklingsteamene involveres INFORMANT 1: Beslutninger på forskjellige nivåer -det kan være flere beslutningsnivåer. Av og til kan det besluttes lokalt, og så eskaleres når det går feil. Den mekanismen som er gull, er at får vi ned ting svart på hvitt, istedenfor møte etter møte med svada. I seg selv er prosessen gull, men må differensiere mer mer, prosjektene må ta flere beslutninger selv selv. Informant 9: Men må dokumenteres samme sted. Noen prosjekter lager hver sin logg <Klargjøring: det kommer ikke fram i presentasjonen at flere Informanter også har pekt på at de opplever at de er i stand til å ta flere beslutninger selv enn det de gis anledning til> INFORMANT 8: Forutsetter tillit INFORMANT 1: Vi må tillate at folk tar beslutninger, heller lese gjennom etterpå.

		INFORMANT 2: Da har vi en liten vei å gå… INFORMANT 8: Vi må klare å ta beslutninger så nærme det det gjelder som mulig INFORMANT 1: Man må forstå hvilke interessenter som må være involvert eller ikke. INFORMANT 4: Det krever at folk kan gjøre de vurderingene da. <Foreslår justering av presenterte funn: Prosess/rammeverk er nyttig i seg selv, men for mange beslutninger tas på for høyt nivå?> INFORMANT 8: Forutsetter tillit. Informant 9: Noen ganger i prosjekt har man lyst til å logge en beslutning, eller få en bekreftelse fra noen, men tror at man må løfte beslutningen, prosessen føles langt unna. INFORMANT 1: Ikke alle har tilgang til ARKBESL (Arkitekturbeslutningslogg) <Metoden beskriver egentlig at beslutninger skal tas på «riktig» nivå – men dette er kanskje ikke kjent nok ute i prosjekter og kontor?> INFORMANT 1: Vi må misjonere for metoden og gi flere tilgang (…) INFORMANT 2: Det positive er at det er dokumentert og åpent, men det som er utfordringen med det er at de som sitter tett på, og så snakker man andre som har noe med det å gjøre, men i en arkitekturbeslutningsgruppe bruker man mye tid på å forklare og forstå alle problemstillingene til de som sitter der osv. det kan være tidkrevende da. Det bør være gjennomtenkt hva som er nødvendig å løfte, hva kan man beslutte på et lavere nivå INFORMANT 1: Og man trenger ikke gjøre alternativanalyse alltid, noen ganger bare dokumentere en beslutning.
Arkitekturbeskrivelse		

A	B	Svar
1	2.2	Det er en svakhet, hvordan vi får produsert målbilder og få dem forankret, men uten at det koster skjorta. Mye som starter opp, men hvordan får vi det til å fungere. Må kvalitetssikres med så mange interessenter, den totale utsjekken blir for stor. Det er veldig krevende, kanskje vi skal levere noe vi kaller 1.0 litt før, ta heller innspill til neste versjon.
1	4.1	For kompliserte maler – for målbilder, epos og alt mulig – ikke alt er arkitektur- vi lager en komplisert mal, som har med seg alt mulig kult, skremmer vettet at av de som skal gjøre noe på et lite område. Når malen bare egner seg for det mest avanserte vi skal gjøre.
1	4.1	Det andre er at vårt arkitekturmateriell (innhold) er for krevende å ta fram og godkjenne – det hindrer oss i å bruke det ute – det blir en kost. Noe som er forankra bredt, men får det ikke godkjent. Jeg kan jo ikke selv godkjenne det jeg lager. Noen som sier ja, det er bra nok for 1.0. Målbilde for [domene], fikk mye støy, selv om det var mye forankring, klart bra nok til 1.0 – vi kunne sagt bra nok, la [forvaltningskontor] jobbe videre. Setter seg heller fast i en grøt, og da kan jeg ikke bruke dette ute.
1	4.1	Hjulpet [Det at du nevner at det er for vanskelig å få på plass godkjente målbilder, innebærer det at du mener at slike målbilder ville bidratt til bedre/mer smidig prosess?] – Vi bruker mye tid for å få fram målbilder, de skal ha så høy kvalitet, kan ta til takke med noe litt lavere kvalitet – da kan vi bruke dem, og bedre smidighet i arbeidet med konseptfase, tidlige faser, og i prosjekter som skal ta fram nye målbilder. Tilstanden ville gjort det enklere. Tidlige avklaringer, når man jobber med løsningsarkitektur osv. Når materialet er bedre kan vi være bedre arkitekter, vi kan også jobbe likere. Mer enhetlig arkitekturstyring.
1	4.2	Er det noe innhold som hjelper?

		- Jeg bruker IKT prinsipper og kvalitetskrav, retningslinje for sikkerhet, målbilde for brev og arkiv, aktørid – ved at man har noe som er metoden eller retningslinjen eller målbildet, det blir faktisk godt mottatt, og man slipper en del diskusjoner rundt det man skal jobbe med, i stedet for at man diskuterer alt. Det gjør det enklere å være arkitekt. Og gjør ting mer enhetlig.
2	3.1	På arkitektur er det kanskje litt uklart hvilke krav stilles, hvilke arkitekturprodukter skal leveres, hva er obligatorisk, hva er til hjelp, levere til hvem når, godkjenning. Og også bruk av EA som verktøy – det er jo så mye flott der som man må få til å bruke, men verktøyet fungerer ikke så godt som det er nå. Ting låser seg, får ikke gjort noe. Men får mye hjelp, men når tidspresset er stort blir sånne ting uoverkommelig og slitsom å forholde seg til. Sånne ting som man bare vil skal virke. I den grad man ønsker å få ting enhetlig – tydeliggjøre hvilke modeller skal tas fram, hvordan brukes.
3	3.7	Hva som er nå-situasjonen, dokumentere så vi forstår hvor vi er, hva kan gjenbrukes, her begynner vi å ha mye på tjenesteorientering, kommunikasjon og de greiene der har vi god fart på nå, bare peke på.
4	3.5	[...]men når vi skal sette det sammen, da mangler vi oversikten, når vi putter det sammen i en svær leveranse, da kan det være noe vi savner der. Hver enkelt bestanddel er godt nok. Men jeg savner jo også fokus på de mer fysiske applikasjonskomponentene tidligere i prosessen. Det er de som faktisk har avhengighetene i produksjonssetting. Gjelder ikke bare arkitektur, det gjelder generelt, f.eks. Teknisk ressurser i kontorene ikke har forståelse av komponentene de er avhengige av.
4	3.4	(Tenker) Det jeg tror at mange med meg savner er ... Man er interessert i arkitekturskissene og avhengighetene, ikke alle detaljer, det er noen ganger for lite arbeid som viser totaloversikten for et epos, det er lett å se hvilke kontor som er involvert, ikke alltid så lett å se hvilke applikasjoner som er involvert, arkitekturskissene er varierende, men på forskjellig nivå, ikke konsekvent bruk av entiteter, mange ganger er de fraværende eller ligger «et annet sted», det å si hva er det denne egentlig gjør – der er det for lite, mangler «managment summary», samtidig som man får veldig mye detaljinformasjon, som for mange ikke er så interessant før det dukker opp

		et problem, Når du skal vurdere femti epos vil du kunne bruke fem minutter på å kunne forstå avhengighetene, og at dette gjøres på en standardisert måte.
6	3.1	Vi har heller ikke veldefinerte dokumentasjon som kan brukes i praksis – noen tegner litt i EA, men ingen på kontoret bruker det eller oppdaterer det. Modellerer ofte målbilder i steden for hvordan det faktisk ser ut. Det er ikke tydelige målbilder og vi her ikke med på å forme utviklingen av systemene, vi bare bygger på den tekniske gjelden. Det flyter litt på leverandørens premisser uten at vi har kontroll selv
6	4.1	Hvor ting blir dokumentert. Vi har ikke varig dokumentasjon, arkitekturen ligger i Confluence på et epos, som dør, vi har ikke felles repository der alle jobber med arkitekturen. Finne opp hjulet på nytt hver gang, kartlegge verdikjede, ingen som har dokumentasjon på det, i alle fall ikke som er holdt ved like.
8	2.1	Fordi jeg tenker på meg selv som en utvikler, er jeg veldig for å prototype ting, skisser er også kode, å skrive kode, så fort som mulig egentlig. Den koden man skriver da er bare en slags dokumentasjon, braindump av det man tenker der og da, ikke nødvendigvis noe som skal bevares videre, bare en annen måte å tenke på, som kan fungere bra i å kommunisere hensikt. Er konkret og enklere å få feedback på. Selv om man ikke ender opp der, ender kanskje opp med en annen teknisk løsning, så vil første gangen man ser en webapp man har pælma sammen sette i gang gode diskusjoner.
9	3.1	En av hindringene også for å kunne jobbe godt er også at systemdokumentasjon som ikke ligner grisen i måneskinn, man lager nye Gliffy-figurer hele tida, burde bare hentet ut fra EA og dokumentert endringen på det, burde hatt felles struktur for hele NAV, veldig vanskelig å se, mye i hodet på folk, dokumentert forskjellig, når man skal jobbe på tvers av områder er det utfordrende å finne ut av hvordan ting henger sammen, hvis man må ned kunne man like gjerne sett i koden. Mye fokus på lavnivå, detaljert systemdokumentasjon.

9	5.1	Systemdokumentasjon. Den synes jeg ofte ikke ser ut i måneskinn, det virker ikke som man har midler eller tid til å ta den til et nivå der den er brukbar, noen har bra systemdokumentasjon, f.eks. KES, når jeg lurer på hvordan fungerer denne prosessen her, har den integrasjon mot KES, det finner jeg i systemdokumentasjonen. Hvis jeg lurer noe tilsvarende på [forvaltningskontor] er det utrolig vanskelig å finne fram i, gamle Word-dokumenter som er arvet fra prosjektet, mindre og mindre brukbar for hver dag. God systemdokumentasjon, hvordan henger ting sammen, omtrent hvilke behandlingsregler brukes her og der, ikke pseudokode. Man gjør seg avhengig av leverandør, som har ting i hodet, dukker ned i koden, og i praksis lager samme gliffy-figurer om igjen og om igjen. Føles ikke tilgjengelig, leverandøren sitter på dette. Da kunne jo kundesiden gjort mer konsekvensanalyse selv. Med en gang du ikke har en god systemdokumentasjon blir du veldig avhengig av leverandør og bistandstimer på det.
	DV	INFORMANT 1: Vi tenker noen ganger for mye på å beskrive nåsituasjonen, sannheten ligger i Fasit og sånne steder. Modellerer det som ikke er sannheten, og da blir det problemer. Vi må finne ut hva som er viktig å modellere, det er viktigere i smidig arkitektur å ha visjonene å styre etter. Er ikke så glad i arkitekturrepositories lenger, vi klarer aldri å vedlikeholde dem, bruke views inn i Fasit i steden, avdekke avhengigheter INFORMANT 8: Amen, <Flere nikker>
	DV	INFORMANT 8: Det er jo også viktigheten man tillegger verktøyene, det er en egenverdi i seg selv, siden det tok lang tid, tillegger det mer verdi enn det har. Bruker alt for mye tid på modeller som er nesten riktig. Hvis det tar kapasiteten din vekk fra å ta valg der og da, være til stede…
	DV	INFORMANT 8: Sparx etc ikke alle har tilgang, til å lese, de greiene der hindrer kommunikasjon, gjør at utviklere som har behov kommer ikke til å lese den. Også notasjon som man ikke nødvendigvis forstår <Archimate>, den kan godt være korrekt, men kommuniserer ikke bra. Det er mange stakholders til en figur. Jeg tror vi burde ha en annen tooling, Ardoq eller lignende. Noe som gir views som er tilgjengelig for alle, kan kommunisere til alle. Det er masse dokumentasjon på Confluence og i EA repository, som ikke alle kan forstå. INFORMANT 4: Man må forstå målgruppen for dokumentasjon, bruker tid på å lese noe som kunne vært kommunisert enklere, one-pager som oppsummerer hva som skal gjøres, hvilke systemer er involvert, løfte opp informasjonen, så kan folk heller dykke ned i

		detaljer ved behov. For mye fokus på dokumentasjon for dokumentasjonens skyld, ikke for målgruppene INFORMANT 1: Og kompliserte maler, man tror man ikke har frihet til å beskrive de som trengs innenfor malen INFORMANT 4: Det er jo også et spørsmål om diffrensiere INFORMANT 8: Mye arkitekturdokumentasjon gjør man jo for å kommunisere et mål eller en beslutning, så det må være egnet til å kommunisere, og det har man glemt litt. Hvordan vi bruker Confluence, det må jo være unikt i verdenssammenheng, vi mangler noen som rydder opp, for det er jo så viktig. INFORMANT 4: Man bør skille mellom dokumentasjon nå, og hva som vi skal ta med oss videre. Vanskelig å finne ut nå hva som dokumentasjon av den faktiske situasjonen, og hva som bare var en del av en konsekvensvurdering eller noe. INFORMANT 3: Litt mer fokus på dokumentasjon i samhandling – hvordan brukes den i forskjellig kontekst. Det man diskuterer rundt det kaster man, dokumentasjonen den må vi ta vare på, siden det kommer folk seinere som ikke kan spørre. Snu handlingen rundt dokumentasjonen til det viktige. INFORMANT 4: Hovedleveranser – vi sjekker at systemdokumentasjon er levert, men ikke at den er god INFORMANT 9: Hvem er brukerne av dokumentasjonen og hva er behovene deres? INFORMANT 1: Mye rundt dokumentasjon blir annerledes hvis man er til stede i teamet. INFORMANT 8: Det skjer jo fort at man etablerer en applikasjonsarkitektur, men hvordan den faktisk implementeres, det er sjelden en-til-en, og sjelden med feedback loop for å oppdatere arkitekturen. Da er det bare leverandørene som vet hvordan det egentlig ser ut. INFORMANT 1: det å holde modeller i live det er krevende, mer må genereres basert på sannhet, det produksjonsnære. Så må vi ha målbilder og dokumentasjon underveis, det siste kan vi kaste når vi har skuta på rett kjøl INFORMANT 4: Ta stilling til trenger vi dette i evigheten eller noe vi trenger her og nå

Arkitekturforståelse		
A	**B**	**Svar**

1	2.1	Viktig å tenke på endringsevne og forvaltbarhet, at det er enkelt og forståelig. En viktig rolle som arkitekt da er det med formidlingsevne og kommunikasjonsevne, forklare så folk forstå hvor du vil, så de de griper det og «gjør det til sitt». Det er utrolig viktig i de fasene.
1	2.1	Det viktige er å finne ut av ting i samarbeid tidlig. Ikke sitte på sidelinje og tegne arkitektur, men få tilbakemeldinger fra de som eier applikasjonen og forstår hvordan den fungerer. Søke etter det enkle og gode, ikke komplisere. Viktig å tenke på endringsevne og forvaltbarhet, at det er enkelt og forståelig. En viktig rolle som arkitekt da er det med formidlingsevne og kommunikasjonsevne, forklare så folk forstå hvor du vil, så de de griper det og «gjør det til sitt». Det er utrolig viktig i de fasene.
1	2.2	Noe som jeg vil nevne: Arkitekturformidling, som jeg driver med, og ser hvor viktig det er. * Hvorfor? Det hjelper ikke at man har laget et målbilde som noen er kjent med, når andre som burde vært det ikke er informert. Viktige ledestjerner i arkitekturarbeidet må være kjent for «massene» prosjektledere, utviklere osv. Hvor er det man ønsker å gå. Et godt eksempel, hvis man ikke vet hvor ting er og finner det lett følges ikke metodene, det skjer ting som ikke burde skje.
1	3.6	God arkitektur forenkler. Man lager ikke arkitektur for at det skal bli vanskeligere – vi prøver å søke etter forenkling, så lenge man tilfredsstiller funksjonelle krav og kvalitetskrav. Og så lenge at du ikke forenkler så mye at det blir vanskelig å endre/forvalte.
1	4.1	Ikke så mye hinder i organisasjon som før, blant annet fordi vi har begynte med OAM [Områdearkitektmøte, møter mellom arkitekter og TA i et forvaltningsområde, og arkitekter fra IKT-arkitekturseksjonen] (det er metode!). Samme med møter med sikkerhetsseksjonen.

1	4.2	Jeg tenker at det at man er med og gjør workshoper med [forvaltningskontor], med leverandører og kontoret og KES og vi, og diskuterer hvordan ting skal være, det tenker jeg er smidighet. Jeg tror på å samle forskjellige roller og funksjoner på tvers i en tidlig fase, for å få det samme tankegodset, da blir arkitekturen en samarbeidssak. Der synes jeg vi er på vei til noe.
1	5.1	Arkitekturvalg må eies av de som skal utvikle, ikke måtte lese seg til det. Det hjelper ikke bare å komme inn i et møte og si hvordan ting skal være, de må være med. Teamet må ha eierskap. Hvis jeg tenker på hvordan NAV skal være, så kan vi ikke gå rett til devops, men vi kunne bygge team som forankrer på tvers, der leverandøren er med, så senere i prosessen så vet de hva de vil. Ikke bare skriftlig,men også muntlig.
1	5.1	Jeg tror veldig på tverrfaglige team, forretning, arkitektur, utvikling, forvaltning – jobbe sammen, vi passer på visse ting, andre med andre ting – det er kanskje der smidigheten på arkitektur oppstår. De som skal utføre det føler seg delaktige, selv om vi langt på vei må skissere den. Jeg tror det blir best i den møteplassen. Vi bør jobbe mer sånn. Jeg har lite sans for overleveringer, her er det jeg har gjort, nå er det din tur – det er vannfall, hviskeleken. Vi kan ikke ha hviskeleken, vi må jobbe ting sammen Arkitekturvalg må eies av de som skal utvikle, ikke måtte lese seg til det. Det hjelper ikke bare å komme inn i et møte og si hvordan ting skal være, de må være med. Teamet må ha eierskap. Hvis jeg tenker på hvordan NAV skal være, så kan vi ikke gå rett til devops, men vi kunne bygge team som forankrer på tvers, der leverandøren er med, så senere i prosessen så vet de hva de vil. Ikke bare skriftlig,men også muntlig.
1	3.1	Viktig å være ute, og være ydmyk for den kompetansen som sitter på mer detaljnivå i kontor og prosjekter. Samtidig som man vet at man har med seg en del kunnskap selv om helheten. Men i møtepunktet, det er der mulighetene skjer. Jeg lærer noe hver gang, samtidig som jeg lærer dem noe, så kommer vi fram til nye muligheter. Hadde ikke funnet løsning på utbetalingsreskontro alene. Vi må være ute, ikke sitte i et hjørne og modellere. Man må noen ganger trekke seg litt tilbake og tegne, men så gå ut igjen og sjekke: «Er det sånn vi forstår dette?».

2	2.3	Det er litt ensom rolle å ha noen ganger. Veldig mange spørsmål, det er bra, men mye ... Det hadde vært greit å ha flere på lag, så man ikke hele tiden står alene om å forsvare alt som gjøres, samtidig som man må orientere seg i så mange ulike områder.
2	3.2	Savner litt i forhold til kommunikasjonen, den gode møteplassen, hvor greier vi å skape de gode arenaene for å forstå hva de ulike miljøene bringer inn, hva betyr det, hvordan kan vi utnytte det. Det er mye fokus på når man gjør noe som går i mot, arkitekturbeslutninger, det er viktig, men kunne vært mer fokus på hvor går veien, hvordan kan vi bruke dette på en god måte, hvor ha dialog, hvor lærer vi av hverandre.
2	3.2	Savner at man har litt dårlige arenaer for å lære, arkitekturforum som fungerer som informasjonskanal fra IKT-arkitektur, og det er jo bra, men det blir jo ikke noe samhandlingsforum, dialog. Det her med å ha en form for forum, har kanskje det i utviklingsmøtet, men arkitekturforum der man presenterer det man har tenkt, og får tilbakemelding på det man har tenkt. For å fange de gode innspillene og ansvarliggjøre på tvers.
2	3.2	Men jeg tenker at vi i større grad kunne hatt arkitektteam, ikke være ensom. Det er alltid utrolig lett når noen kommer med forslag til hvordan dette skal gjøre, å ikke bare være kritisk, men å være sparringspartner som bidrar og designer og kommer opp med gode ideer, ikke bare være kritiker.
2	3.5	Nja, det kan i alle fall være feil noen ganger, man jobber litt for fossefall, med veldig mye overlevering, så blir kanskje noe av konteksten borte, så når utvikleren får det, dekker det ikke nødvendigvis behovet. Da er det kanskje arkitekturarbeidet som ikke har fungert, kommunikasjon og arkitektur og det som hører med. God arkitektur innebærer god kommunikasjon. Det handler veldig mye om formidling, formidle mellom ulike grupper mennesker, forstå, omsette, hvis man ikke greier å håndtere det, mister man mye på veien, hjelper ikke med perfekte modeller hvis det som skal bruke dem ikke forstår dem, ikke så perfekte likevel hvis de ikke kan gjenspeile behovet.

2	5.3	Har hatt mer linjerolle før, nå blir rollen mye tydeligere, når man er prosjektets egne arkitekt og må passe på at alle de gode initiativene realiseres på en god måte, og sørge for at det faktisk blir gjort sånn. Ikke tid til å ettergå, utviklerteamene tar snarveier, det er tidspress og det skjer noe, det blir litt glemt hva løsningen skulle være. Ikke så lett å fange, jeg går jo ikke og kontrollerer kode. Og det er jo litt krevende, man blir jo stilt til ansvar for det. Jobber så mye med behovene, lage backlog, vanskelig å gå tilbake og kontrollere hva som har blitt gjort. Men prøver å få det fram i standup. Det er god erfaring, man får med seg mye. Det så jeg i uføre også, mer systemteamkoordinator/analysekoordinator – nyttig med standupen. I noen av teamene der noen konsekvent ikke gidder det, tror det er for detaljert, går glipp av det.
3	2.1	For de funksjonelle må man tenke på de egenskapene de er opptatt av og hvordan det fungerer, hvis det er mer mot teknisk eller drift er det mer på det, hvis det er helhet opp mot arkitektur er det innkapsling av ansvar, hvordan brikke i et større landskap Tilpasse kommunikasjon til aktørene. Ikke så mye formelle metoder, annet enn kommunikasjon.
3	3.4	Men gjort for mye på løsningsbeskrivelser. Ved å jobbe fram en arkitektur som interessenter ikke er inneforstått med, så ting må gjøres igjen.
3	3.6	Ja, det bidrar fordi du blir klar over avhengigheter og viktige egenskaper for kvaliteten i systemene og skapt forståelse for løsningen for forskjellige behov til forskjellige interessenter. Burde nok informere mer, særlig IKT-drift, styringsinformasjon osv, de blir kanskje glemt i kampens hete. -Hvorfor bidrar dette?

		-Fordi man blir klar over hva som skal bygges, og avhengighetene og det er for større prosjekter er det veldig viktig for en realistisk gjennomføringsplan. Og det å scope inn ting, så man kan sette i gang parallelle aktiviteter og ha kontroll på de, ikke gjøre alt sekvensielt. Mest i større prosjekter hos oss er det ikke så mye vi endrer i parallell.
4	3.1	Mens prosessen der noen gjør utredning og beskriver arkitektur som overleveres til de som skal konstruere da blir en del ting borte underveis, det blir rett og slett ikke hensiktsmessig, det er informasjon som mangler, og vi må gjøre en del konsekvensutredning i konstruksjonsfasen for å forstå hvordan det egentlig henger sammen. Kanskje det er gjort tidligere, men informasjonen er ikke tilgjengelig.
4	3.1	føler det er for stort gap mellom det som ligger i arkitekturprosessen og det som blir realisert. Den er litt teoretisk, det er det gapet jeg har forsøkt å fylle, gapet mellom logiske komponenter og det som faktisk produksjonssettes. [...] (Hva er det som ikke fungerer i prosessen?) Den største delen er kanskje differensen mellom det som gjøres i forkant i konsekvensutredninger og det som er faktisk realisering og konstruksjon. Tror ikke den er like stor i alle prosjekter, har litt med bemanning og overleveringer å gjøre. Når ressursene sitter med gjennom hele prosessen blir det helt anderledes. [...] Mens prosessen der noen gjør utredning og beskriver arkitektur som overleveres til de som skal konstruere da blir en del ting borte underveis, det blir rett og slett ikke hensiktsmessig, det er informasjon som mangler, og vi må gjøre en del konsekvensutredning i

		konstruksjonsfasen for å forstå hvordan det egentlig henger sammen. Kanskje det er gjort tidligere, men informasjonen er ikke tilgjengelig
4	3.2	Kanskje ikke så mye med samhandlingen med arkitektene, ikke vært så mye med på det, men kommunikasjonen mellom arkitektur og prosjekter og kontorer og leveranser, men den har blitt litt bedre, men det er litt sånn «de der oppe i åttende, de sitter og tegner og vet ikke hva som skjer», oppfattes som litt for teoretisk og ikke så hands-on. Noen forvaltningskontorer synes vel bare arkitektene lager støy, mener de har full kontroll selv, samtidig som mange av dem har et veldig snevert syn, ser sin egen silo, liten fokus på det som skjer utenfor. Der spiller jo arkitektur en viktig rolle.
5	2.1	Veldig for smidig tilnærming, viktig med overordnede rammer men – fan av å sitte sammen og diskutere løsninger – sitter aldri alene og designer løsning, det gjør jeg sammen med utviklere, flere arkitekter fra oss, diskutere, begynne å tegne, få det ned på papiret – det er den kjedelige jobben. Ikke fan av rigid prosess på det – det er ikke jeg som vet hvordan dette funker i virkeligheten, men utviklere som jobber med i hverdagen – jeg kan legge til ett «ørnesyn» på det. Bruker tavle som verktøy i diskusjoner. Noe basert på UML når det skal ned på papiret, om det er helt riktig UML... men det er det jeg kan.
5	2.2	dårlig skille mellom områder og applikasjoner – når det står en boks der så blir det en boks der – ser ikke at den firkanten der er en del av noen annet – det blir en feil tolkning av arkitekturnivå – høyere nivåer blør nedover på løsningsdesign og infrastrukturdesign, da blir det veldig fastsatt, hindrer at man løser på den mest optimale måten. Ikke nødvendigvis feil, men hvordan det blir oversatt nedover. Veldig ofte i konsekvensvurdering blir det gjort (eller før?) litt usikker – vet ikke helt om det er fordi er ressurser eller bare historikk – det er en forståelse av målbildet som blir borte – intensjonen er ofte veldig god, men blir ikke med videre, det er bare grafen som «blir med videre».

		En ser bare på designet, glemmer hvilke egenskaper man forsøker å oppnå gjennom arkitekturen – det blir borte på veien fra «deres nivå» - noen ganger også et kunstig skille mellom kontorer, «hvem eier klumpen».
5	3.4	Når vi løser små ting går det greit, store prosjekter – jo lenger konsekvensutredning, møter ting kommer på toppen, så får du en Rolls Royce når du kunne klart deg med noe annet – og som ofte ikke er forankret i et faglig behov – forutsetninger som gjøres – man tolker hva «fag antakelig trenger» - krav basert på forutsetninger som kanskje egentlig ikke er til stede.
5	4.1	Vanskelig å pinpointe – det jeg tror er at det stilles for detaljerte krav – det er noe med transisjonen mellom virksomhetsarkitektur til løsningsarkitekt som skurrer – gir en del begrensninger. Men hva som gjør det – hva er intensjonen bak arkitekturen fra IKT-arkitektur, hva er krav fra kontoret, et sted mellom IKT-arkitektur og oss er det for dårlig kommunikasjon. Tror mye av det som skjer på overordnet nivå er fornuftig, må ha kontroll på data som flyter i en så stor organisasjon. Men det er noe i kommunikasjonen nedover som er problematisk – oversetting til løsningsarkitektur.
5	5.1	et eller annet sted skjer det kommunikasjonsfeil, noen steder feil personer i feil roller? Det er intensjoner som blir borte, også uavhengig av personer, på veien til løsningsdesign. Samme utfordring i overlevering til utvikling - det tar for lang tid, prøver å løse det ved å involvere folka tidligere – men det at det tar så lang tid fra noe er påtenkt til det blir implementert at ting går i glemmeboka, nye personer, feil personer, for mange personer, møtekultur...
6	3.1	Hvor skal jeg begynne... Jeg ser jo mye utfordringer, ikke arkitekturen eller arkitektene nødvendigvis, men hvordan vi er organisert og jobber, vannfall, handover, QA... Det som foregår er så overordnet at det ikke ivaretar arkitekturen i systemene. Teori og praksis spiller ikke helt sammen.

6	3.2	Det har blitt bedre med OAM-møtene, men IKT-arkitektur jobber jo så tidlig i prosessen før vi er involvert, så det er ingen som vet helt hva de driver med når vi får ballen. Så kommunikasjonen kunne vært mye bedre, og med mer samarbeid gjennom hele prosessen, og at arktiktene var knyttet til et fagområde og var eksperter på det, ikke så generelle.
6	3.4	Noen ganger eksterne arkitekter som ikke kjenner arkitekturen til NAV så godt, dårlig tid, noen vil ha en fin skisse på områdedesignet, og så låser man seg på en lite optimal løsning – kanskje det er for lite ja. Arkitekturarbeid er jo som regel bra. Det er for oppdelt, og ikke alltid de riktige personer. Jeg ble jo satt til å være løsningsarkitekt på [prosjekt], og har jo ikke peiling... Man tar litt for lett på det, setter ikke på riktig kompetanse tidlig, ber noen skissere noe, noen ganger går det litt på kompetanse. Man bruker ikke arkitektene helt riktig, ønsker mer eksperter innenfor domener og områder som kjenner til NAV, og som kan bidra videre i prosessen, hva har de tenkt, hvorfor har de tatt avgjørelsen, og evt endrer mening underveis når det er mer informasjon. Det hadde vært betryggende. Men når det er stykka opp og delt er det sjelden man forstår hvorfor det er valgt å løse det på den måten
6	4.1	At arkitektene ikke er med lenger inn i konstruksjon, og at vi ikke har arkitekter som er mer tekniske, som programmerer selv og er med på å implementere det de spesifiserer. Det er vanskelig å være smidig med en handover, en overlevering av dokumenter, hvordan skal du gjøre dette. Frustrasjon å ikke forstå hvorfor ting er gjort på den måten.
8	2.4	Demoer i arkitekturarbeid må ikke være working software, kan være «her er det vi tror nå», dele litt tanker hele tiden, og ønske mer å få feedback. I Aura gjør vi jo det. Jeg tror det ville vært nyttig i P2 også, på et sånt nivå, at man snakket sammen litt mer, i dag har jeg tenkt å sitte sammen med [annen løsningsarkitekt] for å snakke om det og det, er det noen som lurer på noe. Prøver å dele sine hindringer og framdrifter i større grad. Det tror jeg på, og det å lage til demoer uten kjempeseremoni.
8	5.1	Har et uavklart forhold til ordet arkitektur, har blitt kalt det lenge, skjønner ikke helt, jeg er ikke utdanna arkitekt, de som er det driver fint lite med det jeg driver med, har aldri skjønt helt hva det betyr, hvis det handler om å designe og tenke ut gode løsninger så er det

		noe utviklere også gjør de bare gjør det på et detaljnivå lenger ned. De løsningene jeg har laget, man er jo oppom alle de nivåene… Det er noe jeg ... Å være smidig i forhold til det her, det er viktig, løsningsarkitekter har mye å lære bort til utviklere, og det år andre veien også. Beste utvikler jeg har møtt, starta som utvikler, ble pusha oppover til han var sjefsarkitekt, så jobba han seg ned igjen, han er helt rå til å kode, han har en helt spesiell måte å tilnærme seg problemet på. Skyldes nok hans erfaring som løsningsarkitekt. Han skulle vi hatt her...
8	DV	Hvis man ikke har kapasitet til å være til stede på standup for det produktet man lager, hva kan være viktigere?
9	2.1	En stor del av jobben er å forankre mot kontorene, ikke bare «vi i prosjektet har bestemt at…» men forankre med områdearkitekter og TAer i kontorene, samle de for å finne felles løsning.
	DV	(om foil om Kommunikasjon) INFORMANT 8: Det nest siste punktet forutsetter jo at organiseringen er sånn [At IKT-arkitektur er separat fra resten]. At det skillet er der er jo noe å jakte etter. Hvis arkitektene faktisk er «service provider» så er det ikke så mye å snakke om i faste møter. Men når arkitektur er et opphøyet team som stiger ned med divine visdom – ikke det beste patternet. INFORMANT 4: Fokus på å finne feil er generelt, er en sykdom vi har her, henger sammen med frykten for å gjøre feil. Istedenfor å tenke «hva er intensjonen bak?» tenker man «her skal jeg finne en feil, da har jeg gjort jobben min som kvalitetssikrer». INFORMANT 1: Må være en bidragsyter som løfter og forbedrer. INFORMANT 4: Gjelder ikke bare IKT-arkitektur, gjelder også SLEM og hvordan vi er rigga her. INFORMANT 1: Kommunikasjon i teamet handler mer om kontinuerlig formidlingen av den kunnskapen du besitter, være en formidler kontinuerlig. På standup, si «hei, det der er ikke egenskaper plattformen støtter» osv, være med der det skjer. Det er en helt annen form for kommunikasjon enn overlevering. INFORMANT 4: Forutsetter at man er der INFORMANT 1: Forutsetter tilstedeværelse og trygghet i rolle og å få tillit fra sjefsarkitekt, og å gi tillit videre.

Andre arkitekturaktiviteter		
A	**B**	**Svar**
1	3.5	[Fra tidligere arbeidsgiver] Der var det ikke gjort arkitekturarbeid. Noen forstod dagens løsninger veldig godt, ville løse det helt likt [i ny løsning]. Det ble merkelige komponenter som kom inn, det så rart ut. Det var en annen arkitekt som løste det, jeg bidro – måtte grave oss ned, vi sparte 40 millioner og fjernet 60 «bokser» - det var på hengende håret at vi fikk endret kursen. Det var flaks. Der var det for lite arkitekturarbeid først, og mer arkitekturarbeid tidlig hadde gjort prosessen betydelig enklere.
3	4.2	Det som slår meg er at om vi skal levere enda mer i parallell må vi opp med andre metoder, da må vi kanskje gjøre flere avklaringer på forhånd, så mer er ferdig definert. Vi må jobbe litt mer så vi har ferdige blokker, f.eks. presentasjonslaget, at vi definerer opp hvordan vi gjør ting, hva passer best for hver problemstilling, applikasjonsinndeling, hvordan vi løser kvalitetskravene, ferdige blokker på hvordan det skal løses. F.eks. På tjenesteintegrasjon, da løser du mange krav når du bare bruker det som er satt opp. Det å kunne peke på en metode eller et prinsipp for hvordan du løser ting. Da kan man få smidig utvikling. Tar involveringen på forhånd. Standardisere det som skal gjentas, jobbe smidig med det funksjonelle.
4	1.4	Men ofte er det leverandøren som har den beste oversikten, særlig over tekniske avhengigheter på konstruksjonsnivå.
5	3.6	Har vært borte i at vi må lage suboptimale løsninger fordi vi tvinges til å gjenbruke noe som ikke er gjenbrukbart, presse ting inn i eksisterende løsninger – leveransesammensetning – levere hele stacket i samme leveranse, øker risiko i forhold til å levere tjenester først, konsumenter etterpå.

5	3.6	Det er vanskelig å si noe om det fordi – jeg tenker på ting som bidrar til å senke risiko – sikkerhetsarkitekturen er jo bra, den senker risiko, men i den andre enden har vi ting som skal øke sikkerheten, bruke policy enforcement, folk skjønner ikke ha de skriver og dermed ikke hva de skal teste.
5	3.6	Men på applikasjonsdesign så vet jeg ikke – det er vanskelig å svare på det. Både og, noe bidrar til å øke noe til å minke – noen ganger legger man for harde føringer, da blir det subooptimale løsninger – man tror man reduserer risiko, f.eks. Standardisere rammeverk – fordi det gir for stramme rammer. Og det er så stort spenn i løsninger i huset – forutsetninger – brukskvalitet, oppetid, interne, eksterne, det gjøres ikke forskjell på om det er mot bruker, inn mot fagsystem – gir for stramme rammer.
8	DV	Det skjer jo fort at man etablerer en applikasjonsarkitektur, men hvordan den faktisk implementeres, det er sjelden en-til-en, og sjelden med feedback loop for å oppdatere arkitekturen. Da er det bare leverandørene som vet hvordan det egentlig ser ut.
Annet - Svar fra informantene som ikke faller inn under kategoriene som blir behandlet i oppgaven.		
A	**B**	**Svar**
1	3.4	KS-regimet er jeg vel ikke så glad i, En del av det man gjør der er arkitekturarbeid for å bevise ovenfor FIN at vi har tenkt nok. Vi skal jo lykkes overfor brukere og saksbehandlere, og understøtte NAVs rolle med å tilby ytelser. Der blir det for mye arkitekturarbeid. Modellen for modernisering feiler litt der, gjelder ikke bare arkitekturarbeid. For lite smidighet.
3	3.1	Det som ikke er forutsigbart er de langsiktige målene til forretning – det er bortimot fraværende. [Prosjekt] som skal omorganisere og skal avvikle [organisasjonsenhet], men det er ikke forutsigbart […], det blir armer og bein. Andre ting, arbeidsgivere, hva er en arbeidsgiver, hvilken rolle har de, hvordan skal NAV forholde seg til dem, hva vil vi framover. Forretningsarkitekturen i prosjekter er som regel litt bedre, men de jobber ofte litt snevert, med en smal problemstilling.
6	4.4	Vi har jo mye last å ta med oss, konteksten gjør det ikke så lett. Vi har dårlig tid, finansiering er jo alltid en issue.

7	3.7	Andre ting jeg føler, ikke på arkitektur, men det mandatet for å kunne banke gjennom beslutninger, inn mot forretning, det er omstendelig i NAV, i [annen virksomhet] er det raske beslutninger, her er det mye mer uklart hvem som har mandatet til å beslutte. Metier hadde det også i rapporten sin, uklare fullmakter.
7	4.3	Nei, egentlig ikke det, mer det med fullmaktene, men IKT-arkitektur har jo sterke mandater, men mer inn mot forretning hvor de stadige omkampene er, mangler det tydelige lederskapet. Fullmaktsstrukturene, hvem kan skjære gjennom.
9	3.1	En ting til som er frustrerende er målpris, jeg ser at det er utstrakt bruk av det, og det virker som at NAV bare aksepterer det, det sier kontrakten, men leverandør presser på og ønsker å bruke målpris, som en konsekvens av dette blir det mye jobb med løsningsbeskrivelsene, blir ikke smidig i tilnærmingen til behovene, siden leverandør ønsker å sikre seg mest mulig, løsningsbeskrivelse som blir utrolig detaljert, og som mest er egnet til å sikre mot at leverandør må håndtere ting som feil. Forskyver risiko fra leverandør til NAV, fastpris blir nesten aldri brukt, lite løpende timer. Det legger på en måte grunnlaget for hvordan man jobber videre, detaljert løsningsbeskrivelse, gir det til et scrumteam, dukker opp i test om tre uker eller noe... Veldig inndeling i at noen kjerneressurser fra leverandør sammen med NAV sitter og finner ut av alt, så er det slutt på involveringen. Jeg synes egentlig man kunne gått mer på løpende timer og dermed som en konsekvens av det måtte NAV jobbe mer sammen med scrum-teamet som skal lage dette her. Krever mer involvering gjennom hele prosessen, ikke bare «lag dette her», trekke seg tilbake. Leverandør ønsker alltid målpris, ønsker å skyve risiko over på NAV.

Vedlegg 3: Presentasjon fra gjennomgang av funn for informanter

Presentasjon av foreløpige funn

– smidige arkitekturprosesser i NAV

Presentasjon for informanter 2. september 2016

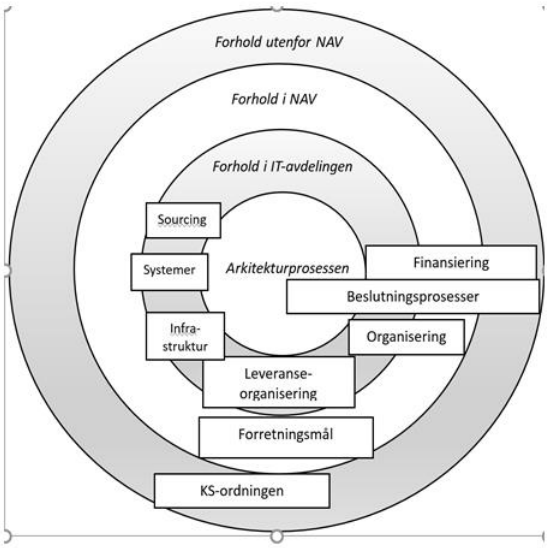
Oversikt over tema

- Rammebetingelser for smidig tilnærming
- Smidig tilnærming til arkitekturarbeid
- Arkitektens rolle
- Arkitekturbeslutninger
- Kommunikasjon
- Arkitekturdokumentasjon

Premisser

- Sammenligner beskrevet praksis (fra intervjuer og dokumentasjon) med empiri og «lessons learned» fra forskning
- «Teknisk» avgrensning av scope for å få håndterbar oppgave: Hovedfokus på IT-avdelingen, forhold utenfor IT-avdelingen behandles som rammebetingelser/bakgrunn.
- Det som er mest interessant (i denne sammenhengen) er det dere har erfart (fortrinnsvis i NAV)
  - Gir empiri fra NAV
- Ønsker tilbakemelding på om dere kjenner dere igjen i oppsummering av funnene, om det er feil/upresist, eller om det er noe viktig som mangler.

Rammebetingelser



Figur 5 - Hindringer og muligheter i omgivelsene

Tema: Rammebetingelser for en smidig tilnærming

- **Hva sier forskningen?**
  - Grad av smidighet som kan oppnås er avhengig av rammebetingelser som prosjektets og systemets egenskaper, organisering, behov for risikohåndtering og menneskelige faktorer som kompetanse og kultur.
  - Kortere prosjekter med kontinuerlige leveranser bidrar til å redusere risiko.
- **Hva sier informantene i NAV?**
  - **Det er mange rammebetingelser som vanskeliggjør en smidig tilnærming**
  - Eksterne rammebetingelser (utenfor IT-avdelingen)
    - KS-regimet medfører mye tidlig arkitekturarbeid, og låser enkelte rammer svært tidlig
    - Interne budsjett- og finansieringsprosesser gjør det utfordrende å finansiere arkitekturforbedringer (refaktoring, utfasing av teknologi etc).
    - Veldig mye må utredes samtidig ved årsskiftet («ketchupeffekt» når budsjettene godkjennes).
    - Det kan være tidkrevende å få beslutninger og føringer fra fagsiden. Forretningsmessige mål er ofte utydelige.
  - Rammebetingelser internt i IT-avdelingen
    - Organiseringen i IT-avdelingen («plan-build-run») gir organisatoriske grenser mellom folk som burde jobbe tett sammen om arkitekturen.
      - Ressurs: Mange dyktige folk, som ønsker å løse problemer.
    - Hovedleveranser gir fossefallsorientering. Alle elementer som inngår behandles likt i prosessen, uavhengig av omfang og risiko.
    - Egenskaper ved systemer og infrastruktur (legacy-systemer, harde bindinger, feil bruk av standardisering er til hinder, «riktig» bruk av standardisering, automatisering er til støtte).

Tema: Smidig tilnærming til arkitekturarbeid

- **Hva sier forskningen?**
  - Behovene for planlegging og smidighet kan balanseres gjennom å fokusere tidlig innsats på områder med høy risiko.
  - Områder med lavere risiko kan utforskes ved hjelp av iterativ tilnærming, støttet av smidige teknikker og tilpassede arkitekturteknikker.
- **Hva sier informantene?**
  - Opplever i hovedsak ikke arkitekturarbeidet som smidig
    - Det er positive erfaringer med mer smidig tilnærming i enkeltprosjekter/områder (digiSYFO, Aura…).
  - Beskrivelsene tyder på at innsatsen i liten grad differensieres basert på risiko.
    - Lite fokus fra arkitektur på de kritiske saksbehandlingssystemene, i andre tilfeller følger man opp det som oppleves som detaljer.
    - Kvalitetssikring i SLEM er «alt over samme lest»
  - Motstridende syn på SLEM
    - SLEM er ikke smidig
      - «Waterfall Scrum»
    - SLEM gir gode rammer for å jobbe smidig.
    - «filosoferering» over om SLEM i seg selv er usmidig, eller hvordan det praktiseres, særlig i HL
      - «Ikke kultur for å gjøre endringer underveis».
  - Smidige elementer i SLEM etterleves ikke alltid
    - Konsekvensvurderinger som behandles som bindende
    - Analyseteam på tvers av kontor som jobber hver for seg, ikke reelle team.

Tema: Arkitektens rolle

- **Hva sier forskningen?**
  - Arkitekten i en smidig organisasjon må være til stede i utviklingsteamet, kommunisere godt med utviklingsteam, og ha kompetanse nok til å kunne diskutere tekniske spørsmål.
  - "The Architect as a service provider"
- **Hva sier informantene?**
  - Seksjon for IKT-arkitektur oppleves generelt som for lite tilgjengelige av utviklingsprosjekter og forvaltning.
  - Arkitektene oppgir at de i dag sjelden har kapasitet til å være til stede i forbindelse med detaljert design (løsningsbeskrivelse) og koding.
    - Gjelder også arkitekter i forvaltning og prosjekt.
  - Arkitektene som er intervjuet har et ønske om å være mer deltakende design- og utviklingsprosessen.
    - Det er gode erfaringer med nye arbeidsformer som støtter dette, workshops, delta i standup.
    - Må endre rolle fra «politi» til samarbeidspartner
  - Det er en opplevd mangel på arkitekter med teknisk forståelse og kompetanse om det enkelte domenet eller systemet
    - Man blir avhengig av leverandørene
    - De fleste intervjuede arkitektene har teknisk bakgrunn/utdanning.

Tema: Arkitekturbeslutninger

- **Hva sier forskningen?**
  - Arkitekturbeslutninger bør tas trinnvis, etter hvert som tilstrekkelig informasjon er tilgjengelig
- **Hva sier informantene?**
  - Arkitekturbeslutningsprosessen i IT-avdelingen fungerer stort sett greit
    - Bidrar til å løse opp i konflikter/få felles forståelse
    - Hjelper «lokale» arkitekter med forankring.
    - Det fungerer godt å dokumentere valg som tas
  - Men:
    - For smidige prosjekter kan prosessen bli for tidkrevende
    - Omfattende involvering av interessenter er ofte det som gjør at beslutninger tar tid.
    - Målbilder og andre førende dokumenter svært vanskelig å få godkjent. Det er også en forventning om at slike dokumenter skal være svært detaljerte.
    - Tidlige beslutninger (konsekvensutredning), gjøres ofte uten involvering av utviklingsteamet.
    - I store prosjekter/programmer tvinges tidlige beslutninger fram, til dels på for lite informasjonsgrunnlag.

Tema: Kommunikasjon

- **Hva sier forskningen?**
  - God muntlig kommunikasjon er en viktig forutsetning for en smidig arkitekturprosess.
  - Overlevering av skriftlig dokumentasjon må alltid ledsages av muntlig kommunikasjon.
- **Hva sier informantene?**
  - Overleveringer av oppgaver og dokumentasjon mellom forskjellige deltakere i arkitektur og løsningsbeskrivelsesprosessen skjer ofte uten (tilstrekkelig) muntlig kommunikasjon.
    - Gir informasjonstap og misforståelser
    - «Vi kan ikke drive med hviskeleken!»
  - Kvalitetssikringsfokus gjør at en del kommunikasjon knyttet til feil og avvik
    - For lite kommunikasjon som har læring og kunnskapsdeling som formål?
  - Kommunikasjon mellom IKT-arkitektur og øvrige er bedre enn den var tidligere, men bør fortsatt forbedres.
  - Positive erfaringer med at de samme ressursene deltar gjennom hele prosessen, reduserer behovet for overleveringer.

Tema: Arkitekturdokumentasjon

- **Hva sier forskningen?**
  - Arkitekturen må uttrykkes og dokumenteres eksplisitt, men omfang og form må tilpasses behovet.
- **Hva sier informantene?**
  - Det er store mengder dokumentasjon tilgjengelig, men informasjonsbehovene i arkitekturprosessen dekkes ikke.
  - De samme tingene dokumenteres om igjen og om igjen.
    - Nåsituasjon beskrives i praksis hver gang man konsekvensvurderer
    - «Man bør standardisere dokumentasjonen av det som gjentas, så man kan fokusere på det funksjonelle».
  - Man blir avhengig av leverandørene for å få innsikt.
  - Det er uklare krav til hva som skal dokumenteres, og hvordan det skal gjøres
    - Omfattende maler
    - Tungvint modelleringsverktøy (Sparx EA)
  - Eksempler på dokumentasjon som mangler/savnes:
    - Oppdatert systemdokumentasjon. Noen steder mangler dokumentasjonen fullstendig.
    - Dokumentasjon som viser sammenhengen mellom systemer som inngår i en løsning, på overordnet nivå.
    - Tekniske avhengigheter
  - Eksempler på dokumentasjon som oppleves som nyttig
    - Arkitekturprinsippper, kvalitetskrav, målbilder, retningslinjer
    - Dokumentasjon av integrasjonsplattform

Vedlegg 4: Referat fra gjennomgang av funn

Gjennomgangen ble gjennomført 2. september 2017.

Tema: Rammebetingelser

- Informant 9: Det må jo ikke nødvendigvis være sånn [at man får ketchupeffekt]– det som mangler er en produktkø man kan ta av, man skal «fore neste sprint» - du kan velge ut fra sannsynlighet, dette kan vi jobbe med, uten å sende det til konstruksjon, men ha en produktkø som er bedre enn i dag. Fordrer styring av produktkøen, ikke umulig.

Tema: Smidig tilnærming til arkitekturarbeid

IKD: Forskjellige syn på SLEM, er det til støtte eller hinder, er det evt metoden eller praksis som er til hinder

- INFORMANT 1: SLEM muliggjør smidighet, men organiseringen rundt ødelegger, ikke SLEM som er problemet
- INFORMANT 3: Det er mest organiseringen rundt [som er problemet], metoden er kanskje litt detaljert, bør tunes
- INFORMANT 1: Vi har en verktøykasse vi kunne tune, men den hindrer ikke
- INFORMANT 8: Er SLEM ikke forstått?
- INFORMANT 1: Tror ikke det er forstått hvordan man jobber smidig
- INFORMANT 8: Filosofere mer: Hvis det i stor grad ikke forstått, er det for komplisert eller er vi dumme?
- INFORMANT 1: Jeg tror kommunikasjonen er mer rundt detaljer i metoden, mer enn om hva vi prøver å oppnå
- INFORMANT 2: Vi har en kultur der kontroll er svært viktig, det er en historikk, prosjekter der man ikke har fått det til. Mange som kontrollerer andre, det får fokus, overlevering og kontroll, jobber ikke etter metoden og sammen.
- INFORMANT 1: Kontrollfokuset er alt for sterkt.
- (Flere nikker)
- INFORMANT 8: Enig, ikke ekspert på SLEM, men forstår det som om kontrollbehovet er implementert i SLEM, eller tolket inn i SLEM. Det er jo litt som organisasjoner som tar i bruk Scrum, det blir rigid, Scrum i seg selv blir viktig, SLEM i seg selv blir viktig. Men det er jo mindset som er viktig. Følge metoder slavisk tjener vi ikke på lang sikt.
- INFORMANT 4: Differensiering er jo viktig da – det henger sammen med kontroll
- INFORMANT 2: Det er jo vanskelig, du sitter i et mønster, det du følger, vanskelig å avvike. Mindsettet må jo endres for at man skal kunne differensiere.
- INFORMANT 1: Vi må begynne å stole på at folk gjør jobben sin. Det er fundamentet. Da må måten man kontrollerer være annerledes
- INFORMANT 3: Risikobasert
- INFORMANT 1: Noen steder er det risiko, man skal etablere fundament for framtiden
- INFORMANT 4: Re-etablere tillit, noen steder er det ikke tillit. Ikke sånn at alt vi gjør kan føre til at folk ikke får penga sine. Det argumentet brukes «over alt»

- INFORMANT 8: Det er jo relevant hvis vi søker en org basert på tillit, er det da behov for SLEM?
- INFORMANT 1: Kanskje ikke i den formen, så detaljert
- INFORMANT 8: Hvis vi har autonome team som eier sin bit av NAV, så …
- INFORMANT 1: Men man ønsker en enhetlig måte å håndtere produktløp på, kunne bytte mellom team osv. Alle kan ikke gjøre alt forskjellig
- INFORMANT 8: NAV IT må ha noen krav til grensesnitt. Produktkø inn til et team er et sånt grensesnitt.
- INFORMANT 4: Vi skal jo jobbe mer forskjellig, det gjør vi ikke i dag. Mye kontroll og sjekklister og telle det ene og andre, overlevering til drift er ikke kvalitetssikring, bare kontroll.
- INFORMANT 1 Også grunnutviklingsavtalene – de er anti-smidig de luxe. Kan ikke gjøres smidig som det praktiseres i dag.

<avbryter – dette er rammebetingelser for prosessen, ønsker ikke å fokusere så mye på det>

- INFORMANT 4: Men det er mer frihet i de kontraktene enn det som brukes i dag?
- INFORMANT 2: Da må jo noen lære det bort.
- INFORMANT 1: Kontorene får ikke hjelp til å håndtere det i praksis. Da gir det ikke så mye mening å snakke om at kontraktene er mer smidige

<…>

INFORMANT 1: Det er andre eksempler på smidig arkitekturarbeid – noen caser i FKON, vi er tett på men delegerer mye, det gjøres i kontorene – der er det ikke arkitekturarbeidet som hindrer framdrift, men andre ting.

INFORMANT 2: Vi bruker jo ikke SLEM i det hele tatt i digiSyfo, men vi bruker jo elementer av det, produktkø og analyse og design. Mye tettere vevd sammen, tar med mer input fra utvikling inn i hvordan man skal jobbe videre. <…> Prøver å definere sånne MVPer (Minimum Viable Product)– når vi er klare nok til å si at vi vet hva vi kan gjøre med det, så det blir en leveranse. Vanskelig å si at alt er helt klart, andre ganger må vi gå tilbake fra utvikling å finne ut av ting, ikke lett å vite når det er passe klart.

Tema: Arkitektens rolle

INFORMANT 4: Jeg tenker jeg blir glad når jeg leser det her, men skal man få til det, gjennom OU kommer samme tilbakemelding, må gjøre mindre av noe annet da, mindre politi og tåkefyrste. Henger sammen med tillit og ansvar.

INFORMANT 8: Hvis man ikke har kapasitet til å være til stede på standup for det produktet man lager, hva kan være viktigere?

<klargjøring, Faber viser til at et team til sammen kan dekke funksjonene, ikke alle arkitekter må kode eller kommunisere med alle>

INFORMANT 1: Vi tenker noen ganger for mye på å beskrive nåsituasjonen, sannheten ligger i Fasit og sånne steder. Modellerer det som ikke er sannheten, og da blir det problemer. Vi må finne ut hva som er viktig å modellere, det er viktigere i smidig arkitektur å ha visjonene å styre etter. Er ikke så glad i arkitekturrepositories lenger, vi klarer aldri å vedlikeholde dem, bruke views inn i Fasit i steden, avdekke avhengigheter

INFORMANT 8: Amen,

<Flere nikker>

INFORMANT 8: Det er jo også viktigheten man tillegger verktøyene, det er en egenverdi i seg selv, siden det tok lang tid, tillegger det mer verdi enn det har. (Dette vil jeg ikke siteres på) Bruker alt for mye tid på modeller som er nesten riktig. Hvis det tar kapasiteten din vekk fra å ta valg der og da, være til stede…

INFORMANT 1: Vi skal være sammen med teamet og hjelpe dem å beslutte mellom alternativer. Ingen ting som hjelper mer enn en god alternativanalyse. Vi må tørre å feile, når vi ikke tør å ta beslutninger, da hindrer vi smidighet.

INFORMANT 4: Vi skal ikke ha rapp over fingeren når vi feiler! Det må være lov å feile!

INFORMANT 3: Organiseringen hindrer oss, systemene er koblet sammen dumt, organiseringen hindrer oss. Arena, litt av alt, på alle måter, systemene er kobla sammen for mye. Lappeverk av funksjonalitet – tørre å si det var dumt, bygge på nytt..

Tema: Arkitekturbeslutninger

INFORMANT 8: Er det riktig å si at det fungerer greit når ikke utviklingsteamene involveres

INFORMANT 1: Beslutninger på forskjellige nivåer -det kan være flere beslutningsnivåer. Av og til kan det besluttes lokalt, og så eskaleres når det går feil. Den mekanismen som er gull, er at får vi ned ting svart på hvitt, istedenfor møte etter møte med svada. I seg selv er prosessen gull, men må differensiere mer mer, prosjektene må ta flere beslutninger selv selv.

Informant 9: Men må dokumenteres samme sted. Noen prosjekter lager hver sin logg

<Klargjøring: det kommer ikke fram i presentasjonen at flere Informanter også har pekt på at de opplever at de er i stand til å ta flere beslutninger selv enn det de gis anledning til>

INFORMANT 8: Forutsetter tillit

INFORMANT 1: Vi må tillate at folk tar beslutninger, heller lese gjennom etterpå.

INFORMANT 2: Da har vi en liten vei å gå…

INFORMANT 8: Vi må klare å ta beslutninger så nærme det det gjelder som mulig

INFORMANT 1: Man må forstå hvilke interessenter som må være involvert eller ikke.

INFORMANT 4: Det krever at folk kan gjøre de vurderingene da.

<Foreslår justering av presenterte funn: Prosess/rammeverk er nyttig i seg selv, men for mange beslutninger tas på for høyt nivå?>

INFORMANT 8: Forutsetter tillitt.

Informant 9: Noen ganger i prosjekt har man lyst til å logge en beslutning, eller få en bekreftelse fra noen, men tror at man må løfte beslutningen, prosessen føles langt unna.

INFORMANT 1: Ikke alle har tilgang til ARKBESL <Arkitekturbeslutningslogg>

<Metoden beskriver egentlig at beslutninger skal tas på «riktig» nivå – men dette er kanskje

ikke kjent nok ute i prosjekter og kontor?>

INFORMANT 1: Vi må misjonere for metoden og gi flere tilgang

INFORMANT 4: Handler også om å gi tillit til eksterne også. Leverandører eller innleide konsulenter.

INFORMANT 8: Konsulenter er litt «second-class citizens»

INFORMANT 2: Det er jo rart da, om vi leier inn folk som arkitekter må vi jo stole på dem.

INFORMANT 2: Det positive er at det er dokumentert og åpent, men det som er utfordringen med det er at de som sitter tett på, og så snakker man andre som har noe med det å gjøre, men i en ark.itekturbeslutningsgruppe bruker man mye tid på å forklare og forstå alle problemstillingene til de som sitter der osv. det kan være tidkrevende da. Det bør være gjennomtenkt hva som er nødvendig å løfte, hva kan man beslutte på et lavere nivå

INFORMANT 1: Og man trenger ikke gjøre alternativanalyse alltid, noen ganger bare dokumentere en beslutning.

Tema: Kommunikasjon

INFORMANT 8: Det nest siste punktet forutsetter jo at organiseringen er sånn [At IKT-arkitektur er separat fra resten]. At det skillet er der er jo noe å jakte etter. Hvis arkitektene faktisk er «service provider» så er det ikke så mye å snakke om i faste møter. Men når arkitektur er et opphøyet team som stiger ned med divine visdom – ikke det beste patternet.

INFORMANT 4: Fokus på å finne feil er generelt, er en sykdom vi har her, henger sammen med frykten for å gjøre feil. Istedenfor å tenke «hva er intensjonen bak?» tenker man «her skal jeg finne en feil, da har jeg gjort jobben min som kvalitetssikrer».

INFORMANT 1: Må være en bidragsyter som løfter og forbedrer.

INFORMANT 4: Gjelder ikke bare IKT-arkitektur, gjelder også SLEM og hvordan vi er rigga her.

INFORMANT 1: Kommunikasjon i teamet handler mer om kontinuerlig formidlingen av den kunnskapen du besitter, være en formidler kontinuerlig. På standup, si «hei, det der er ikke egenskaper plattformen støtter» osv, være med der det skjer. Det er en helt annen form for kommunikasjon enn overlevering.

INFORMANT 4: Forutsetter at man er der

INFORMANT 1: Forutsetter tilstedeværelse og trygghet i rolle og å få tillit fra sjefsarkitekt, og å gi tillit videre.

Tema: Arkitekturdokumentasjon

<Dette er et område det var utfordrende å oppsummere – det er mye forskjellig som har dukket opp her. Dere nevnte tidligere å bruke verktøy for å generere dokumentasjon - Kunne man lest mer ut av systemene?>

INFORMANT 8: Sparx etc ikke alle har tilgang, til å lese, de greiene der hindrer kommunikasjon, gjør at utviklere som har behov kommer ikke til å lese den. Også notasjon som man ikke nødvendigvis forstår <Archimate>, den kan godt være korrekt, men

kommuniserer ikke bra. Det er mange stakholders til en figur. Jeg tror vi burde ha en annen tooling, Ardoq eller lignende. Noe som gir views som er tilgjengelig for alle, kan kommunisere til alle. Det er masse dokumentasjon på Confluence og i EA repository, som ikke alle kan forstå.

INFORMANT 4: Man må forstå målgruppen for dokumentasjon, bruker tid på å lese noe som kunne vært kommunisert enklere, one-pager som oppsummerer hva som skal gjøres, hvilke systemer er involvert, løfte opp informasjonen, så kan folk heller dykke ned i detaljer ved behov. For mye fokus på dokumentasjon for dokumentasjonens skyld, ikke for målgruppene

INFORMANT 1: Og kompliserte maler, man tror man ikke har frihet til å beskrive de som trengs innenfor malen

INFORMANT 4: Det er jo også et spørsmål om differensiere

INFORMANT 8: Mye arkitekturdokumentasjon gjør man jo for å kommunisere et mål eller en beslutning, så det må være egnet til å kommunisere, og det har man glemt litt. Hvordan vi bruker Confluence, det må jo være unikt i verdenssammenheng, vi mangler noen som rydder opp, for det er jo så viktig.

INFORMANT 4: Man bør skille mellom dokumentasjon nå, og hva som vi skal ta med oss videre. Vanskelig å finne ut nå hva som dokumentasjon av den faktiske situasjonen, og hva som bare var en del av en konsekvensvurdering eller noe.

INFORMANT 3: Litt mer fokus på dokumentasjon i samhandling – hvordan brukes den i forskjellig kontekst. Det man diskuterer rundt det kaster man, dokumentasjonen den må vi ta vare på, siden det kommer folk seinere som ikke kan spørre. Snu handlingen rundt dokumentasjonen til det viktige.

INFORMANT 4: Hovedleveranser – vi sjekker at systemdokumentasjon er levert, men ikke at den er god

Informant 9: Hvem er brukerne av dokumentasjonen og hva er behovene deres?

INFORMANT 1: Mye rundt dokumentasjon blir annerledes hvis man er til stede i teamet.

INFORMANT 8: Det skjer jo fort at man etablerer en applikasjonsarkitektur, men hvordan den faktisk implementeres, det er sjelden en-til-en, og sjelden med feedback loop for å oppdatere arkitekturen. Da er det bare leverandørene som vet hvordan det egentlig ser ut.

INFORMANT 1: det å holde modeller i live det er krevende, mer må genereres basert på sannhet, det produksjonsnære. Så må vi ha målbilder og dokumentasjon underveis, det siste kan vi kaste når vi har skute på rett kjøl

INFORMANT 4: Ta stilling til trenger vi dette i evigheten eller noe vi trenger her og nå.

Vedlegg 5: Oversikt over hindringer og muligheter

Kontekst - Organisasjon	
Hindringer	**Muligheter**
• IT-avdelingens organisering gir uønskede skiller mellom grupper som bør arbeide tett i en smidig prosess. • Arkitektene fra Seksjon for IKT-arkitektur oppleves som lite tilgjengelige for forvaltningskontorene og utviklingsteamene • Finansieringsmodell og budsjettprosesser gjør det vanskelig å jobbe smidig.	• Organiseringen av IT-avdelingen endres ved årsskiftet 2016/17, noe som kan gi en modell som bedre understøtter smidig tilnærming. • Kommunikasjonen mellom Seksjon for IKT-arkitektur og prosjekter og forvaltningskontor oppleves som bedre enn den var tidligere.

Kontekst – Prosjektenes egenskaper	
Hindringer	**Muligheter**
• Samme krav til planverk, rapportering etc stilles til alle prosjekter, uavhengig av størrelse. Gir mye «overhead» for små prosjekter. • Flere mindre leveranser/prosjekter «pakkes» sammen i en hovedleveranse. Gir bindinger mellom leveranser som ellers ikke har noen sammenheng, og fører til fossefallstilnærming.	• IT-avdelingen ønsker å reduser omfanget til hovedleveransene, som oppleves som en hindring for smidighet av mange av respondentene, og i større grad la hvert prosjekt produksjonssette egne endringer.

Kontekst – Kompetanse og kultur	
Hindringer	**Muligheter**
• Prosjektene etterlyser arkitekter med mer teknisk kompetanse. Mye av teknologikompetansen sitter hos leverandørene. • «Skepsis» til konsulenter kan være til hinder for kunnskapsdeling og samarbeid. • Kontrollkultur gir fokus på detaljer og kvantitativ kvalitetssikring. • Ingen av arkitektene som er intervjuet har erfaring med «lettvekts» arkitekturteknikker.	• Enkeltpersoner som får ting til å fungere på tross av hindringer, positiv innstilling. • IT-avdelingen rekrutterer flere med utviklingskompetanse, slik at noen av hindringene for smidighet som er knyttet til kunde/leverandørskillet mellom utviklere og andre potensielt blir svakere.

Kontekst – Systemenes egenskaper	
Hindringer	**Muligheter**
• En del eksisterende systemer har teknologi eller utforming som gjør det vanskelig å jobbe smidig med endringer av arkitektur • Sterke bindinger mellom systemer vanskeliggjør en smidig tilnærming.	• Standardisering på utvalgte områder som sikkerhet og integrasjon gjør det enklere å jobbe smidig med funksjonalitet. • Automatisering av IT-prosesser som systemoppsett, test og utrulling gjør det enklere å jobbe smidig.

Arkitekturprosessen - Målsetninger	
Hindringer	**Muligheter**
• Arbeidsprosessene (praksis) oppleves ikke som smidige av de involverte. • Innsats i arkitekturarbeid differensieres i liten grad etter risiko. • Arkitekturarbeidet er ikke iterativt. • Prosessen er dårlig egnet til å håndtere endringer underveis i prosjektene. • Det er vanskelig å ta i bruk ny teknologi.	• Ønske blant arkitektene om skifte tilnærming, mer bidragsyter, mindre «politi». • Endringer i retningslinjer for kvalitetssikring er endret, gjør at ansvaret for arkitekturprosessen delegeres mer.

Arkitekturprosessen – Analyse, syntese, evaluering, implementering, forvaltning	
Hindringer	**Muligheter**
• Flere av de smidige elementene i SLEM etterleves ikke i praksis. • Kommunikasjon om SLEM er detaljorientert, og vektlegger ikke i stor nok grad den smidige tankegangen og målsetningen. • Hovedleveransene inneholder kontrollelementer som ikke støtter opp under smidighet. • Arkitekturprosessen og utviklingsprosessen oppleves som adskilt • Arkitektene oppleves i noen tilfeller som for detaljorienterte, og mer opptatt av formaliteter enn av å finne løsninger. • Arkitektene oppgir at de i dag sjelden har kapasitet til å være til stede i forbindelse med detaljert design (løsningsbeskrivelse) og koding.	• Leveransemetoden SLEM er basert på smidige prinsipper, og inneholder elementer som støtter opp under smidighet, f.eks. tverrfaglige analyseteam. • Gode erfaringer med å løse oppgaver i samarbeid, på tvers av organisasjonsgrenser. • En felles prosess som gir forutsigbarhet – folk vet hva som forventes av dem • Positive erfaringer med å benytte smidige teknikker, f.eks. at arkitekt deltar i utviklingsteamets standup • Erfaringer fra smidig pilot-prosjekt og fra autonomt utviklingsteam kan brukes som grunnlag for ny felles praksis • Utforske arkitekturmuligheter gjennom koding (pilotering, prototyping)

Arkitekturbeslutninger	
Hindringer	**Muligheter**
• Det oppleves at for mange beslutninger tas på for høyt nivå, og at prosessen oppleves som «langt unna». Utviklingsteam er som regel ikke involvert i beslutninger. • Beslutninger kan ta tid, siden mange interessenter skal involveres. Prosessen kan være for tidkrevende for smidige prosjekter • I noen tilfeller mangler det vilje til å delegere beslutninger. • Prosessen som «verktøy» er ikke godt kjent. Noen oppfatter at den bare kan brukes for beslutninger på høyt nivå. • Beslutninger som burde vært delt, dokumenteres «lokalt».	• Den dokumenterte prosessen fungerer godt for mange formål, blant annet å løse opp i uenigheter og interessekonflikter. • Det oppleves som positivt at beslutninger blir dokumentert, og at det finnes en felles beslutningslogg. • Mange av informantene sier de har innflytelse på beslutninger. • Flere av informantene sier at de er i stand til å ta flere beslutninger enn de får anledning til. • Beslutninger og ansvar delegeres i større grad enn tidligere til områdearkitektene. • Den definerte arkitekturbeslutningsprosessen setter ikke begrensinger for hvilket nivå beslutninger skal tas på.

Arkitekturbeskrivelse	
Hindringer	**Muligheter**
• Ikke tydelige nok krav til hva som skal utarbeides av arkitekturdokumentasjon. • Det mangler tydelige målbilder for mange av systemene. Det er krevende å få godkjent slike målbilder. • Det finnes store mengder dokumentasjon av arkitektur og systemer, men dokumentasjonen ser ikke ut til å dekke behovene. • Dokumentasjonen er ofte ufullstendig, eller det er usikkerhet rundt om den er oppdatert. Systemdokumentasjonen er ofte «i hodet til leverandøren». • Dokumentasjonen som tas fram som en del av analyseprosessen blir for detaljert, slik at det er vanskelig å få oversikt over sammenhenger.	• Dokumentasjon av prinsipper og krav for arkitektur oppleves som nyttige. • Man kan i større grad bruke kode for å uttrykke arkitektur. • Generere dokumentasjon av nåsituasjon ved hjelp av verktøy (basert på kodebaser, konfigurasjonsdata etc).

Arkitekturforståelse	
Hindringer	**Muligheter**
• Utfordringer med å tolke/oversette abstrakt arkitekturdokumentasjon til en faktisk forståelse av hvordan systemene henger sammen. • Informasjon og kunnskap følger ikke med gjennom prosessen – «hviskeleken». • Overleveringer av arkitekturbeskrivelser fra en gruppe til en annen skjer ofte uten muntlig kommunikasjon. Intensjonen bak blir borte. • Arkitekter jobber noen ganger isolert, uten å kunne diskutere med andre.	• Ønske om mer direkte kommunikasjon og samarbeid. • Ønske blant arkitekter om å jobbe mer smidig og være mer til stede. • Bruke «demoer» for å få tilbakemeldinger på arkitekturideer og dokumentasjon.

Andre arkitekturaktiviteter	
Hindringer	**Muligheter**
• Det er utfordrende å analysere hvilken påvirkning endringer har på arkitekturen, siden dokumentasjonen i mange tilfeller er mangelfull. • Man er avhengig av utviklingsleverandør for å gjennomføre *Analyse av påvirkning*. • Pålegg om gjenbruk av rammeverk oppleves noen ganger som en hindring for smidighet • Refaktorering av arkitektur er lite i bruk	• Gjenbruk av integrasjons- og sikkerhetsarkitektur gir handlingsrom til å konsentrere designarbeidet rundt det som varierer (funksjonalitet) • Ta i bruk standardverktøy for å gjenvinne arkitekturdesign fra kode og konfigurasjonsdata

Annet	
Hindringer	**Muligheter**
• Forretningsmessige mål kan noen ganger være uklare, og lange avklarings/beslutningsprosesser blir da en hindring for framdrift. • KS-prosessen tvinger fram omfattende arkitekturarbeid tidlig i prosessen. • Avtaler og prismodeller for utvikling. Bruk av målprisavtaler.	• Gjenbruk av integrasjons- og sikkerhetsarkitektur gir handlingsrom til å konsenterere designarbeidet rundt det som varierer (funksjonalitet) • Avtalene som regulerer utvikling gir handlingsrom for andre tilnærminger enn de som vanligvis velges i dag.

Vedlegg 6: Sammenligning av forventet og empirisk mønster

Tema	Innsikt	Funn	Diskusjon
Rammebetingelser for smidig tilnærming	Prosessens kontekst er avgjørende for evnen til smidighet.	• Det er identifisert hindringer, men også muligheter, for en mer smidig tilnærming innenfor alle kategoriene. • I tillegg er det identifisert andre hindringer i og utenfor IT-avdelingen.	• Funnene viser at det er mange opplevde hindringer for en mer smidig praksis i omgivelsene til arkitekturprosessen. • Flere av rammebetingelsene er under endring, i det som tilsynelatende er en positiv retning for mer smidig tilnærming. Arkitektfagmiljøet kan bruke momentet til endringer i prosessene.
Organisasjonen	Organisasjonens modenhet utgjør driver og begrensning for prosjektenes, og dermed prosessenes, egenskaper (Kruchten, 2013)	• IT-avdelingens organisering gir uønskede skiller mellom grupper som bør samarbeide. • IT-avdelingen omorganiseres fra 1.1.2017. Utviklingen skal organiseres i produktteam. • Arkitektene er fortsatt organisert i en separat avdeling, men skal være en «ressurspool» som skal være aktivt deltakende i prosjekter og forvaltning. • Det kan være utfordrende å skaffe finansiering til tiltak som understøtter framtidig smidighet i systemene.	• Ny organisering av utvikling er mer i tråd med en smidig tilnærming, og forutsetter at arkitektene blir mer aktivt deltakende. • Daglig samarbeid mellom ansatte og innleide konsulenter bør legge vekt på faglig samarbeid. • Dersom det ikke er mulig å finansiere tiltak som understøtter systemenes smidighet, er det ikke mulig å oppnå en reelt smidig prosess.
Prosjekt	Store prosjekter begrenser muligheten til smidig tilnærming, og øker behovet for å uttrykke arkitektur på forhånd. (Dybå & Dingsøyr, 2009), (Nord & Tomayko, 2006), (Boehm, 2011)	• Utviklingsprosjekter er ofte omfattende. • Uavhengige endringer organiseres inn i samme leveranse (prosjekt) • IT-avdelingen forsøker å redusere omfanget av hovedleveranser, og bruke flere frittstående, mindre leveranser	• IT-avdelingens praksis med å organisere flere mindre endringer inn i samme leveranse (prosjekt), er i strid med anbefalinger fra forskninger. • Mindre prosjekter vil gi bedre rammebetingelser for en smidig tilnærming (Dybå & Dingsøyr), og potensielt redusere risiko (Jørgensen).

Tema	Innsikt	Funn	Diskusjon
		• Styringsprosesser, planverk og krav til rapportering er ikke tilpasset små, smidige prosjekter.	• Mindre prosjekter vil også redusere behovet for omfanget av arkitektur som må uttrykkes på forhånd (Boehm)
Kompetanse og kultur	• Kompetanse og kultur hos teamdeltakere og interessenter er et viktig suksesskriterium for smidige prosesser. (Waterman et al., 2015), (Yang et al., 2016), (Dybå & Dingsøyr, 2008) • Arkitektene må ha god teknisk kompetanse. (Rendell, 2009), (Faber, 2010)	• Risikoaversjon og «kontrollkultur» påvirker arkitekturprosessen. • Flere av arkitektene som er intervjuet uttrykker et ønske om å være mindre «politi» og mer en aktiv bidragsyter. • Noen av informantene mener internt ansatte er skeptiske til konsulenter, og at dette er et hinder for smidighet. • Det etterlyses mer teknisk kompetanse hos arkitektene og andre internt ansatte. It-avdelingen er i ferd med å rekruttere flere ansatte med teknisk profil. • Ingen av informantene har kjennskap til arkitektursentriske lettvektsprosesser. • De fleste informantene har noe erfaring fra smidige prosjekter eller smidige teknikker.	• Det kan være sammenheng mellom manglende tilgang til teknisk kompetanse tilgjengelig, og sjekkliste-orientert kvalitetssikring. • Flere ansatte med teknisk kompetanse, og annen prioritering av tidsbruk kan gi mulighet til mer kvalitative bidrag. • Arkitektenes rolle bør være å forhindre grep som er til hinder for videre endringsevne. Det forutsetter systemkunnskap og «hands on» deltakelse. • Det er ikke gitt at alle arkitektene kan gå inn i en slik rolle. En mulighet er at arkitekturteamet utfyller hverandre (Faber, 2010)
Systemene	Systemenes alder, størrelse eller risikoprofil kan være til hinder for smidighet. Riktig valg av infrastruktur og systemutforming kan gi bedre støtte for smidighet. (Yang et al., 2016), (Bloomberg, 2013), (Waterman et al., 2015), (Kruchten, 2013)	• Avhengigheter mellom systemene oppleves som en hindring for smidig tilnærming • FKON-prosjektet bidrar til å redusere teknisk gjeld, og forbedre endringsevnen i systemene. • Etablering av automatisering av test, utrulling og miljøoppsett er en støtte for smidig tilnærming.	• Det er ikke gitt at alle systemer lar seg endre på en smidig måte. Variasjoner i systemenes størrelse, alder og kritikalitet kan gi behov for å tilpassede prosessvarianter.

Tema	Innsikt	Funn	Diskusjon
Målsetninger for en smidig arkitekturprosess	Arkitekturprosessen kan balansere behovene for endringsevne og risikohåndtering, gjennom å prioritere områder med høy risiko tidlig. (Poort, 2012), (Waterman et al., 2015). En smidig prosess må resultere i et smidig system, ellers vil systemet være til hinder for videre *Vedlikehold og videreutvikling*. (Beck et al., 2001) (Bloomberg, 2013), (Buschmann & Henney, 2013).	• «Alt» har vært kvalitetssikret og gått gjennom samme prosess fram til våren 2016, fra våren 2016 er analysene av endringer delegert, og arkitekter trekkes inn ved behov. • Framtidig smidighet i systemene vektlegges ikke i stor nok grad i prosjektene.	• Praksis som er beskrevet ivaretar ikke målsetningene for en smidig arkitekturprosess. • Det er gjort tilpasninger for å delegere ansvar, men det er ikke sikkert den nye tilnærmingen ivaretar risiko på en god nok måte • Responsibility Driven Architecture (Blair et al., 2010) kan være en god tilnærming for å balansere behovene for kontroll «på tvers» og smidighet. • Framtidig smidighet i systemene ivaretas ikke på en god nok måte.
Arkitekturprosessen – analyse, implementasjon og vedlikehold	En smidig arkitekturprosess kan understøttes av smidige teknikker, og lettvektsvarianter av arkitektursentriske metoder. (Yang et al., 2016), (Madison, 2010), (Cleland-Huang et al., 2013), (Nord & Tomayko, 2006).	• Informantene opplever i hovedsak ikke arkitekturarbeidet som smidig. • Metodeverket SLEM er basert på smidige prinsipper, men det er en utfordring at metodens intensjoner og mål ikke er godt nok forstått av alle. • Arkitektene deltar mest i de tidlige fasene, og er sjelden tilgjengelige i forbindelse med konstruksjon og produksjonssetting. • Noen av informantene forteller om positive erfaringer med mer smidig tilnærming og bruk av smidige teknikker.	• Det er et avvik mellom praksis og den definerte metoden. Den definerte metoden inneholder både smidige og usmidige elementer. • Man bør i større grad kommunisere intensjonen og prinsippene bak SLEM, slik at metoden kan brukes til å understøtte en mer smidig praksis. • SLEM bør videreutvikles, basert på erfaringer fra prosjekter og team som eksperimenterer med mer smidige metoder. • Arkitektene bør være ambassadører for de smidige prinsippene. • Bruk av arkitektursentriske lettvektsmetoder, og smidige teknikker i arkitekturarbeidet kan gjøre det lettere å integrere arkitekturarbeidet i smidige prosjekter.
Arkitekturbeslutninger	Arkitekturbeslutninger bør tas trinnvis, basert på en overordnet plan, og	• Den definerte arkitekturbeslutningsprosessen	• Dagens praksis der arkitektene bare deltar tidlig i prosessen kan bidra til å

Tema	Innsikt	Funn	Diskusjon
	involvere viktige interessenter. (Poort, 2012), (Blair et al., 2010).	oppleves som et nyttig verktøy av mange, blant annet fordi den bidrar til å løse opp i uenigheter. • Prosessen kan være tidkrevende, blant annet fordi mange interessenter skal sette seg inn i problemstillingen. • Det er en opplevelse av at mange beslutninger tas på for høyt nivå. Utviklere deltar sjelden i arkitekturbeslutninger. • Arkitekturbeslutningsprosessen legger til rette for delegerte og lokale beslutninger, men dette er ikke godt kjent.	presse fram flere tidlige beslutninger enn nødvendig. • Beslutninger bør i større grad delegeres til de som har praktisk kunnskap, og utsettes til «siste forsvarlige tidspunkt». • Ved å etablere beslutningsplaner som er beskrevet av (Blair et al., 2010) kan man oppnå dette samtidig som sentrale interessenter får en trygghet. • Å dele opp beslutninger kan redusere antallet interessenter som må involveres i hver enkelt beslutning. • Arkitektene deltar der de kan tilføre verdi og kunnskap til prosessen, etter mønster fra «tjenesteyter»-rolle beskrevet av (Faber, 2010)
Arkitekturbeskrivelse	• Arkitekturvisjon, -prinsipper og -krav må understøtte forretningens mål. (Leffingwell, 2011), (Bloomberg, 2013) • Arkitektene må ikke detaljstyre design, men definere handlingsrommet til utviklingsteamet, og ha tillit til utviklingsteamets kompetanse. (Erder & Pureur, 2016), (Madison, 2010), (Abrahamsson et al., 2010) • Designdokumentasjon bør være høynivå, teknologiuavhengig, og dokumentere rasjonalet bak sentrale designvalg. (Selic, 2009). • Kode kan brukes som virkemiddel for design (Leffingwell, 2011), (Faber, 2010)	• Det er store mengder dokumentasjon, men informantene beskriver likevel mangel på nødvendig dokumentasjon. • Mange steder mangler målbilder og tilsvarende førende dokumenter. • Det er uklare krav til hvordan man dokumenterer arkitektur og design. • Designdokumentasjon er ofte svært detaljert • Arkitektene forholder seg sjelden til kode.	• Dokumentasjon må målrettes i større grad, og sees i sammenheng med hvilke behov den skal dekke, og hvem det kommuniseres til. • Designdokumentasjon har kort levetid, og kan suppleres med muntlig kommunikasjon • Systemdokumentasjon har lengre levetid, og det er ikke alltid mulig for mottakeren å stille spørsmål. • Systemdokumentasjon kan delvis genereres basert på kode og konfigurasjonsdata. • Målbilder/visjoner bør utvikles i samarbeid mellom arkitekter og utviklingsteam, for å sikre bedre *Arkitekturforståelse*.

Tema	Innsikt	Funn	Diskusjon
Arkitekturforståelse	• Arkitekter må tilrettelegge for kommunikasjon mellom utviklingsteam og viktige kravstillere. (Madison, 2010), (Blair et al., 2010). • Arkitekten må delta i utviklingsteamet, og ha kompetanse nok til å kunne diskutere tekniske spørsmål. (Faber, 2010), (Madison, 2010), (Blair et al., 2010)	• Arkitektene har sjelden tid til å delta i utviklingsarbeidet. • Overleveringer av oppgaver og dokumentasjon skjer ofte uten muntlig kommunikasjon. • Intensjonen bak arkitekturen går ofte tapt på veien fram til utvikler. • Form på dokumentasjon er ikke forstått av målgruppen • Det er et ønske om mer direkte og muntlig kommunikasjon om arkitektur mellom aktørene.	• Manglende arkitekturforståelse kan være et resultat av utfordringer under flere av de andre temaene, som kontekst for prosessen, implementasjon av selve prosessen og utforming av arkitekturbeskrivelser. • Kommunikasjon og lav grad av deltakelse legger ikke til rette for god arkitekturforståelse. • Flere av arkitektene uttrykker et ønske om å være mer til stede, men oppgir tid som en hindring. Ved å prioritere innsatsen etter risikovurdering, og hvor det er behov for hjelp, kan man frigjøre tid.
Analyse av påvirkning	Analyse av arkitekturpåvirkning kan være en tidkrevende aktivitet.(Yang et al., 2016)	• En vanlig konsekvensutredning er i stor grad analyse av påvirkning. • Utfordrende å analyser påvirkning, pga manglende dokumentasjon, dette gir avhengighet til utviklingsleverandør	• *Analyse av påvirkning* kan bli enklere dersom systemene har bedre dokumentasjon.
Arkitekturgjenbruk	En svakhet med smidige metoder er at man ikke nødvendigvis legger til rette for arkitekturgjenbruk. (Durdik, 2011)	• Standardisering og krav til gjenbruk av uegnet funksjonalitet, oppleves som en hindring for smidig tilnærming • Standardisering av infrastruktur som integrasjon og sikkerhet oppleves som en støtte for smidig tilnærming.	• Det er viktig at standardisering og gjenbruk er virkemiddel for å understøtte effektivitet og reduserte kostnader, ikke et mål i seg selv.
Arkitekturgjenvinning	*Ikke beskrevet i gjennomgått litteratur*	• Det er utfordrende å vedlikeholde sentrale oversikter over avhengigheter mellom systemer • Dokumentasjon genereres i liten grad basert på kode.	• Kan vurdere å ta i bruk verktøy som kan generere dokumentasjon basert på koden som er satt i produksjon. Kan frigjøre tid, og legge til rette for bedre *Analyse av påvirkning*

Tema	Innsikt	Funn	Diskusjon
Arkitektur-refaktorering	Behov for arkitekturrefaktorering kan identifiseres gjennom inspeksjoner av arkitekturen etter hver sprint. (Sharifloo et al., 2008)	• Noen av informantene mener at en tilnærming der man gjør mindre designarbeid på forhånd, og baserer seg mer på refaktorering underveis vil kunne redusere risiko. Det er imidlertid lite erfaring med dette i NAV.	• Refaktorering er en viktig del av smidige metoder. Dersom Sharifloos tilnærming skal tas i bruk forutsetter det at produkteiere som skal prioritere hva som gjøres under utvikling forstår behovet.

Vedlegg 7: Informasjonsskriv til informanter

Forespørsel om deltakelse i forskningsprosjektet

Smidige arkitekturprosesser

Bakgrunn og formål

Jeg arbeider med en masteroppgave på studiet «IT og ledelse» ved Institutt for Informatikk på Universitetet i Oslo.

Som en del av masteroppgaven skal jeg gjennomføre en studie på egen arbeidsplass, det vil si IT-avdelingen i Arbeids- og velferdsdirektoratet. Temaet for oppgaven er hvordan arkitekturprosessene kan tilpasses krav om raskere IT-leveranser. Jeg ønsker å undersøke hvilke hindringer og muligheter som finnes i IT-avdelingen for å ta i bruk en smidig tilnærming i arkitekturarbeidet.

Du er valgt ut til å delta fordi oppfyller et eller flere av disse kriteriene

- Fyller en arkitektrolle i linje eller prosjekt (løsningsarkitekt, områdearkitekt, teamarkitekt hos leverandør)
- Jobber med å definere og utforme metoder og prosesser for arkitektur
- Påvirkes av arkitekturarbeidet i rollen du utfører, f.eks. ved å være avhengig av avklaringer, analyser eller beslutninger fra arkitekturfagmiljøet.

Hva innebærer deltakelse i studien?

Din deltakelse i studien innebærer at du stiller i et intervju om temaene i studien, enten alene eller som del av en gruppe. Intervjuene vil ha en varighet på 60-90 minutter.

Intervjuene vil dreie seg om dine erfaringer relatert til arkitekturarbeid i IT-avdelingen, din opplevelse av hvordan dagens prosesser er til hinder og/eller til hjelp for hurtige leveranser.

Det vil bli gjort lydopptak av intervjuet. Lydopptakene brukes for å transkribere intervjuene i etterkant. De vil ikke brukes til andre formål enn denne studien.

Hva skjer med informasjonen om deg?

Alle personopplysninger vil bli behandlet konfidensielt, og vil ikke være tilgjengelig for andre enn min veileder ved UiO og meg.

Personopplysningene det er aktuelt å lagre er:

- Ditt navn
- Din rolle/stilling i organisasjonen
- Informasjon om utdanning og relevant arbeidserfaring

I den ferdige rapporten vil ikke ditt navn framkomme, men du vil kunne gjenkjennes gjennom beskrivelse av arbeidsoppgaver eller lignende. NAV og IT-avdelingen vil ikke anonymiseres i

rapporten.

Prosjektet skal etter planen avsluttes 1. juni 2017. Lydopptak slettes og transkriberte notater vil anonymiseres etter dette.

Frivillig deltakelse
Det er frivillig å delta i studien, og du kan når som helst trekke ditt samtykke uten å oppgi noen grunn. Dersom du trekker deg, vil alle opplysninger om deg bli anonymisert.

Dersom du ønsker å delta eller har spørsmål til studien, ta kontakt med meg:

- Ida Kristine Dørum, tlf.: 94 33 44 86, epost: ida.kristine.dorum@nav.no

Du kan eventuelt også ta kontakt med min veileder ved UiO, Bendik Bygstad, epost: bendikby@ifi.uio.no.

Studien er meldt til Personvernombudet for forskning, Norsk samfunnsvitenskapelig datatjeneste AS.

Samtykke til deltakelse i studien

Du ansees for å ha gitt samtykke til deltakelse ved å gi et positivt svar på denne henvendelsen på epost til ida.kristine.dorum@nav.no

Vedlegg 8: Godkjenning fra NSD

Bendik Bygstad
Institutt for informatikk Universitetet i Oslo
Postboks 1080 Blindern
0316 OSLO

Vår dato: 22.04.2016 Vår ref: 47969 / 3 / STM Deres dato: Deres ref:

TILBAKEMELDING PÅ MELDING OM BEHANDLING AV PERSONOPPLYSNINGER

Vi viser til melding om behandling av personopplysninger, mottatt 15.03.2016. Meldingen gjelder prosjektet:

47969	*Smidige arkitekturprosesser*
Behandlingsansvarlig	*Universitetet i Oslo, ved institusjonens øverste leder*
Daglig ansvarlig	*Bendik Bygstad*
Student	*Ida Kristine Dørum*

Personvernombudet har vurdert prosjektet og finner at behandlingen av personopplysninger er meldepliktig i henhold til personopplysningsloven § 31. Behandlingen tilfredsstiller kravene i personopplysningsloven.

Personvernombudets vurdering forutsetter at prosjektet gjennomføres i tråd med opplysningene gitt i meldeskjemaet, korrespondanse med ombudet, ombudets kommentarer samt personopplysningsloven og helseregisterloven med forskrifter. Behandlingen av personopplysninger kan settes i gang.

Det gjøres oppmerksom på at det skal gis ny melding dersom behandlingen endres i forhold til de opplysninger som ligger til grunn for personvernombudets vurdering. Endringsmeldinger gis via et eget skjema, http://www.nsd.uib.no/personvern/meldeplikt/skjema.html. Det skal også gis melding etter tre år dersom prosjektet fortsatt pågår. Meldinger skal skje skriftlig til ombudet.

Personvernombudet har lagt ut opplysninger om prosjektet i en offentlig database, http://pvo.nsd.no/prosjekt.

Personvernombudet vil ved prosjektets avslutning, 01.06.2017, rette en henvendelse angående status for behandlingen av personopplysninger.

Vennlig hilsen

Kjersti Haugstvedt

Siri Tenden Myklebust

Kontaktperson: Siri Tenden Myklebust tlf: 55 58 22 68
Vedlegg: Prosjektvurdering

Dokumentet er elektronisk produsert og godkjent ved NSDs rutiner for elektronisk godkjenning.