Masteroppgave i informasjonssystemer

Fakultet for økonomi og samfunnsfag
Høgskolen i Agder - Våren 2007

Agile Systemutviklingsmetoder i praksis

En flercasestudie blant bedrifter på Agder

Atle Sørensen
Erlend Sørgaard Egeland

Atle Sørensen
Erlend Sørgaard Egeland

Agile Systemutviklingsmetoder i praksis

En flercasestudie blant bedrifter på Agder

Masteroppgave i informasjonssystemer
Våren 2007

Høgskolen i Agder

Institutt for informasjonssystemer, Fakultet for økonomi og samfunnsfag

Forord

Denne oppgaven er en del av mastergradsprogrammet ved institutt for informasjonssystemer ved Høgskolen i Agder, og er skrevet som avslutningen på studiet i løpet av vårsemesteret 2007.

Ved å samarbeide med fem forskjellige bedrifter på Sørlandet har det blitt mulig å få frem de resultatene vi har gjort i løpet av den studien vi har gjennomført. Vi har i denne forbindelse tatt for oss bruk av agile systemutviklingsmetoder i praksis og ulike faktorer som påvirker hverandre når slike metoder skal taes i bruk.

Vi vil selv si at arbeidet med oppgaven har vært lærerikt og nyttig for oss, med tanke på at vi har fått økt innsikt i hvordan bedrifter faktisk arbeider med dette emnet i praksis. I tillegg til dette har vi også arbeidet i feltet med ekte prosjekter og ikke kun med fiktive prosjekter som tidligere har vært vanlig for oss i forskjellige studiesammenhenger. Dette er da for øvrig erfaringer vi ser kan være nyttige for oss i ettertid i et kommende arbeidsliv. Vi besitter nå en innstegskunnskap om hvordan mennesker arbeider i slike bedrifter, hva som er viktig når man arbeider i team og dermed hva vi selv kan fokusere på når vi skal ut i jobb.

Samarbeidet på oppgaven kom i stand fordi vi har sett det hensiktsmessig å arbeide to personer sammen slik at vi kan rette hverandre underveis og holde motivasjonen oppe gjennom hele prosessen. Dette samarbeidet har fungert svært bra, og mye av dette har sin forklaring i at vi tidligere har arbeidet sammen på andre prosjekter. Vi vet dermed hvor vi kan komplettere hverandre i det arbeidet vi gjør.

Til slutt ønsker vi å benytte oss av muligheten til å takke dem som har bidratt til at oppgaven har blitt som den har blitt. Først og fremst ønsker vi å takke de fem utviklingsbedriftene som har muliggjort gjennomføringen av intervjuene våre. Vi sender også en stor takk til vår veileder Even Åby Larsen for kontinuerlig konstruktiv kritikk for å forbedre oppgaven, og også for tips og triks gjennom hele prosessen. I tillegg ønsker vi å takke våre foreldre for kreative innspill, korrekturlesning og støtte. Vi ønsker dessuten å takke våre medstudenter ved masterstudiet for gjensidig motivasjon, støtte og kreative innspill. Til sist vil vi takke alle andre som har bidratt med informasjon, teknikker og andre innspill som har gjort oppgaven mulig.

Kristiansand 30. Mai 2007

Erlend Sørgaard Egeland

Atle Sørensen

Sammendrag

Agile systemutviklingsmetoder ansees av mange som ganske nye og forholdsvis løse. Det er tidligere blitt gjort forskning på hvordan disse bør brukes i teorien, hva som er viktig når man skal velge ut en slik metode, og når man i teorien skal bruke den. Det er mange faktorer som påvirker metoder, og det finnes også en del holdninger til hvorfor de fungerer eller ikke. Bruken av metoder er også en kjent problemstilling, hvor vidt metodene kan brukes eller ikke. Mesteparten av den forskningen som tidligere er gjort, har blitt foretatt av personer som har preferanser enten for eller i mot slik systemutvikling, og rundt metoder generelt. Men hva er da holdningene rundt bruken av slike metoder i praksis og hva er det som er avgjørende faktorer for at bedrifter faktisk velger å benytte seg av slike metoder?

Vi har i denne oppgaven undersøkt hva det er som faktisk påvirker valget av agile metoder, hva slags elementer som faktisk brukes i praksis, og hvordan disse påvirker hverandre. Det vi har forsøkt å oppnå med dette, er å skape et mer helhetlig bilde av denne situasjonen i motsetning til hva som tidligere er gjort. Ved å gjennomføre en flercasestudie av 5 ulike bedrifter på Sørlandet har vi kunnet belyse disse spørsmålene. Resultatet av disse intervjuene er en modell som illustrerer sammenhengene mellom disse faktorene. Modellen består av 9 forskjellige faktorer som er essensielle for de bedriftene vi har snakket med. Faktorene er:

- Evaluering
- Test
- Leveranse
- Dokumentasjon
- Metode
- Team
- Krav
- Kommunikasjon
- Kundeforhold

Disse ni faktorene er en generalisering av de funnene vi har gjort i våre intervjuer. Ved å undersøke disse ni faktorene har vi lokalisert 12 sammenhenger. Mange av disse sammenhengene kan bli bekreftet av tidligere teori, men vi har også funnet nye sammenhenger i forhold til hva som er sagt fra før. Spesielt vil vi trekke frem involvering og samarbeid med kunder, og det sterke fokuset på kommunikasjon og hvordan man kommuniserer som viktige funn. Ut over dette har vi også fått frem at det er enkelte av de faktorene vi har funnet som ansees, av de bedriftene vi har snakket med, en del mer avgjørende enn hva tidligere forskning har påpekt.

Vi anser våre funn som et bidrag i form av noen nye avdukninger innen faktisk bruk av agile systemutviklingsmetoder. Vår modell vil være nyttig for å kunne se sammenhenger og viktige faktorer i agil utvikling. I tillegg gir den en plattform for videre forskning innen området.

Innhold

Figurer

Vedlegg

1 Introduksjon

Utviklingsmetoder er helt essensielle innen programvareutvikling; man kan ved å velge feil metode risikere at hele utviklingsprosessen feiler, og at hele prosjektet er i fare. Det har i flere år vært en jakt på den universelle metoden, som passer i alle utviklingskontekster, team og organisasjoner. Stadig nye metoder har blitt utviklet, der alle har sine styrker og svakheter. Dette er slik vi tolker Martin et al.(2003). Agile metoder har fått en større og større tilhengerskare de siste årene, ettersom krav og forretningsmiljø er i stadig forandring. I tillegg er fokus på tilpasning av metode og produkt i tillegg til kvalitet på produktet blitt viktige begrep innen systemutvikling. Motivene for bruk av metoder kan variere, et motiv kan være sitatet under:

"We may be doing it wrong, but we're doing so in the proper and customary manner"

Fitzgerald et al. (2002, p. 3)

Andre motiver for å bruke utviklingsmetoder kan være å tilfredsstille kundene bedre, kutte kostnader og øke effektivitet, forbedre dokumentasjon eller på grunn av spesielle krav som blir satt til utviklerne.

"Development organizations are faced with a plethora of choices regarding issues such as resources, type of architecture and approach. There are no absolute "right" answers to these issues; timing, industry orientation and specific enterprise requirements will dictate the proper response. Moreover, "consumerization of IT" trends are introducing more chaos and application variability into the mix. "

Hotle (2006b, p. 2)

Det finnes, som det blir pekt på av i sitatet over, et utall forskjellige utviklingsmetoder. Flere av disse har blitt utviklet for å løse spesielle problemer som utviklere har kommet over. Man kan derimot vanskelig finne en perfekt metode som passer perfekt inn i alle utviklingskontekster. Det er derfor vanlig å ta utgangspunkt i en metode og videreutvikle denne, ved at man legger til og trekker fra teknikker og verktøy som man ser man kan ha nytte av i en bestemt situasjon. Hva som er viktigst når man velger metode, kommer veldig an på konteksten til utviklingen - noen prosjekter krever mye kommunikasjon, er kritiske for bedrifter, krever rask levering eller har veldig vage krav.

Det er skrevet en del litteratur om hva som er viktig for agile systemutviklingsmetoder og hva som er fordelene med de ulike metodene. Det er derimot ikke fokusert veldig på å lage en større oversikt over hva som påvirker agil utvikling og hvilke elementer som blir ansett som viktige. Vi ønsker i denne studien å ta for oss elementer i agil systemutvikling og hvordan disse henger sammen, hvordan disse elementene blir påvirket og påvirker hverandre både når det gjelder valg av metode, men også i løpet av selve utviklingsprosessen. Det vil være en fordel å kunne ha en modell å gå ut fra når man skal orientere seg i utvalget av metoder, og for å være oppmerksom på hva som påvirker utviklingen. Denne modellen som sier noe om disse ulike faktorene vil vi komme tilbake til når vi presenterer den i kapittel fem..

1.1 Problemstilling

Under gjennomføringen av en initiell litteraturstudie og fra tidligere kunnskap/erfaringer, fant vi mange ulike artikler innen agile utviklingsmetoder. En del av kjennetegnene på de ulike

artiklene fra agil utvikling er at de er ofte skrevet av personer med tydelige preferanser enten for eller imot agil utvikling. Det er foretatt flere studier innenfor feltet utviklingsmetoder, men ingen direkte studier hvor man undersøker de underliggende holdningene og grunnen til å bruke agile utviklingsmetoder. Vi ønsker å se på de bakenforliggende grunnene og holdningene til agil utvikling. I tillegg ønsker vi å se på sammenhengende mellom de ulike elementene som finnes i en agil utvikling.

Forskningsspørsmålet vårt blir da:
Hvilke elementer brukes i praksis i en agil systemutviklingsmetode? Hvordan påvirker disse hverandre, og hva er motivasjonen for å bruke dem?

1.2 Motivasjon

Agil programvareutvikling har, hvis man kan si det slik, truffet en nerve i programvaremiljøet. Det finnes de som argumenterer for det, de som er mot det, og de som prøver å blande sammen dette med den tradisjonelle eller plandrevne formen for programvareutvikling. I tillegg er det også mange som undrer seg om hva agilitet egentlig er (Williams & Cockburn, 2003). På bakgrunn av dette ser vi også vår motivasjon for å velge den vinklingen vi har gjort i vår mastergradsoppgave. Vi har tidligere vært inne på vår problemstilling og ser dette som en av drivkreftene til at vi velger å gjennomføre vår studie på akkurat dette emnet. Vi ser at det mangler en større oversikt over elementer som inngår i og påvirker agil systemutvikling. Når vi prøver å lage en slik modell, vil det være med på å forstå bruk av agile metoder bedre samtidig som det kan gi en bredere oppfatning av hvordan systemutvikling fungerer.

1.3 Posisjonering

Bruk av metoder er et kjent fenomen innen programvaremiljøet. Dette kan enten være noe som kun er i utviklerens hode eller som er nøye spesifisert i dokumenter og retningslinjer (Cockburn, 2002; Fitzgerald et al., 2002). Metoder kan være store og rigide med mange regler og retningslinjer, eller små rammeverk uten mange detaljer (Cockburn, 2002). Flere store utviklingsprosjekt feiler å levere hva som var planlagt og ønsket, eller feiler på grunn av store endringer, i følge Standish Group (1994). De siste årene er det blitt mer fokus på agile metoder og fokus på å kunne møte stadig raskere endringer fra markedet. Vår oppgave vil fokusere på hvordan man bruker agile metoder og hva slags holdning utviklere har til disse metodene. Basert på vår oppgave vil det være mulig å se på hvordan man driver utvikling i dag og på eventuelle forbedringer man kan gjøre i fremtiden. Vi ønsker å se objektivt på bruken av agile metoder, i motsetning til mye av litteraturen som til nå er publisert, som er veldig subjektiv og skrevet av personer med stor tilknytning til det agile miljøet.

Vår oppgave vil fokusere på SMB-markedet i bedrifter rundt Kristiansandsområdet. Dette er bedrifter som har relativt små prosjekter, og har store forutsetninger for å innføre agile utviklingsprosesser. Ut over dette vil vi også forsøke å finne ut noe som kan være generaliserbart for bruk av metoder generelt i praksis, men hovedsaklig for agile metoder.

1.4 Oppgavens struktur

Oppgaven består syv forskjellige kapitler, hvor hvert av disse tar for seg et unikt område. Først gir vi en introduksjon til oppgaven. Her plasserer vi feltet (inkluderer også posisjonering og vår motivasjon for oppgaven), presenterer problemstilling.

Det andre kapittelet tar for seg forskningsmetoden vår, hvor vi beskriver hva vi skal gjøre i følge metoden. Dette vil da si teoretisk, før vi så beskriver hvordan vi har gjennomført studiet i praksis.

Tredje og fjerde kapittel er en litteraturstudie. Her definerer vi sentrale begreper og konsepter, før vi så gir en presentasjon av tidligere relevant forskning på området.

Femte og sjette kapittel tar for seg hva vi har funnet gjennom studien. Først presenterer vi konkret i kapittel fem hva vi har funnet, før vi så i kapittel seks diskuterer disse resultatene opp mot den tidligere forskningen vi presenterte i kapittel tre og fire. I tillegg diskuterer vi her hva av resultatene som kan sees som et bidrag til det feltet vi tar for oss.

Siste og sjuende kapittel er konklusjonen. Her gir vi en oppsummering av hva vi har funnet før vi så presenterer hva som er begrensningene i vår forskning og hva som kan forskes videre på av våre resultater.

2 Forskningsmetode

Forskningsmetoden er en beskrivelse av hvordan vi ønsker å legge opp studien vi skal gjennomføre. Vi har ikke fokusert på en konkret metode, men vi har valgt åtte steg (Remenyi et al., 1998) som vil være grunnlaget eller malen for stegene vi skal gå gjennom i løpet av forskningen vi skal gjennomføre:

1. Studere litteratur
2. Formalisere forskningsspørsmål
3. Etablere forskningsmetode
4. Samle data
5. Analysere data
6. Utarbeide konklusjoner
7. Forstå begrensningene av forskning
8. Produsere anbefalinger og retningslinjer

Bakgrunnen for at vi har valgt denne typen metode (Remenyi et al., 1998), er at vi ikke har sett det hensiktsmessig å bruke en metode med mange regler. Vi har sett det som bedre for det arbeidet vi skal gjøre å ha en start hvor vi har retningslinjer som sier noe om hva vi skal følge. I tillegg til dette har vi, som vi skal komme inn på senere, valgt en modell som viser analysen av data (Jakobsen, 2000) som vi tilegner oss. Årsaken til dette er at vi til en viss grad ikke var klar over hva slags type funn vi ville komme til å finne gjennom studien. I ettertid har vi sett at det vi har funnet har vært slik at vi har kunne bruke tekniker fra grunngitt teori (e: grounded theory) (Esteves et al., 2002). Dette har da vært fordi en slik teknikk muliggjør produksjonen av det vi etter hvert har sett at vi vil komme til å kunne forklare. Vi har da brukt grunngitt teori som et supplement. Dette vil da si at vi har de åtte stegene vi presenterte tidligere som ryggrad og bruker teknikker fra grunngitt teori for å analysere data.

Vi vil videre i kapitlet (dvs. avsnittene som kommer) beskrive helt konkret hva vi kommer til å gjøre arbeidet vårt i løpet av forskningsprosessen. Hvordan dette har blitt gjennomført i praksis, kommer vi tilbake til i slutten av dette kapitlet.

2.1 Studere litteratur

Ved å studere litteratur i forkant og i tidlige faser av oppgaven vår ønsker vi å få en oversikt over temaet utviklingsmetoder. Det å starte bredt innenfor området utviklingsmetoder og deretter snevre inn og sortere ut artikler vil sikre at vi får med oss viktige områder innen litteraturen.

Starten av litteraturstudiet vil altså være et studium av et bredt spekter av litteratur for å få oversikten over hva som finnes av gjort arbeid i feltet vi skal gå gjennom. Etter hvert som vi får gjort noen analyser av de dataene vi har funnet, vil vi luke ut det vi har funnet av litteratur som ikke er relevant for de funnene vi har gjort. I tillegg vil vi spe på med litteratur som vi har i ettertid funnet ut er interessant, men som ikke var med i den oversikten vi startet med. Til sist vil vi foreta en omorganiseringsprosess som strukturer litteraturen slik at den blir kategorisert i lik linje med de funn vi har gjort, fordi dette vil forenkle lesingen av dokumentet og da også gi det en bedre konstruksjon slik at det blir ryddig og lettere å lese.

2.2 Formalisere forskningsspørsmål

Dette spørsmålet er et av de viktigste elementene i oppgaven vi skriver, i og med vi her sier hva vi ønsker å oppnå med det vi skriver. Formaliteten av spørsmålet vil utvikle seg og bli mer nøyaktig etter hvert som oppgaven får en mer spisset form, vi får oversikt over litteratur og får informasjon fra de ulike respondentene vi har vært kontakt med og dermed få kokt ned forskningsspørsmålet til en eller et par setninger.

2.3 Etablere forskningsmetode

Hva slags metodologi som blir brukt, kan bli påvirket av tid, penger, forskningsspørsmål og dyktigheten til forskerne (Remenyi et al., 1998). Ettersom dette er en masteroppgave, har alle disse faktorene spilt inn, da spesielt tid og penger. Vi har valgt å basere oss på en kvalitativ analyse av utvalgte bedrifter innenfor Kristiansandsområdet. Årsaken til dette er at vi som sagt har begrensede ressurser og vil da ikke ha muligheten til å reise rundt ut over nærmiljøet. I og med at vi ønsker å se på holdningene og bruken av agile metoder, har vi valgt å utføre forskningen som en intervjurunde. Ved å gjennomføre intervju med flere ulike bedrifter vil vi kunne lokalisere holdninger og faktisk bruk, innenfor ulike utviklingsmiljø. Motsetningen til dette vil være en casestudie der vi får holdninger fra kun en part, eller en kvantitativ studie der det er vanskelig å få en begrunnelse rundt svarene som blir gitt.

Vårt ståsted i denne oppgaven vil være basert på fenomenologi (e: phenomenology). I motsetning til positivistisk tankegang innser vi at det ikke vil finnes et entydig svar på vårt forskningsspørsmål. Svarene vi vil få i intervjuene vil avhenge av menneskelige oppfatninger og variere fra bedrift til bedrift. Likevel ønsker vi å kunne komme opp med noen generelle oppfatninger og meninger som vi har funnet i intervjuene fordi det gir et innblikk i hva slags holdninger næringslivet har.

Metoden vi har brukt er til en viss grad grunngitt teori. Avviket fra hvordan grunngitt teori faktisk er beskrevet i teorien, er at vi har tatt utgangspunkt i en tidligere forskningsartikkel (Patel et al., 2006), hvor man har funnet ut hva som er viktig med tanke på agilitet. Vi har i den forbindelse ikke vært like åpne i starten av forskningsprosessen, slik som det blir beskrevet at grunngitt teori skal brukes.

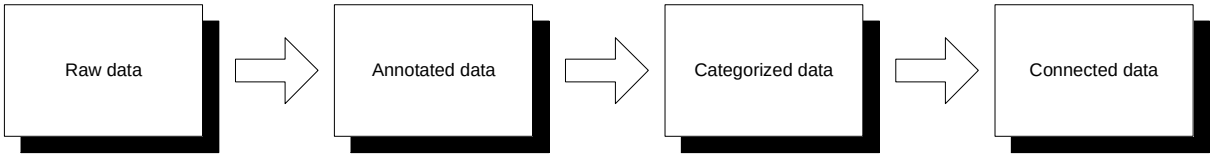
2.4 Samle data

Innsamling av data vil skje ved hjelp av intervjuer med ulike utviklingsbedrifter i Kristiansand og omegn. Personer som vi intervjuer vil stort sett være prosjektledere og personer som jobber innen og med utvikling innenfor hver bedrift. Intervjuene vil bli tatt opp med diktafon, så skrevet ned og bli sendt til intervjuobjektene for godkjenning.

Blir det for mye struktur på intervjuet, kan det skje at man ikke fanger opp eller misforstår fenomener som er viktige for respondenten (Ryen, 2002). Vi har på bakgrunn av dette valgt å ha et relativt åpent intervjuoppsett. Selv om vi har valgt å fokusere på de agile prinsippene og holdninger til disse, har vi ikke direkte spurt om holdninger rundt dette. Grunnen til det er at hvis man konfronterer respondentene med direkte utsagn, kan det føre til at de blir påvirket til å være enig eller uenig med utsagnene, istedenfor å gi oss svar på hvordan de faktisk oppfatter ulike fenomen. Vi har laget et oppsett som beskriver ulike temaer vi ønsker å gå innpå i intervjuet, og stikkord rundt spørsmål vi kan stille. På bakgrunn av disse temaene vil vi tilstrebe en åpen kommunikasjon med respondentene, for på den måten avdekke holdningene deres rundt temaene.

2.5 Analysere data

For å analysere de data og funn som kommer ut av dette studiet har vi valgt å bruke en modell presentert av Jakobsen (2000), se Figur 1: Jakobsen (2000) forskningsmodell.



Figur 1: Jakobsen (2000) forskningsmodell

Denne modellen gir oss muligheten til å kunne starte med rå data på den ene siden, slik som vi tilegner oss ved våre møter / intervjuer med de ulike bedriftene. Videre kan vi kommentere disse data og funn, samt plassere disse i kategorier, dette gjelder da steg 2 og 3 i modellen til Jakobsen (2000). Siste steget i modellen (steg 4) vil være å koble sammen data og funn, dette vil lede til at man kan produsere generaliserte data som kan kobles til andre prosjekter og ikke bare til de bedriftene vi har tatt utgangspunkt i. Vi vil i analysearbeidet bruke teknikker fra grunngitt teori.

Grunngitt teori er en forskningsmetode for å bygge teorier på bakgrunn av innsamlede og analyserte data (Esteves et al., 2002). Grunngitt teori skal hjelpe forskeren med å forklare fenomen man finner i datainnsamlingen, og deretter sette dette opp mot relevant teori.
Det finnes tre hovedprosedyrer i grunngitt teori (Esteves et al., 2002):

- Åpen koding: Merking av konsepter som representerer atskilte hendelser og andre instanser av fenomenet som studeres.
- Aksial koding: Data er satt sammen og kategorisert etter åpen koding. Dette skjer ved hjelp av å finne sammenhenger mellom dataene og kategorier.
- Selektiv koding: Man velger en hovedkategori og knytter denne sammen med de andre kategoriene, man validerer knytningene og fyller ut kategorier som trenger mer utvikling.

Beskrivelsen av hvordan disse fasene er gjennomført, blir presentert senere i dette kapitlet når vi går nærmere inn på hvordan vi har foretatt studien i praksis i den delen hvor vi beskriver dataanalysen vi har foretatt oss.

2.6 Utarbeide konklusjoner

Det er viktig at man presenterer leseren for noe nytt i forhold til hva som fantes av tidligere kunnskap på området, for å kunne bidra til det store kunnskapstreet (e: add to the body of knowledge). Med andre ord skal man gjennom en diskusjon overbevise dem man skriver for om det man kommer frem til i konklusjonen (Remenyi et al., 1998). Det er dette vi vil gjøre i denne fasen av oppgaven, hvor vi vil se resultatet av den selektive kodingen vår. Ved å se sammenhenger mellom kategorier ønsker vi å komme med en modell eller generelle punkter ved bruk av og holdninger til agile metoder. Etter utarbeidelsen av denne forklaringsmodellen vil vi knytte denne modellen, inkludert dens sammenhenger, opp mot den teorien vi tidligere har presentert. Vi vil da se på om modellen vår støtter tidligere forskning, om det er noe som er motstridig i forhold til forskningen som er tidligere gjort, og til sist om det er noe som inngår i modellen som er nytt i forhold til hva den tidligere forskningen har tatt for seg.

Konklusjonen vil drøfte om teorien rundt agile metoder faktisk blir brukt og blir ansett som fordelaktig av personer som driver programvareutvikling. Dette vil da også være en oppsummering av de funn som vi mener er de viktigste vi har kommet frem til, og en beskrivelse av hva som faktisk er folks holdninger til og motivasjon for å bruke agile systemutviklingsmetoder i praksis.

2.7 Forstå begrensning av forskning

Når man arbeider med en problemstilling, får man etter hvert som arbeidet går sin gang en bredere forståelse av emnet. Dette vil da si at dersom man skulle begynne på arbeidet på nytt, ville man kunne utvide hva man så etter og også fokusere på andre områder for å gjøre en bedre studie. Slikt skaper rom for at man da forstår begrensningene av sin egen forskning. (Remenyi et al., 1998). For å kunne reflektere over eget arbeid er det viktig at man klarer å forstå begrensninger av forskningen. Dette innbærer at man har en diskusjon rundt eget arbeid og tema, hvor dette er basert på funn (kan også kalles resultater). Her diskuterer man rundt de funn man har gjort, men også i forhold til hva man eventuelt kunne ha forbedret.

2.8 Produsere anbefalinger og retningslinjer

I mange mastergradsoppgaver og doktorgradsavhandlinger vil flere konkludere på en måte hvor man konverterer funn til anbefalinger. Slike anbefalinger kan bli sett på av andre forskere innen samme felt for å kunne kontrollere enigheten blant disse (Remenyi et al., 1998). På bakgrunn av konklusjonene i forskningen bør man foreslå noen retningslinjer eller punkter som ledere kan innføre for å bedre lønnsomhet eller effektivitet (nødvendigvis ikke i forhold til kapital, men for eksempel for å oppdrive bedre utviklingsteam og økt produktivitet). Disse retningslinjene kan eventuelt måles opp mot hvor enig ledere er i å bruke dem eller opp mot relevant teori. Vi vil også komme inn på hva vi har sett som av begrensninger i vår forskning og foreslå det som er mest interessant her som retningslinjer for hva andre eventuelt kan ta med seg videre til annet arbeid og også mulige forbedringer som bør bli gjort i en mulig ny studie. Forskningen som til nå er gjort har store preferanser for og imot bruken av agile systemutviklingsmetoder.

2.9 Gjennomføring av studien

I dette underkapitlet vil vi gå gjennom hvordan den praktiske delen av studien vår har foregått. Helt konkret vil dette si at vi presenterer intervjuobjektene vi har benyttet oss av, hvordan og hvorfor vi har valgt disse, hvordan vi har gjennomført intervjuene i praksis (oppbygning av spørsmål og gjennomføring i praksis) og til sist hvordan vi har analysert de data vi har samlet inn i denne prosessen. Kort sagt er dette et kapittel hvor vi rapporterer det vi har gjort i forbindelse med den flercasestudien vi har fortatt.

2.9.1 Datainnsamling

Datainnsamlingen vår har bestått av intervjuer med ulike involverte i utviklingsfirmaer. Vi vil i dette avsnittet dokumentere hvordan denne innsamlingen har blitt gjennomført i praksis.

Intervjuobjekter

For å kunne sette oppgaven vår inn i en kontekst og for å kunne tolke resultatene på best mulig måte er det viktig å ha fokus på intervjuobjektene. Derfor vil vi i dette avsnittet gå igjennom de fem intervjuobjektene våre og beskrive dem. En slik beskrivelse vil hjelpe i analysearbeidet og hjelpe oss å sette de resultatene vi får inn i en kontekst. I tillegg har vi valgt å anonymisere intervjuobjektene slik at deres utsagn ikke skal kunne kobles direkte til deres organisasjon (f eks via navn).

Tabell 1: Oversikt over intervjuobjekter

Objekt 1	o Liten utviklingsbedrift, 6 ansatte, 2 utviklere, 2 designere, en altmuligmann og daglig leder/selger o Arbeider mest med relativt små webutviklingsprosjekt, og publiseringsløsninger o Kontor i Kristiansand o Arbeider ikke etter noen formell dokumentert metode o Intervjuet utvikler og designer
Objekt 2	o Liten utviklingsbedrift, 6 utviklere i tillegg til to implementeringsteam, daglig leder og 2 regnskap/ brukermanual/ designere o Utvikler presentasjons-, prosjekt- og beslutningsstøtteverktøy for store internasjonale kunder o Kontor i Kristiansand, implementeringsteam over hele verden der verktøyene skal installeres o Arbeider etter egendesignet metode basert på RUP o Intervjuet utviklingslederen
Objekt 3	o Mellomstor utviklingsbedrift, ca. 20 ansatte i Kristiansand inkludert testere, utviklere og designere o Hovedkontor i Kristiansand, utviklere også i India og Oslo o Utvikler klient- og serverhyllevare for oljeindustrien. Store internasjonale kunder o Intervjuet prosjektleder for utvikling av serverprogramvare og prosjektleder for utvikling av klientprogramvare o Teamet som utviklet serverprogramvaren brukte MSF for agile (spiral og fossefallsmetode) o Teamet som utviklet klientprogramvaren brukte Scrum , til punkt og prikke
Objekt 4	o Mellomstor lokal utviklingsbedrift med ca. 40 ansatte o Hovedkontor i Kristiansand o Utvikler spesialtilpasset programvare for ulike kunder, både lokalt i Kristiansandsområdet og nasjonalt o Intervjuet prosjektleder i et av prosjektene o Brukte Scrum som metode
Objekt 5	o Offentlig etat basert i Sør Norge o Ca. 250 ansatte på flere lokasjoner o Tilpasser og utvikler både nettløsninger ut mot publikum og interne systemer o Intervjuet to prosjektledere o Brukte Oracle metoder Classic og Fastback avhengig av størrelse og kontekst på prosjektet

Valg av intervjuobjekter

Tilgang til intervjuobjekter kan være en stor utfordring, spesielt for personer som kommer fra akademiske miljøer og hvor tidsrammen for forskningen er kort (Remenyi et al., 1998). Ettersom dette er en masteroppgave og vi har hatt kun ett semester å gjennomføre forskningen på, har det vært en utfordring å få et representativt utvalg intervjuobjekter til å stille opp. Det finnes flere typer utvalg, enkelhets- eller tilgangsutvalg (e: convenience sample), som er det vi vil se på (Remenyi et al., 1998). Ettersom vi sendte ut e-post til en rekke utvalgte bedrifter og fikk svar fra seks stykker, fem av disse på et tidlig tidspunkt og den sjette på et sent tidspunkt, har vi benyttet oss av et tilgangsutvalg. Grunnet dette har vi kun benyttet oss av fem av de seks vi fikk tilgang til, i og med det ble for sent å kjøre et siste intervju da den sjette bedriften omsider gav respons (grunnet sykemelding hos kontaktpersonen).

Intervjuobjektene våre faller på mange områder inn i samme kategori, ettersom de er små og mellomstore bedrifter, som utvikler programvare ved hjelp av agile metoder. Det er allikevel en del forskjeller på objektene når man ser nærmere på dem, blant annet organisasjonskultur, modenhet, noen forskjeller på størrelse og organisasjonsstruktur.

Oppbygning av intervjuer
Det finnes to typer intervjuer: åpne intervjuer og semistrukturerte intervjuer (Remenyi et al., 1998). Semistrukturerte intervjuer fungerer slik at man setter opp en intervjuguide, som lar forskeren ha en viss struktur over intervjuet. Ved forskning på flere bedrifter er det spesielt viktig å ha en intervjuguide. Vi lagde en løs intervjuguide (se vedlegg G), hvor vi spesifiserte temaer som vi ønsket å dekke i intervjuet. Vi spesifiserte også noen spørsmål under hvert tema, som vi brukte som en huskeliste for å sørge for å få relativt sammenlignbare resultater.

Gjennomføring av intervjuer
Intervjuene ble foretatt hos bedriftene, med prosjektledere eller andre som var involvert i utviklingsarbeid. Intervjuene ble tatt opp på bånd og skrevet ned i ettertid. Under selve intervjuet var det en av oss som hadde hovedansvar for å stille spørsmål og den andre hadde hovedansvar for notering (i form av stikkord) og observasjon. I tillegg noterte den med ansvart for dette underveis ned hovedpunkter i intervjuet som vi mente var interessante. Som sagt prøvde vi å få en mest mulig åpen tone, og forsøkte å få intervjuobjektene til å holde en åpen samtale, hvor de fortalte om hvordan de drev systemutvikling og hva de syntes var viktig. Vi brukte temaene og spørsmålene for å få i gang dialogen og rette intervjuobjektene inn på de temaene vi ønsket svar på. Bakgrunnen for at vi har valgt oss ut de temaene vi har gjort, er basert på en initiell litteraturstudie og erfaringer og kunnskap fra tidligere kurs ved HiA.

Det har variert hvor vellykket dette har vært, ettersom det er stor forskjell på hvor ”snakkesalige” intervjuobjektene har vært. Noen intervju har fungert meget bra, hvor vi har kommet med stikkord rundt temaer og intervjuobjektet har snakket relativt fritt, mens andre ganger har vi måttet stille mer direkte spørsmål

Dokumentering av intervjuer
For å kunne referere på en ryddig måte til hva som ble sagt i intervjuene har vi sett behovet for å dokumentere det som har blitt sagt av hvert respektive intervjuobjekt. Dette er fordi det vil være enklere å gå tilbake i intervjuene og se hva som ble sagt, i motsetning til dersom man skal sitte og spole frem og tilbake på en båndopptaker. Motsatt av fordelen her er at man må vie mye tid til en slik dokumentasjon (Jakobsen, 2000). På en annen side ser vi det som at fordelene her veier opp for ulempene, og har som sagt da valgt å gjennomføre en slik dokumentasjon.

Som vi har nevnt tidligere, har vi gjort opptak av intervjuene og har ut fra dette foretatt en dokumentasjon av hva som ble sagt. Fremgangsmåten for dette har vært slik at vi har hørt på intervjuene og skrevet ned helt konkret hva alle parter har sagt.

Dataanalyse
Som vi har vært inne på tidligere, har vi planlagt å benytte oss av teknikker fra grunngitt teori som fremgangsmåte for å hente ut data fra intervjutekstene vi har samlet inn. Hvordan vi har gjennomført dette arbeidet i form av forskjellige former for koding, blir presentert videre i dette delkapitlet.

Åpen koding
Åpen koding skjedde ved at vi hver for oss gikk igjennom de nedskrevne intervjuene og markerte ord som vi mente var viktige, og som intervjuobjektene pekte på som viktige. Etter begge var ferdige med denne prosessen, satte vi oss sammen og sammenlignet resultatene. Resultatet av den åpne kodingen var et sett med ord og utsagn som vi mente representerte intervjuene best mulig.

Aksial koding
Ved å selektere og kategorisere ordene man fant i åpen koding skal man her komme frem til et sett hovedkategorier man kan plassere data i. Vi valgte å utforme kategorier fra hvert enkelt intervju, basert hovedsaklig på den åpne kodingen, og plassere data inn i disse kategoriene. Etter at alle intervjuene var gjennomgått og kategorier satt opp for hvert intervju, begynte vi å analysere intervjuene opp mot hverandre. Det ble i denne fasen lagt vekt på å generalisere dataene, og på den måten reduser antall kategorier. Dette er en krevende jobb ettersom man må se etter fellestrekk og prøve å trekke sammen utsagn. Vi hadde noen utfordringer her ettersom hvert intervjuobjekt bruker forskjellige ord om samme fenomen. Etter flere gjennomganger kom vi frem til et sett med ca. 20 kategorier av varierende størrelse. Kategoriene ble organisert i diagrammer med dataene under seg. Se vedlegg (se vedlegg J) for organisering av den aksiale kodingen.

Selektiv koding
Den selektive koding vår har skjedd når vi har begynt å bli litt mer kritisk til hva vi skal velge ut som interessante funn i intervjuene. Her har vi sett på hvordan vi kan slå sammen enkelte av de kategoriene for å forenkle det vi til slutt skal komme frem til i forklaringsmodellen vi har konstruert. Den forenklingen vi gjorde, foregikk i form av at vi satte opp alle kategoriene etter aksial koding i et diagram som kan minne om et klassediagram, men kun med klasser og uten relasjoner. Her så vi i sammenheng hva som kunne slåes sammen til en kategori og hvilke attributter i de forskjellige klassene som kunne slåes sammen for å redusere antallet og øke oversikten. . På bakgrunn av intervjuene og den aksiale kodingen fant vi sammenhenger mellom kategoriene. Flere av disse sammenhengene ble kuttet, etter hvert som vi så hvordan de påvirket hverandre indirekte, både for å forenkle modellen og for å øke forklaringsevnen til denne. Dette var starten på den forklaringsmodellen vi har kommet frem til.

3 Definisjoner

For å plassere feltet vi skal jobbe innen har vi valgt å starte med noen definisjoner. Vi vil ta for oss hva tradisjonell systemutvikling er og hva agil systemutvikling er. Til slutt i dette kapitlet vil vi ta for oss hva en utviklingsmetode er og kommer så med noen eksempler på metoder for hver av disse typene utvikling (tradisjonell og agil).

3.1 Tradisjonell systemutvikling

Tradisjonelle metoder, eller også ofte kalt for plandrevne metoder, som for eksempel fossefallsmetode (e: waterfall) og varianter av denne (SDM, SSDM, SADM, Navigator, ForeFront, Method/1, Summit), har ofte fokus på å drive systemutvikling på en helt annen måte enn den typen vi ser på i denne oppgaven (nemlig agil utvikling). Fokuset disse tradisjonelle metodene har, er at man skal dele kunnskap via dokumentasjon og rollebaserte team, og at man har detaljbaserte planer for hele utviklingsprosessen. Dette vil med andre ord si at man har stor fokus på dokumentering og produksjon av dokumentasjon. Det har også blitt påpekt at man har en beskrivelse av hva som skal bli gjort, hvordan man skal gjøre det og også hvilken tid man har til rådighet helt konkret til å gjøre det man skal gjøre. Denne dokumentasjonen er til stede slik at man skal være sikker på at alle krav, alt design og all administrasjon blir fanget opp og gjort noe med (Chau et al., 2003).

3.2 Agil systemutvikling

I motsetning til tradisjonell programvareutvikling, eller plandrevet utvikling som man også kan kalle det, er agil systemutvikling en form for utvikling der man hilser endring velkommen gjennom alle steg i prosessen, og ikke hogger kravene i sement ved starten av prosjektet. Motivasjonen for å utvikle utviklingsmetoder som støttet denne tankegangen kom frem på midten av 90-tallet fra utviklere som på et vis hadde sett seg lei på den plandrevne måten å utvikle programvare på (Williams & Cockburn, 2003).

Ordet agil og agile metoder og prosesser blir ofte brukt innen programvareutvikling. Hva som er agile metoder blir presentert senere i oppgaven, men først og fremst må vi definere hva agil er. Ifølge Ordnett.no (2007) betyr ordet agil rask/bevegelig/smidig. Dette er også noe som kjennetegner flertallet av agile metoder.

Cockburn (2002) karakteriserer ordet agil med å være effektiv og manøvrerbar. Han definerer en agil prosess som en lett (i form av få regler), men tilfredsstillende prosess. Det blir her også sagt at det ikke er et spørsmål om agile metoder kan bli brukt i et utviklingsprosjekt, men snarere hvordan kan vi holde oss mest agile.

Agile metoder er relativt nye innen applikasjonsutvikling. Hotle (2006a) hevder den første agile metoden var Dynamic Systems Development Methodology (DSDM), som ble utviklet av en gruppe selskaper og publisert i 1994. Andre agile metoder som kan nevnes er:

- EXtreme Programming (XP)
- Scrum
- Crystal methods
- Lean Development
- Adaptive software development
- Rapid application development (RAD)
- Dynamic Systems Development Methodology (DSDM)

I tillegg til de overnevnte metodene finnes det en rekke andre metoder med ulike variasjoner. Grunnen til at agile metoder først ble utviklet, var problemer som oppstod med eldre mer strukturerte metoder, slik som fossefallsmetoden. Problemene med tidligere strukturerte metoder har vært at de er store, formelle og tunge å administrere. Det er også en stor fare for forsinkelser og problemer med testing og endrede brukerkrav. Siden utviklingen deles opp i moduler og hele systemet ikke blir testet sammen før alt er ferdig, risikerer man at store feil ikke oppdages før svært sent i utviklingen.

"Agile development combines creative teamwork with an intense focus on effectiveness and manoeuvrability."

Cockburn & Highsmith (2001a, p. 120)

Dette sitatet er et godt eksempel på hva agile metoder prøver å omfavne. Et av hovedpunktene i agilitet er at metoden skal ha evne til å være manøvrerbar og møte de stadig skiftende kravene fra kunder og omgivelser.

"Agility is dynamic, context-specific aggressively change embracing, and growth-oriented. It is not about improving efficiency, cutting costs, or battening down the business hatches to ride out fearsome competitive "storms." It is about succeeding and about winning: about succeeding in emerging competitive arenas, and about winning profits, market share, and customers in the very center of the competitive storms many companies now fear."

Goldman et al. (1995, p. 42)

Som man kan se av sitatet over, dreier ikke agile metoder seg om å effektivisere og kutte kostnader, men om å tilfredsstille kundenes krav til produktet og å utvikle best mulig produkter med høy kvalitet.

"In determining our process for developing software it is important to remember that the primary purpose of a development project is to deliver is a software system that generates value to a business. Models and documents are essentially by-products of the development; they do not generate value directly. If we are to optimize the flow of value from software development then we need to think seriously about eliminating process inventory and waste."

Davies (2005)

Men som en motsetning til Goldman et al. (1995) mener Davies (2005) at man skal bruke dette for å oppfatte det motsatte. Dette vil da si at det er forholdsvis delte meninger om akkurat denne delen av motivasjonen for å innføre slike typer metoder.

3.3 Systemutviklingsmetode

I de tidligere dager arbeidet man med systemutvikling slik at man ikke hadde spesielt mange krav, og at det var utviklerne som bestemte hva det var det var behov for når det skulle utvikles et system. Metoden man benyttet her var da en metode som la stor vekt på utviklingsprosessen og ikke videre på analysen, og som ofte er referert til som "Code & Fix modellen". Videre som utviklingen og behovet for analyse økte, datamaskiner ble mer brukt i ulike situasjoner har metodene utviklet seg til å bli mer komplekse (Fitzgerald et al., 2002).

Utviklingsmetoder eller metodologier kan defineres på ulike måter, og med ulik detaljeringsgrad. Cockburn (2002) definerer metode som: ”A cooperative game of invention and communication, which is goal seeking, finite and cooperative.” Dette er en relativt løs definisjon som ikke spesifiserer hva en metode skal gjøre.

Fitzgerald et al. (2002) definerer metode som følgende: ” A coherent and systematic approach, based on a particular philosophy of systems development, which will guide developers on what steps to take, how these steps should be performed and why these steps are important in the development of an information system.”

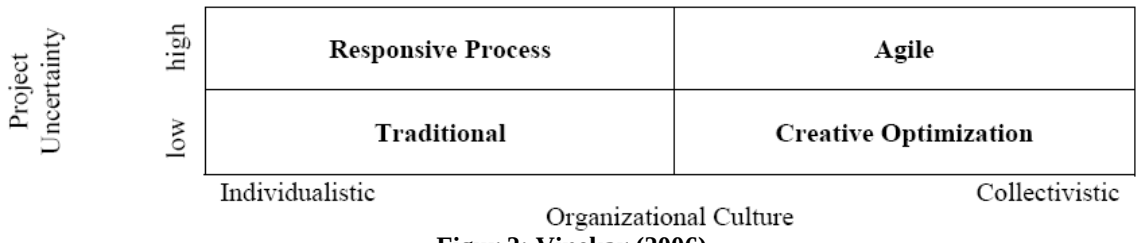
Disse to definisjonene kan på mange måter kalles to ytterpunkter. Men de utfyller hverandre på en måte og viser at det finnes rom for flere tolkninger av metode. Disse definisjonene kan brukes om de fleste metoder som brukes, også innen agile metoder som blir fokusert på i denne oppgaven.

En annen definisjon på utviklingsmetode angir hvordan metoden bør bli brukt:

“Any formalised method (…) is better thought of as a guide to organisation for the achievement of the vision rather than a prescriptive basis for project planning and action.”
Madsen et al. (2006, p. 13)

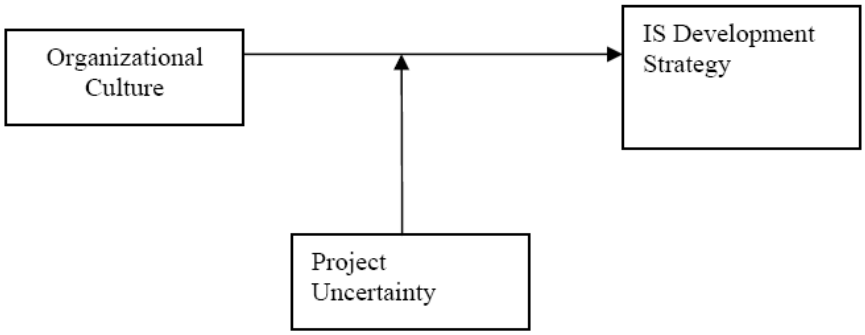
Madsen et al. (2006) fokuserer på at utviklingsmetoden skal være et hjelpemiddel for visjonen til bedriften. Et eksempel her er at hvis visjonen er at man raskt skal levere fungerende system, bør man bruke en metode som støtter opp om dette, for eksempel Scrum eller XP.

Figuren (Figur 2: Vinekar (2006))nedenfor viser forskjellen på agile metoders og tradisjonelle metoders plassering i forhold til organisasjonskultur og usikkerhet i prosjekter. Det som i figuren kalles ”responsive Process” (som på norsk kan oversettes med en motagelig prosess), vil være preget av endring av arbeidsprosesser, raske iterasjoner og høy brukerinvolvering, og den være prosessdrevet. ”Creative Optimization” vil være basert på lite iterasjon, lav brukerinvolvering, prosessene vil være menneskedrevne, og det vil være fokus på taus kunnskap. Dette er i utgangspunktet kun en teoretisk beskrivelse av metode i praksis. Som flere forskere peker på, vil ikke alle organisasjoner bruke en metode til punkt og prikke, men tilpasse den til eget bruk. Denne modellen kan være med å forklare og sette tydeligere skiller mellom ulik bruk av metode, men vil ikke påvirke våre synspunkt videre i oppgaven.



Figur 2: Vinekar (2006)

Det som imidlertid er interessant er modellen (se Figur 3: Vinekar (2006))om hva som påvirker valg av metode. Ut fra modellen kan vi se at organisasjonskultur og usikkerheten på prosjektet definerer hva slags utviklingsstrategi man velger. Sett fra vårt synspunkt kan det være interessant å se på egenskapene til intervjuobjektene våre, og hva slags prosjekter de arbeider med, og hvordan dette påvirker valget av metode, enten bevisst eller ubevisst.



Figur 3: Vinekar (2006)

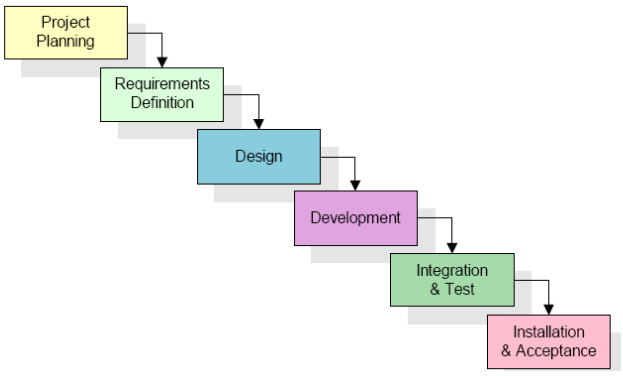
Som vi kan se ut fra dette (angående organisasjonskultur) vil det også være noe som er tilstede som en faktor som kan påvirke i hvor stor grad man involverer kundene i de prosjektene man gjennomfører i en utviklingsorganisasjon.

3.3.1 Eksempel på tradisjonell utviklingsmetode

Fossefallsmetoden (e: waterfall) eller System Development Life Cycle (SDLC), er et godt eksempel på hva en tradisjonell systemutviklingsmetode er. Dette er en metode som har mange rigide regler, og en god del faser man må følge. Modellen har følgende faser:

- Forstudium
- Kravspesifikasjon
- Vurdering av alternativ
- Spesifikasjon av funksjonalitet
- Systemarkitektur
- Programmering og testing
- Innføring av system
- Prosjektvurdering

Det finnes litt variasjoner på hva fasene man går gjennom kalles, men i bunn og grunn ender man opp på det samme. Figuren (Figur 4: SDLC/Fossefallsmodellens faser) under viser hvordan gangen mellom fasene fungerer. Det er viktig å påpeke at man i praksis ikke alltid klarer å følge en slik metode uten å arbeide iterativt.



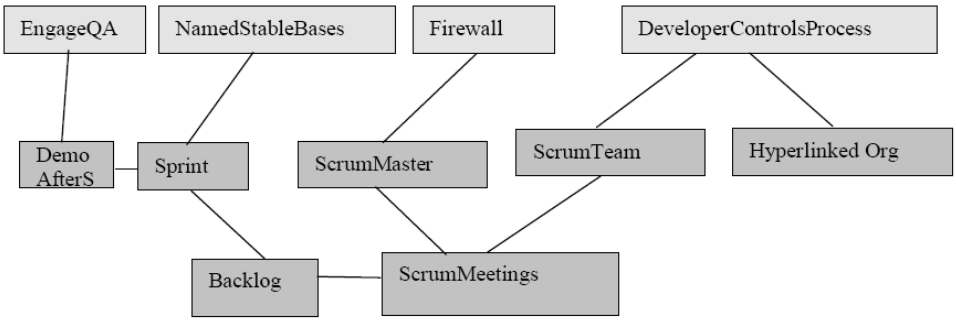
Figur 4: SDLC/Fossefallsmodellens faser

3.3.2 Eksempel på agile utviklingsmetoder

I og med at vi har fokusert på agile utviklingsmetoder i denne studien, vil vi ikke gå nærmere inn på konkrete tradisjonelle metoder enn vi har gjort tidligere. Vi har i stedet valgt å fokusere på å presentere et par eksempler på agile metoder for å belyse teamet vi tar for oss. De metodene vi vil presentere videre nå i de kommende avsnittene er Scrum og Crystal-familien.

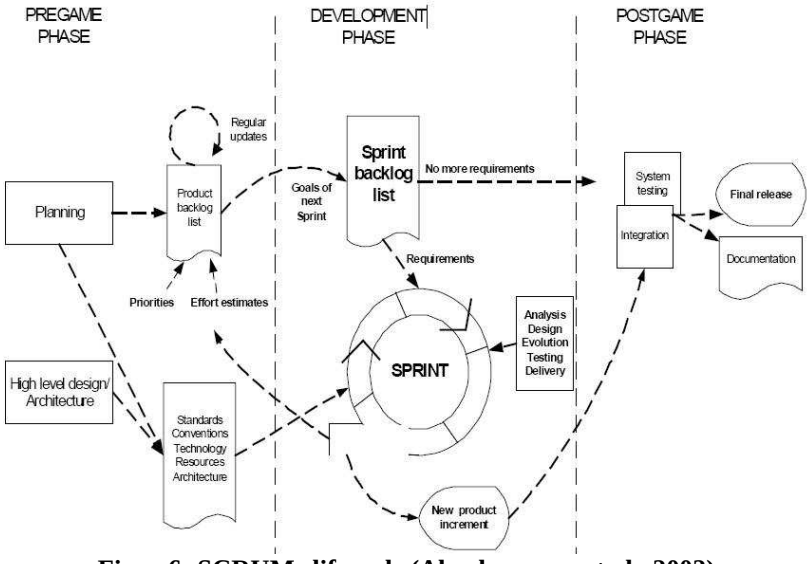
Scrum
Scrum er en agil systemutviklingsmetode som har som hovedfokus å levere fungerende programvare ved raske og korte leveranser (såkalte sprinter), foretrukket lenge på disse vil være rundt en måneds tid (Beedle et al., 2000; Mahnic & Drnovscek, 2005). Eksempel på dette har vi også funnet i Sutherlands eksempel (2004), hvor et prosjekt som strakk seg over en halvårsperiode hadde seks individuelle sprinter, som da altså vil bli varende ca. en måneds tid.

I motsetning til i tradisjonelle metoder er det ikke en forutbestemt måte man deler opp i analyse, design, utvikling osv, men her styrer man dette via fokusering på fremgang i møter, hvor man diskuterer de forskjellige sprintene. Under hver sprint holdes det daglige møter kalt Daily Scrum, og man har på slutten av en sprint et demomøte som hovedsakelig skal engasjere kunden og tilfredsstille utviklerne ved at de har fått gjort noe på prosjektet (Beedle et al., 2000).



Figur 5: Elementer i SCRUM (Beedle et al., 2000)

Dette er en illustrasjon (se Figur 5: Elementer i SCRUM (Beedle et al., 2000)) som viser sammenhengen mellom de ulike elementene som inngår i metoden. Som vi ser her, har vi mange forskjellige elementer, og vi har snakket blant annet om møter, sprinter og demoer. For å bruke disse i praksis har vi også en illustrasjon som viser hvordan livssyklusen i en SCRUM-prosjekt arter seg.



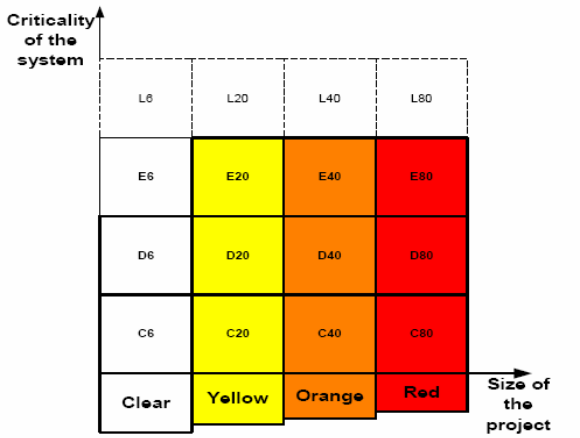
Figur 6: SCRUMs lifecycle (Abrahamsson et al., 2002)

I figuren (se Figur 6: SCRUMs lifecycle (Abrahamsson et al., 2002)) ser vi også at man ikke gjør som i tradisjonell utvikling og deler inn i de fasene man mange er kjent med fra hva som ofte er referert til som SDLC. Man har i motsetning her, som vi har vært inne på, ordnet det slik at man går gjennom alt hver gang man gjennomfører en sprint (fase) helt til man ikke har krav som skal endres lenger.

Crystal-familien

Vi har også valgt å ta med denne familien av metoder i og med den er noe av det mest agile man kan oppdrive blant det store utvalget av agile metoder. Dette er hvertfall slik vi forstår skaperen av metoden Cockburn (2002; 2004), og derfor føler vi det nødvendig å presentere denne typen metoder som et eksempel på agile metoder.

Crystal-familien er delt inn i ulike farger etter hvor stort prosjektet er. De ulike kategoriene spenner fra 6 personer («Clear») til 80 personer («Red»), og kan også strekke seg enda lenger oppover alt etter hvor mørk farge man benytter i betegnelsen. I tillegg deles det opp i hvor kritisk prosjektet er. Jo viktigere prosjektet er for bedriften, jo mer formaliserte og rigide utførelser av oppgaver.



Figur 7: Dimensjoner i Crystal-familien (Cockburn, 2002)

De ulike dimensjonen er rangert etter farge og kode alt etter hvor stort prosjektet er (som vi har vært inne på) og hvor kritisk det er. De ulike bokstavkodene står for følgende: **C** er Comfort, **D** er Discretionary money, **E** er Essential money og **L** er Life (Abrahamsson et al., 2002; Cockburn, 2002).

Det er to overordnede verdier som alle Crystal-metodene er bygd opp etter. Den ene er at de konsentrerer seg veldig om personer og kommunikasjon, den andre er at alle Crystal-metodene er svært tolerante. Det vil si at man kan skreddersy dem slik man ønsker (Cockburn, 2002).

Cockburns (2002) filosofi er at man skal strekke en metode oppover med tanke på størrelse, istedenfor å nedskalere en større metode. Dette er gunstig dersom man har relativt små grupper (4-8 personer) og varigheten til prosjektene sjeldent overstiger 12 måneder, ganske likt med SCRUM (Sutherland, 2004). Hvis de i noen faser trenger mer folk, kan man strekke metoden uten å gjøre de store forandringene.

Hvis man heller ønsket å skalere ned en større metode som for eksempel Crystal Orange, vil man bruke ”for mye” tid til å organisere prosjektgruppen, og man vil risikere å produsere for mange formelle dokumenter. Vi ser samtidig farene med å strekke metoden for langt, hvis et prosjekt for eksempel skulle vokse i antall ansatte og tid. Man risikerer at mye informasjon går tapt, ettersom Crystal Clear baserer seg på at deltakerne sitter i samme rom, eller i nabokontor. En annen ulempe som setter store krav til tilpasning av metoden, er siden metoden er såpass fleksibel, er den ikke optimalisert for noen slags utviklingsprosjekt. I tillegg er dette en relativt ny metode som ikke har blitt utprøvd i veldig mange prosjekt. Det er derfor viktig at man tenker nøye igjennom og bruker en del tid på hvordan man vil tilpasse metoden.

Crystal Clear
Crystal Clear er det «medlemmet» av Crystal-familien som benyttes på prosjekter med få deltakere og som strekker seg over forholdsvis kort tid. Metoden legger også opp til at alle deltakere skal sitte i samme rom, eventuelt nabokontor, for enkelt å kunne utveksle informasjon og kunne samarbeide bedre. Siden personene er samlet i nærheten av hverandre, har ikke Crystal Clear noen kommunikasjonsstruktur eller veldig rigide systemvalideringsprosesser. I denne metodikken er det enkelte ting som inngår, men den er ikke detaljert og kompleks. Det er følgende elementer som inngår i Crystal Clear:

- Roller
- Policystandarder
- Produkter/Leveranser
- Verktøy

Det er også mulig å trekke inn elementer fra XP (eXtreme Programming), Scrum eller ASD (Adaptive Software Development) dersom man føler disse fungerer bedre i sin utnyttelse av metoden, gjelder for eksempel for policystandarder. Dette gjelder for øvrig kun hvis elementene man trekker inn er tilsvarende for de man velger å bytte ut.

Metodikken er i utgangspunktet ikke egnet for kritiske systemer, men kan tøyes til å gjøre dette dersom man ikke har behov for å bruke en tyngre metodikk, for eksempel Crystal Orange. For øvrig er metodikken svært tilpasningsvennlig og oppfordrer til å benytte mye kommunikasjon i prosjektene, som også er felles for alle metodikkene i Crystal-familien (Cockburn, 2002, 2004).

4 Tidligere forskning

Vi vil i dette kapitlet gå igjennom tidligere relevant forskning innen feltet vi skal studere. Oppbyggingen av dette kapittelet er basert på funnene vi har gjort i våre intervjuer med de ulike bedriftene. Hvert element i modellen vår som vi skal beskrive senere, er tatt med i tidligere forskning. Vi vil på denne måten kunne ta for oss og diskutere hvert element vi har funnet opp mot tidligere forskning på dette området.

Oppfattelse av agile metoder kommer an på mange faktorer, slik som utdanning, arbeidsmåter, arbeidsmiljø, med flere. Mange har nok på bakgrunn av dette allerede gjort seg opp seg en formening om bruk av agile metoder. Definisjonene av disse påstandene kan diskuteres, men det er utvilsomt viktige punkter å tenke igjennom hvis man ønsker å benytte seg av agile metoder. Dette er kun noen myter eller punkter å tenke på, og det finnes enda flere viktige punkt å tenke igjennom når man utvikler agilt. Hotle (2006a) lister opp en liste med typiske misoppfattelser som kan være nyttig oppklaring for mange utviklere:

- **Agile metoder er nye – Feil**: Gartner peker på DSDM som den første dokumenterte agile metoden, og denne ble publisert i 1994. Men allerede før dette fantes det agile metoder
- **Agile metoder finnes overalt – Feil og rett**: Gartner estimerer at kun 5 % av utviklede program skjer med agile metoder.
- **Man trenger ikke legge om hele bedriften for å utvikle agilt - Feil**: For å få en optimalisert og fungerende agil metode må alle arbeidsmetodene legges om.
- **Milepæler/ faseinndeling fungerer bra med agile metoder – Feil og rett:** Agile metoder går igjennom milepæler og faser meget hyppig, når hver iterasjon er gjennomført. Men å innføre milepæler og faser på samme måte som en fossefallsmetode fungerer ikke.
- **Man må innføre nye budsjetterings- og finansieringsformer ved innføring av agile metoder – Rett:** Ved innføring av agile metoder går man vekk fra å sette opp rigide krav på forhånd og regne ut hva dette vil koste. Man fokuserer heller på produktutvikling og tett kontakt med kunden. Dette fører til at man kan kansellere prosjektet tidligere eller skalere bemanningen fortløpende.
- **Hvis jeg er agil, slipper jeg å modellere – Mest feil**: For å formalisere utviklingsprosessen og optimalisere en agil metode er det viktig å bruke modelleringsteknikker. Dette foregår etter ”akkurat nok” prinsippet. Man produserer ikke mer modellering enn man må.
- **Agile metoder krever ikke en konsistente prosesser – Feil**: Agile metoder prøver å ikke fokusere på store tunge prosesser. Selv om agile metoder kan se kaotiske ut fra utsiden, finnes det standardiserte og konsistente prosesser.

Agile metoder har blitt diskutert i en rekke artikler. Mange artikler har dreid seg om hva som karakteriserer agile metoder. En av definisjonene kan være den som blir brukt av Beck et al (2001). Dette er et manifest som er utarbeidet av en rekke fremtredende ledere innen området agil systemutvikling. De karakteriserer agile metoder som:

- *Individuals and interactions* over processes and tools
- *Working software* over comprehensive documentation
- *Customer collaboration* over contract negotiation
- *Responding to change* over following a plan

Agile metoder fokuserer på de verdiene som er uthevet med kursiv til venstre. Disse fire utsagnene er hva som blir karakterisert som kjerneverdier til agile metoder, og som man alltid bør strebe mot å oppfylle når man jobber med agile metoder.

En av kildene for å finne informasjon om både slike metoder og om hva man bør ha i fokus i agil systemutvikling, er Agile Alliance og Agile Manifesto. Her startet 17 ledende personer innen metodologier en gruppe hvor de satte sammen et sett av verdier og prinsipper som de selv mente var god programvareutvikling (Beck et al., 2001). De prinsippene som er nevnt her er tolv stykker:

1. Vår høyeste prioritet er å tilfredsstille kunden gjennom tidlige og kontinuerlige leveringer av verdifull programvare.
2. Ønsker endring av krav velkommen, selv i sene faser av utviklingen. Agile prosesser utnytter endringer til kundens konkurransefordel.
3. Leverer fungerende programvare ofte, fra et par uker til et par måneder med et fortrinn av den kortere tidsskalaen.
4. Foretningsmennesker og utviklere arbeider sammen daglig ut gjennom prosjektet.
5. Bygger prosjekter rundt motiverte individer. Gi dem miljøet og støtten de trenger og stol på at de får jobben gjort.
6. Den mest effektive (e: efficient & effective) metoden å fremføre informasjonen til og innad i utviklingsteamet er ved ansikt-til-ansikt samtaler/kommunikasjon.
7. Fungerende programvare er det primære målet for fremgang.
8. Agile prosesser promoterer bærekraftig utvikling. Sponsorene, utviklerne og brukerne burde være i stand til å vedlikeholde en konstant fart på ubestemt tid.
9. Kontinuerlig oppmerksomhet på teknisk dyktighet og godt design forsterker agilitet.
10. Enkelhet, kunsten av maksimering av mengden arbeid ikke gjort er det essensielle.
11. Den beste arkitekturen, kravene og designen dukker opp av selvorganiserte team.
12. Ved normale intervaller, reflekterer teamet over hvordan de skal bli mer effektive, så justerer og finjusterer de sin oppførsel i henhold til dette.

For vårt fokus i denne oppgaven er det svært interessant hva et utvalg svarte på spørsmål om oppfatningen av agile prinsipper/verdier (se tabell Tabell 2: Rangering av prinsipper for agil utvikling (Patel et al., 2006)). Det som blir trukket frem som viktigst av de agile prinsippene for utviklingsorganisasjonene som ble spurt var:

- Kunder og utviklere skal arbeide tett og hyppig gjennom hele prosjektet
- Ansikt til ansikt er den mest effektive formen for å overbringe informasjon
- Tilfredsstille kunder med tidlige og hyppige leveringer av fungerende programvare

Tabell 2: Rangering av prinsipper for agil utvikling (Patel et al., 2006)

		Importance for Org (%)		
	Principle	High	Medium	Low
1	Achieve customer satisfaction through early and continuous delivery of valuable software.	60	24	16
2	Changing requirements should be welcomed, even late in development for the customer's competitive advantage	40	34	26
3	Working software should be delivered frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale	25	42	33
4	Working software should be the primary measure of progress	37	34	29
5	Business people and developers should work together frequently throughout the project.	84	13	3
6	Projects should be built around motivated individuals who should be provided with the appropriate environment and support to get the job done	53	39	8
7	Face-to-face conversation is the most efficient and effective method of conveying information to and within a development team	60	26	14
8	The needs of the development team must be prioritized in line with cost, quality and time	35	18	47
9	The principles of good design should be strived for at all times	44	28	28
10	A development environment and its products should strive for and exhibit simplicity.	23	48	29
11	Architectures and designs should emerge and not be imposed.	26	21	53
12	Frequent reflection should be used for continuous improvement of development process and product.	23	44	33

Tabellen under (Tabell 3: Agile themes (Patel et al., 2006)) viser en oversikt over funnene i undersøkelsen hvor vi ser hvilke agile prinsipper som ble rangert høyest i praktisk bruk. (Patel et al.(2006). Man har kategorisert de ulike prinsippene i overordnede kategorier som er viktige innenfor systemutvikling. Kategoriene er knyttet opp mot de respektive agile prinsippene, men det kan nok hevdes at disse også vil være viktige kategorier også utenfor rendyrkede agile metoder.

Tabell 3: Agile themes (Patel et al., 2006)

Agile Themes	Relating Principles	Rating the take up of principles in Practice	Justification of Rating
Communication and Collaboration	7,5	High	Both the frequency of communication and mode of communication being practiced adhere to Agile principles of daily face-to-face communication.
Team involvement	6,8	High	98% of the Individuals are empowered to make decisions and over 64% take part in tailoring the process but team skills is the least used criteria/characteristic to tailor software development processes.
Refection	12	Medium	89% of the respondents carry out some form of reflection, but only at the end of the project not throughout the project.
Frequent delivery of working software	1,3,4	Medium	The range of frequency in delivering working software to customers is wide (between 2 weeks – 3 months). In addition, 17% of the respondents only deliver on the final release. Project Milestones and Deliverables still remain the primary measure of progress (54%) while only 27% of the respondents measure progress through working software.
Managing Changing Requirements	2	Low	Although 98% of the respondents cater for change, only 38% do so using dynamic prioritization through iterative and incremental development.
Design	9,10,11	Void	These principles were not well understood by respondents.

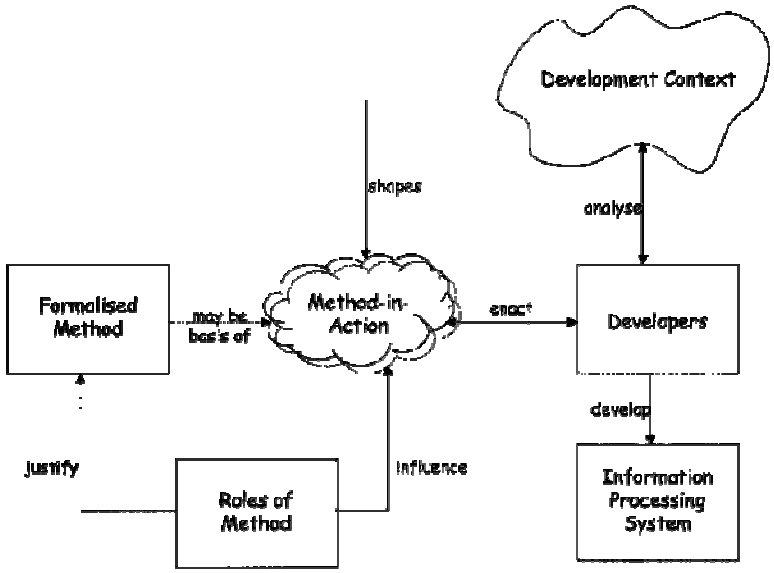
Tabellen over er (Tabell 3: Agile themes (Patel et al., 2006)) hentet fra en undersøkelse som er blitt gjort tidligere, som kan ligne på den vi gjennomfører nå. Den er forskjellig fra vår undersøkelse i form av at man ikke har noe videre fokus på hvorfor disse holdningene faktisk er til stede. Her tar man bare for seg hvilke oppfatninger erfarne prosjektledere og lignende har i forbindelse med agile utviklingsmetoder, i form av en tallfestet rangering. Man har heller ikke fokus på hvordan disse ulike kategoriene påvirker hverandre.

4.1 Metode

Når vi snakker om agil systemuvikling som tema for vår oppgave, er det naturlig at vi kommer inn på metoder når vi er inne på hva som er forsket på tidligere innen dette feltet. Det gjelder ikke kun for agile metoder, men også for metoder generelt når vi nå nevner metoder. I forbindelse med forskning på metoder har vi valgt å ta utgangspunkt i hva tidligere forskning sier om tilpasning av metoder, skalering av metoder, modenhet for bruk av metoder, innføring og bruk av metoder og hva som er bakgrunnen for valg av metoder.

4.1.1 Tilpasning

Figur 8: "A framework for ISD method use" (Fitzgerald et al., 2002) illustrerer hvordan metoden spiller inn i utviklingen av et informasjonssystem. Som vi kan lese ut av figuren, skiller Fitzgerald et al. (2002)mellom formalisert metode (e: formalised method) og metode i bruk (e: method in action). Dette bygger på oppfatningen av at ”ingen” praktiske utviklingsprosjekt følger en formalisert metode til punkt og prikke. Man trekker inn en del teknikker man har erfaring god erfaring med fra tidligere, og kutter vekk de teknikkene man ikke har så god erfaring med, eller som ikke passer inn i prosjektet. Dette kan sammenlignes med Cockburn (2002) sitt utsagn om at utviklingsmetoder er personlige erfaringer som ikke fungerer for alle personer og alle prosjekter.



Figur 8: "A framework for ISD method use" (Fitzgerald et al., 2002)

Man kan ut fra denne figuren trekke konklusjonen at forsøk på å følge en metode slavisk vil være meget vanskelig, for ikke å si umulig. Hvis man ser på figuren, er det mange aspekter som spiller inn, og som utviklerne må ta hensyn til.

Metoden i bruk er ”under press” fra både den formaliserte metoden som utviklerne skal bruke i henhold til formelle planer og beskrivelser.

- Metodens rolle i utviklingen: Dvs. om den skal være et utgangspunkt for videre arbeid eller om man legger stor vekt på at metoden følges slavisk. Rollen til metoden kan bestemme hvilken metode som blir valgt og hvordan den valgte metoden blir brukt.
- Konteksten (e: development context) omfatter teknologi, kultur og forandringsstrategier. Den påvirker både utviklerne og metoden i bruk.
- Utviklere spiller sterkt inn på både utviklingskonteksten og metoden i bruk. Dette har sammenheng med kulturen og kompetansen til utviklerne. Utviklerne blir samtidig påvirket tilbake av metoden i bruk og konteksten.
- Formalisert metode er den basisen på papiret som man bygger en metode i praksis på, enten dette er en sammensetning av flere metoder hvor man bruker det man vil, eller at det er en metode hvor man bruker det som passer inn i prosjektet man holder på med.

Summen av alle disse faktorene påvirker direkte eller indirekte utviklerne og dermed også hvordan systemet som utvikles blir seende ut.

Man kan også se den andre siden av modellen som sitatet nedenfor viser. Dette vil da skje dersom man blir for opptatt av å følge dette rammeverket for å finne den beste metoden og dermed gjør for mye arbeid av dette i stedet for å fokusere på å levere et godt produkt på riktig tidspunkt og opprettholde tilfredsheten til en kunde.

“Obviously, it [the methodology] is not followed rigorously. That would lead to far too much paperwork”

Fitzgerald (1997, p. 5 (206))

“Most organizations employ a software development process but implement it in a flexible/configured manner.”

Patel et al. (2006, p. 3)

Dette (sitatet over) viser at det er svært vanlig å tilpasse metoden man innfører, og at man ikke føler metoden slavisk. Det sier også noe om at det blir støttet opp om figuren som tidligere viste at det var forskjellige faktorer som påvirket hvordan en metode blir brukt i praksis.

Undersøkelsen vi presenterte tidligere av Patel et al (2006) er kun en pekepinn på hvordan metoder eller prosesser blir plassert i organisasjoner. Denne undersøkelsen fikk frem to hovedelementer om programvareutvikling generelt:

1. Flesteparten av organisasjoner anvender systemutviklingsmetoder, men de implementerer dem på en fleksibel eller konfigurert måte.
2. Utviklingsprosesser er støttet av den sterke bruken av iterative sykluser, men endringer er implementert på strukturert måte og passer da nødvendigvis ikke inn.

Noe annet interessant en har funnet her er at flesteparten sier de benytter strukturert metode/prosess, hvor av 56 prosent bruker en form de har produsert selv mens kun 18 prosent bruker kommersielt navngitte hvor RUP, SSADM og DSDM dominerer. Det er også interessant å påpeke at bedriftene velger å tilpasse ”merkenavns”-metodene (82 prosent), og at noen bruker forskjellige metoder til forskjellige prosjekter (79 prosent). Det er også stor variasjon i forbindelse med hvor mye ressurser som brukes på tilpasning av metoden (varierer fra 1 prosent av prosjekttiden og opp til 40 prosent av prosjekttiden).

4.1.2 Skalering

Når man snakker om agile metoder og store prosjekter, vil mange kanskje stusse litt, pga den manglende strukturen som kan finne sted i slike metoder. Men det har vært utført prosjekter med flere hundre deltakere hvor man har tatt i bruk det man kan kalle for det agile rammeverket. Til tross for dette er det også gjort studier som sier at man har en øvre grense på 50 deltakere når man skal arbeide agilt. På en annen side er dette tallet ikke noen form for tvang eller regel, men en retningslinje. Det vil si at man kan benytte seg av og innføre agilt arbeid med flere deltakere enn dette, men det sies at dersom man skal arbeide ”ett hundre prosent” agilt, bør man forholde seg til denne øvre grensen (Williams & Cockburn, 2003). Andre har også dokumentert at det er svært lite som sier noe om prosjekter med mange deltakere der det er brukt en eller annen form for agil systemutvikling. Flesteparten av studiene som er gjort tar for seg prosjekter som har enten under tolv deltakere eller mellom tolv og tjuefem deltakere. Det er også viktig å påpeke at det blir sagt i forbindelse med dette studiet at det er viktig hva slags størrelse prosjektet har (Lindvall et al., 2002).

4.1.3 Innføring og bruk

Hanssen og Fægri (2006) gjennomførte en case studie over to år, i en norsk utviklingsbedrift, som gikk fra å utvikle med fossefallsmodellen til en evolusjonær metode. Denne metoden (evo) fokuserte på tett samarbeid med kunder og hyppige iterasjoner. Fokuset på forskningen var hvordan man forholder seg til kunder i denne overgangen. Artikkelen lister opp ni punkter som ble ansett for viktige å ta hensyn til når man innførte og utviklet ved hjelp av agile metoder.

1. Engasjere kunder - Det er ikke prosjektledere, men kunder som skal være med på utviklingen. Det er viktig å velge riktige kunder, og få klart frem hva som ønskes av tilbakemeldinger.
2. Interessenters praktiske roller – Interessentens to hovedroller i et prosjekt er å lage detaljerte og målbare mål for utviklingen, og å teste løsningen som er produsert.
3. Forsknings og utviklings erfaring – utviklere satte stor pris på interaksjonen med kundene, og ble også hardt rammet hvis kundene trakk seg ut av samarbeidet. Utviklerne fikk følelse av at noen faktisk brydde seg om det arbeidet de gjorde
4. Spesifisere kvalitetsmål – dette er et av hovedpunktene i evo. Kundene setter mål som reflekteres i deres hverdag. For eksempel den tiden det tar for å kjøre en spørring. Dette ble regnet som fordelaktig siden kundene kunne snakke ”brukerspråk” og utviklerne fikk kvantitative målbare mål å jobbe etter. Dette resulterte igjen i at man var friere i utviklingen, man var sikker på når målet var nådd, og kundene gav konstruktiv tilbakemelding
5. Testing – Testing under hele utviklingsprosessen og et formelt rammeverk for håndtering av testing ble poengtert som viktig.
6. Ukentlige iterasjoner – Evo legger opp til ukentlige tester, dette ble ansett som noe lite. Det blir derfor understreket viktigheten av riktig lengde på iterasjonene, dette bør skaleres til størrelsen og kompleksiteten på prosjektet.
7. Interessentinvolvering, overvåking og ledelse – Det er viktig som sagt i punkt 3 at hvis en kunde faller fra, må denne erstattes. Hvis ikke dette punktet overholdes, mister man hele fordelen med iterativ utvikling, og man går tilbake til tilnærmet tradisjonell utvikling
8. Synlighet i utviklingsprosessen – Det er en stor fordel at kundene ser hvordan utviklingsprosessen blir gjennomført, og at de ser fremgangen i prosessen.

9. Omfang og estimering – Etter innføring av evo estimerer nå bedriften fortløpende mellom hver iterasjon. Ved å estimere og korrigere hyppig kan man gi mer korrekt tilbakemelding til kundene og andre interessenter, og man kan sørge for at prosjektet er i rute.

Dette var punkter som ble spesielt lagt merke til i denne casen, men det vil være aktuelt også for andre som benytter agile metoder. Sannsynligvis vil det også være viktige punkter å tenke igjennom også for bedrifter som bruker tradisjonelle metoder.

4.1.4 Valg

Cockburn (2000) peker på fire viktige faktorer for å velge metode:

1. En større gruppe trenger en større metodologi – Størrelse på metode defineres som antall elementer metodologien omfatter (roller, produkter, standarder osv.).
2. Et mer kritisk system, hvor uoppdagede feil vil føre til større skade, trenger større allmenn synlig korrekthet (større tetthet) i konstruksjonen.
3. En liten økning i metodens størrelse og tetthet øker kostnadene og størrelsen på administrasjonen rundt metoden.
4. Den mest effektive formen for kommunikasjon (overføring av ideer) er interaktiv og ansikt til ansikt, for eksempel ved bruk av tavler (e: whiteboard – i denne sammenheng)..

I tillegg til disse punktene peker også Cockburn (2000) på to andre faktorer som spiller inn:

- Prosjektets prioritet – Med dette menes hva sponsorene til prosjektet ønsker - vil de ha et billigst mulig system, må det utvikles raskt, må det være 100 % feilfritt, eller er det viktig å ha åpenhet rundt utviklingsprosessen.
- Metodedesignerens erfaringer og særegenhet – Alle metodedesignere har basert metoden sin på tidligere erfaringer og redsler for hva som kan gå galt. Man må derfor se på hva metoden støtter best opp, og hvordan dette fungerer i ens eget prosjekt.

"A small, rigorous methodology may look the same as an agile methodology, but it won't feel the same."

Cockburn & Highsmith (2001b, p. 131)

Basert på dette utsagnet kan si at det er ikke navnet på metoden eller hvordan den er designet i teorien som har noe å si om hvordan den er å bruke, men hvordan arbeidsrutinene i metoden blir brukt. Man kan her trekke ut at agile utviklingsmetoder har en veldig annerledes måte å arbeide på i forhold til tradisjonelle metoder.

4.2 Leveranser

Leveranser er en viktig del av agil systemutvikling, noe som sitatet under bekrefter. Et av hovedfokusene i agil utvikling er at man skal levere programvare raskt, noe som også blir bekreftet av Beck et al. (2001) i deres beskrivelse av agile utviklingsmetoder.

"The goal of software development is to produce software that meets the needs of your project stakeholders in an effective manner. The primary goal is not to produce extraneous documentation, extraneous management artifacts, or even models. Any activity that does not directly contribute to this goal should be questioned and avoided if it cannot be justified in this light."

Ambler (2006)

I tillegg bekreftes også dette med leveranser som fokus i det agile manifestet, som har som et av sine prinsipper at leveringer av programvare skal skje ofte, gjerne med noen ukers eller toppen en til to måneders mellomrom (Beck et al., 2001). Det samme sier Light (2006) i Gartners prinsipper for agil utvikling der man skal ha fokus på leveranser som går raskt for å først og fremst levere hva kunden ønsker. Man skal derimot ikke ha stort stress med tidsfrister fordi dette kan stjele fokus fra selve leveransen.

Utover dette er det også tidligere praksis som bekrefter dette, hvor flere bedrifter ble spurt om rangeringen av dette manifestet, hvor fungerende programvare kom høyt opp i listen, mens på en annen side kom hyppigheten rimelig lavt (Patel et al., 2006). Disse svarene viser at det er leveranser som er i fokus, men det er ikke alltid hyppigheten har den store plassen.

4.3 Team

Som vi tidligere har vært inne på, har de tolv agile prinsippene et visst fokus på samarbeid innad i utviklingsgruppen, samt samarbeid med kunden. I det studiet som ble gjort ble det sagt at når man arbeider så tett at alle vet hva alle gjør, har dette faktisk en effekt på innsatsen hos utviklerne som deltar i dette teamet. Resultatet av en slik arbeidssituasjon var faktisk at man fikk akselerert farten på det arbeidet de ulike utviklerne gjorde (Sutherland, 2001). Dette var dog et prosjekt hvor man benyttet seg av Scrum som utviklingsmetode, en metode som tvinger en til å møtes daglig i en form for morgenmøte, så om dette er presentabelt for hva som skjer i andre prosjekter eller hva vi ønsker å finne ut i vår studie, gjenstår å se. Men det er uansett en pekepinn på at noe av det agile manifestet fungerer i praksis.

Cockburn (2002) peker på at ”oppskriften” på å bli agil ligger i å identifisere noen optimale prosedyrer (e: sweet spots) for effektiv applikasjonsutvikling. Videre blir det også nevnt 5 viktige punkter for å bli agil, som da er følgende:

1. To til åtte personer i hvert rom. Dette fører til at informasjon kan overføres raskt og enkelt, og uten at det blir for mange forstyrrende element.
2. Brukereksperter til stede på utviklingsplassen. Ved å ha brukere tilgjengelig blir tiden fra et spørsmål dukker opp til en løsning eller definisjon har kommet på plass så liten som mulig. Dette hindrer også at misforståelser kan oppstå.
3. En måneds utviklingsfaser. Etter hver måned skal det være en ny fungerende prototype klar til testing. Dette sikrer at prosjektet er på riktig vei, man unngår feil og misforståelser, og feil kan rettes fortløpende.
4. Fullt automatiserte tester, både funksjonstester og enhetstesting. Dette hjelper både på programmets design og kvalitet, siden det kan testes ofte, og det blir rapportert at utviklerne slapper mer av.
5. Erfarne utviklere. Erfarne utviklere kan være to til ti ganger mer effektive enn uerfarne utviklere. Man kan derfor kutte antall utviklere, noe som gjøre det lettere å innfri punkt nummer en.

Agile prosjektteam fungerer lik som et økosystem. Hvis et medlem slutter, kompenseres det for dette; blir teamet spredd over større avstander, finner man andre måter å kommunisere på (Cockburn & Highsmith, 2001b). Dette vil muligens føre til at man blir mindre agil, men man vil hele tiden prøve å kompensere for å få en optimal utviklingsprosess.

Det har også blitt sagt at prosjektmedlemmer kommer og går, og det er derfor viktig at man har regler for koding og kommunikasjon (Light, 2006). Dette viser da at det er tiltak som kan settes i verk for at kommunikasjonen i teamet går som den skal. I tillegg vil det være enklere for nye medlemmer å sette seg inn i utviklingsprosessen.

4.3.1 Oppbygning av teamet

I store organisasjoner oppstår det gjerne mange forskjellige roller, og de tradisjonelle metodene bruker disse rollene i forbindelse med sine rollebaserte systemer. Der har man team som består av samme typen roller. Ved bruk av agile metoder på har man den andre siden ofte kryssfunksjonelle team, som vil si at man for eksempel ofte rullerer på rollene innad i teamet. Man kan for eksempel også ha ekspertmedlemmer (for eksempel på programmering eller testing) som er spredd utover flere forskjellige team (Chau et al., 2003).

En fordel med rollebaserte team er at dersom man har mennesker som arbeider på uavhengige produkter, vil man kunne jobbe asynkront fra hverandre, dersom det ikke er en høy flyt av kunnskap mellom de forskjellige subteamene (Chau et al., 2003).

Men det er også påpekt at å sette to mennesker sammen er en svært vanskelig oppgave. Mange har sagt at det vil være mindre kostnader med formell opplæring enn dersom man setter sammen en ekspert og en nybegynner for å foreta opplæringen på en slik måte. På en annen side har det også blitt sagt at det kan være det motsatte i forbindelse med kostnader, så dette er noe som avhenger av situasjonen i bedriften eller prosjektets kontekst (Chau et al., 2003). Andre påker også at det er lite form for formell opplæring i et team som benytter seg av en agil systemutviklingsmetode, her bruker man hverandre som mentorer og protesjeer (Lindvall et al., 2002).

I tillegg til dette er det også påpekt noen egenskaper personellet bør ha dersom man velger å benytte seg av en agil utviklingsmetode. Hva som er de viktigste egenskapene er i følge Lindvall et al. (2002) følgende:

- Besitte erfaringer fra den virkelige verden på den teknologiske biten av prosjektet
- At de tidligere har utviklet systemer som er like det de holder på med nå
- Gode menneskelige ferdigheter (e: people skills)
- Gode muntlige og skriftlige evner (e: communication skills).

Dette var listet opp som egenskaper i en undersøkelse der 25-33% påpekte disse, så om det er gjeldende for alle prosjekter skal vi ikke påstå, men det gir en pekepinn på hva som muligens kan forventes av teammedlemmer i agile utviklingsteam.

4.3.2 Tillit innad i teamet

Det har blitt påpekt at det er stort behov for tillit i et utviklingsteam, dette gjelder for eksempel i forbindelse med kildekode og lignende. Teknikker for å øke tillit innad i teamet kan være XPs parprogrammering, kollektivt eierskap til kode, onsite customer og/eller såkalte stand-up meetings. Dette er en ting som agile metoder støtter opp om ved at de for eksempel oppfordrer til gjensidig tillit, respekt og forståelse mellom utviklerne, men også med respekt for kunden (Chau et al., 2003).

I tillegg til det som nå har blitt sagt om tillit er det også viktig at man lar teamet få ansvar. Det har blitt pekt på at ansvar i forhold til kostnader og leveringsestimater er viktig for teamet og ha ansvar for. Utover dette bør også teamet ha den tilliten at de blir gitt ansvaret for en eventuell økning eller redusering av funksjonalitet (Light, 2006).

4.3.3 Kompetanse i teamet

Det å vite hvem som har kompetanse om hva i et team er det som blir kalt for kompetanseadministrasjon. I et agilt arbeidsmiljø løser man dette med daglige møter (slik som man har i Scrum med daily scrum). Her får hvert medlem av teamet presentere hva de har gjort siden sist møte (kan gjøres individuelt eller parvis). Dette fremmer at man får mulighet til å vite hvem som gjør hva, hvem som har gjort hva, og hvem man skal gå til dersom man har et problem eller man trenger å få gjort noe på det prosjektet man holder på med (Chau et al., 2003). Vi ser på dette ikke kun som kompetansebygging, men også teambygging, og at man får en økt form for deltakelse fra teammedlemmer i prosjekter, samt at man hever frekvensen av kommunikasjon innad i teamet.

I tradisjonelle metoder har man på den andre siden ingen direkte form for teknikk for å løse denne problematikken. Her blir kompetanse definert etter hvem som har skrevet dokumenter og ikke på bakgrunn av hvem som kan presentere hva de har gjort, slik man gjør på en daglig basis i agile metoder (Chau et al., 2003).

Man bør også la alle medlemmene (utviklerne) i et team ha mulighet til å bruke og endre koden i alle klassene, men de er også da ansvarlige for problemer som kan oppstå (Light, 2006). Dette viser at det er mulig å spre kompetansen som ligger i teamet, ved at noen kan bruke andres kode til for eksempel å løse et problem de har eller løse andres problemer ved å endre deres kode. Det samme gjelder for rutiner for gjennomgang av kode, hvor det er pekt på at man bør ha rutiner for at utviklerne i teamet skal gå gjennom hverandres kode (Light, 2006).

Cockburn & Highsmith (2001b) peker på to ulike kompetanser i prosjektteam: Individuell kompetanse og teamkompetanse. Uten å ha en grunnleggende individuell kompetanse er det vanskelig for utviklere å komme frem til et godt produkt. Derfor er det i agile metoder lagt vekt på parprogrammering (XP) og mentoring. Ved teamkompetanse menes teamets mulighet til å kommunisere og samarbeide. Individuell kompetanse er viktig, men ved å samarbeide vil utviklingsteamet kunne mangedoble resultatene sine (Cockburn & Highsmith, 2001b). Faktorer som påvirker dette er: hvor agil organisasjonen er og hva slags miljø utviklerne arbeider i (kunder, konkurrenter, krav o.l).

4.3.4 Team & suksess

Harrison & Copllien (1996) har utført en studie på hva som karakteriserer vellykkede utviklingsprosjekt. Slike karakteristikker er selvfølgelig bare en del av aspektene som bør være på plass, og selv om disse er på plass, vil det ikke si at man har et vellykket prosjekt.

- Enkel organisasjonsstruktur – Det vil si færre roller i hvert prosjekt, men ikke nødvendigvis små prosjekt med få ansatte
- Arbeid flyter innover – Utviklere og designere blir tatt med på utviklingen av programmet, de har kontroll over hva som utvikles
- Jevn fordeling av arbeidsmengde og kommunikasjon
- Iterasjon, iterasjon – Høy grad av prototyping og nær kontakt med kunder
- Kompensasjon/ belønning for suksess – I tillegg til penger kan dette være feiringer eller ”happenings” innad i utviklingsteamet.

Denne studien fokuserte ikke bare på agile systemutviklingsmetoder, men på alle typer prosjekt. Det som kan være interessant resultat her, er at flere av disse resultatene er aspekter som agil systemutvikling fokuserer på, for eksempel iterasjon, enkel struktur og jevn fordeling av arbeidsmengde og kommunikasjon.

En karakteristikk for å oppnå vellykket utvikling er fremsatt av Duggan et al.(2006). De fokuserer på hvordan en metode bør brukes og hvilke aspekter man bør fokusere på når man utvikler programvare.

"Boosts in application development productivity and quality come most readily from consistent execution of two approaches: a balance of effective management processes and repeatable low-level development processes."

Duggan et al. (2006, p. 1)

4.4 Krav

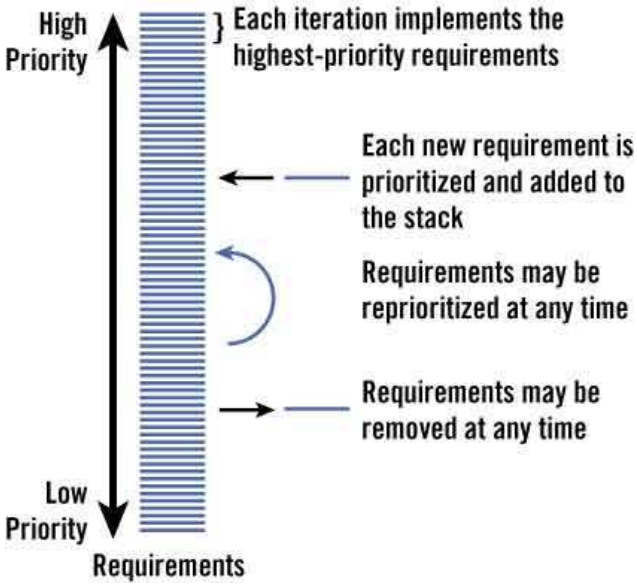
Det er blitt pekt på flere viktige punkter for å være agil i prosessen med å innhente krav. Noe av dette er viktigheten av å automatisere endringer og endringsledelse, mer og mer "sourcing" over flere ledd i utviklingen krever klarere forståelse for hvilke krav som er viktig. En løsning på dette kan være å innføre et verktøy eller programvare for å kunne administrere endring i krav (Light, 2005).

Light (2005) trekker også frem fire viktige punkt som bør tenkes igjennom for å fastslå hvilken strategi for agil endringsledelse man bør velge.

1. Dynamikken i markedet.
2. Størrelse på markedet, og hvordan tilbyderne har posisjonert seg.
3. Hvor er mulige vekstområder i markedet?
4. Hva er best praksis for å levere kvalitet, samtidig som å forbedre leveringssykluser?

I forhold til krav er det også viktig at man etablerer en klar forståelse av hva som er endringer og hva som ikke er det. Man bør ha kontroll på dette og derfor til tider også gå tilbake til tidligere versjoner for å opprettholde denne kontrollen. Dette gjelder også spesielt når man etter hvert begynner å levere mer programvare og kunden kommer med innspill. Da bør man sørge for at man kontinuerlig forbedrer kravspesifikasjonen og rutinene man har (Light, 2006).

Kravspesifikasjonen vil endre seg i løpet av utviklingen, dette er en av forutsetningene til agile metoder. For å håndtere disse endringene peker Ambler (2004) på viktigheten av å prioritere krav. Som vi kan se av illustrasjonen under (Figur 9: Agil håndtering av krav, Ambler (2004)), kan det når som helst komme nye krav, kravene kan endre prioritering, eller krav kan bli kuttet. Selv om kravene endres hele tiden, foreslåes det at man kan fryse kravene før hver iterasjon, slik at en endring som kommer midt i en iterasjon, behandles som et helt nytt krav. Ambler peker også på viktigheten at man har en øverste autoritet, som kan bestemme prioriteringen av kravene. Kunder og utviklere vil ofte ha ulikt syn på hva som bør prioriteres, kundene har kanskje ikke oversikt over den tekniske biten som tar for seg hva som må være tilstede for å bygge på, og utviklerne har kanskje ikke nok innsikt i forretningsprosessene til å se hva som er viktigst for bedriften. Det er her viktig at alle blir behandlet mest mulig likt, og at man prioriterer kravene med objektive øyne.



Figur 9: Agil håndtering av krav, Ambler (2004)

Agil kravspesifikasjon baserer seg på høy kommunikasjon, helst ansikt til ansikt. Utviklere og kunder sitter sammen og diskuterer krav og hvordan systemet skal se ut. Davies (2005) fokuserer på bruken av brukerhistorier som en stor fordel innen agile metoder. Ved å lage bestemte historier som systemet skal kunne utføre, sikrer man at man har høy forståelse mellom kunder og utviklere, i tillegg til at disse historiene er et godt utgangspunkt for å skrive testscenarier. En agil måte å utvikle kravspesifikasjon på, er å skrive detaljerte tester samtidig som man utvikler kravspesifikasjonen. Davies (2005) peker på at ved å gjøre dette sikrer man at man leverer det som kunden og kravene faktisk definerer.

Dette er noe som også støttes opp av andre, hvor det blir påpekt at man skal prioritere nødvendige krav først, og ”kjekt å ha krav” senere i prosessen. Man må også da ha en tett dialog med kunden og hyppig gjøre revideringer av de kravene man har (Light, 2006).

4.4.1 Best practices for å holde seg agil i kravspesifikasjonen

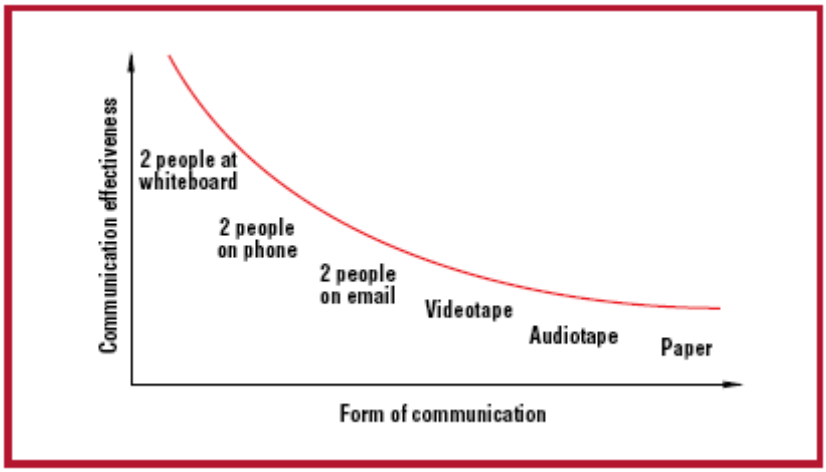
Agile teknikker for å utvikle kravspesifikasjon er ofte ikke så veldig detaljerte. Man bygger på tidligere erfaringer og tilpasser dette til gjeldende prosjekt. Det er derfor ofte vanskelig å si noe spesifikt om hva man bør gjøre. De understående punktene er et forsøk på å finne generelle retningslinjer for utvikling av kravspesifikasjon og hvordan men best holder seg agil i denne prosessen (Ambler, 2007).

- Interessenter deltar aktivt i utvikling av kravspesifikasjon
- Bygg på inklusive modeller – Lag enkle prototyper eller skisser under utviklingen av kravspesifikasjonen
- Start bredt i kravspesifikasjonen og så bli smalere etter hvert
- Utvikl krav akkurat tidsnok (e: just in time) – innse at kravene endrer seg og frys kravene akkurat tidsnok før de skal implementeres
- Målet ditt er å implementere krav, ikke dokumentere dem
- Lag kravspesifikasjonen plattformuavhengig frem til et visst punkt
- Smått er bedre, mindre krav er enklere å estimere og enklere og lede
- Ikke bruk mye tid på å spore krav og endringer gjennom hele utviklingen, hvis ikke dette er høyst nødvendig

- Forklar teknikken rundt utarbeidelse av kravspesifikasjonen, for eksempel modelleringsteknikker
- Bruk kundeterminologi
- Ha det gøy
- Sørg for støtte fra ledelsen
- Gjør interessenter om til utviklere, ikke at de skal gjøre utviklingen, men at de tenker på utviklingen av prosjektet ikke bare forretningsmessige sider

4.5 Kommunikasjon

Kommunikasjon er rangert høyt opp blant teknikker i praktisk utviklingsarbeid. Cockburn (2000) kategoriserer former av kommunikasjon på denne måten. Den mest effektive måten å kommunisere på er ansikt til ansikt, og effektiviteten synker helt ned til man sender hverandre papir. Et tillegg til denne figuren kan være videokonferanser og telefonkonferanser med flere personer.



Figur 10: Kommunikasjon i agile team

Der tradisjonelle metoder mener at man kan bruke dokumentasjon for å kommunisere innad i utviklingsteamet mener den agile fremgangsmåten at man kan ta en annen synsvinkel. Agile systemutviklingsmetoder har det syn at man kan erstatte slik dokumentasjon med uformell dialog innad i utviklingsteamet. En slik dialog vil man også kunne ha mellom teamet og kunden, hvor man har mer fokus på taus kunnskap i motsetning til eksplisitt kunnskap. Årsaker som har blitt pekt på til at dette er tilfellet er at man kan produsere kildekode og endre kravspesifikasjoner frekvent i motsetning til å dokumentere mye er en måte å flytte kostnader (Chau et al., 2003). Dette vil slik vi ser det være en måte å flytte kostnadene til noe som tjener kunden på en bedre måte.

4.6 Evaluering

Hvordan man evaluerer det arbeidet man gjør når man driver med systemutvikling, mener vi har en betydning for det arbeidet vi gjør. Vi har tatt for oss evaluering av team og metode, og hva som tidligere er blitt sagt om hvorfor dette er viktig.

4.6.1 Team

Om det kan kalles kontinuerlig læring eller kontinuerlig evaluering er litt opp til en selv. Men det vi kan si er at agile metoder oppfordrer til dette ved ulike teknikker/tiltak. Metodene oppfordrer til at man har møter etter endte faser/steg/sprinter, refleksjonsarbeidsgrupper, revideringsmøter og andre lignende typer aktiviteter.

Disse foregår ikke kun ved prosjektets slutt, men også underveis i gjennomføringen av prosjektet slik at man kontinuerlig kan forbedre seg selv og lære av sine feil (Chau et al., 2003). Med andre ord så evaluerer man seg selv, teamet og prosjektet og lærer av hva man har gjort feil og gjør et forsøk på å forbedre seg selv til neste steg eller neste steg eller også til neste prosjekt.

4.6.2 Metode

Det er stort fokus på hvilken metode man skal bruke og hvordan den skal brukes. Men det er ikke så stort fokus på hvordan man kan måle om metoden faktisk fungerer. Hotle (2005) har kommet opp med en liste over hvilke punkter man bør fokusere på, ved evaluering av metoden:

- Produktivitet – eksempler som kan brukes her, er funksjonalitet delt på arbeidsinnsats eller antall linjer koding per arbeidstime
- Produktkvalitet – Man kan måle antall defekter per funksjon som er blitt utviklet. Dette bør måles etter at produktet er levert til kunde, og helst mellom 30-90 dager etter levering
- Prosesskvalitet – en måleenhet her kan være omarbeid. Hvis det blir oppdaget feil i kravspesifikasjonen eller i testfasen ellers, kan man måle tiden det tar å rette opp dette igjen. Det blir spesielt pekt på at dette kan være et godt utgangspunkt ved Return Of Investments (ROI) analyser.
- Enhetskostnad – Henger sammen med første punkt. Måleenheter her er kostnad per funksjon eller linje kode
- Kundetilfredshet – kan utføres ved hjelp av spørreskjema. Det er viktig å påpeke at det er ikke nødvendigvis ”harde” måleenheter som er viktigst, men også hvordan kundens subjektive oppfattelse er. Måleenheter kan være:
    - Leverte man det kunden ønsket?
    - Leverte man det man ble enige om?
    - Fulgte man tidsskjema?
    - Leverte man programvare når bedriften trengte det?
    - Var programvaren god nok til vanlig bruk?
    - Generell tilfredshet med utviklingen og produktet.
- Varians i forhold til plan – Hvordan fungerte prosjektet i forhold til det som ble planlagt? Punkter som bør være med: Varians i forhold til budsjett, timeplan og arbeidsinnsats.

Slik rammeverk vil ikke fungere for alle organisasjoner, men det er bedre å gjøre en begrenset evaluering enn å ikke gjøre en evaluering i det hele tatt (Hotle, 2005).

4.7 Kundeforhold

At kunden er i senter er det mye av det vi har sett på tidligere som bekrefter, men det er også noe man skal passe seg for i denne forbindelse. Det har blitt pekt på at man bør ha en formell avtale som er skriftlig dokumentert. Denne bør ta for seg hvor mye og når kunden skal involvere seg i prosjektet, for å svare på spørsmål og delta på møter (Light, 2006).

Når det er sagt, kommer Cockburn & Highsmith (2001a) med et utsagn satt på spissen når det gjelder agil utvikling: Dårlige kunder som ikke vet hvordan og hva slags system de ønsker, resulterer i dårlige system. Dette utsagnet kan tolkes som at man fraskriver utviklere ansvar, og heller legger dette ansvaret over på kundene. Dette utsagnet er noe søkt, men det gir en viktig pekepinn på hva som er viktig i et agilt uviklingsmiljø. Det viser også tydelig viktigheten av å ta hensyn til kunder når man velger utviklingsmetode.

Det er ikke alle kundegrupper eller system som det passer å utvikle agilt. Et velkjent fenomen fra utviklere er syting over at kunden vet ikke hva de vil ha før de faktisk får det. Ved å utvikle agilt å gjennomgå flere iterasjoner, har man i hvert fall større mulighet til å avdekke hva kunden faktisk vil ha, ifølge agile metoder.

Det er også gjort en undersøkelse som sier noe om hva som skal til for å få vellykket systemutvikling, og her scorer kundeinvolvering høyest av alle. I følge Standish Group (1994) var den gjennomsnittlige kostnadsoverskridelsen på 189 %, og den gjennomsnittlige tidsoverskridelsen på 222 %. 365 personer innen applikasjonsutvikling ble spurt om hva som var viktig for at et utviklingsprosjekt skulle være vellykket, i tabellen (Tabell 4: Project success factors) under ser vi resultatet. Brukerinvolvering, støtte fra ledelse og klare krav til applikasjonen ble trukket frem som de tydeligst viktigste faktorene. Det er ikke spesifisert hva slags utviklingsmetoder respondentene brukte, men disse faktorene kommer tydelig til uttrykk innen agil utvikling. Brukerinvolvering er det første punktet i ”the agile manifesto”, hvor det uttrykkes at brukerne skal være med i utviklingen. Støtte fra ledelsen kan man finne igjen i prinsippet om at forretningsmennesker skal jobbe sammen med utviklerne. Klare krav er en av grunnene til at agil utvikling har funnet sted, ved å gjennomgå ofte prototyping og presentere programvare til kundene sikrer man at tilbakemeldinger og riktige krav.

Tabell 4: Project success factors

Project Success Factors	% of Responses
User involvement	15,9&
Executive Management Support	13.9%
Clear Statement of Requirements	13.0%
Proper Planning	9.6%
Realistic Expectations	8.2%
Smaller Project Milestones	7.7%
Competent Staff	7.2%
Ownership	5.3%
Clear Vision & Objectives	2.9%
Hard-Working, Focused Staff	2.4%
Other	13.9%

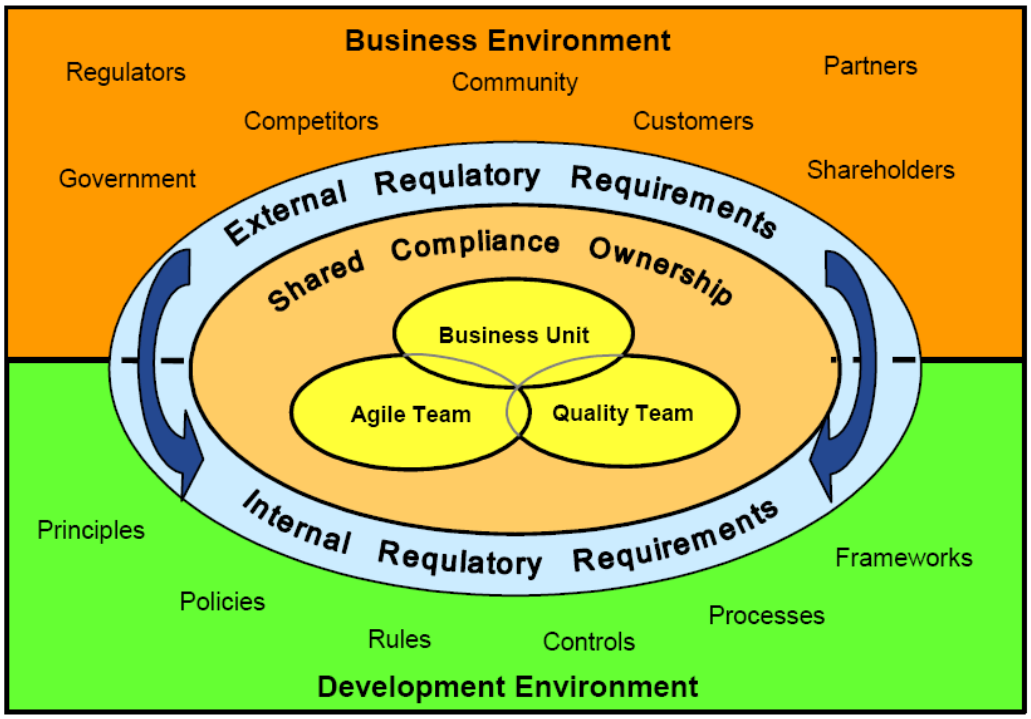
4.7.1 Kundeinvolvering ved test

Det er mange som har hatt sin skepsis til kritiske systemer og lignende i forbindelse med bruk av agile utviklingsmetoder, i relasjon til dokumentering og testing av krav. Men på en annen side er det også metoder som tar for seg testing som en del av seg selv, som til en viss grad er drevet av testing, slik som for eksempel XP. Den gjør det også mulig for kunden å delta aktivt i prosessen slik at han/hun vet at det blir levert det som det er behov for. Dette foregår slik at det er kunden selv som skriver akseptansetesten slik at de kan teste på det som er kritisk nødvendig for han/henne (Lindvall et al., 2002).

Med en slik form for kundeinvolvering får man også gitt en mulighet for at kunden selv kan kontrollsjekke at man får opprettholdt den kvaliteten han/hun ønsker i det produktet som blir levert. Det er også i tråd med ett av de agile prinsippene som utvilsomt har som fokus at man skal levere et produkt som har god kvalitet, fungerer og er det som kunden ønsker.

4.7.2 Kontekst

Figur 8: "A framework for ISD method use" (Fitzgerald et al., 2002) presenterer hvordan en utviklingsmetode forholder seg til omgivelsene. Norton (2006) viser i modellen (se Figur 11: Norton (2006) – The agile team and compliance environment) under hvordan agile utviklingsmetoder og utviklingsprosjekt forholder seg til omgivelser, hvor det er tatt med en del flere eksempel. De tre innerste gruppene representerer de som skal og bør føle eierskap med utviklingsprosessen. Elementene som finnes utenfor sirkelen vil komme med innspill, krav og tilbakemeldinger. Det viktige er at kvalitetsgruppen, agilitetsgruppen (utviklere) og forretningsgruppen (kunder) jobber sammen og ser viktigheten av hverandre, i tillegg til de krav som kommer fra de utenforstående partene.



Figur 11: Norton (2006) – The agile team and compliance environment

Man vil aldri få nøyaktig lik utviklingskontekst, og utviklerne blir som regel og byttet ut over tid. Man kan også se ut fra figuren hvilken rolle metoden skal ha i prosjektet, påvirker hvordan metoden i praksis vil brukes. Ettersom man da har fire faktorer som påvirker hvordan metoden i praksis ser ut, er det innlysende at man må tilpasse metoden etter prosjektet, og ikke blindt kopiere en metode fra tidligere prosjekt og følge dette slavisk. Cockburn (2002) foreslår istedenfor at man tar med seg de ulike teknikkene videre i andre utviklingsprosjekt og syr disse inn i metoden som i praksis skal bruke.

4.8 Dokumentasjon

Vi har tidligere vært inne på leveranser i forbindelse med hva som er gjort av forskning på det område vi nå tar for oss. Det er da også normalt at vi tar med dokumentasjon som en egen seksjon. Dette er fordi dokumentasjon til en viss grad også kan bli sett på som en leveranse, men ikke i den form som vi tidligere har presentert leveranser på.

"If you decide to keep seven models, then whenever a change occurs (a new/updated requirement, a new approach is taken by your team, a new technology is adopted, ...) you will need to consider the impact of that change on all seven models and then act accordingly. If you decide to keep only three models then you clearly have less work to perform to support the same change, making you more agile because you are traveling lighter. Similarly, the more complex/detailed your models are, the more likely it is that any given change will be harder to accomplish (the individual model is "heavier" and is therefore more of a burden to maintain). Every time you decide to keep a model you trade-off agility for the convenience of having that information available to your team in an abstract manner (hence potentially enhancing communication within your team as well as with project stakeholders). Never underestimate the seriousness of this trade-off."

Ambler (2006)

Utsagnet over tar utgangspunkt i det forfatteren kaller agil modellering. Her fremgår det tydelig at det er viktig at man ikke produserer dokumentasjon i større grad enn det man har behov for. Dette er også i samsvar med at man skal ha fokus på leveranser i agil systemutvikling.

Case study fra Defense Information System Application (DISA)
DISA hadde rykte på seg for å levere kvalitetsprogramvare, men fikk krav utenfra om at de måtte kutte utviklingskostnadene. Løsningen ble å omorganisere utviklingen fra tradisjonell utvikling, med et stort antall utviklingsverktøy og mye dokumentasjon, til få utviklingsverktøy og agil systemutvikling. Omleggingen var en stor suksess, man oppnådde over 70 % reinvesteringsandel og stor tilfredshet blant ansatte. Baum (2005) fant fem kritiske suksessfaktorer for innføringen av metoden:

- Velg gode og få utviklingsverktøy, og lær disse godt
- Jobb tett sammen med kunder for å bygge gode applikasjoner
- Lag meningsfylte utviklingsstandarder for å fremheve uniformt utseende og fellesfølelse
- Bygg ombrukbar metode, og lagre dette i tilgjengelige bibliotek
- Invester i ansatte og utdanning.

Disse punktene skal hjelpe bedrifter å øke inntjening og effektivisere utviklingen. De er utviklet på et høyt nivå og skal kunne brukes ved innføring av flere typer agile metoder. Det er ikke universelle pekepinner, men det gir indikasjoner på at man kan oppnå en hel del ved innføring av agile metoder og nedskjæring i mengden av dokumentasjon.

Som man kan se av denne studien støttes det opp om at man ikke skal lage dokumentasjon bare for å lage den. Det er imidlertid viktig å påpeke at studien kun er et casestudium, så man skal ikke ta det for god fisk at det er generaliserbart. Hva vi mener med det er at det er ikke sikkert det vil ha samme effekt for alle å gå fra mye dokumentasjon til lite dokumentasjon slik det gjorde i eksemplet vi viste her (fra DISA)

*”So in an agile development world, when you're developing a brand new product, think very hard about what your app is *really* all about. Could you take it to market with all the important features, or with features that are less functionally rich, in a fraction of the time?*

Apart from reduced cost, reduced risk and higher benefits by being quicker to market, you also get to build on the first release of the product based on real customer feedback.”

Waters, (2007a)

Sitatet over viser at man skal ha fokus på kvalitet når man driver utvikling av programvare på et agilt vis. Dette trenger det nødvendigvis ikke være dokumentert opp og i mente slik mange av de tradisjonelle tilnærmingene ønsker at man skal. Man kan her tolke ut av det som blir sagt i sitatet at man har en solid filosofi i teamet og prosjektet om man skal opprettholde kvaliteten som kunden ønsker, det vil si med andre ord at man da sikrer kundens ønsker og behov. Man må sette da tid opp mot funksjonalitet, og velge ens prioriteringer ut fra dette.

4.9 Test

Det er kostbart å rette feil jo lenger ut man kommer i utviklingsprosessen. Det er også stadig økende press på raske leveringer innen programvare. Ved å planlegge og teste gjennom hele utviklingsprosessen kan man oppdage feil tidligere, øke kvaliteten på det som blir utviklet og redusere kostnader. Det har blitt pekt på 10 elementer som en huskeliste til utviklere; dette gjelder ikke kun agile prosesser, selv om kundesamarbeid, hyppige leveranser/utgivelser og testing blir ansett som meget viktig innen agile metoder (Duggan, 2005).

1. Klare ledelsesprosesser med tidlig involvering av utviklere, operasjonsarbeider og forretningspersoner
2. Formulere krav som kan testes eksplisitt
3. Automatisering som støtter opp om hyppige ”builds”
4. Automatisert testing
5. Peer inspeksjon av kode, design og test cases
6. Opprette et eget produksjonskontroll team som skiller utviklere fra implementeringsarbeidere
7. Gruppering av forandringer som skal testes og innføres
8. God infrastruktur rund endrings og konfigurasjonsledelse
9. Fokus og vedlikehold av infrastruktur forandringer mellom utgivelser (e: release)
10. Konsistent behandling av endringer og konfigurasjon i alle softwareprosjekt i hele bedriften

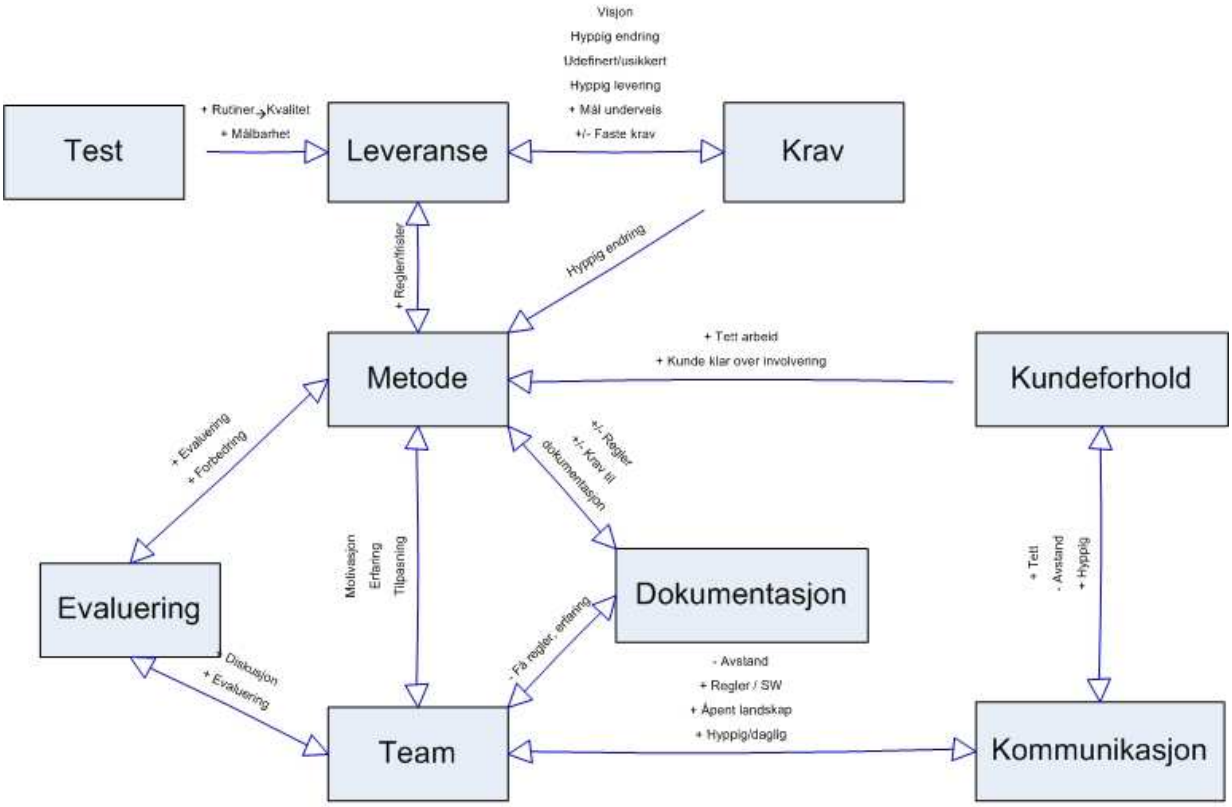
I tillegg til dette er det også blitt sagt at man bør tenke på testingen når man utvikler krav. Det er fordi det er viktig at man får krav som er mulig å teste, og fordi man også må tenke på hvordan man skal få testet de kravene man utvikler (Light, 2006).

5 Resultater

Dette er en beskrivelse av hva vi faktisk har funnet i den studien vi har gjort. I likhet med forrige kapittel er også dette et rapporterende kapittel, men vi vil også være noe innom diskusjon og drøfting her i forbindelse med modellen som vi har kommet frem til. Hovedsakelig tar kapitlet for seg forklaringsmodellen vår og en oppsummering av de funnene vi har gjort.

5.1 Forklaringsmodell

Vi har kommet frem til følgende forklaringsmodell som et resultat av den studien vi har gjort. Modellen viser sammenhenger mellom hva som påvirker i forbindelse med hvorfor man velger å bruke en agil systemutviklingsmetode og hvordan man faktisk bruker den. Den peker også på problematikk og fordeler med slike metoder sett fra brukernes side.



Figur 12: Forklaringsmodell for bruk av agile utviklingsmetoder

En konkret og mer detaljert beskrivelse av hva denne modellen beskriver kommer videre i dette delkapitlet. Her vi vil gå inn på hvordan vi har gått frem for å komme frem til den modellen vi har laget, hva som ligger bak de elementene som er i modellen (dvs. rektanglene/kategoriene), en beskrivelse av de sammenhengene (påvirkende faktorer) som vi har funnet.

5.2 Fremgangsmåte for konstruksjon av forklaringsmodell

Vi har gått gjennom flere steg for å konstruere den modellen vi har kommet fremt til (se Figur 12: Forklaringsmodell for bruk av agile utviklingsmetoder). Modellen er konstruert på bakgrunn av åpen koding og aksial koding (som vi har vært inne på tidligere, se kapittel 2).

Fremgangsmåten vi har benyttet er at vi har brukt den selektive kodingen som et steg i prosessen. Vi har i forbindelse med dette videreutviklet/forenklet det initielle klassediagrammet vi startet med aksial koding. Det klassediagrammet vi da kom frem til viste seg å ha 14 kategorier og ble basisen for produksjonen av et kart eller en modell som skulle vise sammenhengene mellom de ulike kategoriene i modellen.

Modellen eller kartet vi kom frem til i denne prosessen ble en form for idèmyldringskart som ikke inneholdt piler, men som beskrev hva som påvirket hva i modellen. Konstruksjonen av denne modellen (se vedlegg H & I), ble gjennomført først i et modelleringsverktøy, så skrevet ut for revidering. Denne revideringen foregikk i form av tegning med penn og papir for å kunne se sammenhenger og i hvilken retning disse sammenhengene påvirket hverandre. Modellen gjorde også at vi nå hadde kuttet ned antallet kategorier fra siste versjon i klassediagrammet (som var 14 stykker) til 12 stykker.

Nå som kategoriene var redusert ytterligere, kunne vi foreta en ny revidering og se gjennom alle sammenhengene og belyse hvilke sammenhenger som kunne ligge til grunn for nok en sammenslåing av kategorier. Det vi kom frem til var at vi kunne slå sammen noen kategorier til og endte da opp med 9 stykker (slik modellen vår viser, se Figur 12: Forklaringsmodell for bruk av agile utviklingsmetoder). Dermed var vi kommet frem til forklaringsmodellen i første versjon. Selve modellen som vi har presentert har gått gjennom revideringer den også, men dette er kun små justeringer og ikke i den grad vi har tidligere presentert. Det som har vært essensen av disse endringene har vært at noen få relasjoner har blitt kuttet og noen andre har blitt opprettet i stedet, slik at vi har fått en modell som gjenspeiler det vi har sett i intervjuene, slik vi tolker dem.

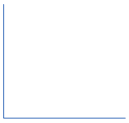
Utviklingen av den endelige modellen har vært en svært iterativ prosess, som da har innebåret ivrig diskusjon både oss i mellom og med veileder for hvordan vi skal komme frem til de forskjellige sammenhengene og få kuttet ned kategorier. Dette ble gjort fordi det ville vært umulig å kunne trekke sammenlikninger og presentere det på et fornuftig vis i form av slik modellen vår så ut på et tidlig stadium, med mange forskjellige kategorier.

5.3 Modellens elementer

I modellen (Figur 12: Forklaringsmodell for bruk av agile utviklingsmetoder) som vi har presentert tidligere, inngår en del elementer. Hva de forskjellige symbolene står for i modellen, blir presentert under.

Dette symbolet beskriver en kategori, som er et resultat av kodingen vi har gjort gjennom analysen vår (prosessen her er forklart tidligere i kapittel 2).

Dette symbolet forklarer i hvilken retning sammenhengen påvirker kategoriene, enten fra den en til den andre siden, og/eller gjensidig.



Symbolet (streken/pilen) er brukt for å knytte sammen kategorier som har effekt på hverandre og/eller gjensidig påvirker hverandre.

Videre i dette kapittelet kommer en forklaring av hva vi legger i de forskjellige elementene (eller kategoriene). De forskjellige kategoriene er følgende:

- Evaluering
- Leveranser
- Test
- Dokumentasjon
- Team
- Kommunikasjon
- Kundeforhold
- Metode
- Krav

Disse 9 kategoriene er som sagt basert på den aksiale kodingen, se vedlegg J.

5.3.1 Evaluering

Intervjuobjektene våre fokuserte både på viktigheten av å evaluere systemutviklingsmetoden man brukte og at man evaluerte teamet. Det ble poengtert viktigheten av å kontinuerlig evaluere hvordan man arbeidet, for så i neste steg kunne justere arbeidsrutiner for å finne den mest hensiktsmessige måten å jobbe på.

"Det (metoden) er ikke noe man kan kjøpe i butikken. Og det er viktig at man har fokus på å bli bedre i jobben sin og forbedre prosessen"

Intervjuobjekt 2

Dette er et godt eksempel på at i tillegg til å evaluere teamet og hver enkelt medarbeider, er det også viktig å evaluere selve prosessen man går igjennom.

Kategorien evaluering henspeiler til evalueringer gjort innad i utviklingsprosessen, mot hvordan teamet arbeider. Dette er i samsvar med de 12 agile prinsippene, som peker klart på at teamet med jevne mellomrom reflekterer over arbeidet de gjør og justerer arbeidsrutinene etter dette.

Evaluering består av følgende attributter:

- Intern gjennomgang
- Fokus på hver iterasjon
- Diskusjoner i plenum innad i utviklingsgruppen
- Evaluering av metode og arbeidsrutiner

I tillegg til den evalueringen internt i teamet er dette også et speilbilde av hvordan utviklingsmetoden som blir brukt i bedriften utvikler seg, samt også arbeidsrutinene, som vi også kan kalle for metoden i praksis.

Denne kategorien gjenspeiler ikke evaluering av produktet som blir levert til kunden (se leveranser og krav).

5.3.2 Leveranser

Leveranser kan være både dokumentasjon rundt programvaren som blir utviklet og fungerende programvare. I denne sammenhengen påpekte de fleste av våre intervjuobjekter at de brukte leveranser som fungerende programvare. Innholdet og hyppigheten på leveransene var viktige tema for våre respondenter.

"(…) vi produserer veldig fort resultater til kunden som han kan forholde seg til"

Intervjuobjekt 2

Her blir resultater brukt om hva man viser til kunden, men det er viktig å vise noe til kunden. En leveranse er som regel en prototype eller del av programmet. Men det er også viktig at man ikke nødvendigvis trenger å levere dette til kunde. Man kan også teste interne leveranser, som blir brukt innad i prosjektteamet

Leveranser er en viktig del av agil utvikling. Leveranser inneholder attributter som:

- Prototyper
- Endelig produkt
- Intervall på leveranser
- Interne og kundereleaser
- Tidspunkt på leveranser
- Funksjonalitet på leveranser

De agile prinsippene peker på at fungerende programvare er måleenheten for hvor langt man er kommet i utviklingen, og for at leveranser skal skje hyppig. Leveranser er som regel det viktigste man ønsker å få ut av en utviklingsprosess. Det finnes flere ulike typer av leveranser, både intern og kundeleveranse. I tillegg er det trukket frem viktigheten av tidspunkt og intervall på leveransene, og hvordan disse leveransene blir brukt. Krav og metode er viktige kategorier som påvirker leveransene.

5.3.3 Test

Med tester mener vi testing av hver leveranse. Man tester for å sikre et best mulig produkt og for å oppdage og renske vekk feil. Rutiner for testing og feilretting blir trukket vekk som viktig deler av utviklingen. Det finnes flere ulike tester, og man kan drive testing på flere ulike stadier i utviklingen. Testingen sitt hovedmål er å minimere feil i leveransene og forsikre utviklerne at de har levert det kunden ønsker.

"Vi har et prosjektoppsett som er allergisk mot feil."

Intervjuobjekt 2

Dette sitatet sier noe om hvor viktig man ser på retting av feil. De fleste innser at det er så å si umulig å levere programvare 100 % feilfri, men man streber etter å luke ut så mange av feilene som mulig før hver leveranse.

Rutiner rundt testing og fokus på målbare tester er viktig ved agil utvikling. Flere intervjuobjekt har fokusert på viktigheten av:

- Rutiner for testing
- Målbarhet
- Brukertester

5.3.4 Dokumentasjon

Veldig mange av våre intervjuobjekt har fokusert på viktigheten av å ikke lage for mye dokumentasjon, og dokumentasjon som ikke blir brukt. Samtidig er de også opptatt av at man må ha en viss dokumentasjon av programmet. Denne sterke fokuseringen på akkurat nok dokumentasjon finner man også igjen i de agile prinsippene. På bakgrunn av dette mener vi at dokumentasjon vil være en nyttig del av modellen, og dette kan påvirke andre deler av modellen.

"(..) dokumentasjon har vært et problem tidligere (…) etter erfaring blir mange store metoder for mye byråkrati"

Intervjuobjekt 3

Dette sitatet oppsummerer mye av skepsisen og fokuset rundt dokumentasjon. De fleste av våre intervjuobjekt var "frykt" for å lage for mye dokumentasjon som ikke ble brukt til noe, videre i utviklingen. Dette var ansett som bortkastede ressurser.

Denne kategorien gjenspeiler nyttigheten av nok dokumentasjon av kode og arkitektur. Det er fokus på å lage nok, men ikke for mye dokumentasjon. Attributter som har blitt listet opp er:

- Regler for dokumentasjon
- Prosesser
- Forbedring
- Unngå å lage dokumenter som ikke blir lest
- Ansvar for dokumentasjon

Dokumentasjon blir tatt frem som viktig med tanke på å formalisere testing og vedlikehold av applikasjoner. De bedriftene som bruker agile metoder er ofte veldig opptatt av ikke å lage for mye dokumentasjon, men det blir også ansett som viktig at man ikke kutter ut dokumentasjon helt. Det er ofte metoden som definerer hvor mye dokumentasjon som skal bli laget, men også teamet har påvirkning på dokumentasjonen. Denne kategorien er viktig siden man prøver å utvikle akkurat nok dokumentasjon.

5.3.5 Team

Team blir definert her som alle som er involvert i prosjektet fra utviklers side. Dette kan være utviklere, designere, prosjektledere, selgere osv. I tillegg definerer denne kategorien hvordan teamet jobber sammen, kommuniserer og administrasjon av teamet:

"(…) det er mer givende for dem (utviklerne) å være med på et SCRUM-prosjekt, for der får teammedlemmer mer ansvar enn i et tradisjonelt prosjekt (…) dette innebærer og at man må ha noen av deltakerne med stor erfaring (…)"

Intervjuobjekt 4

Ved å bruke Scrum som er en agil metode, gir man teamet mer ansvar. Ettersom teamet da blir mer ansvarlig for utviklingen, er det naturlig at teamet er en viktig faktor i vår modell. Ikke bare opp mot selve utviklingen, men også mot kunde og administrasjon.

Attributter som blir nevnt som viktige i sammenheng med team er:

- Definerte møter
- Estimater og budsjettering
- Kompetanse til teammedlemmer
- Erfaring til teammedlemmer
- Aktiviteter
- Størrelse og kontekst til teamet
- Geografisk lokasjon for teammedlemmer og arbeidsmiljø
- Personlige egenskaper til teammedlemmer

Teamet er en viktig del av utviklingsprosessen. Basert på medlemmenes bakgrunn, erfaring og kompetanse blir leveransene formet. Selv om man har en god utviklingsmetode, vil man støte på problemer hvis teamet ikke er godt nok sammensatt eller har samarbeidsproblemer. En viktig kategori som påvirker teamet er kommunikasjon, dette gjelder både internt og ut mot kunder.

5.3.6 Kommunikasjon

Kommunikasjonskategorien skiller seg ut fra de andre kategoriene. Kommunikasjon er ikke noe håndfast, men kommer an på menneskene som arbeider i utviklingen, i motsetning til for eksempel leveranser som er veldig håndfaste. Men kommunikasjon ble ansett såpass viktig av våre respondenter at vi valgte å legge til kommunikasjon som en egen kategori.

"Det er veldig mange kommunikasjonsutfordringer midt oppe i andre prosesser. Kommunikasjon er nøkkelen her"

Intervjuobjekt 1

Kommunikasjon er et viktig stikkord i agil systemutvikling. Et av de agile prinsippene sier klart at forretningsmennesker og utviklere skal sitte sammen. Denne kategorien peker på viktige aspekter og utfordringer for å få kommunikasjonen til å fungere:

- Geografisk lokasjon av kunder og utviklere
- Regler for kommunikasjon
- Programvare og tekniske løsninger for kommunikasjon
- Fysisk utforming av arbeidsmiljø

Kategorien kommunikasjon kan sees på som bindeleddet mellom kunder og teamet eller prosjektdeltakerne. Intervjuobjektene trakk frem kommunikasjon som en hovednøkkel for å få til en god utviklingsprosess. Spesielt i agil systemutvikling hvor man er avhengig av nær kontakt, for raskt å kunne endre retning var det viktig at kundene var tilgjengelig. Et eksempel på dette er morgenmøter, hvor alle, både utviklere og kunder, kommer sammen for å diskutere fremgangen og eventuelle problemer som har oppstått rundt prosjektet.

5.3.7 Kundeforhold

Denne kategorien omfavner involvering av kunder, med det menes at de deltar i prosjektet. Dette blir oppfattet som meget viktig, og kan enten spesifiseres formelt i en avtale eller baseres på tillitt. Kundene blir da nødt til å bidra i prosjektet for å sørge for fremdriften og endringer i prosjektene

"Siden vi er et konsulentfirma, og jobber etter kundens premisser, er det opp til kunden å bestemme hvordan de lar oss jobbe (…)"

Intervjuobjekt 4

Det er viktig å understreke at utviklingsfirma jobber for å tilfredsstille kunden. Man får betalt for å utvikle et produkt som kunden ønsker. I tillegg til dette er det oppfattet som viktig fra bedriftenes side at kunden involverer seg tilstrekkelig i prosjektet.

Hva slags kunde og hvordan kunder involverer seg i prosjektet, har blitt tatt opp som en viktig brikke av intervjuobjektene våre. Det er generell stor variasjon på kundene, men viktige momenter som blir nevnt er:

- Kundene har tillit til utvikler
- Kundene involverer seg i prosjektet
- Synlighet innad i prosjektet og arbeidsrutinene er viktig for kunden
- Kunden kommer med mange endringer
- Avstand til kunden kan være vanskelig
- Formell avtale om involvering er viktig

5.3.8 Metode

Metodekategorien inkluderer formell metode, metode i praksis, metodevalg og arbeidsrutiner. Det kan argumenteres for å skille disse momentene, men i våre intervju fant vi en så klar sammenheng mellom de ulike kategorien at vi har valgt en overordnet kategori som definerer alle disse momentene.

Metode er en av hovedkategoriene i vår modell. Metodekategorien påvirker mange andre kategorier og blir igjen påvirket av mange kategorier. Denne kategorien inneholder først og fremst elementer hentet fra agil systemutvikling, men kan også sees på som en kategori for generelle utviklingsmetoder. Metode inneholder i tillegg som sagt arbeidsrutiner og hvordan disse blir gjort i praksis, så denne kategorien vil ikke kun dreie seg om teoretisk bruk av metode.

"(…) hva slags navn metoden har syns ikke jeg er viktig. Jeg syns det viktigste er at man har en metode, selv om kanskje ikke den metoden er så god, enn at man ikke har det."

Intervjuobjekt 3

Det ble også påpekt at det er viktig at man har et formelt navn på metoden, men dette var ikke like mye nevnt som det at man hadde en formell prosess, som ikke nødvendigvis hadde et navn, eller var noe man hadde laget selv (se sitat under).

"Fordelen med å bruke faglig baserte metoder er jo at de har litt i ryggen og bygger på noe, og at man ikke finner opp kruttet selv."

Intervjuobjekt 4

Om man følger en metode til punkt og prikke eller tilpasser arbeidsrutinene sine, er opp til hver enkelt bedrift. Det viktige å trekke frem er at selv om man ikke har et formelt navn på metoden eller følger denne tett, vil dette (metodens oppsett) påvirke utviklingen.

. Viktige punkter som definerer metode er:

- Formell metode
- Valg av metode kommer an på
  - Utviklere
  - Krav
  - Kunde
  - Erfaringer
  - Dokumentasjonskrav
  - Størrelse på organisasjon
  - Kontekst til prosjektet
- Metoder blir ofte brukt som rammeverk
- Metoden tilpasses organisasjonen og prosjektet
- Roller beskriver bruk av metoden
- Bruk av metoden er en lærende prosess
- Agilitet og iterasjoner er viktig med tanke på valg av metode
- Regler for utviklere

5.3.9 Krav

En av de viktigste grunnene for å velge agile metoder, sett fra våre respondenter synsvinkel, er å være klar for stadig skiftende krav. Det er nettopp for å sette seg i stand til å oppfylle kundenes stadig skiftende ønsker at man har tatt i bruk agile metoder. Det ble også påpekt at definisjonen av kravene har et ord med i saken. Årsaken til dette er at om definisjonen var vag, spilte det inn på valg av metode.

"Q1: Så det er egentlig helt frem til testfasen at det kommer endringer?
A: Ja sånn er egentlig hele bransjen. Alltid nye ting kommer inn."

Intervjuobjekt 2

"Hvis vi hadde brukt fossefallsmetoden hadde det gått rett til skogen (…)) vi kunne aldri oppfylt kravene siden de endrer seg hele tiden"

Intervjuobjekt 2

En av de viktigste motivasjonene for å velge å bruke en agil metode er stadig skiftende krav. Det blir pekt på at kravene endres etter hvert som kunden og utviklere får en bredere forståelse av hva systemet skal utføre. De agile prinsippene sier klart at man skal ønske endringer velkommen for å være i stand til å utvikle et best mulig produkt.

Det ble fokusert på følgende egenskaper rundt krav:

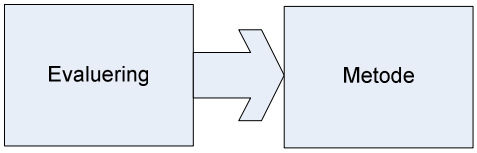
- Hyppige endringer
- Kravene er ofte udefinerte eller vage i begynnelsen
- Det er en fordel at kravene på et tidspunkt blir fryst, for å prioritere og fokusere på dem i neste iterasjon
- Prototyper hjelper til å gjøre kravene klarere
- Utarbeidelse av krav er en kontinuerlig prosess i utviklingen
- Kravspesifikasjonen hjelper utviklere og kunder til å få et felles begrepsapparat
- Kravene må prioriteres
- En god kravspesifikasjon krever god fagkunnskap

Kravkategorien henger tett sammen med leveranser og metode. Kravkategorien fokuserer først og fremst på kravspesifikasjonen, men også krav og ønsker fra kundene som ikke er formelt dokumentert kan komme inn her. Ved å vise prototyper og jobbe tett med kunden vil man få endringer i krav. Dette bør utviklerne være inneforstått med, og jobbe for å tilfredsstille, ikke motarbeide endringer i krav. Selv om et intervjuobjekt mente endringer i krav var et problem, svarte de fleste at dette regnet de med og var en av grunnene til at de valgte agile systemutviklingsmetoder.

5.4 Forklaring av sammenhenger i modell

Når vi nå har beskrevet de forskjellige elementene i modellen vår, kommer vi naturlig inn på hvordan disse er relatert til hverandre. Dette avsnittet beskriver hver av sammenhengene mellom de forskjellige elementene og hva som gjør at de er relatert til hverandre eller har en effekt på hverandre. Dette avsnittet er basert på utsagn fra intervjuobjektene våre og fokuserer på hvordan de ulike kategoriene henger sammen.

5.4.1 Evaluering av metode, en kontinuerlig forbedring av prosessen



Figur 13: Sammenheng mellom metode og evaluering

Evaluering henger sammen med metode ved at man en eller annen gang i utviklingen må evaluere det man holder på med. Denne sammenhengen bygger på evaluering av selve metoden og de arbeidsrutinene man utvikler programvare med, og ikke evaluering av leveransene.

"Vi tilegner oss etter hvert arbeidsrutiner. Som vi følger, som vi kontinuerlig forbedrer"

Intervjuobjekt 2

Evalueringen kan gjennomføres strengt og formelt, for eksempel en gang i halvåret gjennomgår firmaet sin metode og arbeidsrutiner, eller det kan være noe mer uformelt, slik som gjennomgang mellom hver sprint i Scrum.

"Ja det er det sikkert, og den (metoden) er jo veldig fersk som sagt. Vi har brukt et år ca på å komme i gang med den, lage arbeidsplanen vår. Det kommer til å bli revidert kontinuerlig, eller med jevne mellomrom, dette rammeverket."

Intervjuobjekt 2

Her kan utviklerne og ledere komme med tilbakemeldinger på bruk av metoden og ulike elementer i metoden. Det er også viktig hvordan man bruker metoden og hvor modent firmaet er. Som vi ser av sitatet over, pågår det ofte en kontinuerlig evaluering, ved at man utvikler arbeidsrutiner.

*”**Q1: Så dere tar det som fungerte best fra tidligere prosjekt og tilpasser dette?***
A1: Ja det kan man si, men vi har utviklet oss mye innen den tiden vi har eksistert som selskap.”

Intervjuobjekt 1

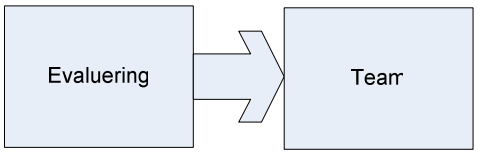
En ytterste konsekvens av evaluering av metoden er at man bytter metode og innfører nye arbeidsrutiner.

”(…) en del metodeansvarlige i forskjellige avdelinger (…) De skal da ta imot innspil om endringer osv. for så å møtes i det vi kaller et metodeforum for å gå gjennom revideringen.”

Intervjuobjekt 5

Stikkordet for evaluering mot metode er kontinuerlig forbedring av metoden og arbeidsrutinene.

5.4.2 Evaluering av team, styrking av struktur og fremgang



Figur 14: Sammenheng mellom evaluering og team

I tillegg til å evaluere metoden og arbeidsrutinene som brukes, evalueres også teamet. Evalueringen av teamet blir gjennomført både internt i teamet og av prosjekteiere, oppdragsgiver og prosjektleder. Evaluering av teamet gjøres typisk etter hver leveranse eller iterasjon. Man ser her på hvor langt man har kommet i prosjektet, hva som er laget av funksjonalitet og lager en oversikt over aktiviteter.

”Ja, det er jo product backlog som egentlig bare er en liste på høyt nivå over funksjonalitet, sprint backlog som er delt opp i aktiviteter og i tillegg som ikke er en del av scrum en status rapport som jeg avga til min oppdragsgiver og til min leder, det er for brukte timer sist uke og totalt slik at vi ikke gikk over budsjettrammen.”

Intervjuobjekt 4

For å kunne holde en viss kontroll med budsjett er det viktig at man kan vise til fremgang i prosjektet og vise til hva man har gjort i løpet av siste iterasjon eller uke. Dette er et viktig verktøy for ledere og prosjekteiere, som hvis det er liten fremgang eller unormalt høyt timeforbruk, må se på sammensetningen av teamet og eventuelt sette inn mer ressurser

”(…) får teammedlemmer mer ansvar enn i et tradisjonelt prosjekt hvor prosjekt leder har ansvar for planlegging, estimering, oppfølging mens i scrum så er det teamet selv som sier at vi skal klare det og det målet i neste sprint (…)”

Intervjuobjekt 4

Ettersom teamet også selv er ansvarlig for planlegging og estimering, må man også evaluere egen innsats. Etter hvert som prosjektet skrider frem ser man hvor mye arbeidsmengde teamet tåler og blir oppmerksom på styrker og svakheter både individuelt og som team.

”(…) teamet blir mye mer sammensmeltet. Ved at vi kjører et morgenmøte hvor alle rapporterer og går igjennom hva de skal gjøre og kan diskuterer hvis man har sittet fast på et problem i en dag. Kan da diskutere i plenum med en gang i stedet for å sitte på det, i forhold til når man har utviklere som er lite åpne som gjerne kan sitte på et problem i en uke uten å gå og spørre noen om hjelp. Man ”tvinger” utviklere til å snakke, for det er ikke alltid at utviklere er så utadvent.”

Intervjuobjekt 3

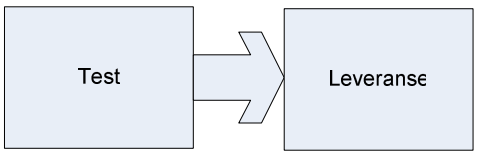
Som vi ser over, er det via en metode at man kan foreta hyppige evalueringer av teamet, noe som gjør at man blir mer samarbeidsdyktige. Det kan i hvertfall tolkes slik ut fra det som blir sagt i sitatet over at man ved kontinuerlige møter hvor man snakker om problematikk, vil forbedre hvordan man arbeider sammen som et team.

”(…) det er viktig at man har fokus på å bli bedre i jobben sin og forbedre prosessen.”

Intervjuobjekt 2

Det å evaluere teamet sitt hele veien, enten på individnivå eller på teamnivå er viktig. Dette bekreftes også av sitatet vi har vist over.

5.4.3 Hyppig testing av leveranser, et redskap for økt kvalitet



Figur 15: Sammenheng mellom testing og leveranser

Testing og retting av feil er et fokus som svært mange bedrifter er opptatt av. Det finnes ulike former for testing, og disse påvirker leveranse på ulike måter. Kundene er som regel involvert i akseptansetest og brukertest, tidligere tester blir utført av utviklerne. Testing gjennomføres gjennom hele iterasjonen, mens brukertest og akseptansetest pågår i slutten av hver iterasjon, og feil rapporteres ofte etter leveransen har blitt overlevert kunden. Testingen utføres som regel først og fremst for å kunne se om man leverer det man skal, i tillegg til å prøve å øke kvaliteten på leveransene. Hva som er kvalitet kan diskuteres, men slik vi har fått inntrykk av defineres det oftest som å levere det man skal med minst mulig feil.

”(…) her har vi gjentatte ganger funksjonstest, for å se om vi leverer det vi skal.”

Intervjuobjekt 2

Som vi ser av dette sitatet brukes testing for å evaluere leveransen og se om det som leveres er det kunden har spesifisert på forhånd. Her er det jo selvfølgelig også viktig at testene og kravspesifikasjonen er designet slik at det er mulig å utføre meningsfulle tester. Dette vil da si at det er viktig at testene som utføres er målbare.

”Vi har prøvd å si klart ifra at dette skal ikke fungere men allikevel kommenterer kundene det når de skal teste systemet. Det er derfor etter vår erfaring best å slippe kunden til senere i utviklingsprosessen når systemet er relativt ferdig.”

Intervjuobjekt 1

Som ”intervjuobjekt 1” påpeker, er det viktig med realistiske krav til testene. Hvis man tester på uferdige system, er det viktig å forklare kundene hva som skal testes og hva som til nå ikke er laget ferdig. Dette sier imot hva andre tidligere har sagt om agil utvikling (Beck et al., 2001) og andre personer vi har snakket med i denne studien.

”(…) Etter dette kommer en intern release som man kan teste på (…)”

Intervjuobjekt 3

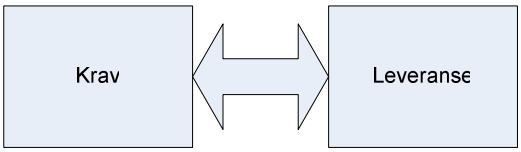
I tillegg bør en også påpeke at det ikke trenger å være ferdige produkt eller noe som leveres til kunden som nødvendigvis må testes på. Det er vel så viktig å luke ut feil tidlig i prosessen. Det er i den interne, kontinuerlige testingen man oppdager og retter flest feil

”(…) etter hver sprint (iterasjon) hadde vi en leveranse som ble testet med en gang og så justerte vi kravspeken etter dette.”

Intervjuobjekt 4

Kravene endrer seg stadig og leveransene må derfor også endres. Det er derfor viktig at testene følger med slik at man kan fange opp eventuelle feil i endring av krav, om endringene har fungert eller for å fange opp eventuelle nye endringer som kunden ønsker.

5.4.4 Leveranser og krav, en gjensidig påvirkende faktor



Figur 16: Leveranse og krav påvirker gjensidig

Kravene og leveransene påvirker hverandre jevnbyrdig. Etter hver leveranse vil kunden få bedre forståelse av systemet og kravene vil endre seg. Hyppighet, størrelse, form målene til leveransen påvirker hvordan kravene vil endre seg.

Leveransene er ofte meget avhengig av kravene. Hvis kravene er usikre og kunden ikke vet helt hva de ønsker av produktet vil det være en stor fordel å ha hyppige leveranser som gjør at prosjektet kan styres underveis, og man kan videreutvikle kravene underveis. Et annet moment er at hvis kravene endrer seg mye pga. skiftende forretningsmiljø, er det fornuftig å ha mindre leveranser og så styre prosjektet etter hvert.

”Men dette prosjektet her hos (…) har det vært veldig lite spesifisering på forhånd sånn at målet har blitt til underveis, og det passer jo egentlig veldig bra til Scrum. Det var egentlig også grunnen til at vi valgte metoden (…)”

Intervjuobjekt 4

Et kjennetegn som har blitt trukket frem er at gjennom agil utvikling med flere leveranser er at målet blir til underveis. Man har lite spesifisering på forhånd, og får kontinuerlig input underveis. Dette fører til større fleksibilitet i det ferdige produktet.

”(…) når du spesifiserer en IT-leveranse kjenner du aldri alle forutsetningene og klarer ikke å formulere alle ønskene dine på papiret. Gjelder også forventninger og ønsker fra kunden, omgivelser, infrastruktur som endrer seg og derfor veldig vanskelig å spesifisere ting på forhånd (…) hadde en ramme for mye arbeid og investeringer de kunne tåle (…). Helt klart et prosjekt som det var hensiktsmessig å ikke spesifisere alt på forhånd for da hadde vi bommet i forhold til hva de forventet seg.”

Intervjuobjekt 4

Det er ikke bare kunde og leveranse alene som spiller inn på kravene også omgivelser og infrastruktur blir nevnt som viktige momenter som er med på å endre krav. Ved å prøve å spesifisere et komplekst udefinert mål på forhånd hadde man risikert å bomme på målet.

”(…) ikke helt visste hva de ville ha og det hadde blitt ganske tungt å spekke det i detalj på forhånd (…) *Du vil aldri klare å spå hva kunde ønsker eller du vil kunne få levert*.”

Intervjuobjekt 4

Som man kan se av dette sitatet, vet kunden ofte ikke helt hva de vil ha. Man sammenligner det å spesifisere et slikt system på forhånd med å spå fremtiden. Det er for det første vanskelig å forstå hva kunden ønsker, men også vanskelig å definere og forstå hva du egentlig har mulighet til å klare å utvikle.

”Sånn har Scrum fungert veldig greit, for da har du noe ferdig noe i løpet av 1,5 måneder, og kan endre fokus.”

Intervjuobjekt 3

Fokus blir trukket frem som viktig, man kan for eksempel fokusere på grensesnitt den ene måneden, for så å fokusere på applikasjonsstruktur neste måned. I tillegg til å fungere bra for å skifte hva som er viktig å levere til kunden innenfor hver måned, hjelper dette også utviklerne til ikke å gå lei eller å kunne fokusere på en ting av gangen.

”At du får levert kjapt, og vist det til kunde. Det er veldig viktig, fordi det er mange prosjekter som feiler pga det tar et halvt år til et år før kunden får se noe, og så var det ikke det som var det de skulle ha og da går det et nytt år for å gjøre endringer og da ender det med å bli kansellert pga endringer i behov”

Intervjuobjekt 3

Å levere raskt til kunden er viktig. Dette sikrer at kunden ser at det er fremgang i prosjektet men også at kunden får mulighet til å forandre mening og komme med innspill underveis.

”Det er klart hvis det er en ny ting, så er det viktig at alle faktisk blir enige om hva som faktisk skal lages. Da må man kartlegge og analysere først, så teste ut litt om vi er på sporet og så lager vi ferdig et system som kan testes.”

Intervjuobjekt 5

Dette er et utsagn som kan minne litt om tradisjonell utvikling. Men det er ikke noen motsetning å bli enige først analysere og utvikle for test. Det kommer derimot an på hvor lang hver slik iterasjon er.

”(…) i store prosjekt, så ombestemmer kunden seg en 50-60 ganger før de faktisk kommer frem til det de vil ha.”

Intervjuobjekt 2

Dette er bare et eksempel på hvor mange endringer som faktisk er vanlige i utviklingsprosjekt. Dette eksemplet er kun store endringer som blir skikkelig dokumentert. I tillegg kommer det flere mindre endringer som ikke blir dokumentert. 50-60 endringsmeldinger blir regnet som gjennomsnittlig, det kunne gjerne være opp mot 200 endringsmeldinger pr. prosjekt.

”(…) kunden ombestemmer seg prosjektet flytter etter, og utviklingsteamet løper etter. Så det er den pendelen, en liten forandring på toppen blir en stor forandring i bunnen.”

Intervjuobjekt 2

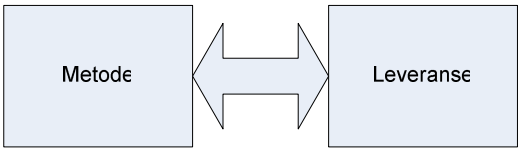
Endringer i kravene kan virke små for kundene og andre utenforstående men blir ofte opplevd som store problemer for utviklerne. Det er viktig å ta i betraktning at selv en liten endring kan føre til store ekstra utgifter. Selv om man utvikler med tanke på at det kommer endringer, er det viktig at man setter av tid til disse endringene og ikke tar for lett på dem.

”Når designet er spikret er veldig mye av utviklingen bundet frem til levering. Og det ser vi som en styrke, at det er bundet i et tidsrom.”

Intervjuobjekt 1

Avhengig av hva slags utviklingsprosjekt man jobber i, er man nødt til å ha en viss binding av krav. Det er uhyre vanskelig og tidkrevende å endre kravene og utviklingen hver dag. Derfor bør man binde kravene mest mulig mellom hver iterasjon og ikke implementere disse midt i en iterasjon. De bør prioriteres og gjøres klar for implementasjon i neste iterasjon, hvis de da ikke er veldig viktige for systemet.

5.4.5 Metoden gir leveransesyklus, og leveranser definerer metodevalg



Figur 17: Metode og leveranse, gjensidig påvirkning

Metoden som blir valgt, har som regel definert hvor ofte man skal levere et produkt. Leveranser trenger ikke nødvendigvis å være ut til kunde det kan også være internt. Fordelen med denne tilnærmingen er at man har mulighet til å fokusere på forskjellige deler av leveransen mellom hver levering. Dette sikrer at man kan holde prosjektmedlemmene motiverte og enklere stykke opp og prioritere leveransene. Leveransene påvirker også metoden, hvis kunden ønsker rask levering av visse ting i prosjektet, eller det er hensiktsmessig å spesifisere og prioritere leveransene underveis.

”Metoden (…) er en litt tyngre prosess, litt mer dokumentasjon og det kan vi tillate oss, siden vi har en lengre syklus for hver release.”

Intervjuobjekt 3

Her har man valgt en metode på bakgrunn av hvor ofte man slipper en ny del av programvaren. Ved at man har slått fast at man skal ha lengre tid mellom hver slipp av programvaren, kan man bruke litt tyngre metoder.

"Den metoden vi har styrt prosjektet etter er Scrum, men den sier mer om hvordan sørge for å styre fremdrift"

Intervjuobjekt 4

Dette sitatet styrker opp under hvordan metoden definerer leveransene. Ved at metoden styrer fremdriften, vil den også si noe om hva som skal produseres og hvor hyppig dette skal produseres.

"Så da tenkte jeg å lansere Scrum for å få litt mer struktur og litt hyppigere lanseringer."

Intervjuobjekt 2

Her blir tydelig metoden dratt frem som en viktig faktor for å styre hvor ofte man skal levere. I motsetning til sitatet over er det metoden her som styrer hvor ofte man skal ha leveranser, ikke leveransene som styrer hvilken metode man bruker. Det er tydeligvis individuelt fra bedrift til bedrift, hva som styrer leveransene.

"(…) men du vil ikke overskride i tid, du vil heller da bare kutte bort funksjonalitet (…) Man kutter enten funksjonalitet eller utvider scopet."

Intervjuobjekt 4

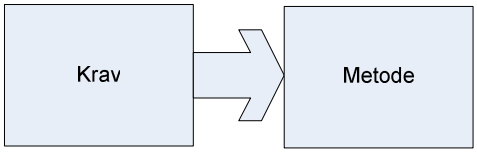
Sitatet viser en viktig faktor i agil systemutvikling. Det blir ansett som meget viktig at man leverer en leveranse på riktig tid. Man utsetter ikke en leveranse et par uker, da leverer man heller mindre funksjonalitet eller eventuelt utvider omfanget, slik at man går igjennom en ny iterasjon for å få med den resterende funksjonaliteten. Her følger man leveransene som metoden spesifiserer til punkt og prikke, og kutter heller i andre deler av utviklingen.

"Det (metodedokumentet, red. adm.) beskriver hvordan vi bruker metoden, samtidig som den er mer detaljert for den sier konkret hvilke leveranser vi skal ha, hva skal vi produsere i hver fase"

Intervjuobjekt 5

I tillegg til å spesifisere når en leveranse skal finne sted, spesifiserer også metoden ofte detaljert hva som skal leveres. Dette kan være dokumenter, modeller, prototyper og lignende.

5.4.6 Stabilitet i kravene, et grunnlag for valg av metode



Figur 18: Krav påvirker valg av metode

Kravene blir ofte trukket frem som hovedgrunnen til valg av agile metoder. Kravene blir stadig endret, på grunn av endring i konkurransesituasjon, tekniske endringer eller ved at man kommer opp med nye ideer. En endring i kravene i en fossefallsmetode kan ha katastrofale konsekvenser, hvis denne kommer sent i utviklingen. Man er da nødt til å gå tilbake og endre hele analysen og i verste fall bygge opp hele arkitekturen på nytt. Hvis man derimot er i den luksussituasjonen at man har en stabil kontekst med stabile krav, og kunder som er veldig klare på hva de ønsker seg, kan det være aktuelt å bruke en fossefallsmetode, men slike situasjoner er ofte ikke så lette å finne i praksis.

"Man starter med en visjon og da ville man i starten ikke kunne være i stand til å skrive spesifikasjon hvis man skulle jobbe på en slik måte (fossefall)"

Intervjuobjekt 3

Det blir trukket frem at ulempen med tradisjonelle metoder er at kravene blir satt tilnærmet en gang. Hvis man da bare har en visjon eller en vag idé om hvordan systemet skal se ut og hva det skal gjøre er det vanskelig å lage en fungerende kravspesifikasjon.

"Hvis vi hadde brukt fossefallsmetoden hadde det gått rett til skogen (…) vi kunne aldri oppfylt kravene siden de endrer seg hele tiden"

Intervjuobjekt 2

"(…) fossefallsmetoden er basert på å sette kravene ca. en gang. Men når kravene endrer seg hver dag opptil flere ganger da går det ikke an å bruke fossefallsmetoden. Da hadde vi tapt for lenge siden."

Intervjuobjekt 2

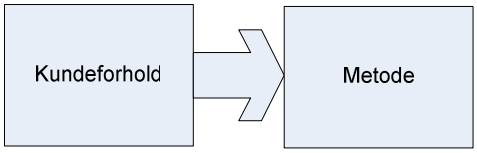
Her er det veldig stor fokus på at kravene stadig endrer seg. Denne bedriften går såpass langt og sier at de ville risikere å gå konkurs hvis de skulle fulgt fossefallsmetoden. Slik kan vi se hvor mye krav faktisk betyr for valg av metode.

"(…)men samtidig er det også (synes jeg) risikabelt å kjøre et fossefallsprosjekt hvor man i stor grad bommer på målet. Du vil aldri klare å spå hva kunde ønsker"

Intervjuobjekt 4

En av grunnene til at man velger å ikke bruke en fossefallsmodell, er at det blir ansett for å være for risikabelt å spesifisere ting i forkant. Dette kan føre til at man ikke får levert det man skal. Agile metoder blir derimot trukket frem som tryggere for å nå kravene og målene man har satt, siden man utvikler det trinnvis og kan forandre på dette underveis.

5.4.7 Involvering av kunde, en viktig faktor for metode



Figur 19: Kundeinvolvering påvirker metoden

Kundeforholdet har en påvirkning på hva slags metode man velger og hvordan denne blir brukt i praksis. Det har blitt påpekt at det er hvor tett kunden har mulighet til å jobbe sammen med utviklere og konsulenter som har noe å si for hva slags type metode man velger.

"Hvis du ikke kan kjøre et prosjekt der kunden deltar kontinuerlig vil man heller kjøre en del spesifisering i starten og heller kjøre noen brukertester underveis."

Intervjuobjekt 4

Her kan man se at kundeinvolveringen vil ha en påvirkning på metoden, i motsetning til om man velger en metode som er tung i dokumentasjon i starten av prosjektet. Det blir også omvendt hvis man har en metode som er typisk agil og ikke har stor fokus på dokumentering i prosjektets start.

"Men dette prosjektet her hos (…) har det vært veldig lite spesifisering på forhånd sånn at målet har blitt til underveis, og det passer jo egentlig veldig bra til Scrum. Det var egentlig også grunnen til at vi valgte metoden, pluss at (…) har vært veldig flinke til å bidra underveis til for eksempel å ta beslutninger."

Intervjuobjekt 4

"Ja her har vi nå 4 fra (…) og 2.3 fra (…), mer eller mindre men ikke på heltid som jobber dette og møtes hver morgen for "daily Scrum" så da jobber vi ganske tett (2 kontorer mellom hverandre med mye interaksjon)."

Intervjuobjekt 4

Dette bekrefter det vi sa over at når man har et prosjekt med en kunde som er kontinuerlig involvert underveis(et eksempel på tett arbeid med kunde), har dette påvirkning på hva slags metode man velger.

"Ja jeg fortalte på introduksjonen hva som skal til for å kjøre et Scrum-prosjekt. Dvs hvilke aktiviteter som er med, møter som er med. Det var bare å presentere dette på kickoff og der og innsalget så understreket vi at det er viktig at prosjekteier eller en fra (…) er med hver morgen på daily scrum. Det var de helt inneforstått med og det var det de trengte, så var det samme for prosjektdeltakere og kunde hvordan man lærte seg prosjektgangen."

Intervjuobjekt 4

Sammenhengen vi har satt opp mellom kundeforhold og metode blir da bekreftet av dette sitatet, med at kundene må være klar over at det kreves involvering av dem. I tilfellet nevnt ovenfor ble dette presentert tidlig i prosjektet slik at det var til stede for at metoden skulle fungere i praksis.

”Det var ikke en formell avtale, så hvis det hadde gått galt, for eksempel hvis de ikke involverte seg så måtte jeg ha tatt det opp med min oppdragsgiver som var en styringsgruppe for prosjektet og sagt at nå får vi ikke involvering fra kunden og da klarer vi ikke å gjennomføre prosjektet.”

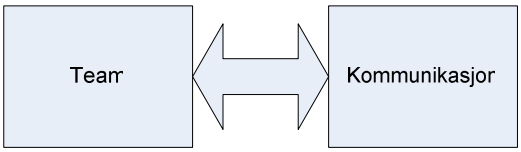
Intervjuobjekt 4

”(…) de sleit litt med kundeinvolveringen når de ikke var på fysisk samme sted. Da må man kanskje gjøre en litt mer formell avtale på forhånd, med at kunden er nødt til å stille opp (…)”

Intervjuobjekt 4

Men det ble også påpekt at dersom det skled litt ut (det tette samarbeidet med kunden), burde man ha en formell avtale med oppdragsgiver (se sitat over), for å kunne tvinge frem daglig samarbeid (til en viss grad).

5.4.8 Team og kommunikasjon, en nødvendighet for god praksis



Figur 20: Team og kommunikasjon påvirker hverandre begge veier

Forholdet mellom kommunikasjon og team er slik at de påvirker hverandre begge veier. I modellen har vi satt opp at regler/sw er en positiv påvirkning, her mener vi regler for kommunikasjon og programvare.

”Man kan komme opp i nesten 200 unike endringsmeldinger. Sånn at det å fokusere på hva vi skal løse og skjønne hovedfunksjonaliteten er viktig tidlig. Alle disse små ønskene noterer vi på små lapper eller i et system (team foundation server fra Microsoft).”

Intervjuobjekt 2

”Vi har en slik ”groupware” løsning som heter Lotus Notes, så der skriver vi hva vi har gjort. Dersom det er noe vi lurer på angående et krav så sender vi det her til den brukeren det gjelder og så får vi svar tilbake når det er avklart. Samme gjør vi ved testing, da sender vi en sånn melding som vi spør om dette er klart eller kan du teste dette, så tester de det og så sender de tilbake om det var greit eller ikke.”

Intervjuobjekt 5

Sitatet viser til regler om hvordan kommunikasjonen skal foregå, i henhold til at det brukes en gruppevareløsning. En bekrefter da at programvare har en påvirkning på hvordan kommunikasjonen foregår i teamet.

Hyppig og daglig kommunikasjon ligger også relatert til at åpent landskap har positiv påvirking på teamet. Teamet påvirker også den daglige kommunikasjonen i form av hvor åpent landskapet er og hvor hyppig de kommuniserer med hverandre.

”Det fungerer veldig greit, men litt frem og tilbake. I noen faser er dette en stor fordel.”

”(…) teamet blir mye mer sammensmeltet. Ved at vi kjører et morgenmøte hvor alle rapporterer og går igjennom hva de skal gjøre og kan diskuterer hvis man har sittet fast på et problem i en dag. Kan da diskutere i plenum med en gang i stedet for å sitte på det, i forhold til når man har utviklere som er lite åpne som gjerne kan sitte på et problem i en uke uten å gå og spørre noen om hjelp. Man ”tvinger” utviklere til å snakke, for det er ikke alltid at utviklere er så utadvent.”

Intervjuobjekt 3

Her ser man at når man tvinger frem kommunikasjon som er hyppig og daglig mellom teammedlemmene, får man en positiv effekt på teamet; det samme beskriver sitatene under.

”(…) når vi er såpass små, kan vi ha daglig kontakt med alle i teamet.”

Intervjuobjekt 3

”(…) det kommer an på hvordan man arbeider og. Slik som her på huset er det fordelaktig at vi sitter ved siden av hverandre. Mens hvis man er utenfor kontoret, må man følge mye mer opp. De har heller ikke mulighet til å sjekke inn og ut kode.”

Intervjuobjekt 3

Den siste sammenhengen vi har sett er en negativ sammenheng, i form av at avstand mellom de som er medlemmer av teamet vil ha en negativ påvirkning på kommunikasjonen i teamet.

”(…) det som er vanskeligst er å få alle utviklerne til å oppdatere arbeidsoppgavene sine hver dag. Det er en del rapporter i scrum som blir veldig ubrukelige dersom ikke utviklerne oppdaterer disse. For eksempel hvis du kommer en mandag og ser det er 50 timer igjen på et prosjekt og ser etter to dager så er det fortsatt 50 timer så sender du en mail og plutselig er det 20 timer igjen. Det er en oppgave for meg som jobber sammen med folk i Oslo, for vi kan ikke ha morramøte sammen, så da blir det mange samtaler på skype.”

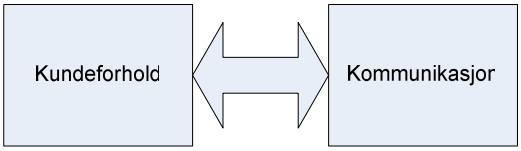
Intervjuobjekt 3

”I mitt team da, så har vi ikke lenger daily scrum pga de andre sitter i Oslo, så jeg må purre litt på de for å oppdatere arbeidsoppgavene sine (…)”

Intervjuobjekt 3

Her ser man et eksempel på at avstand er problematisk i forbindelse med at den daglige kommunikasjonen blir svekket, og man får da mindre oversikt. Det sier også noe om at det åpne landskapet er viktig, i og med at det viser seg at det er avgjørende til en hver tid å vite hvor lenge folk har igjen på oppgavene de holder på med.

5.4.9 Kundeforhold og kommunikasjon, tett arbeid påvirker gjensidig



Figur 21: Kundeforhold og kommunikasjon, gjensidig påvirkning

Kundeforhold og kommunikasjon er også en relasjon som påvirker hverandre gjensidig. Her er essensen i sammenhengen at dersom man jobber tett og hyppig med kunden, bedrer det kommunikasjonen og omvendt (bedre kommunikasjon gir bedre kundeforhold), og at avstand mellom deg og kunden har en negativ effekt.

"Det andre teamet har jo valg MSN for Agile fordi de jobber mye mot India og det er ikke Scrum optimalisert for (avstandsprosjekter), men det er ingenting som stopper deg fra å gjøre det da."

Intervjuobjekt 3

"(…) jeg vet at han ene som er med på prosjektet her har kjørt et scrum-prosjekt hvor kunden satt i Oslo med daily scrum over telefonkonferanse. Og da var det litt vanskelig og det er lettere for kunden å skyve bort møter (…)"

Intervjuobjekt 4

Her vises at det er en negativ sammenheng med avstand og kommunikasjon, men det viser også at det lar seg gjøre. Det er dog viktig å påpeke at det ikke er en frekvent positiv sammenheng, men verd å vise fordi det bekrefter ytterligere at avstand ikke er en optimalisering i forbindelse med sammenhengen mellom kunden og kommunikasjon.

"Det som er viktig for oss er spesielt for en av våre største kunder, er at vi har høy synlighet inn i prosjektet vårt. Sånn at vi veldig fort produserer resultater til kunden som han kan forholde seg til. (…) Vi sitter faktisk ute hos kunden og forsøker å gjøre det som overhodet mulig. Så fort vi har noe, ut til kunden å diskutere skjermbilder. Helst hver uke."

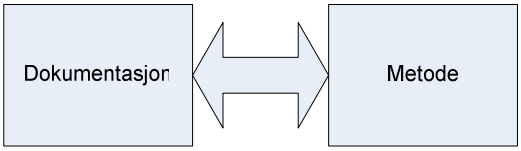
Intervjuobjekt 2

"Ja her har vi nå 4 fra (…) og 2.3 fra (…), mer eller mindre men ikke på heltid som jobber dette og møtes hver morgen for "daily Scrum" så da jobber vi ganske tett (2 kontorer mellom hverandre med mye interaksjon)."

Intervjuobjekt 4

Sitatet viser at det er viktig for kunden at det er tett kommunikasjon med utviklingsteamet, og det omvendte er også tilfelle. Denne kommunikasjonen, som skjer tett og hyppig, bidrar til å bedre kundeforholdet, og dersom man oppnår et slikt kundeforhold, blir man også bedre på tett og hyppig kommunikasjon.

5.4.10 Valgt metode definerer dokumentasjon og omvendt



Figur 22: Gjensidig påvirkning for metode og dokumentasjon

Modellen viser at metode påvirker dokumentasjon og omvendt, i form av regler og krav til dokumentasjon, både i en positiv og negativ sammenheng. De kravene og reglene som stilles til dokumentasjon påvirker valg av metoden, og de reglene og kravene metoden har påvirker hva slags dokumentasjon som blir produsert.

"Ja kanskje som et rammeverk, men det er samtidig strenge regler for hvordan dette daily scrum skal foregå med hvor mange spm man skal stille til hvert medlem og ikke gå utenfor i masse snakk. Det er også regler for dokumentene som product backlog, hvor strukturen er gitt så sånnsett er det jo ganske strengt. Men det er jo fritt i forhold til testing, kvalitet, dokumentasjon, det er helt opp til hvert enkelt prosjekt. Så sånn sett kan du si at scrum-metoden løper litt fra dette ansvaret."

Intervjuobjekt 4

En bekreftelse på at hva slags metode man har, hvilke regler og krav som stilles her har en påvirkning på hva man lager av dokumentasjon. Sitatet under viser at regler og krav som stilles påvirker hva som faktisk lages av dokumentasjon..

"Kunne hatt litt flere forskjellige steg i implementasjonen, kunne hatt litt flere regler. Legger gjerne opp til at man skal bruke XP eller andre former for retningslinjer for dokumentasjon og slikt. Kunne gjerne hatt en genial løsning på en "scrum ad-on"/scrum til det neste steget."

Intervjuobjekt 3

Det er viktig å påpeke at de som har sagt dette også har nevnt at i slike prosjekter som de kjører, er det viktig at man har med en del erfarne deltakere, på grunn av det lille som blir sagt om dokumentasjon og gjennomføring i metoden.

Dokumentasjon er ikke kun nevnt som negativt, det er også nevnt at det vil kunne være en fordel at man ikke har for mye dokumentasjon, noe som er en årsak til at man velger en metode som ikke har for mye krav og regler til dokumentasjon.

"Faren med at man har for mye retningslinjer kan jo være at man produserer noe som faktisk ikke blir brukt, for eksempel et dokument eller en rapport som aldri blir lest er bortkastede timer."

Intervjuobjekt 4

"(…) det har vært svært lite dokumentasjon på akkurat dette prosjektet her pga vi har hatt mer fokus på å koke det ned og få fokus på funksjonalitet (…) Scrum har litt få regler, men fokuserer på å få ting bedre og bedre og bedre."

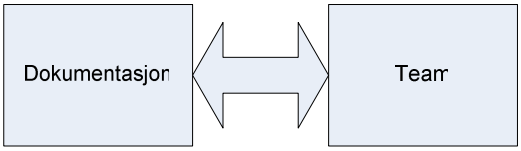
Intervjuobjekt 3

Men det blir også pekt på, som vist i modellen, at det kan til tider være positivt med litt mer dokumentasjon, og et av argumentene for at man skal velge en metode med krav til dette vil være ved bruk av uerfarne utviklere, slik som sitatet under viser.

"Jeg har tidligere jobbet i et annet konsulentselskap som lagde sin egen versjon av RUP og gav det et eget navn, hvor de hadde planer, standarder etc for hvordan du skulle gjøre alt mulig. Dette er noe som i noen tilfeller kan være greit å støtte seg på, spesielt kanskje hvis du har lite erfaring selv eller har et team som har lite erfaring. Det blir da, tror jeg, lettere for uerfarne å komme inn å jobbe på prosjekter."

Intervjuobjekt 4

5.4.11 Dokumentasjon og team, erfaring og standarder påvirker hverandre



Figur 23: Team og dokumentasjon påvirker hverandre

Det som vi har sett påvirker i denne sammenheng er at erfaring og regler som har noe å si. De to elementene påvirker hverandre begge veier, ved at få regler til dokumentasjon kan ha en negativ påvirkning i forhold til teamet dersom man har for få retningslinjer og for lite erfaring hos teammedlemmene. Dersom man da har lite erfarne utviklere i teamet påvirker det hvor mye dokumentasjon man har behov for. Dette har vi vist før i sammenheng med dokumentasjon og metodevalg, men sitatet viser også sammenhengen mellom teamoppbygning og dokumentasjon.

"Jeg har tidligere jobbet i et annet konsulentselskap som lagde sin egen versjon av RUP og gav det et eget navn, hvor de hadde planer, standarder etc for hvordan du skulle gjøre alt mulig. Dette er noe som i noen tilfeller kan være greit å støtte seg på, spesielt kanskje hvis du har lite erfaring selv eller har et team som har lite erfaring. Det blir da, tror jeg, lettere for uerfarne å komme inn å jobbe på prosjekter."

Intervjuobjekt 4

Det ble også svart litt tidligere (dvs. før sitatet over) følgende når vi spurte om det var noen savn med metoden de benyttet seg av i det prosjektet de holdt på med da vi snakket med dem.

"Jeg vil si det må være litt mer retningslinjer for dokumentasjon, testing, kommentering osv."

Intervjuobjekt 4

"Den store fordelen med det er at vi etter hvert er i ferd med å få et felles begrepsapparat for hele (…). Så uansett hvem vi snakker med, så vet de hva vi snakker om, for eksempel hvis jeg sier endringsforslag så vet alle hva jeg snakker om. Mens tidligere så snakket ikke vi på MVA-siden om endringsforslag, vi brukte kanskje begreper som "aa" eller "cr" (change request), mens vi nå er i ferd med å få et felles begrepsapparat. Når vi snakker om testing har vi definert at de hovedtestene vi har, eller testfasene som er systemtest, akseptansetest og produksjonstest, mens innenfor de testene kan du så klart ha stresstest, ytelsestest osv. Men dette er hovedfasene av testing, og tidligere var systemtest for noen noe helt annet enn det var for oss."

Intervjuobjekt 5

Dette viser en effekt på teamet, i relasjon til regler for dokumentasjon. Sitatet viser at med regler som er felles for alle som deltar, øker forståelsen og alle er mer på bølgelengde i forhold til hva slags dokumentasjon det er snakk om. Det er ikke den økte dokumentasjon som har effekten, men begrepene man snakker om som har en effekt på teamet.

Det er også sagt at det er sammenheng mellom regler for dokumentasjon i metoden som gjør det positivt for teamet, dvs. at man utvikler seg etter hvert når man får innført en metode i en bedrift som ikke har en metode. Så til tross for at mange rakker ned på dokumentasjon, er det også påvist noen positive sider med dette i intervjuene vi har gjort.

"Det er sammenheng mellom dokumentasjon og testing. Er det riktig eller er det galt når testen blir kjørt. Sjekk dokumentasjonen, men da finnes det ikke dokumentasjon. Men hvordan funker det da, det husker jeg ikke, da må vi bli flinkere til å dokumentere. Det er en syklus. Den prosessen tror jeg de fleste firma går igjennom før de begynner å bruke en formell prosess."

Intervjuobjekt 3

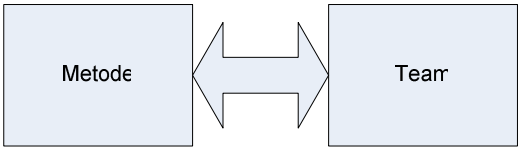
Ved å presentere dette viser også at det krever erfaring hos utviklere for å kunne benytte en metode som dikterer lite, dermed vil det også indirekte ha en påvirkning på teamsammensetningen slik vi tolker det intervjuobjektene har sagt.

"Nei det hadde ikke fungert, man vet ikke nok på forhånd om hva man vil ha. Man starter med en visjon og da ville man i starten ikke kunne være i stand til å skrive spesifikasjon hvis man skulle jobbe på en slik måte. Hvis du skal lage noe du har lagd før eller noen andre har lagd før og vet for eksempel hvilke moduler etc som må med så kan være mulig å bruke en slik en tror jeg, men jeg ville uansett bruke scrum og hatt en lang sprint fordi man da har mye mer klart i starten og trenger ikke bruke tid på for eksempel kartlegging. Men man kan være i utgangspunktet mye flere folk i en stabil metode som fossefall, men du kan bruke scrum også med mange mennesker på et stort prosjekt skulle jeg tro, hvis du deler opp i mange moduler og flere team som jobber individuelt."

Intervjuobjekt 3

Sitatet over ble fremmet i forbindelse med et spørsmål om en fossefallsmetode ville være mulig for det teamet vi da snakket med. Noe som viser at dersom man dokumenterer i forkant og bruker en metode som krever dokumentasjon kan man også gå utenom dette. Igjen tolker vi det som at dersom man skal gå utenom noe slikt vil det være med et utgangspunkt i at man har erfaring i teamet og derfor kan bruke en metode som krever mindre dokumentasjon, noe vi har vært inne på tidligere.

5.4.12 Metode og team, en gjensidig tilpassende faktor



Figur 24: Team og metode påvirker hverandre begge veier

Begge disse to elementene i modellen vår har påvirkning på hverandre i form av motivasjon, erfaring og tilpasning. Dette vil da si at metoden tilpasser seg til teamet og at teamet tilpasser seg metoden.

"Så vi har valgt å holde oss til de utviklingsmetodene som er agile metoder og er tilpasningsdyktige. Så det vi gjør er at vi bruker agile prosesser som betyr at vi kan konfigurere utviklingsprosessen til hvert enkelt prosjekt. Vi tar litt av det vi ser fungerer pluss at vi tilpasser konfigurasjonen til det nye prosjektet, avhengig av hva vi som utviklingsorganisasjon kan sette inn i prosjektet."

"Vi skjeler veldig til RUP, når vi bygger prosessene våre. Uten at vi nødvendigvis følger RUP. Vi tilpasser det til vår organisasjon og det er sånn RUP skal brukes."

Intervjuobjekt 2

"(…) kan man tilpasse metoden til et større team, med rigide regler. Nå har ikke vi gjort det her, eller vi har en del regler sånn som at vi tvinger folk til å ved gjennomgåelse av kode må det være knyttet opp til en bug eller lignende. Dette er viktig for å dokumentere endringer. Så utviklingsverktøyet støtter opp om at metoden blir fulgt korrekt."

Intervjuobjekt 3

Som vi kan se her er det en sammenheng mellom metoden og teamet, hvor metoden blir tilpasset til både prosjekt og organisasjon, som da i bunn og grunn slik vi oppfatter intervjuobjektene er det samme som teamet.

"Vi har en prosessmal som ligger på vår team foundation server som vi bruker i utviklingen. Disse er helt like, dette pga vi tilpasser oss metoden i tillegg til at vi tilpasser metoden til oss. Denne modellen oppdateres kontinuerlig."

Intervjuobjekt 2

Motivasjon er også en faktor som påvirker hva slags metode man velger. Det har blitt pekt på flere årsaker til at man velger metoden, og en av dem var (slik sitatet under viser) at man har en formell måte å gjøre ting på i teamet, at dette er etablert, og ikke at man bruker et formalisert oppsett med et konkret navn. For øvrig var dette også flere av objektene som pekte på dette som en årsak til metoden.

"(…) hva slags navn metoden har syns ikke jeg er viktig. Jeg syns det viktigste er at man har en metode, selv om kanskje ikke den metoden er så god, enn at man ikke har det."

Intervjuobjekt 3

Det ble også påpekt at det er viktig at man har et formelt navn på metoden, men dette var ikke like mye nevnt som det faktum at man hadde en formell prosess, som nødvendigvis ikke hadde et navn, eller var noe man hadde laget selv (se sitat under).
"Fordelen med å bruke faglig baserte metoder er jo at de har litt i ryggen og bygger på noe, og at man ikke finner opp kruttet selv."

Intervjuobjekt 4

En annen grunn til at folk har valgt den metoden de har gjort, går på at kunden ikke vet hva de vil ha, bare at de vil ha det. Dette var en årsakssammenheng som gikk igjen i flere av intervjuene, men vi har kun valgt å ta med ett sitat som illustrerer denne sammenhengen.

"Hovedgrunnen var jo at vi så på det som et prosjekt der (…) ikke helt visste hva de ville ha og det hadde blitt ganske tungt å spekke det i detalj på forhånd."

Intervjuobjekt 4

Den siste sammenhengen vi har sett frekvent gjennom de intervjuene vi har gjort er at det er en sammenheng mellom erfaring og hva slags metode folk velger å benytte seg av. Noe som da var interessant var at begge de bedriftene som bruker Scrum baserte sitt valg helt eller i stor grad på erfaring fra tidligere.

"(…) dette prosjektet som ble det første prosjektet vi benyttet Scrum på. Dette var fordi jeg hadde erfaring med dette fra tidligere arbeidsgiver."

Intervjuobjekt 4

Dette viser da at erfaring generelt og gode erfaringer spesielt med en metode vil være en faktor for om den blir brukt i praksis.

"Jeg vil aldri i mitt liv jobbe med noe annet enn scrum, så hvis folk jeg jobber sammen ikke kan dette så får de lære seg det."

Intervjuobjekt 3

I tillegg har også erfaringen med metoden i dette eksempelet en effekt på teamet ved at man får teamet til å bruke noe man har gode erfaringer med. Det er dog muligens noe bastant å ikke ville lære seg andre metoder, men det er en sammenheng, mellom tidligere erfaring og valg av metode. Den er ikke veldig ofte gjentatt i våre intervjuer, men den er verd å synliggjøre, føler vi.

Hvordan alle disse sammenhengene lar seg relatere til tidligere teori rundt emner vi utforsker er det vi vil ta for oss når vi i neste kapittel starter diskusjonsdelen.

6 Diskusjon

Vi vil i dette kapitlet ta for oss sammenhengen mellom våre funn og modellen vi har laget og tidligere forskning. Videre i dette kapitlet vil vi å presentere sammenhenger og ulikheter mellom vår modell og tidligere forskning og komme opp med eventuelle forklaringer på disse. En slik fremstilling vil være med på å validere modellen vi har kommet frem til ved at vi kan sammenligne resultatene med tidligere funn. I tillegg vil vi fokusere på de resultater vi har kommet frem til, men ikke funnet gode forklaringer på i litteraturen.

6.1 Kommunikasjon, kundeforhold, krav, leveranse og team

Dette kapitlet presenterer de kategoriene vi har funnet å være mest fremtredende og mest fokusert på i våre intervjuer. Årsaken til at vi har tatt med disse kategoriene er at vi ser ut fra intervjuene at de har mer betydning enn hva de andre kategoriene har.

Kommunikasjon

Kommunikasjon blir ansett av mange av våre intervjuobjekter som nøkkelen til en god utviklingsprosess. Dette blir også ansett som viktig ikke bare innad i utviklingsteamet, men også ut mot kunder. Kategorien inneholder regler for kommunikasjon, utfordringer rundt kommunikasjon og hva bedriftene anser som den beste måten å kommunisere på. Elementer som har blitt poengtert her er

- Det er viktig at man sitter tett sammen både innad i teamet og med kunden.
- Man må ha visse regler for kommunikasjon innad i teamet
- Større avstander var utfordringer for kommunikasjon, og kommunikasjonen måtte administreres mer

Kommunikasjon er også fremhevet tydelig i den agile metodelitteraturen. Blant annet Cockburn (2000)og Beck et al. (Beck et al., 2001) peker på hvor viktig kommunikasjon ansikt til ansikt er. Funnene våre rundt kommunikasjon stemmer relativt godt overens med den litteraturen vi har funnet. Funnene våre tyder imidlertid på at kommunikasjon er et område som dekker enda mer enn hva som har blitt beskrevet i litteraturen til nå. Dette er fordi det er lite litteratur som tar for seg en sammenheng mellom agile systemutviklingsmetoder og kommunikasjon i den grad vi har sett. Det er mange utfordringer rundt det å optimalisere kommunikasjon. Blant annet gjelder dette å finne den beste måten for å kommunisere mellom kunde og utviklere på. Selv om det blir påpekt at man bør kommunisere ansikt til ansikt, er ikke dette alltid mulig i praksis. I tillegg har vi funnet utfordringer for å initiere kommunikasjon og hva slags forum som skal brukes. Dette er et område der våre funn tyder på at det vil være fordelaktig å utvikle flere retningslinjer og gode praktiske virkemidler.

På en annen side, i forhold til det vi har sett av teoretisk arbeid, er kommunikasjon noe som i mye større grad blir sett som viktig i de intervjuene vi har gjort. Det finnes, som nevnt over, eksempler på at det som blir sagt i teorien blir bekreftet i våre intervjuer, men det er ikke i så stor grad som objektene påpeker. Dette kommer vi tilbake til når vi beskriver sammenhengene mellom kommunikasjon og de andre kategoriene relatert til denne i modellen.

Kundeforhold

For mange av dem vi har snakket med er kundeforhold en svært viktig faktor. Det er her mye som er avgjørende for at man kan bruke den typen metoder som man faktisk bruker. Årsaken her er i grunn ganske lik med det man ser i forbindelse med kommunikasjon, at man har et tett forhold med kunden. Som vi har vært inne på tidligere, går dette på at man kan ha daglig kontakt med kunden, vise kunden hyppig den fremgang som skjer i prosjektet, og at kunden er aktiv med og tar beslutninger i utviklingsprosessen.

Kundeforholdet er noe som da er med på å binde samen den gruppen av kategorier eller faktorer. Disse anser vi som det mest viktige i det vi har funnet ut i forhold til hva vi har illustrert i forklaringsmodellen. Beck et al. (2001) og Light (2006)peker på viktigheten av involvering og samarbeid med kunder. Våre funn tyder på at man faktisk velger metode på bakgrunn av kunde. Dette er en sammenheng som ikke er blitt fokusert på i tidligere litteratur som vi har funnet.

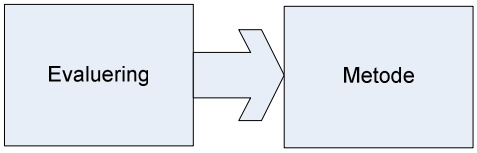
Krav
Krav ble ansett for å være den faktoren som påvirket valget av agile systemutviklingsmetoder mest. Mulighet for å endre krav og skifte fokus ble ansett som helt nødvendig for å kunne levere det kunden ønsket. Prioritering av krav slik som Light (2006) poengterer blir også av våre intervjuobjekter fokusert på, også punkter som å starte bredt, bruke kundeterminologi og viktigheten av modelleringsteknikker (Ambler, 2007) ble trukket frem som viktige punkter i utarbeidelsen av krav.

I tillegg kom flere av våre intervjuobjekt med relativt negative utsagn rundt tradisjonelle utviklingsmetoder. Det ble her påpekt at ved å utvikle tradisjonelt og sette kravene tidlig, ville man ikke være i stand til å oppfylle kundens ønsker.

Leveranse
Leveranser blir i agile systemutviklingsmetoder karakterisert som fungerende programvare, og levert i hyppige intervaller (Beck et al., 2001), noe våre respondenter på mange områder var enige i. Men det ble samtidig trukket fram at det var viktig å ha tilstrekkelig dokumentasjon og design rundt systemet. Respondentene var på mange måter enige med ambler 2006, med at man burde passe seg for å lage for mye dokumentasjon. Hyppig levering er også trukket frem som viktig innenfor agil utvikling, som nevnt tidligere, og dette ble også ble fremhevet som viktig av bedriftene vi intervjuet. Hyppig levering til kunde var ansett som meget viktig, men også interne leveranser ble fokusert på som svært avgjørende. Fordelene med tidlige og hyppige leveranser var at kunden fikk mulighet til å få noe håndfast å teste på, og at man dermed kom opp med mer gjennomtenkte krav. Hvis man hadde hatt vage krav tidlig i utviklingen, ville også tidlige leveranser hjelpe til å klargjøre kravene underveis.

Team
Gjennom vår studie fant vi ut at utviklingsteamet av intervjuobjektene ble ansett som en meget viktig faktor. I tillegg til å være viktig for kommunikasjonen og det ferdige produktet, fant vi at teamet var avgjørende for hva slags metode man valgte. Disse funnene kan til en viss grad støttes opp av Lindvall et al. (2002). Men spesielt for vår undersøkelse fant vi flere indikasjoner på at prosjektlederen var den avgjørende faktoren for hva slags metode som ble brukt. Det ble ansett som viktig at teamet fungerte bra sammen og hadde god kompetanse innenfor systemutvikling. I tillegg var det viktig at teamet kontinuerlig forbedret arbeidsprosessene sine. Ved å gi teamet ansvar for estimering og utarbeidelse av planer anså våre intervjuobjekter at man kunne øke motivasjonen i teamet. Denne tankegangen finner man igjen i de agile prinsippene (Beck et al., 2001) ved at man skal bygge prosjekter rundt motiverte individer.

6.2 Evaluering av metode, en kontinuerlig forbedring av prosessen



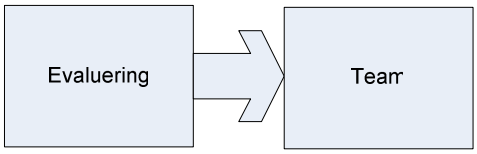
Figur 25: Sammenheng mellom metode og evaluering

Evaluering av metode ble tatt frem som viktig av våre intervjuobjekter med tanke på at man kontinuerlig bør forbedre seg. Det ble fokusert på at både rammeverket som metoden er basert på, men ikke minst arbeidsrutinene ble stadig evaluert. Noen firmaer hadde dette nedfelt formelt, slik at man gjorde dette i bedriften. Andre fokuserte mer på at hver ansatt forbedret sine arbeidsrutiner og kom opp med bedre løsninger for å arbeide. Hvordan dette blir gjort, kommer etter våre funn på an på hvordan bedriftens størrelse og kultur er. Små bedrifter med mindre struktur trenger ikke store formelle evalueringer. I større bedrifter er det vanskeligere å fange opp eventuelle feil, og man må da strukturere dette for å sørge for at det blir tatt opp. Det mest agile og hensiktsmessige tror vi vil være å gjøre dette uformelt eller kontinuerlig i hverdagen. Men som sagt hvis det er en stor organisasjon, er det nødvendig å formalisere dette mer. En slik formalisering vil man kunne gjøre ved at man har veldokumenterte rutiner som er innarbeidet i organisasjonen slik at man kontinuerlig vil foreta slike evalueringer.

Flere av firmaene tilpasser også metoden mye i praksis i forhold til hvordan den er beskrevet i teorien. Dette krever også selvfølgelig evaluering, og man må se hvordan man faktisk bruker metoden opp mot hvordan den er beskrevet. Ved å evaluere på denne måten har man mulighet til å styre utviklingsprosessen og forbedre resultatene.

Evaluering av metode fant vi ut ble først og fremst utført innad i teamet. Man evaluerte om man hadde levert det man skulle og i forhold til kunders ønsker og tidsplan. Bruken av slik evaluering henger også tett sammen med testing av leveranser, noe som vi kommer tilbake til i et senere kapittel. Disse punktene blir også tatt frem som viktige av Hotle (2005), som også trekker frem en del målbare punkter som man kan bruke ved evaluering av metode. Vi har ikke funnet noen utstrakt bruk av dette i våre intervjuer, men ser absolutt nytten av dette. Men man må i tillegg ta i betraktning at en kvantitativ evaluering av metode vil være vanskelig og krevende slik mange av våre intervjuobjekter arbeider. Årsaken til dette er at det er problematisk å finne kvantifiserbare mål. I tillegg har flere av våre bedrifter kun brukt denne ene metoden i sin utvikling. Da vil det bli vanskelig å finne noen brukbare mål å evaluere opp mot. En løsning, slik vi ser det, kan muligens være å sette opp noen mål på forhånd om hvordan man ønsker at metoden skal skaffe resultat.

6.3 Evaluering av team, styrking av struktur og fremgang



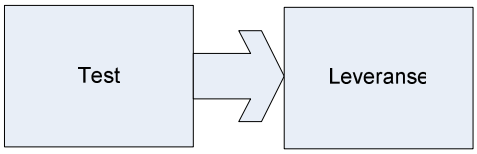
Figur 26: Sammenheng mellom evaluering og team

Evaluering av team henger på mange måter sammen med evaluering av metode. Men ved evaluering av metode er det større fokus på at teamet evaluerer innsatsen sin opp mot involvering i prosjektet. I tillegg tar denne evalueringen for seg hvordan teammedlemmene fungerer sammen og mot kunde. Evaluering av teamet ble trukket frem som svært viktig, spesielt av de bedriftene som brukte Scrum som utviklingsmetode. Grunnen til dette er at teamet her får større ansvar for hver iterasjon, og dermed er det viktig at teamet fungerer på en tilfredsstillende måte. Dette kan sammenlignes med funn Harrison & Copllien (1996) gjorde med at og som fikk dem til å konkludere med at man bør ha enkel organisasjonskultur og at arbeid flyter innover.

To av intervjuobjektene brukte scrum som utviklingsmetode, de fokuserte spesielt på Scrum sine ”daily scrum møter”. Dette gjør at kommunikasjonen øker innad i teamet og også ut mot kunder. I tillegg blir det tatt frem som meget positivt at man får mulighet til å oppdatere hverandre hver dag under disse møtene. Her kan man uformelt komme med forbedringer og forslag til endringer. Slike møter blir også fremhevet av Chau et al. (2003), som et meget godt verktøy for kompetanseheving innad i teamet og mot kunder. Ut over dette ble det presisert at disse møtene var nyttige på grunn av mange utviklere er litt ”asosiale”. Derfor vil disse møtene tvinge utviklere til å snakke og evaluere, noe som de kanskje ikke ville tatt initiativ til hvis man ikke hadde hatt slike møter. Det ble også trukket frem at det er viktig at teamet underveis evaluerer seg selv og arbeidsmetodene. Dette er det samme som det tolvte agile prinsipp som sier at man skal evaluere kontinuerlig.

Funnene våre rundt evaluering og team stemmer godt overens med det tolvte av de agile prinsippene. Ved normale intervaller evaluerer og justerer teamet hvordan de skal bli mer effektive. Et viktig punkt å understreke her er ikke at det gjøres, men hvordan det skal gjøres. Det høres veldig bra ut å evaluere teamet, men hvis man ikke får frem det utviklerne faktisk mener, er denne evalueringen unyttig. Våre funn peker mot at et daglig kort møte, slik som det skisseres i Scrum, blir funnet meget nyttig når det gjelder evaluering av teamet. Dette forumet får folk til å snakke, og man får tatt tak i problemer. Årsaken til dette tror vi har sammenheng med at disse møtene skjer ofte og er forholdsvis uformelle.

6.4 Hyppig testing av leveranser, et redskap for økt kvalitet



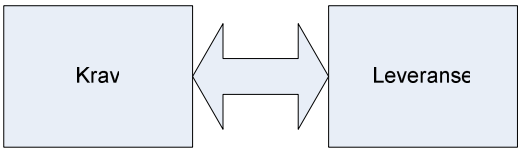
Figur 27: Sammenheng mellom testing og leveranser

Tester er sentrale innenfor agile systemutviklingsmetoder. I bedriftene vi har intervjuet, brukes testing fortrinnsvis til to ting: For å sjekke om man har levert det man skulle og for å oppdage endringer i krav og justeringer i forhold til kunders ønsker. Det er også viktig å poengtere at man ikke nødvendigvis trenger å involvere kunder i testing, spesielt ikke tidlig i utviklingsfasen. Flere bedrifter har hyppige iterasjoner hvor de kommer med interne leveranser som de kan teste på.

Det var også én bedrift som hadde negativt inntrykk av å slippe kunder til for å teste for tidlig i systemutviklingen. Med dette mente de det ble fokusert for mye på elementer som fortsatt ikke var implementert. Grunnen til dette kan være at man ikke hadde noen spesifisert metode for utvikling her, og man manglet kommunikasjon ut mot kunden. De påpekte selv at det var en kommunikasjonsutfordring å formidle til kunden hva som skulle være ferdig og hva som ikke var implementert ennå. Andre intervjuobjekter fokuserte derimot på at man hadde en veldefinert prosess rundt testing, hva som skulle testes, og hvordan testene skulle utføres. Flere bedrifter påpekte at testing var et av deres hovedfokusområder. De strebet hardt for å forbedre testrutiner og tilbakemeldinger på feil, for å kunne levere bedre kvalitet til kunder. Dette finner man igjen hos Cockburn (2002), som peker på at et av punktene for å være mest mulig agil ligger i fullautomatiserte tester. Dette skal sikre at man testingen blir utført ofte og riktig.

Testing for å sjekke om man leverer det man skal, er en viktig faktor. For å gjøre dette er det viktig at man faktisk har mulighet til å kunne teste på ulike måleenheter. Kravspesifikasjonen ble her brukt aktivt for å sjekke om man leverte det man skulle, og testing ble brukt for å endre kravspesifikasjonen etter hvert som kunder kom med endringer etter utførte tester. Det ble av noen respondenter påpekt mangler rundt rutiner og dokumentasjon i agile metoder. Flere trakk frem at agile metoder ikke legger opp til spesifikke teststrategier eller andre teknikker for testing. Dette er noe man må utvikle på egenhånd, ta inn fra andre metoder, teknikker eller man kan innføre komplett metoder for slik testing. Men det ble påpekt at det hadde vært en fordel hvis agile metoder hadde kommet med noen forslag til teknikker som kan brukes, for eksempel lagt mer opp til å bruke Scrum sammen med XP.

6.5 Leveranser og krav, en gjensidig påvirkende faktor



Figur 28: Leveranse og krav påvirker gjensidig

Mye av hva som har blitt sagt om leveranser i teorien har, som vi har presentert tidligere, fokusert på at man skal levere hva kunden ønsker, levere det kjapt osv. Hvis man tar utgangspunkt i de agile prinsippene, vil man ha fokus på selve produktet og ikke på andre elementer som kan stjele fokus fra dette (Ambler, 2007; Beck et al., 2001; Light, 2006). Dette bekreftes også av hva vi har funnet ut i vår studie. Men det er ikke nødvendigvis kun det at kunden skal kunne se en ny del av produktet til enhver tid som er årsaken til at utviklingsbedrifter velger å benytte seg av hyppige leveranser. En årsak som vi har funnet er at dersom man har hyppige endringer i krav på lik linje med at man har hyppige leveranser, vil man ved en rask leveranse raskere kunne endre fokus og få på plass nye krav.

Udefinerte krav tidlig i utviklingsfasen blir også trukket frem som en av motivasjonene for å bruke agile metoder. Dette er en faktor som vi ikke har funnet igjen i den litteraturen vi har studert. Mange bedrifter har påpekt at de starter med en visjon, en skisse eller lignende, og at det dermed vil være hensiktsmessig at man kan skifte fokus raskt. Dette bidrar da til at man kan raskere få konkretisert hva man faktisk skal gjøre, og få en bedre forståelse av hva slags krav det faktisk er man står overfor. Indirekte er dette også koblet til metode, det vil si at man velger en metode basert på hva slags krav man har. Metoden definerer da hva slags leveranser man skal ha.

På en annen side har vi også fått erfart at man må ha en prioritering av kravene til en viss grad, slik at man ikke ender opp med en evig syklus innen dette. Dette har blitt påpekt av flere av dem vi har snakket med, hvor viktig det er å få inn krav hele tiden i prosjektene (slik som agil systemuvikling sier). Men det blir også presisert at det er viktig at man setter grenser for når man faktisk skal kunne komme endringer, eller om dette for eksempel skal komme med i en senere leveranse (eller om det er plass i det hele tatt). Teorien har foreslått at man fryser krav før iterasjoner, slik at man får med det som er i den iterasjonen, og alt som kommer som endringer etter dette, betraktes som nytt (Ambler, 2004; Ambler, 2007). Man vil da kunne endre fokus i neste iterasjon til de nye kravene eller ønskene fra kunden som kom inn i tidligere faser.

Det er da imidlertid også viktig at man ikke lar dette gå for langt; til tross for at man fryser kravene, vil det være et behov for et stopp en gang. Vi har vært inne på at teorien sier at man bør fryse kravene før iterasjoner, men det nevnes lite at man en gang skal ha en stopp. Men det er dog viktig å påpeke at man skal ved agil utvikling alltid ha fungerende programvare (Beck et al., 2001) som vil tilsi at man kan stoppe når som helst. I løpet av intervjuene har vi fått nevnt at det som er viktig med slike metoder at man enten setter stopp når tiden er ute og dermed kutter funksjonalitet, eller at man utvider prosjektet slik at man får tid til de krav som ellers måtte bli kuttet bort. Light (2006) fremhever at man skal ha fokus på leveranser som går raskt og som leverer det kunden ønsker. Men man må ikke fokusere for mye på tidsfrister, ettersom dette kan ta fokus bort fra leveransen. Ved å kutte funksjonalitet eller utvide prosjektet, som våre bedrifter sa de gjorde, sørger man for å ikke ”jobbe seg i hjel” før

tidsfrister. Dette blir også bekreftet av Beck et al (2001) når det blir snakket om at man ikke skal benytte seg av skippertak og heller holde en konstant fart. Som en motsetning til dette har den statlige etaten vi snakket med, påpekt at man ikke i alle situasjoner vil kunne ha mulighet til å utvide tid eller kutte funksjonalitet, noe som da betyr at man må jobbe hardt for å kunne nå målet man har innenfor tidsfristene. Men på motsatt side så sier de også fra samme stedet at de ikke alltid tar med endringer som kommer i siste liten, for det er viktig å prioritere kravene og se hvor viktige endringene er. Dette vil da si at bedriftenes holdninger til dette ikke er helt konsistente, men det er viktig å påpeke at å jobbe i skippertak ikke er en frekvent nevnt tendens i de intervjuene vi har gjort.

Når vi nå er inne på krav og deres påvirkning på leveranser, har vi imidlertid ikke sett igjen noe konkret fra hva Ambler (2007) har påpekt som viktige punkter i forhold til å holde seg agil i utarbeidelsen av en kravspesifikasjon. Det som går igjen i det vi har sett, er i bunn og grunn at man starter med visjoner, skjermbilder eller lignende som er basisen for hvordan man bygger opp krav og leveranser i prosjekter bedriftene vi har vært i kontakt med kjører. Grunnen til at vi ikke spesifikt har funnet igjen disse punktene, kan være at man ikke ser dem som viktige eller at bedriftene ikke har noen rutiner eller retningslinjer for dette.

Det har også blitt påpekt som viktig at kunder deltar aktivt, dette sørger for at man kan endre kravene, og kundene får mer kontroll over kravene. Bruk av kundeterminologi og å gjøre interessenter om til utviklere ble trukket frem som utfordringer (Ambler, 2007). Noen av intervjuobjektene våre poengterte at det var vanskelig å kommunisere modeller og UML diagram til prosjektorienterte mennesker. Dette er en interessant problemstilling, ettersom man kan diskutere hvem som skal tilpasse seg hvem. Ideelt sett så hadde både utviklere hatt kundekompetanse og kunder utviklerkompetanse, men i praksis viser dette seg vanskelig. Man bør derfor tenke igjennom hvordan man best kan formulere kravspesifikasjonen slik at begge parter blir inkludert. Vi fant ikke noen konkluderende svar på dette spørsmålet, men fikk indikasjoner på at en løsning på dette vil være å forbedre kommunikasjonsprosessen mellom kunde og utviklere.

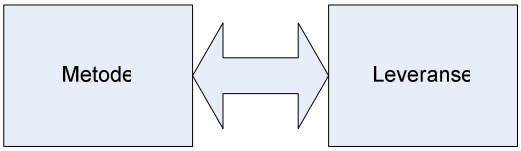
Når det gjelder hva som er viktigst for vår gruppe i forhold til Patel et al (2006) sin gruppe, ser vi for vår del at både fungerende programvare og hyppige leveranser har en viktig posisjon. De vi har snakket med har påpekt flere grunner til at det er viktig å levere raskt, og å levere fungerende programvare. At man har tidlige leveranser og krav som endrer seg hyppig, bidrar til at man har større sjanse til å tilfredsstille kundenes ønsker. Hadde man hatt endelige krav som ble stilt i starten, ville dette bli en stor utfordring. Det er vanskelig å trekke noen bastante konklusjoner om hva bedriftene vi har intervjuet mener er viktig rundt krav og leveranser. Det vi kan si er at det er en pekepinn på hvorfor det Patel et al (2006) sin undersøkelse viste var viktig faktisk er viktig. Men i tillegg finnes det også andre ting som vi har funnet i vår undersøkelse som ikke hadde høy prioritet i undersøkelsen til Patel. Tabellen i undersøkelsen vår (se Tabell 2: Rangering av prinsipper for agil utvikling (Patel et al., 2006)) viser at respondentene i Patel et al. (2006) sin undersøkelse rangerte behandling av krav meget lavt. Vi har derimot i vår undersøkelse funnet at dette blir ansett som meget viktig, og at det er en av hovedgrunnene til at agil utvikling blir valgt.

I relasjon til hva Light (2005) har pekt på i forbindelse med endringsledelse har vi ikke sett noe videre av i våre intervjuer. Det vil da si at man ikke har noen bestemt strategi i forhold til hvordan man ønsker å håndtere endringer som kommer inn i løpet av utviklingsprosessen.

På en annen side kan det hende at man har en strategi for hvordan man vil gjøre dette, men vi har ikke fått dette fortalt, og det stemmer heller da ikke overens med det vi har sett i teorien. Det teorien beskriver angående programvare for å håndtere dette, har vi sett lite til i det intervjuobjektene har sagt. Det nærmeste vi kommer her er at en har nevnt at de noterer ned endringsmeldinger fra kunder. Mer bruk av programvare i utviklingsprosessen kommer vi i stedet tilbake til når vi snakker om dokumentasjon.

Det er heller ikke gitt noen indikasjoner på hva Davies (2005) sier om å bruke historier til å illustrere krav sammen med kunden, for å øke forståelsen. Ingen av dem vi har snakket med har påpekt dette som et virkemiddel i forhold til å få bedre forståelse av udefinerte og diffuse krav. Man fokuserer heller, i følge våre intervjuobjekter, på å illustrere krav til kunden gjennom hyppige leveranser. På denne måten får kunden noe håndfast å forholde seg til. Dette gir kunden anledning til å ombestemme seg og endre kravet til noe mer konkret slik at de får levert hva de vil ha.

6.6 Metoden gir leveransesyklus, og leveranser definerer metodevalg



Figur 29: Metode og leveranse, gjensidig påvirkning

Hvordan metode og leveranse henger sammen har sammenheng med hvor hyppig man ønsker eller har behov for å levere som påvirker hva slags metode man velger. Hvilken metode man velger definerer også hva man skal levere og når man skal levere det. Vi har sett at når det var definert fra bedriftens side at man skulle ha lengre sykluser mellom hver levering, valgte man en tyngre prosess med mer dokumentasjon. På motsatt side valgte de som hadde hyppige leveringskrav mindre og lette metoder, for eksempel Scrum. Dette er en metode som er optimalisert for raske leveringer. Ettersom man velger ulike agile metode på bakgrunn av leveringstid, er dette et interessant tema, som vi vil forklare i neste avsnitt.

Dersom man har et internt krav om når en ny versjon skal være på markedet, finner man en metode som er akkurat omfattende nok til å produsere en ny versjon i løpet av denne tiden. I andre bedrifter fungerer dette annerledes, hvor de velger en metode og lar denne metoden styre når man kommer med en ny versjon. Hvilken av disse tilnærmingene som er ”riktig”, kan diskuteres. Cockburn (2000) peker på at man må ta i betraktning teamets størrelse, kritiskheten til systemet, prioriteten til systemet og erfaring hos deltakere. De 12 agile prinsipp (Beck et al., 2001) sier at man skal levere fungerende programvare ofte, intervallene spenner fra et par uker til et par måneder. Det vi ikke har funnet litteratur på er noe som beskriver eksplisitt at man velger metode på bakgrunn av lengden på hver leveransesyklus. Spesielt er det også at noen av de vi har snakket med ville valgt å benytte agile metoder med lang leveringssyklus. Her ville det nok vært naturlig for mange å velge en tradisjonell utviklingsmetode, men man velger heller en agil metode med lang iterasjon. De som har sagt dette mener det er fordi de ønsker den fleksibiliteten som agile metoder gir dem.

Flere andre respondenter sa og at selv om de fikk krav om å levere i lengre sykluser, ville de brukt agile metoder med lange iterasjoner. Eventuelt ville de levert interne versjoner hyppig og en kundeversjon til slutt, noe vi også finner igjen i Patel et al. (2006).

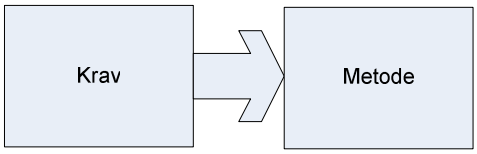
Grunnen til dette var ifølge respondentene våre det samme som Cockburn & Highsmith (2001b) påpeker, nemlig at en liten, men streng metode kan ligne agile utviklingsmetoder, men de føles og oppfører seg ikke likt som agile metoder. Hvis forslag om agile metoder med lange iterasjoner fungerer, slik som våre respondenter foreslo, vil dette være med på å understreke fleksibiliteten til agile metoder. Vi har derimot ikke funnet noen relevant litteratur som har undersøkt et slikt case, men i teorien ville det være meget interessant å teste ut.

Måling av leveranser i fungerende programvare, slik som er beskrevet i de agile prinsipp, virker som et utbredt fenomen blant våre bedrifter. Dette var ikke tilfellet i Patel et al (2006) sin undersøkelse, hvor man styrte etter milepæler. Fordelen med denne måten å levere på var at man kunne endre fokus mellom hver iterasjon. Det blir også trukket frem viktigheten av å kunne vise kundene et fungerende produkt tidlig i prosessen. I tillegg blir det hevdet at kunder får både mer tillit til utviklerne, og at en konkret leveranse gir større muligheter til å diskutere med kundene. Man kan samtidig hevde, slik Hotle (2006a) gjør, at agile metoder på mange måter bruker milepæler, men at disse passeres meget hurtig. Milepæler blir brukt annerledes i agile metoder med tanke på at slutten på hver iterasjon kan ansees som en milepæl. På denne måten kan man holde en seremoni i slutten av en iterasjon. Eller man kan markere hver iterasjon på noenlunde liknende måter, som agile metoder gjør når man går fra en fase til en annen. Slike seremonier eller ”happenings” kan være nyttig å ha for å motivere teamet (Harrison & Copllien, 1996).

Det er stor forskjell på hvordan metoden påvirker leveransene direkte i bedriftene. Noen bruker metoden til å definere relativt strengt hva som skal leveres i hver fase av utviklingen. Andre har derimot relativt få definerte leveranser i metoden. Det er også noe forskjell på hvordan man definerer en leveranse. En bedrift definerte designdokumenter og lignende som leveranser, mens andre mente at leveranse kun var fungerende programvare. Ser man til metodelitteraturen, fokuserer agil utvikling på at leveranser skal være fungerende programvare, mens tradisjonell utvikling, i tillegg til fungerende programvare, definerer også dokumenter som leveranser. Ambler (2007) peker på at i agil systemutvikling er målet ditt å implementere krav, ikke dokumentere dem.

Alle leveranser eller aktiviteter som inngår i utviklingen, og ikke direkte bidrar til målet om å utvikle fungerende programvare, bør unngåes (Ambler, 2004). Blant våre bedrifter ble det derimot ansett som positivt at en metode har en viss styring av leveranser. Man trekker frem at metode styrer leveransene og gjør det enklere for både utviklere og kunder å se hva som faktisk skal leveres i hver iterasjon. I motsetning til de agile prinsippene som kun ser på fungerende programvare som en leveranse, ble det trukket frem at andre leveranser også var fordelaktig å ha med i definisjonen av metode. Eksempler på dette kan være designmodeller, prototyper, endringsforslag og kravspesifikasjon. Løsningen her er nok, som Ambler og foreslår, at man er kritisk til å lage for mye dokumenter. Men det er allikevel viktig å påpeke at man ikke må fornekte at leveranser utenom fungerende programvare kan være nyttig i visse situasjoner.

6.7 Stabilitet i krav, et grunnlag for valg av metode



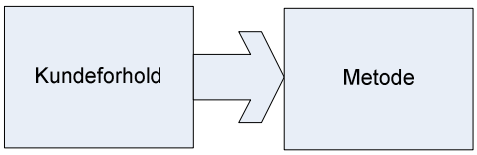
Figur 30: Krav påvirker valg av metode

I teorien er det mye som har blitt påpekt som bakgrunn for valg av metode. Dette gjelder ikke bare for valget i seg selv, men også for når metoden blir tilpasset i bruk. Cockburn (2000) peker på en rekke faktorer som ligger til grunn for at man skal velge den metoden man faktisk velger, blant annet størrelse på teamet, størrelse på metode, kritiskhet i system og kommunikasjonsformer.

Det vi ser går igjen i begrunnelsen for valg av metode er blant annet at man har hyppige endringer i kravene, kravene er svært lite spesifikke i starten av prosjektet og man vet langt fra alle kravene på forhånd. Flere av de vi har snakket med sier at man bruker dette som en bakgrunn for hvordan de velger metoder. Det som da er årsaken er at dersom man har såpass hyppige endringer som intervjuobjektene beskriver, vil det være behov for å ha en metode som lar deg endre krav og tilføre krav kontinuerlig. Det vil i tillegg være en årsak som passer inn rundt agile systemutviklingsmetoder. Som det også blir pekt på i intervjuene, vil ikke denne typen behandling av krav passe særlig godt inn i en type metode på linje med en fossefallsmetode. Det at man har veldig vage krav tidlig i utviklingsprosessen, blir også trukket frem som en av hovedgrunnene til at man velger agile metoder. Man lar da kundene komme opp med nye endringer mens man utvikler, og mens kundene ser hvordan det nye systemet kommer til å se ut.

Dessuten er det viktig å nevne at man bør ha en agil metode i en situasjon der kravene endrer seg hyppig. Men dersom man ser på dette i et perspektiv hvor valg av metode er i fokus, er dette en ny vinkling og et nytt argument vi ikke har sett tidligere. Dette kan ansees som et supplement til de fire faktorene for valg av metode vi finner hos Cockburn (2000).

6.8 Involvering av kunde, en viktig faktor for metode



Figur 31: Kundeinvolvering påvirker metoden

I følge våre funn har tett arbeid med kunde mye å si for hvordan metoden blir brukt i praksis. I tillegg spiller dette en rolle i forhold til hvilken metode man velger å bruke. Om man da bør ha en formell avtale om hvor mye en kunde skal delta i prosessen eller ikke, er noe usikkert. Light (2006) peker på at man bør ha det, dette blir også påpekt av ett av våre intervjuobjekter. Men i intervjuet blir det sagt at kun dersom man antar/det ville vise seg at kundeinvolveringen skulle bli et problem, ville man bedt om en formell avtale. Det er også viktig å nevne at våre intervjuer har gitt indikasjoner på at det er viktig med tett arbeid, for metoden vil ikke kunne bli fulgt i praksis slik den er ment til å gjøre (særlig ved Scrum), i og med enkelte har brukt metoden ved prosjekter som foregår over distanser.

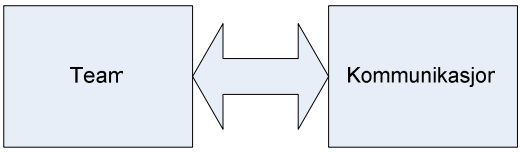
Dette er spesielt viktig når det er snakk om avstandsforhold mellom utviklingsbedrift og kunde. På en annen side har også noen av de vi har snakket med sagt at det nødvendigvis ikke vil være en umulighet, men det vil være en utfordring. I sum ser vi av dette vil det være fordelaktig at man har et tett arbeid med kunden når man benytter seg av agile metoder. De 12 agile prinsipp stadfester og klart at det er viktig at utviklere og forretningsmennesker jobber tett sammen (Beck et al., 2001).

På en annen side er det også viktig at man også tar hensyn til kunden når man velger metode, slik som det blir sagt i teorien (Cockburn & Highsmith, 2001a). Kravene/uttalelsene?? fra teorien er noe mer bastante enn hva vi har funnet ut i våre intervjuer, men intervjuene våre viser til at det er viktig å se på hva slags type involvering man kan få til. Det har blitt pekt på at lite involvering krever tyngre metoder (i forhold til spesifisering) enn i prosjekter hvor man har svært stor involvering. Spesifiseringen har man her i starten av prosjektet og kjører brukertester underveis for å finne ut endringer. Lindvall et al. (2002) foreslår en løsning hvor brukeren selv skriver testen for å delta aktivt i testen. Dette har vi ikke funnet noen klare indikasjoner på blant våre bedrifter. Men på en annen side vises det imidlertid at kundeinvolvering vil ha en effekt på hva slags metode man velger, og hva det har å si for denne i praksis. Hanssen & Fægri (2006) påpeker også at engasjering av kunde er viktig for å bruke agile utviklingsmetoder. I våre intervjuer fant vi også at når man innførte agile metoder, syntes kunden at dette var noe nytt og spennende i forhold til tradisjonelle metoder. En slik holdning ble ansett som positivt fra både kundenes og utviklernes side, de var fornøyd med at de fikk såpass stor involvering i prosjektet.

Utover dette har det også blitt sagt av noen av intervjuobjektene at det er viktig å gi beskjed tidlig om hva som kreves av kunden, særlig i forbindelse med tett og daglig samarbeid. Som i utgangspunktet er det i samsvar med det Norton (2006) mener i sin modell hvor det er viktig at både utviklere og kunder ser viktigheten av hverandre og det å jobbe tett sammen. En annen viktig faktor ved kundeinvolvering som vi fant, var at kunden var flink til å ta beslutninger underveis. Dette vil være viktig når utviklere og kunder skal forstå viktigheten av hverandre (Hanssen & Fægri, 2006; Norton, 2006).

Men det er nevnt i faglitteraturen at det ikke kun er kundeforhold som påvirker bruk av metoden, men også andre krefter i konteksten (Fitzgerald et al., 2002; Norton, 2006). Vi har ikke funnet klare uttalelser som kan støtte opp om denne teorien. Det som våre intervjuobjekt har fokusert på, er at kunden og mottakerorganisasjonen er de viktigste påvirkerne utenfor teamet og utviklingsorganisasjonen. Dette vil selvfølgelig ikke si at de ikke blir påvirket av regler, samfunnet og andre utenforstående, men det ble ikke nevnt spesifikt i våre intervjuer.

6.9 Team og kommunikasjon, en nødvendighet for god praksis



Figur 32: Team og kommunikasjon påvirker hverandre begge veier

Teamet er i vår oppgave definert som utviklere og andre involverte fra utviklersida. I tillegg til hvordan teamet er satt sammen er det også interessant å se hvordan teamet arbeider. Alle intervjuobjektene våre rapporterte at de satt og jobbet sammen i åpne landskap eller med kun få kontorer mellom hverandre. Dette trakk man frem som en stor fordel ettersom man kunne kommunisere raskt og enkelt. Dette er også dokumentert av Sutherland (2004) som peker på at dette fører til mer effektivt arbeid. Ettersom dette ble gjennomført av samtlige av intervjuobjektene våre, kan det tyde på at denne metoden å arbeide på begynner å bli ansett som en slags ”best practice” på utviklingsområdet. Det ble ikke rapportert om noen negative opplevelser med denne arbeidsformen.

De som hadde teamet spredt over større avstander, hadde større utfordringer rundt kommunikasjon og involvering av teamet. Man måtte bruke større tid på administrasjon og oppfølging av ansatte. Men vi kan si oss enige med Cockburn & Highsmith (2001b), som hevder at agile team oppfører seg som økosystem. Selv om man fikk større avstander og muligens ikke god nok involvering mellom teammedlemmene, fant man andre måter å jobbe på. Scrum er optimalisert for å møtes hver dag, som et av våre intervjuobjekter har sagt, men det var ingenting som hindret deg fra å jobbe over avstand. Dette ville da være en faktor som forårsaker en del mer administrasjon. Det er lite tvil om at det er best å kommunisere ansikt til ansikt for eksempel på en tavle, som Cockburn (2000) hevder.
Det som blir poengtert er at kommunikasjonen skjer hyppig. Selv om det er fordelaktig å gjøre det ansikt til ansikt, er det like viktig å ha hyppig kommunikasjon når man sitter med avstand mellom hverandre. Spesielt Scrums daglige møter ble ansett som svært verdifulle, når det gjaldt å kommunisere innad i teamet. Her hadde man mulighet til å oppdatere seg på hva andre holdt på med, og bli klar over/spesifisere hvilke problemer man hadde. I tillegg hadde man mulighet til å kommunisere med kunder. Det ble også påpekt at det var viktig at disse møtene ble administrert strengt. Med andre ord vil dette si at man kun snakket om akkurat det som er nødvendig for å oppdatere de andre, og at noen ikke ”kuppet” hele møtet og pratet om ting som er unødvendig for alle å høre. Et av intervjuobjektene som brukte Scrum, selv om teamet var spredd over store distanser, rapporterte at de hadde sluttet med ”daily scrum”. Man hadde da et større møte litt sjeldnere. Problemet som oppstod her var at man måtte bruke større ressurser på å administrere utviklerne. Andre problemer her var oppdatering av timelister og at man ikke visste like godt hva hver utvikler jobbet med på et gitt tidspunkt.

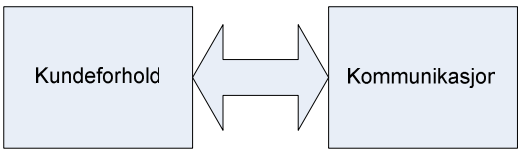
Såkalt gruppevare (e: groupware) ble også trukket frem som viktig når det gjaldt kommunikasjon innad i teamet. De fleste bedriftene hadde en løsning på dette (henholdsvis Microsoft Team Foundation Server eller Lotus Notes). Disse verktøyene ble brukt aktivt for å kommunisere innad i teamet. Man la opp løsninger eller endringer man hadde støtt på, la opp tester og kode som måtte gjennomgåes og lignende. Dette funnet kan støttes opp av de agile prinsippene som sier at man skal gi motiverte medarbeidere de verktøyene de trenger for å få jobben gjort. Det vi fant stemmer med dette, men vi ser at det er muligheter for å fremme bruken av dette på en sterkere måte.

Disse systemene er store og komplekse, og det virket på noen av funnene våre som at rutiner rundt bruken av disse muligens ikke var optimalisert. Det er lite litteratur på dette området, hvordan både agile og tradisjonelle utviklingsmetoder benytter seg av og gir muligheter for å optimalisere bruk av slike løsninger.

Generelle regler for kommunikasjon vil og være fordelaktig når man når en viss størrelse i prosjektet. Prosjekter som er såpass små at man møter og snakker med alle i teamet hver dag trenger ikke veldig strukturerte regler. Men flere bedrifter bør vurdere å formalisere kommunikasjonen bedre. Selv om man kanskje går litt på bekostning av det å være agil, er det viktig at man kan formaliserer kommunikasjon til en viss grad. Light(2006) peker på viktigheten av å ha regler rundt kommunikasjon, ettersom teammedlemmer kommer og går. Dette kommer jo selvfølgelig an på kulturen i bedriften og hvor lange hvert prosjekt er. Agile prosjekter som varer over kort tidsrom, vil muligens ikke være fullt så bundet av å ha formelle regler, i motsetning til de som går over lengre tid. Kommunikasjonsregler i den daglige arbeidssituasjonen til teamet ble ikke trukket frem som spesielt viktige av våre intervjuobjekt, likevel ble det ansett som viktig at den formelle kommunikasjonen fungerte, for eksempel når det gjaldt å sjekke inn og ut kode og lignende.

Litteraturen har mye av de samme funnene som vi har funnet. Det vi på en annen side ikke har funnet støtte for, er vektlegging av teamkompetanse og samarbeidskompetanse i teamet. Det kan virke som om dette blir tatt som en selvfølge, og at man ikke er så fokusert på dette. Ingen av våre respondenter trakk frem måter man hevet samarbeidskompetansen innad i teamet på. Det kan på vår intervjurunde virke som om det å samarbeide og forholde seg til hverandre, ble støttet opp ved hjelp av generelle kommunikasjonsprosesser. Til tross for dette finner man igjen i intervjusvarene? noen av de samme momentene som teorien nevner i kategorien kommunikasjon, hvor det fokuseres sterkt på at medlemmene kommuniserer godt. I forhold til teorien kan våre funn tyde på at noen agile metoder mangler noe når det gjelder teknikker som brukes for å forbedre kommunikasjon. Etter hva vi har fått inntrykk av gjennom intervjuene, vil det være fordelaktig at man formaliserer dette på en eller annen måte. Et eksempel kan være å bruke morgenmøter slik Scrum gjør.

6.10 Kundeforhold og kommunikasjon, tett arbeid påvirker gjensidig



Figur 33: Kundeforhold og kommunikasjon, gjensidig påvirkning

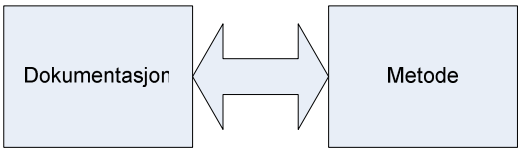
At kundeforholdet påvirker kommunikasjonen og omvendt, har vi vært inne på i forklaringen av sammenhengen vi ser i vår forklaringsmodell. Men hvordan står teorien i forhold til hva vi har sett hos de bedriftene vi har snakket med? I følge de agile prinsippene skal man arbeide tett sammen med kunden og kommunisere daglig med han/henne (Beck et al., 2001). Dette har vi også sett er viktig i følge våre intervjuer, noe som da med andre ord er med på å bekrefte at teorien stemmer. Effektiviteten av kommunikasjonsformer, slik som Cockburn (2000) sier, har vi og funnet bekreftelse på i intervjuene. Et eksempel er at det blir nevnt at det er enklere å skyve bort daglige møter dersom man arbeider med møter over telefon, enn dersom man sitter med gangavstand fra hverandre i samme bygget. Jo nærmere man kommer hverandre, jo bedre vil man kommunisere. Med andre ord vil praksisen her bekrefte det teorien sier.

Videokonferanse ble trukket frem som et viktig verktøy blant de bedriftene som hadde teamet spredt over store avstander. Vi har ikke funnet noen teori som tar for seg denne spesifikke bruken av multimedia, men vil anta at videokonferanse vil ligge tett opp til Cockburn (2002) sin modell med to personer ved en tavle som den mest effektive formen for kommunikasjon.

Det er påpekt at det er viktig for enkelte at man har høy synlighet inn i utviklingsprosessen. Det vil si at man for eksempel sitter ute hos kunden og diskuterer skjermbilder og lignende. Dette gjøres fordi det er viktig for kunden at han/hun ser hva som blir gjort og fort får muligheten til å gjøre forandringer og ha kontroll på fremgang. Dette er også blitt bekreftet av Hanssen & Fægri (2006). Dette aspektet lar kunden komme tettere innpå utviklingsprosjektet og produktet, og man vil oppnå økt kommunikasjon slik vi tolker hva som blir sagt. Også dette stemmer overens med hva som blir presentert i den teoretiske modellen.

Tillitt fra kunden til utviklerorganisasjonen er et aspekt som ble trukket frem som essensielt for at agil systemutvikling skulle fungere. Intervjuobjektet mente at det var viktig at kunden stoler på at utvikleren greier å levere et fungerende system innenfor tidsfristen. Dette spiller en stor rolle ettersom man ikke har klare krav og tidsfrister som blir laget tidlig i prosjektet. I verste fall er det mulig at man faktisk ikke får levert et fungerende system. Når man på en annen side driver tradisjonell utvikling er det høyst usikkert om man får levert eller oppfylt de kravene man har satt seg i analysefasen. Vi har ikke funnet teori som omhandler tillitsaspektet spesielt, men det vil være en naturlig forlenging av samhandling og kommunikasjon som blir påpekt i de 12 agile prinsipp (Beck et al., 2001). Man kan selvfølgelig også diskutere hva som fordrer mest tillit, tradisjonelle eller agile metoder, men det kan nok føles noe tryggere for kunder å ha en ferdig spikret tidsplan tidlig i utviklingen.

6.11 Valgt metode definerer dokumentasjon og omvendt

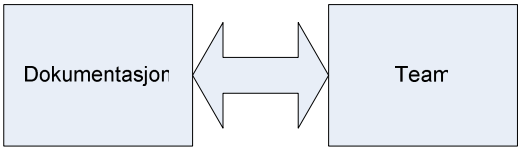


Figur 34: Gjensidig påvirkning for metode og dokumentasjon

Agile metoder karakteriseres som å være en lett, men tilfredsstillende prosess (Cockburn, 2002). Flere av våre bedrifter sier at agile metoder løper litt ifra ansvaret rundt dokumentasjon og prosesser i metoden. De mener at veldig mye er overlatt til utviklerne, og agile metoder kan i noen tilfeller være litt for lite å støtte seg til. Man kan da argumentere for eller imot hvordan man skal bruke agile metoder. Flere vil nok hevde at dette er en av styrkene til agil utvikling at man har såpass stor fleksibilitet fordi man kan bruke ulike arbeidsmetoder, som man selv mener er hensiktsmessige. Men det er allikevel et tankekors når flere utviklere savner flere retningslinjer. Det er veldig vanskelig å definere hvor en metode bør legge seg her. Med for mange regler og dokumentasjon mister man fordelene rundt å det være agil, mens for få regler kan føre til at utviklerne ikke ser noen hjelp i metoden. Svaret kan være som en respondent foreslo: å integrere for eksempel XP i Scrum. Scrum ville da være den overordnede metoden man styrte etter, mens man brukte teknikkene for testing og programmering fra XP. Dette er på mange måter slik agile metoder skal benyttes, men det bør muligens utvikles videre for å sørge optimalisering av en slik prosess.

Fordelene som ble trukket frem med tilstrekkelig dokumentasjon, var at det var enklere for teamet å arbeide sammen, spesielt hvis det kom nye utviklere eller uerfarne utviklere til. Ved å ha tilstrekkelig dokumentasjon rundt arbeidsprosesser og metode har man ”noe å støtte seg til” hvis man er usikker på hvordan man skal utføre ulike elementer i metoden. Chau et al.(2003) og Lindvall et al. (2002) fokuserer på hvordan opplæring fungerer i team. Mentoring og jobbing i par blir trukket frem som styrker rundt agile metoder. Ved å jobbe på denne måten er tanken at både nyansatte og andre i teamet vil lære av hverandre. I vår undersøkelse har vi ikke funnet noe som tyder på at dette har blitt brukt i stor utstrekning. Det har derimot blitt pekt på at det er viktig å ha tilstrekkelig dokumentasjon rundt metoden, for å sørge for at nye personer kan fungere bra i teamet. Det har også blitt påpekt at noe slikt gjelder mennesker som er nye i faget, som da har behov for mindre dokumentasjon enn erfarne utviklere. Dette er også dokumentert i Cockburn (2002) og i de 12 agile prinsippene, ved at erfarne utviklere kan være opp til ti ganger mer effektive enn uerfarne utviklere. Løsningen i de bedriftene vi intervjuet var tett kommunikasjon, og spesielt morgenmøter ble trukket frem som viktige. Dette ble ansett som et forum der utviklerne kunne stille spørsmål og få en oversikt over hva de ulike personene i teamet gjorde.

6.12 Dokumentasjon og team, erfaring og standarder påvirker hverandre



Figur 35: Team og dokumentasjon påvirker hverandre

Dokumentasjon er en viktig del av systemutvikling. Vi har identifisert at dokumentasjon blir brukt i to ulike kontekster: En der man kan ha dokumentasjon rundt metoden om hva man skal bruke metoden og en annen der man kan ha dokumentasjon som dokumenterer den programvaren man utvikler. Det blir også påpekt viktigheten av å ikke lage unødig dokumentasjon. Bedriftene vi har intervjuet anser ubrukt dokumentasjon for tapt arbeidstid som heller eventuelt kunne blitt brukt til å forbedre produktet.

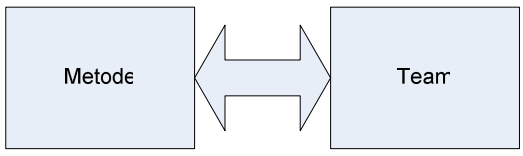
Den største bedriften vi intervjuet anså også dokumentasjonen rundt metoden til å være veldig viktig for å skaffe et felles begrepsapparat. Dette gjeldt både innad i teamet og ut mot kunder. De mente at uten en felles forståelse, som var dokumentert i metoden, ville det bli vanskelig å kommunisere innad i teamet. Eksempel på dette er at når man snakket om systemtest, så visste alle hva det dreide seg om. Uten en dokumentasjon rundt dette kan begrepet bety noe forskjellig for ulike utviklere. Grunnen til at man fokuserte mye på dette her, er antakelig størrelsen på organisasjonen, og at man ikke alltid hadde mulighet til å jobbe ansikt til ansikt. Det er lite fokus på slikt begrepsapparat i den agile litteraturen, men dette kan ha noe å gjøre med at agile metoder legger opp til tett interaksjon, og at personer møtes ansikt til ansikt. Men det er uten tvil viktig at man har like begreper, for å unngå misforståelser. Selv om man sitter sammen, kan man fort komme opp i situasjoner der man bruker samme uttrykk, men snakker om to vidt forskjellige ting.

En annen viktig faktor som spiller inn rundt dokumentasjon og team er størrelsen på teamet og modenheten til bedriften. Mindre, nystartede bedrifter med få ansatte har som regel ikke så mye dokumentasjon rundt metoden. Dette er også støttet opp av Cockburn (2000). Man må ofte igjennom en prosess mens bedriften og systemet vokser; etter hvert som ting blir større og mer uoversiktlige, blir man tvunget til å ha mer dokumentasjon rundt utviklingen og programvaren. Ifølge Cockburn (2002) vil det være mest hensiktsmessig, å strekke en mindre metode for å passe et større prosjekt, istedenfor å skalere ned en større metode. Å skalere ned en større metode vil bety for mye administrasjon for et lite team. Det vil derfor være hensiktsmessig for en mindre bedrift å holde på en liten metode så lenge som mulig, uten at det selvfølgelig går ut over produktet.

Ambler (2007) peker på at jo mer kompleks metode man bruker, jo større utgifter vil endringer bety. Har man for eksempel syv modeller, vil en endring måtte foretaes i alle disse modellene; dette er tidkrevende, spesielt hvis man ser i forhold til hvis man bare hadde syv modeller. På bakgrunn av dette kan man trekke som konklusjon at man absolutt ikke bør lage flere modeller enn høyst nødvendig. Det koster ikke bare å utvikle disse, men faktisk minst like mye, om ikke mer, å vedlikeholde dem. I tillegg vil små krav være enklere å estimere og lede.

En amerikansk casestudie av Baum (2005) pekte på at man ved å gå fra tradisjonell utvikling med mye dokumentasjon til agil utvikling med mindre dokumentasjon kunne få store kostnadsbesparelser. I tillegg har Waters (2007b) og de 12 agile prinsippene (Beck et al., 2001) pekt på viktigheten av å minimalisere dokumentasjon. Dette er også noe vi har funnet i vår studie. Flere av bedriftene er veldig skeptiske til å få for mye dokumentasjon inn i utviklingsprosessen, man fokuserer heller på funksjonaliteten rundt produktet. Faren med denne tilnærmingen er at man dokumenterer for lite. Dette kan være farligere enn for mye dokumentasjon. Årsaken til dette er at hvis en del av programmet ikke fungerer og man ikke har dokumentert skikkelig, risikerer man å måtte endre eller lage hele denne delen på nytt. I tillegg risikerer man at vedlikehold av programvaren blir svært vanskelig. Etter vår mening er det veldig stort fokus på akkurat nok dokumentasjon. Dette er bra og selvfølgelig nyttig for å få ned utviklingskostnader. Men det er viktig at man også ser farene ved for lite dokumentasjon. Man må ikke se seg blinde på å kutte i dokumentasjonen helt til man får en ufullstendig og lite fordelaktig dokumentasjon.

6.13 Metode og team, en gjensidig tilpassende faktor



Figur 36: Team og metode påvirker hverandre begge veier

Team og metode henger tett sammen. Metoden definerer hvordan teamet arbeider, og måten teamet arbeider eller ønsker å arbeide påvirker metoden. Dette finner man også igjen i Fitzgerald et al. (2002) sin modell (se Figur 8: "A framework for ISD method use" (Fitzgerald et al., 2002)), hvor det beskrives hvordan metoden i bruk (e: method in action) påvirkes av både utviklere og den formelle metoden. Eksempler på hvordan metoden påvirker teamet er formelle regler for samarbeid og kommunikasjon, koderegler og leveranser. Metoden blir påvirket eksempelvis av erfaringen til utviklerne, arbeidsmetodene deres og hvordan man samarbeider i teamet.

Det klareste punktet vi fant i denne sammenhengen var at erfaringen til utviklerne var avgjørende for hva slags metode som blir brukt. Flere av intervjuobjektene våre hadde ”tatt med seg” metoden fra tidligere arbeidsgivere og/eller utviklingsprosjekt. De hadde hatt gode erfaringer med utviklingsmetodene fra før, og valgte å innføre disse også i de nye organisasjonene. Det er også viktig at man har noen erfarne utviklere i teamet, spesielt når man bruker Scrum, ettersom teamet selv står ansvarlig for estimering og planlegging. Men vårt inntrykk er at spesielt prosjektleders erfaring veide tungt når man valgte metode. Faren som vi kan se her er at man ”ser seg blind” i valget av metode. Hvis man for eksempel har hatt en god erfaring med en metode, vil man selvfølgelig bruke denne på nytt. Det som er faren her er dersom denne metoden passer dårlig til prosjektet. Da vil man, i motsetning til positive erfaringer, oppleve stor reduksjon i produktivitet og oppfylling av krav.

En annen grunn som ble trukket frem som viktig i valg av metode var det å motivere ansatte. Dette støttes også opp om av ulike prinsipper for agil systemuvikling (Beck et al., 2001; Light, 2005). Bedriftene i denne studien påpekte at der teamet selv fikk større ansvar for det de skulle levere, gjorde det at de fant arbeidet sitt mer meningsfylt, i motsetning til hva de gjorde om de ble overvåket og fortalt hva de skulle gjøre av en prosjektleder. Motivasjon ble også brukt som en positiv faktor med tanke på kortere iterasjoner. Her fikk man da fokus på ulike deler av systemet, og deltakerne fikk mulighet til å ha mer varierte arbeidsoppgaver. Vi har ikke funnet noen lignende funn i tidligere forskning som påpeker denne fordelen med korte iterasjoner. Vi ser dog absolutt fordelen med at utviklere har mulighet til variasjon i arbeidsdagen. I tillegg er det nyttig for mennesker å ha mål å se frem til, og å se at disse blir oppfylt. På den måten hjelper korte iterasjoner til at man ser god fremgang i prosjektet og raskere kan oppnå mål man har satt seg.

Hva som påvirker hverandre mest av team og metode er vanskelig å finne entydige svar på. Noen respondenter svarte at de oppdaterte utviklingsmodellen kontinuerlig etter hvert som de kom frem til bedre arbeidsrutiner. Andre igjen brukte utviklingsmetoden når de skulle definere arbeidsoppgaver. Etter hva vi har funnet, og som blir støttet opp om av Fitzgerald et al. (2002), påvirker disse hverandre jevnbyrdig. Vår oppfatning er at disse to kategoriene både bør bli lagt vekt på og skal taes hensyn til og påvirke hverandre. Det vil være bortimot umulig og lite hensiktsmessig å ikke ta hensyn til begge kategorier i utviklingen. Utvikling i praksis har en del å lære av teorien og teorien må bygges på praksis.

7 Konklusjon

Denne studien har hatt fokus på å identifisere vesentlig årsaker til at man bruker agile metoder i praksis og som påvirker ulike elementer i agile utviklingsmetoder og bruken av dem. I tillegg har vi sett på hvordan disse faktorene påvirker hverandre, både når det gjelder valg av metode og underveis i utviklingen. Til slutt har vi også registrert holdninger til de ulike faktorene og utviklingsmetoder som våre intervjuobjekter har fokusert på. Det finnes relativt mye litteratur innen dette området fra før. Vi har funnet igjen mye av den tidligere forskningen i våre intervjuer. Hovedsakelig har dette omhandlet valg av metode, krav, leveranser og kommunikasjon, noe som hjelper til å validere og støtte opp om vår modell. I tillegg har vi sett andre emner fra litteraturen men dette er de mest sentrale. Mangelen i tidligere forskning, slik vi så det, var at det ikke fantes en helhetlig oversikt eller modell over ulike faktorer som spiller inn i utviklingen. Det finnes mye litteratur som tar for seg to eller tre faktorer, men lite som viser hvordan flere faktorer påvirker hverandre. I tillegg har vi funnet flere sammenhenger som ikke har blitt poengtert noe videre i tidligere litteratur.

Vi har intervjuet fem bedrifter som driver med systemutvikling, for å avdekke hva de mener er viktige faktorer innen systemutvikling. På bakgrunn av disse intervjuene har vi kommet frem til ni kategorier som bedriftene mener er viktige for å få en god utviklingsprosess og for å levere et godt produkt. Ut fra intervjuene har vi også kommet frem til hvordan disse kategoriene påvirker hverandre. Man kan diskutere hvorfor disse kategoriene har blitt brukt, men vi har prøvd å generalisere kategoriene så mye som mulig slik at modellen skal være mulig å bruke i andre sammenhenger.

Kundeforhold er en kategori som ikke har blitt fokusert mye på i tidligere litteratur. Det blir påpekt innenfor agil systemutvikling at det er viktig med stor kundeinvolvering, men det er lite dokumentasjon rundt hvordan kunden faktisk påvirker utviklingen. Våre funn tyder på at kunde spiller en stor rolle når man skal velge utviklingsmetode. Det er viktig at man har en kunde som er villig til og ønsker å involvere seg i utviklingen, hvis man ønsker å velge en agil utviklingsmetode.

En av de viktigste grunnene til at man tok i bruk agile metoder var at kravene endret seg veldig ofte, og kravene ofte var vage i starten av prosjektet. Ved å ha hyppige leveranser både internt og ut mot kunder, fikk man større forståelse av hvordan systemet burde fungere, og kravene endret seg etter hver leveranse. Det kom også frem at man valgte metode på bakgrunn av hva slags krav man hadde og hvor ofte man hadde krav om eller ønsket å komme med leveranser.

I tillegg er kommunikasjonen med kunder trukket frem som en meget viktig faktor i utviklingen. Her ble fysisk tett samarbeid og god oppfølging fra kunde pekt på som viktig. Vi anser det som litt bemerkelsesverdig at det ikke finnes noe videre tidligere forskning på kunders involvering i utviklingsprosessen, spesielt siden det faktisk er kunden man utvikler programvare for.

Kommunikasjon er essensielt i agile metoder. Dette gjelder både innad i teamet og også ut mot kunder. Tidligere litteratur har fokusert på at denne kommunikasjonen må være tett, og helst ansikt til ansikt, for å få en best mulig kommunikasjonsprosess. Våre respondenter svarte med at tetthet og hyppighet var viktig. Det ble også trukket frem som viktig at det fantes visse regler rundt dette.

Selv om avstand ble trukket frem som negativt, ble avstandsteam også brukt i noen grad. Det ble ansett for noe mer tungvint å administrere, men fungerte tilfredsstillende når man hadde videokonferanser og interaktive verktøy å kommunisere med.

Utviklingsteamet ble og trukket frem som en viktig påvirkningsfaktor i utviklingen. Basert på teamets kompetanse og tidligere erfaringer valgte man metode. I tillegg spilte kommunikasjonen, kompetansen og motivasjonen til teamet inn på hvordan utviklingsprosessen ble gjennomført. Vi har spesielt merket oss at tidligere gode erfaringer fra utviklingsmetoder har spilt en meget stor rolle når man velger utviklingsmetode i nye prosjekter. Flere av intervjuobjektene våre, har tatt med seg den utviklingsmetoden man brukte hos tidligere arbeidsgiver. Det positive med dette er at man bruker en metode man faktisk kan. På en annen side er det også en risiko ved at man låser seg til en metode og ikke utforsker andre metoder som kan fungere bedre. Dette ender opp i to sider av samme sak, som vi ikke føler vi kan si noe konkret om det er helt positivt eller helt negativt.

Det finnes ikke noe endelig svar på det vi har tatt for oss nå. På en annen side det vi faktisk kan si er at vi har lokalisert at det er noen av de emnene som det er forsket på tidligere som tilsynelatende er viktigere enn andre. Kommunikasjon, kundeforhold, krav, leveranse og team er viktige elementer for å drive agil systemutvikling i følge de vi har snakket med. Det som skiller seg spesielt ut her og som vi kan tolke av intervjuobjektene er det som har mest betydning for dem er kommunikasjon. I forhold til den tidligere forskningen er dette en ny vinkling i og med vi ikke har sett noe arbeid som kan bekrefte en faktor som er svært mye viktigere enn andre. Man skal dog ikke ta alt for god fisk og det er som sagt ikke noe fasitsvar på det vi har undersøkt men det gir rom for interessante tanker rundt agil systemuvikling. Det sier litt om hvor viktig disse fem forskjellige egenskapene faktisk er.

7.1 Implikasjoner for videre forskning

I løpet av denne studien har vi kommet frem til flere vesentlige årsaker, som for eksempel kommunikasjon, team, kundeforhold og krav, som påvirker hvordan man bruker agil systemutvikling. Vi har illustrert hvordan disse kategoriene påvirker hverandre og hvordan oppfatningen rundt systemutvikling har vært i de bedriftene vi har intervjuet. Studien vi har gjennomført er en flercasestudie, og det vil dermed være interessant å prøve ut modellen, fordi man da kan se om funnene våre kan generaliseres.

En annen mulighet er å utarbeide hypoteser og påstander basert på vår modell og teste disse kvantitativt. Dette vil være med på å kunne validere våre funn, og eventuelt forbedre vår modell. Vi har gjennom vår studie fått en viss formening om hva man mener er de viktigste faktorene i modellen, men det vil være interessant å teste ut dette kvantitativt. Man vil da ha mulighet til å rangere de ulike sammenhengene og kategoriene.

Et tankevekkende funn i vår studie er kunders involvering og påvirkningsgrad ved valg og bruk av metode. Det vil være spennende å gjøre en studie på dette området for å avdekke flere faktorer rundt kunders involvering i utviklingsprosessen. I forbindelse med dette ville det være svært interessant dersom man fikk kontakt med de faktiske kundene for å se dette fra deres synsvinkel. Dette vil da gi muligheten til å bekrefte eller avkrefte hva de ulike bedriftene har sagt om dette teamet.

7.2 Begrensninger for vår forskning

Ettersom dette er en masteroppgave, vil det selvfølgelig være begrensinger i tid og omfang. Dette har ført til at et begrenset antall bedrifter har blitt intervjuet. Begrensningen rundt forskningsmetode, hvor vi har brukt en flercasestudie, er at vi har et begrenset utvalg med intervjuobjekter. De bedriftene som har blitt intervjuet, befinner seg på mange måter innenfor samme segment, ved at de er små og mellomstore bedrifter som driver med programvareutvikling.

Vi har tilstrebet å få en mest mulig åpen dialog med intervjuobjektene, og fokusert på å finne ut hvordan de faktisk arbeider. På tross av dette finnes det muligheter for at vi har misforstått intervjuobjektene, eller vi ikke har fått frem viktige punkter i intervjuene.

I tillegg til dette har vi ikke, som nevnt i forrige delkapittel, fått tak i kundene til bedriftene. Årsaken til dette var de var vanskelig å få tak i gjennom bedriftene, noen ville ikke gi oss informasjon om dette og det har også vært et spørsmål om tilgjengelig tid. Det har dermed ikke gitt oss begge sider av saken i forbindelse med hva kunden har å si ved valg og bruk av agile metoder. Konsekvensen av dette er at vi ikke har kommet frem til funn som man kan kalle generaliserbare eller til en viss grad gyldige, i og med vi ikke har hatt kontakt med begge aktørene. Denne valideringen gjelder da ikke for hele studien, men kun for de segmenter som tar for seg der kundene er aktuelle.

7.3 Vårt bidrag til forskningsfeltet

I løpet av denne studien har vi sett på forskjellig litteratur i relasjon med agile systemutviklingsmetoder, gjort en studie rundt bruken av dette i praksis og diskutert denne praksisen opp mot litteraturen. Det vi har produsert som bidrag til forskningsfeltet er en forklaringsmodell som oppsummerer hvordan av metodene faktisk er. Modellen viser hvordan de ulike elementene påvirker hverandre og hva det er som avgjør hvordan man bruker systemutviklingsmetoder i praksis. Den er også representativ for de ulike delene av tidligere forskning som tidligere kun har vært enkeltstående elementer. Ut over dette har vi også fått frem noen elementer som ikke har blitt belyst av teorien som vi også anbefaler til videre forskning.

Litteratur

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile Software Development Methods - Review & Analysis (Vol. 478): VTT Publications.

Ambler, S. (2004). Less Is More [Electronic Version]. Retrieved 1.05.2007 from http://www.ddj.com/dept/architect/184415221.

Ambler, S. W. (2006, September 15, 2006). Agile Modeling (AM) Principles v2. *Agile Modelling* Retrieved 10. mai, 2007, from http://www.agilemodeling.com/principles.htm#TravelLight

Ambler, S. W. (2007). Agile Requirements Best Practices [Electronic Version]. Retrieved 01.05.2007 from http://www.agilemodeling.com/essays/agileRequirementsBestPractices.htm#EvolutionaryModeling.

Baum, C. H. (2005). The Defense Information Systems Agency Focuses Its Application Development on People and Technology (pp. 4). Gartner.

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001). Manifesto for Agile Software Development. Retrieved 20.03, 2007, from http://www.agilemanifesto.org/

Beedle, M., Devos, M., Sharon, Y., Schwaber, K., & Sutherland, J. (2000). SCRUM: An extension pattern language for hyperproductive software development. In N. Harrison, B. Foote & H. Rohnert (Eds.), *Pattern Languages of Program Design 4* (pp. 637-652): Addison Wesley.

Chau, T., Maurer, F., & Melnik, G. (2003). *Knowledge Sharing: Agile Methods vs. Tayloristic Methods.* Paper presented at the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03).

Cockburn, A. (2000). Selecting a projects Methodology. *IEEE software*.

Cockburn, A. (2002). *Agile sofware development*. Boston (U.S.): Pearson Education, Inc.

Cockburn, A. (2004). *Crystal Clear, A Human-Powered Methodology for small teams*. Boston, United States: Pearson Education.

Cockburn, A., & Highsmith, J. (2001a). Agile software development: The business of innovation. *Computerworld*.

Cockburn, A., & Highsmith, J. (2001b). Agile Software Development: The People Factor. *Computer, November 2001*, 3.

Davies, R. (2005). Agile Requirements [Electronic Version]. Retrieved 01.05.2007 from http://www.methodsandtools.com/archive/archive.php?id=27.

Duggan, J. (2005). Develop Your Applications Fast, but Develop Them Well (pp. 5): Gartner.

Duggan, J., Niwa, M., & Hotle, M. (2006). Development Process Frameworks and Methodologies: It Takes Two (pp. 6): Gartner.

Esteves, J., Ramos, I., & Carvalho, J. (2002). *Use of Grounded Theory in Information Systems Area: An Exploratory Analysis*. Paper presented at the European Conference on Research Methodology for Business and Management.

Fitzgerald, B. (1997). The Use of Systems Development Methodologies in Practice: A Field Study. *Information Systems Journal, 7*(3), 201–212.

Fitzgerald, B., Russo, N. L., & Stolterman, E. (2002). *Information systems development*. London: McGraw-Hill.

Goldman, S. L., Nagel, R. N., & Preiss, K. (1995). *Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer*. New York, United States: Van Nostrand Reinhold.

Group, S. (1994). The CHAOS Report [Electronic Version], 4. Retrieved 11.04.2007 from http://www1.standishgroup.com/sample_research/chaos_1994_1.php.

Hanssen, G. K., & Fægri, T. E. (2006). *Agile Customer Engagement: a Longitudinal Qualitative Case Study*. Paper presented at the ISESE'06.

Harrison, N. B., & Copllien, J. O. (1996). Patterns of Productive Software Organizations,. *Bell Labs Technical Journal, 1*(1).

Hotle, M. (2005). Set Up an AD Development Metrics Dashboard. *Gartner*, 8.

Hotle, M. (2006a). Agile Development: Fact or fiction (pp. 5): Gartner Inc.

Hotle, M. (2006b). Findings From the 'Application Development' Research Team Meeting: A Drive for Formal AD Governance (pp. 3): Gartner Inc.

Jakobsen, D. I. (2000). *Hvordan gjennomføre undersøkelser? Innføring i samfunnsviteskapelig metode*. Kristiansand, Norge: Høyskoleforlaget AS - Norwegian Academic Press.

Light, M. (2005). Agile Requirements Definition and Management Will Benefit Application Development (pp. 7). Gartner: Gartner.

Light, M. (2006). Pairing Agility With Quality: Gartner`s 10 Principles of NeoRAD (pp. 4): Gartner.

Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., et al. (2002). Empirical Findings in Agile Methods. *XP/Agile Universe 2002, LNCS 2418*.

Madsen, S., Kautz, K., & Vidgen, R. (2006). A framework for understanding how a unique and local IS development method emerges in practice. *European Journal of Information Systems, 15*, 14.

Mahnic, V., & Drnovscek, S. (2005). Agile Software Project Management with Scrum.

Martin, R. C., Newkirk, J. W., & Noss, R. S. (2003). *Agile Sofware Development - Principles, patterns and practices*. New Jersey, United States: Prentice Hall, Pearson Education Inc.

Norton, D. (2006). Agile Methods in Regulated Environments (pp. 6): Gartner.

Ordnett.no. (2007). Kunnskapsforlagets blå språk- og ordbokstjeneste. Retrieved 27.03, 2007, from http://www.ordnett.no/ordbok.html?search=agil&publications=5&publications=21&publications=9&publications=2&publications=17&publications=23&publications=1&publications=22&publications=10&publications=16&publications=8&publications=3&publications=20&publications=15&publications=19&publications=18&publications=7

Patel, C., Lycett, M., Macredie, R., & Cesare, S. d. (2006). *Perceptions of Agility and Collaboration in Software Development Practice.* Paper presented at the Hawaii International Conference on System Sciences, Hawaii.

Remenyi, D., Williams, B., Money, A., & Swartz, E. (1998). *Doing Research in Business and Management*. London: Sage publications.

Ryen, A. (2002). *Det Kvalitative Intervjuet*. Bergen: Fagbokforlaget.

Sutherland, J. (2001). Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies. *Cutter IT Journal, Vol. 14*(No. 12), 5-11.

Sutherland, J. (2004). Agile Development: Lessons Learned from the First Scrum. *Cutter Agile Project Management Advisory Service. Executive Update, Vol. 5*(No. 20).

Vinekar, V. (2006). *Towards a Theoretical Framework of Information Systems Development Strategy: The Contingent Effects of Organizational Culture and Project Uncertainty.* Paper presented at the Southern Association for Information Systems Conference.

Waters, K. (2007a). Agile Principle #8: Enough's enough! *Agile Developement Blog by Kelly Waters* Retrieved 1st mai, 2007

Waters, K. (2007b). Agile Principle #8: Enough's enough! *Agile Development Blog by Kelly Waters* Retrieved 1st mai, 2007, from http://kw-agiledevelopment.blogspot.com/2007/04/agile-principle-8-enoughs-enough.html

Williams, L., & Cockburn, A. (2003). Agile Software Developement: It's all about feedback and change. In *Agile Methods*: IEEE Computer Society.

Vedlegg

Vedlegg A – Intervjuobjekt 1

Q1: Atle Sørensen
Q2: Erlend Egeland
A1: Utvikler i bedriften
A2: Designer/ utvikler i bedriften

Q1:Hva utvikler dere for tiden?
A1: For det meste utvikler vi webløsninger og publiseringsløsninger, ved bruk av sitebar. Vi driver stort sett med publisering, i tillegg til vedlikehold av applikasjoner, nettbaserte applikasjoner for kunder.

Q1: Så dere driver med customutvikling av applikasjoner?
A1:Ja

Q1: Så det er mest nettløsninger dere driver med?
A1: Ja det er vel kun nett, men også litt utvikling, dette henger litt igjen fra gammelt av. I tillegg har vi litt vedlikehold av SMS applikasjoner.

Q1: Så kjernevirksomheten er å drive webutvikling og drive med publiseringsløsninger?
A1: Ja det vil jo vise seg, vi var veldig brede i begynnelsen, men nå prøver vi å spisse oss inn mot kun publisering og nettløsninger.

Q1: Hvor mange ansatte er dere i selskapet?
A2: Vi er 6 ansatte.

Q1: Hva slags utdanning har utviklerne hos dere?
A2: Siv.ing master i IT, Grimstad
A1: Avhopper fra bachelor i IT, og de andre har litt varierende bakgrunn.

Q1: Det vi egentlig skriver oppgave om er bruk av metoder. Bruker dere noen metoder?
A2: Mange prosjekter er veldig små, de går kun over et begrenset antall timer, og dermed blir det ikke tid til å bruke store formelle metoder. Vi har selvfølgelig prosjektplaner og sånt, men vi rekker ikke å sette oss så veldig mye ned. Vi har et prosjektdokument i begynnelsen, og så følger vi egentlig det.

Q1: Ettersom dere er såpass små blir det kanskje litt overkill med en stor formell metode?
A2: Ja, men i større prosjekt, for eksempel holder vi å avslutte et prosjekt for UDI nå, der hadde vi litt mer. Vi hadde forprosjekt, så godkjente kunden prosjektet og så satte vi i gang.

Q1: Så dere får inn et kravdokument fra kunden og så setter dere i gang og utvikler?
A2:Ja vi fikk først en kravspek fra dem, så fant vi ut det var litt for mye jobb så skalerte vi ned prosjektet og satte i gang og jobbet.

Q1: Hvor lang tid er det dette prosjektet går over?
A2: Totalt sett dreier det seg nok om 350 arbeidstimer

Q1: Jobber hele bedriften på samme prosjektet da?
A1: Vi er tre eller fire som jobber om det.

A2: Vi har litt forskjellige roller i bedriften. To designere, en involvert i UDI prosjektet. To utviklere, en daglig leder/ selger og har kunderelasjoner i forhold til forhandlere og sånt. Og sistemann er en altmuligmann.

Q1: Så UDI prosjektet er det siste relativt store prosjektet dere har hatt?
A1: Ja

Q1: Har dere noen formell metode dere følger?
A2: Egentlig ikke. Du vil neppe finne noe navn på den i litteraturen.

Q1: Så dere har ikke et formelt dokument eller rammeverk som forteller dere hvordan dere skal utvikle ting?
A1: Nei, men vi tilegner oss etter hvert arbeidsrutiner. Som vi følger, som vi kontinuerlig forbedrer

Q1: Men det står ikke dokumentert noe sted?
A1:Nei, ikke annet enn at i prosjektkontraktene står det en fremdriftsplan.. Men ikke noe formelt internt dokument.

Q1: Så dere tar det som fungerte best fra tidligere prosjekt og tilpasser dette?
A1: Ja det kan man si, men vi har utviklet oss mye innen den tiden vi har eksistert som selskap.

Q1: Hvordan var det da dere startet opp? Var det enda mer tilfeldig utviklingsmetode?
A1:Det var dårlig forprosjektering, det var gjerne sånn at man fikk prosjektet og satte i gang og kodet, og fant ut etter hvert hva kunden ville ha. Og vi snappet opp veldig mye i designprosessen.

Q1: Det er jo kjernen av hva vi skriver om. Agile utviklingsmetoder i forhold til tradisjonelle som man innhenter kravspec og så arbeider uten veldig mye interaksjon fra kunden. Så har man jo det man kaller agile metoder hvor man arbeider mer ut fra prototyping og mer interaksjon med kunden og korte iterasjoner.
A2: Hvis kunden har ubegrenset med midler så er jo dette en meget god måte å jobbe på. Vi følger den metoden opp imot andre kunder, men det er noen kunder som vi ikke fungerer som prosjektledere. Vi er leid inn som rene utviklere, og da kommer det bare noen krav som vi må utvikle og løse. Det er veldig få av de prosjektene, men det er noe vi ønsker å holde oss unna.
A1:Det ene prosjektet lagde vi det vi skulle lage som holdt på i to måneder, og så har vi hatt en test periode der kunden har kommet inn og utført akseptansetest og brukertest. Men de fikk ikke noen prototype før vi var ferdig. Vi lagde all funksjonalitet før vi slapp kunden til.

Q1: Fungerte dette bra?
A2: Ja det fungerer veldig greit. Vi har prøvd og hatt kunder inne tidligere, men da har kundene fokusert på ting som vi ikke har begynt å løse enda, selv om vi har spesifisert at dette ikke er fokusert på enda, men skal løses. Kunden oppfatter at det er mye feil i systemet.

Q1: Så det har vært unødvendig mye påpeking av feil og misforståelser?
A2: Ja det har i grunnen det, men dette har antakelig mye å gjøre med kommunikasjon. Vi har prøvd å si klart ifra at dette skal ikke fungere men allikevel kommenterer kundene det når de skal teste systemet. Det er derfor etter vår erfaring best å slippe kunden til senere i utviklingsprosessen når systemet er relativt ferdig.

Q1: Det blir jo ofte hevdet spesielt i agile metoder at det er store fordeler med å utvikle flere prototyper.
A1: Det komme kanskje veldig an på prosjektets størrelse. Hvis du har et kjempesvært prosjekt der du ikke vet hvor du vil. Kan nok kanskje dette fungere som et godt redskap for å styre prosjektet underveis.
A2:Vi har jo ikke så mange rene utviklingsprosjekter. Vi legger til funksjonalitet og tilpasser systemet. Men sånn som i et prosjekt med en medlemsdatabase vi hadde, kunne vi nok når man ser i ettertid utviklet en prototype noe tidligere.

Q1: Dette kommer kanskje også an på kundene og hvor bevisst de er på hva de ønsker av systemet?
A2: Ja, noen kunder vet ikke hva de vil ha før etter vi har laget systemet.
A1: I tillegg bytter mange kunder også mening underveis. Eller de kommuniserer noe annet enn det de faktisk vil ha. Det er veldig mange kommunikasjonsutfordringer midt oppe i andre prosesser. Vi har blant annet erfart at kundene kommuniserer et behov til oss, og vi gjør det vi kan for å løse dette behovet og gjøre det på en mest elegante metode, og så skjønner de ikke poenget med det. ”Hvorfor skal vi ha det?”. ”Fordi dere oppga at dere hadde dette behovet.” ”Men egentlig var ikke det behovet så viktig alikavel.”

Q1: Så på den måten kunne det kanskje vært en fordel og brukt en del prototyping og iterasjoner med kundene?
A1: Tja, jeg vet ikke helt hva som hadde vært mest fordelsaktig der, men det hadde nok vært viktig å skrive endringsmeldinger, og å få de godkjent av kunden. Og når de er godkjent av kunden starter utviklingen. Istedenfor å kaste seg over protoyping og koding. Nå går det kun på at har sluttet med prototyper og spontan programmering. Vi skriver forventet funksjonalitet av systemet, og når dette er godkjent av kunde setter vi i gang med utvikling. Hvis vi er flinke nok til å beskrive systemet trenger vi ikke prototyper.

Q1: Kommer det mange nye innspil etter at dere har vist kundene det ferdige produktet?
A2: Ja det gjør jo egentlig det. Da prøver vi å være klare og å forklare kunden at dette er endringer, og det er forskjell på retting og endringer. Det er ikke alltid kunden ser dette skillet.

Q1: Så det er en utfordring å definere for kundene hva som er endringer og hva som er retting av feil?
A1: Ja det er det, for selv om kunden har spesifisert en ting kommer de gjerne med nye eller forbedrede måter å løse problemet på. Så utfordringen er å vise at dette er en endring og dette er retting av feil. Jeg fikk senest en mail i dag fra en kunde som mener at systemet fungerer på feil måte, mens vi mener det fungerer slik de har definert det. Vi skjønner jo at de gjerne vil

at det skal fungere annerledes men det er ikke slik det er definert i kontrakten. Spørsmålet da er jo hvem av oss som har rett.

Q1: Så det blir en drakamp mellom dere og kunden?
A1: Nja, de har jo ikke spesifisert systemet riktig så vi prøver egentlig bare å forbedre kommunikasjonen med kundene. Kommunikasjon er nøkkelen her.

Q1: Men dette kunne kanskje ha vært oppdaga tidligere hvis dere hadde utviklet prototyper?
A1: For så vidt ikke, for vi har blitt bedt om å kopiere funksjonalitet fra et system til et annet og vi har gjort det, sånn at prototypen eksisterer allerede.
A2: Men vi kunne kanskje oppdaget det for 2måneder siden, så på den måten kunne kanskje kunden oppdaget det tidligere i prosessen.

Q1: Men sånn som dere ser det, er det litt opp til kunden da hva slags metode dere vil bruke?
A2: Det kommer veldig an på type kunde og type prosjekt. En ren webdesign, vil på mange måter være prototyping, ved at vi sender designforslag frem og tilbake. Vi kan ikke bare lage et design for kunden og si at dette er ditt design. Vi bruker en del prototyping i den prosessen.
A1: Det er egentlig kun i designprosessen vi bruker prototyping. Men der bruker vi det til det fulle. Når kunden har godkjent et design så er det spikret i sement. Da begynner vi på en prosess om å lage funksjonaliteten rundt.

Q1: Og det bryr kanskje egentlig ikke kunden seg så mye om?
A1: Nei egentlig ikke, men det er jo fortsatt en del kunder som ønsker endringer. Men som sagt da prøver vi å være flinke med å si at dette er endringer. Det var vi ikke i begynnelsen. Da fikset vi veldig mye og da tapte vi fort litt penger.

Q1: Men som du sa tidligere, hvis kunden betaler kunne dere gjerne benytte dere av agile metoder og prototyping?
A1: Ja vi kunne i grunnen det, som sagt kommer det veldig an på prosjektet.
A2: En av ulempene er kanskje at når kunden kommer til oss har ikke vi noen metode å vise til at vi bruker.

Q1: Det er jo en del firma som streber etter å bli sertifisert på en metode, og det er en del kunder som er interessert i å ha en dokumentert metode. Er dette noe dere har vært borte i?
A2: Ikke enda i hvert fall, spesielt ikke i den typen prosjekter som vi har og hvordan vi jobber. De er veldig fokusert på prosessen vår, og der har vi utviklet en god prosess. Web prosessen vår fungerer gang etter gang. Men på utviklingsprosesser har vi ikke en slik prosess.

Q1: Dere er jo for så vidt ikke så store, men det er jo mange bedrifter som sertifiserer seg i for eksempel rup eller lignende. Dette er ikke noe dere har prioritert?
A2: Vi burde kanskje brukt RUP eller en metode men det har ikke behov for dette enda. Eller på noen av prosjektene ser man mot slutten at hvis vi hadde hatt en objektmodell eller et diagram så kunne vi spart tid. Prosjektene er kanskje litt større enn hva de virker i begynnelsen. Vi har i noen prosjekter satt opp datamodeller om hvordan vi ønsker å løse det. Men det er også et aspekt å ikke lage dokumenter bare for å lage dokumenter.

Q1: Hva er fordelene med metoden eller prosessen som dere bruker?`
A2: Web løsningen, det er veldig åpent i begynnelsen, designerne tar over tidlig og spikrer designet. Når designet er spikret er veldig mye av utviklingen bundet frem til levering. Og det ser vi som en styrke, at det er bundet i et tidsrom. Vi prøvde uten dette før, og det fungerte i hvert fall ikke. Det er en fordel at når vi er enige med kunden hvordan systemet skal se ut, kan vi ”få jobbet i fred”. Med våre tidsfrister er i hvert fall dette positivt.

Q1:Hvordan fungerer løsningen med at dere sitter alle sammen i et åpent landskap?
A1: Det fungerer veldig greit, men litt frem og tilbake. I noen faser er dette en stor fordel.

Q1: Hvordan er det med kundekontakt? Er det i utgangspunktet kun daglig leder som sitter med ansvar for det?
A2: Ja det er i utgangspunktet det, han eller meg er prosjektleder og det er vi som tar imot kravspec og lignende. Det er han som tar seg av kommunikasjon med kunden og han som blir spurt hvis det er noe vi lurer på. Prosjektlederen er som regel også med i innsalget av prosjektet, så det anser vi som en fordel at kunden har en person å forholde seg til. Men når designprosessen starter har designerne direkte kontakt med kunden. Så kommer kunden inn i testperioden igjen når han skal teste ”det ferdige” produktet. Dette er både utvikler og prosjektleder som er med på. Det er viktig ettersom hvis kunden savner en ting må prosjektleder ta ansvar å sjekke om vi har gjort feil i forhold til kravspec eller om kunden ikke har spesifisert dette.

Q1: Tror dere utviklingsprosessen hadde gått raskere hvis dere hadde hatt en formell metode?
A1: På utviklingsprosjektene kunne det nok vært en fordel. I hvert fall med tanke på noen av prosjektene vi har erfart.
A2: Fordelen hadde vært at alle vi hadde visst hva vi skulle gjøre, og kunden også hadde vært inneforstått med hva vi skulle gjøre.

Q1: Altså at dere hadde noe å vise kunden hvilke fase dere var i og hva dere fokuserte på?
A2: Ja, og kunden litt mer med på utviklingen, og kunden vært litt mer åpen. Det generelle problemet med utviklingsprosjekt er tidsaspektet. Det er bortimot umulig å estimere hvor lang tid det tar å lage en løsning som vi ikke har lagd før

Q1: Hva er det kundene er fokusert på når dere forhandler en kontrakt? Er det opptatt av fastpris?
A1: Ja de aller fleste er opptatt av det, og hvis vi da går over budsjett er det ikke så lett å få ut noe mer. Selv om vi sier at det kun er et estimat. Problemet er at selv om de vil ha en fastpris, vil de også gjerne komme med noen innspill og endringer underveis, og hvis vi føyer oss etter det så går prosjektet fort over tida.

Q1: Så det er ofte et problem at kunden endrer på kravene?
A1: Ja i utviklingsprosjekt så er det det. De kommer med tilbakemeldinger og ideer hele veien. Ting de kommer på underveis som hadde vært kjekt og ha med.
A2: Det virker som de ofte har hatt det inne i hodet, men de har ikke sagt det til noen. Og ofte helt logiske ting, de har ofte tatt det for gitt at vi burde tenke på det. Men ettersom vi ikke kjenner arbeidsoppgavene deres i detalj, tenker vi ikke på disse tingene. Det er en del systemer som vi har vært med å utvikle, vi skjønner hvordan vi skal lage systemet men vi skjønner ingenting av hvordan det skal brukes. Man trenger mye fagkunnskap for å bruke

systemet. Vi hadde et møte eller flere med dem og prøvde å sette oss inn i hvordan de brukte systemet, og det er veldig nødvendig. Men når det har oppstått problemer er det vanskelig å vite om det er vi som har gjort feil eller om det var noe feil i matematikken vi fikk av dem. Et eksempel er at de brukte tallet 3,14 mens vi brukte det matematiske tegnet π.

Q1: Når slipper dere kunden fra dere?
A2: I utviklingsprosjekt har vi jo som regel en sluttfase, men som regel slipper vi aldri kunden helt. Vi har som regel en vedlikeholdsavtale, i tillegg til at mye av løsningene våre drifter vi selv på vår egen server.

Q1: Når får kunden tilgang til systemet?
A1: I testfasen kommer kunden som regel inn. Da legger de som regel inn innhold. De får opptrening av oss og tilgang slik at de kan fylle inn informasjon i systemet. I grove trekk kan man gjerne si at ca. en måned etter systemet er satt i drift er kundene kommet over i en relativt stabil driftsfase.

Q1: Du sa tidligere at det kunne være en fordel å ha en metode å vise kunden. Er dette noe kunden er opptatt av?
A2: Ikke de små kundene
A1: De vet egentlig ikke hva en utviklingsmetode er. Det er kun når det gjelder litt større kunder som har IT-utdannede folk som bryr seg om metode.

Q1: Har dere hatt noen prosjekter hvor kunden har vært opptatt av dette?
A1: Egentlig ikke. Vi prøver å starte på en plattform når vi utvikler slik at vi slipper å programmere alt fra bunnen av. Vi bygger på to plattformer, ERedaktør og Sitecore, blant annet nettbutikk er bygd der. Vi lager en basisløsning, eller OTS løsning som vi selger til andre utviklere.
A2: Vi går mer og mer vekk fra utvikling fra bunnen av, ettersom det er mange fallgruver i et slikt prosjekt.

Q1: Så større, formelle bedrifter med IT avdelinger er interessert i utviklingsmetodene deres.
A2: Egentlig er ikke de så interessert heller, men kanskje indirekte. I kontrakten så står det jo en fremdriftsplan, og selv om man ikke kan sette et metodenavn på den så er vel det egentlig metoden vår.

Q1: Så dere har for så vidt en metode da?
A1: Tja hvis man kan navnsette den fremdriftsplanen så har vi nok det.
A2: Vi har jo metode, men det er en hjemmesnekra metode, eller ”best practice metode”. Det har nok noe med hvor mange ansatte vi er, og hvilke typer personer som er med underveis. Men sånn som størrelsen og rollene fungerer nå holder den uformelle metoden vi har nå. Alle vet hvem som gjør hva, vi er på en måte en veldig rollebasert bedrift.

Q2: Hvis dere hadde fått et krav fra en kunde om å bruke en viss metode, hadde det vært noe problem?
A1: Nei egentlig ikke, vi måtte satt prosjektet etter det selvfølgelig. Hvis det var en ny metode vi måtte heve kompetansen på, måtte vi beregne inn de kostnadene.

Vedlegg B – Intervjuobjekt 2

A1: Atle Sørensen
Q1: Representant fra bedriften

Først om utviklingsavdelingen i (…). I mai i fjor begynte vi, da var vi en person, meg. Nå er vi fem utviklere, og to implementeringsteam worldwide. Som implementerer de programmene vi har lagd. Mitt fokus har hele tiden vært en stor kunde, men vi har flere store kunder, blant annet SAS og National Oilwell.

Presentasjonen går ut på hvordan vi har utviklet oss siden mai i fjor. Noen tunge og noen andre ting vi har lært på veien. Det første jeg kan si er at vi bruker ikke et fast oppsett på en utviklingsmetode, men vi har siden de fleste av oss kommer rett fra skolen, en god ide om hva utviklingsmetoder er og hvordan de bør brukes. Så vi har valgt å holde oss til de utviklingsmetodene som er agile metoder og er tilpassningsdyktige. Så det vi gjør er at vi bruker agile prosesser som betyr at vi kan konfigurere utviklingsprosessen til hvert enkelt prosjekt. Vi tar litt av det vi ser fungerer pluss at vi tilpasser konfigurasjonen til det nye prosjektet, avhengig av hva vi som utviklingsorganisasjon kan sette inn i prosjektet. Det som er viktig for oss er spesielt for en av våre største kunder, er at vi har høy synlighet inn i prosjektet vårt. Sånn at vi veldig fort produserer resultater til kunden som han kan forholde seg til. Det er noe som vi prøver å forbedre oss på hver eneste gang, altså prototyping. Vi sitter faktisk ute hos kunden og forsøker å gjøre det som overhodet mulig. Så fort vi har noe, ut til kunden å diskutere skjermbilder. Helst hver uke. Det er spesielt viktig i webutviklingsprosjektene våre, siden dette er noe som skal representere kunden på internett, enten til samarbeidspartnere, underleverandører eller kunder. Derfor er det viktig at kunden kjenner seg igjen i utformingen. Vi har veldig mye diskusjon rundt hvordan ting skal se ut. Hvor knapper skal stå og lignende. Det må vi ta tidlig, siden web programmering fungerer slik at du lager et oppsett og går hjem og programmerer og kommer ut til kunden med ferdig produkt. Vi har også fokus at alle regler, forretningsregler, skal være veldig tidlig klart. Det er noe som vi gjerne skulle sagt at sånn er det, men det er det ikke. Det er en prosess hvor vi forbedrer oss hele tiden.

Jeg har kalt fasene våre for insertion, elaboration, construction og deployment, og det er RUP faser. Vi skjeler veldig til RUP, når vi bygger prosessene våre. Uten at vi nødvendigvis følger RUP. Vi tilpasser det til vår organisasjon og det er sånn RUP skal brukes. Vi er for få mennesker i utviklingsteamet, så en full implementasjon av RUP ville blitt for mye. I starten så har vi et stort fokus på GUI, og hvordan driver forretningen egentlig med, og hva er det de ønsker vi skal løse. Det er vår måte å kommunisere at vi har skjønt hva de prater om det er tidlig å komme opp med GUI, slik at de kan prøve softwaren.

Q1: Men i denne fasen er det minimalt med funksjonalitet?
A1: Minst mulig, men i web-verden må man ha en viss funksjonalitet bak. Slik at hvis man for eksempel skulle få en feilmelding så kommer denne opp.

I starten er det ofte slik, spesielt i store prosjekt, så ombestemmer kunden seg en 50-60 ganger før de faktisk kommer frem til det de vil ha. Vi har faktisk statistikk på dette. Man kan komme opp i nesten 200 unike endringsmeldinger. Sånn at det å fokusere på hva vi skal løse og skjønne hovedfunksjonaliteten er viktig tidlig. Alle disse små ønskene noterer vi på små lapper eller i et system (team foundation server fra Microsoft). Når vi så går videre i mer detalj, gjennom prototyper som er laget går vi igjennom kravene med kunden. Dette er en

kontinuerlig prosess. Så går vi også igjennom selve GUI. Så begynner vi å designe selve applikasjonsarkitekturen. For web-applikasjoner er dette tålig likt., mens for det som har med integrasjon er det ikke likt. For du vet ikke på forhånd hvilke system må jeg samarbeide med hos kunden.

Da må vi fokusere på at vi har et sett med data, hvor kommer dataen fra og hva skal til for at vi kan gjenta prosessen for ny data. Og vi bygger applikasjoner på en sånn måte at vi ønsker en kontinuerlig inndatastrøm, og kun lagrer data en gang. Vi er redd for å ta for eksempel et ERP systems data, lagre de og så manipulere de. Da går vi heller til et datavarehus og henter det derfra.

Q1: Så dere fokuserer veldig på integrasjon?
A1: Ja det er veldig viktig for oss, siden vi ikke genererer noen data selv vi lager program som tar kundens data og representerer de på en annen måte. Det er stort sett det vi gjør. Vi lager prosjektverktøy og beslutningsstøtteverktøy. Og det vi gjør er å lese alle grunndata og representere data. Og så har vi et system, i oljebransjen i Norge, regnes det som fremdriftsbasert, vi har en web basert program som gjør at man kan registrere hvor langt man har kommet i sine prosjekter på bakgrunn av våre data. Vi utviklet dette fra mail til august i fjor, og drifter dette videre. Samtidig har vi også arbeidet med å få ut en modul to av dette programmet. Nå arbeider vi mot underleverandørene til National Oilwell med samme fremdriftsløsninger.

I konstruksjonsfasen som vi har ”stålet” fra RUP så er det da å få implementert alt etter spec. Og til og med her har vi gjentatte ganger funksjonstest, for å se om vi leverer det vi skal. Og det kommer noen endringer fra kundene. Det gjør det ofte, det er bare unntaksvis at det ikke kommer endringer.

Q1: Så det er faktisk helt frem til testfasen at det kommer endringer?
A1: Ja, sånn er egentlig hele bransjen. Alltid nye ting kommer inn.

Q1: Så det er kanskje en av fordelene med å bruke agile metoder?
A1:Ja det er det, hvis vi hadde brukt fossefallsmetoden hadde det gått rett til skogen.

Q1: Hva er hadde vært det verste med fossefallsmetoden? Kravene?
A1: Aldri kunnet oppfylt kravene siden de endrer seg hele tiden. Man måtte bygd et helt nytt system og så kastet det og så bygd et helt nytt system og kastet det. Nå kan vi kanskje lage to websider og viderebruke 75 % av det.

Og så har vi der vi har mest å lære testfasen. Siden jeg begynte her har vi kontinuerlig jobbet med å forbedre oss. Og det må vi bare innrømme at vi ikke er gode nok på det området. Men nå har vi rutiner for testing som kjøres, slik at vi har en sjanse til å oppdage problemer før de oppstår

Q1: Men dere er vel ikke alene om denne utfordringen?
A1: Nei men samtidig er vi såpass små fremdeles at vi har en sjanse til å gjøre noe med det. De store guttene i bransjen har ikke den muligheten. Vi har et prosjektoppsett som er allergisk mot feil. Og da må vi teste utviklingsteamet før vi leverer produktet fra oss. Men vi er blitt mye bedre på dette området. I prosentvis har vi økt med 75 % i feilretting.

Når det kommer til requirements så forsvinner jeg for så vidt ut av utviklingsteamet. Vi står kun for den tekniske biten, så er det noen andre som utvikler brukermanualer. Dette fungerer veldig bra.

Q1: Denne metoden er noe dere har utviklet selv?
A1: Ja denne er selvutviklet.

Q1: Så det er ikke noen utgifter rundt lisenser eller lignende? Eventuelt opplæringsutgifter?
A1: Vi har jo for så vidt opplæringsutgifter ettersom vi lærer noe nytt hver dag. Sånn sett vil vi alltid ha opplæringsutgifter. Men om det skal tilskrives metoden eller erfaring er et annet spørsmål. Ellers er dette ting som jeg har lært på skolen gjennom studiet så sånn sett er det ikke noen utgifter for bedriften. (eventuelt mitt studielån).

Q1: Men for nye ansatte, er det enkelt å komme inn her og lære seg og sette seg inn i metoden?
A1: Ja det er ikke noe problem, ettersom de fleste kommer fra skolen og kan en del av disse tingene.

Q1: Men har dere noe formelt dokument som definerer hva dere gjør i hver fase?
A1: Vi har en slags mal og så tilpasser vi den. Den malen er veldig enkel, og vi legger til og trekker fra til hvert enkelt prosjekt. (se slides).

Q1: Hva er de viktigste styrkene til deres metode?
A1: Tilpassningsdyktighet, absolutt. Det at vi ikke får store utgifter hvis kunden kommer med forandringer. Svakheten er kunden ombestemmer seg prosjektet flytter etter, og utviklingsteamet løper etter. Så det er den pendelen, en liten forandring på toppen blir en stor forandring i bunnen.

Q1: Sånn som for utviklerne, hva er viktig for dem å tenke på når de bruker metoden?
A1: Det er det de gjør er et håndverk, det er ikke en kunst for det skal gjentaes flere ganger. Det er ikke noe man kan kjøpe i butikken. Og det er viktig at man har fokus på å bli bedre i jobben sin og forbedre prosessen.

Q1: Spesifikt for deres metoder er det noen hinder her?
A1: Hvis man kommer fra HiA som en av våre nyansatte gjør så trenger man ikke noe mer forkunnskap enn det. Det er viktig at det skal være intuitivt og enkelt å sette seg inn i.

Q1: Dere bruker da denne metoden som et rammeverk? Det er ikke noe som følges slavisk?
A1:Den blir et rammeverk i utviklingsteamet.

Q1: Det er jo en mulighet når man bruker agile metoder at det blir tatt noen snarveier, med tanke på dokumentasjon.
A1: Ja, men det kan ikke vi for vi har et implementeringskrav fra kunden som må følges. For vi bruker UML for å dokumentere bruksmønster og tekst i tillegg. Hadde vi fulgt fossefallmetoden hadde dette vært en papirmølle, og det kommer det aldri til å bli.

Q1: Hvordan hadde det vært å bruke fossefallsmetoden i deres bedrift? Hadde dette fungert?
A1: Nei, fossefallsmetoden er basert på å sette kravene ca. en gang. Men når kravene endrer seg hver dag opptil flere ganger da går det ikke an å bruke fossefallsmetoden. Da hadde vi tapt for lenge siden. Fossefallsmetoden er en metode som gjør at man har en veldig rigid prosess man skal igjennom, og vi er en dynamisk gjeng som synes det kan være greit å forandre på ting hvis de ikke funker. Det er heller ikke så lett i fossefallsmetoden. Fordelen med vår metode er at vi kan lære underveis.

Q1: inspirasjonen til metoden var RUP sa du. Er det noen andre metoder som har vært vurdert eller som har hatt innvirkning på deres metode?
A1: Jeg har gått på høgskolen og kjenner til de andre alternativene av metoder. Jeg har brukt RUP i tidligere stillinger som utvikler og det har fungert veldig greit. Jeg vet at dette fungerer for andre og at det er utprøvd. ”Why change a winning team”. Men jeg har jo tatt ting fra andre metoder og, som har passet inn.

Q1: Kan du se noen ulemper eller aspekt som kan videreutvikles i deres metode?
A1: Det er vanskelig å kommunisere UML til prosjektorienterte mennesker. Som ikke har samme innsikten i dokumentstrukturen. I tillegg til å produsere UML, skriver vi en del tekst. Derfor er det fortsatt en utfordring å kommunisere eksternt ut mot kunder og prosjektdeltakere i motsetning til hvordan man kommuniserer innad i utviklingsteamet. Dette kan muligens begrunnes med forståelsen av UML og delvis på grunn av svakheter i presentasjonen og kanskje UML ikke er så intuitivt.

Q1: Hva slags type program er det dere utvikler? Skreddersøm eller hyllevare?
A1: Webapplikasjoner slik som vi tilbyr må spesialtilpasses til hver bedrift. Samtidig finnes det en generisk bunn i det.

Q1: Har kunden noen oppfattelse eller innvirkning på utviklingsmetode?
A1: Spesielt de som har et eget utviklingsteam internt, så de ser nok litt på hvordan vi jobber.

Q1: Så de har en viss formening om hvordan prosessen bør være?
A1: De følger i hvert fall med på hvordan vi jobber.

Q1: Så det er en av grunnene til at dere lar kunden få innsyn? Det hadde kanskje vært annerledes hvis kunden ikke hadde hatt peiling på systemutvikling?
A1: Egentlig ikke, det har med at kunden skal føle seg trygg på at vi kan jobben vår. Helt ned til den personlige smertegrensen prøver vi å slippe kundene inn.

Q1: Men hvis kunden ikke hadde noen oppfatning av utviklingsmetode så hadde de kanskje ikke brydd seg så mye?
A1: Vi har for så vidt ingen kunder som ikke har en oppfatning av dette.

Q1: Hvordan er det med sertifisering mot metode, er det noe dere vurderer?
A1: Vi jobber med å sertifisere oss teknologisk, men vi kommer aldri til å sertifisere oss mot RUP eller ISO eller noe lignende. Det har vi ikke behov for. Det vil ikke gi oss noe verdi tilbake og det tar for lang tid. Vi er ikke en slik type bedrift. Vi lager ikke dokumenter som ingen leser. Vi har heller ikke støtt på dette fra kunder.

Q1: Den formelle metoden som du viste oss, er det den samme som faktisk blir brukt?
A1: Ja den presentasjonen lagde jeg tidligere i dag, så det er den faktiske metoden.

Q1: Så dere har ikke stor dokumentasjon rundt metoden?
A1: Vi har en prosessmal som ligger på vår team foundation server som vi bruker i utviklingen. Disse er helt like, dette pga vi tilpasser oss metoden i tillegg til at vi tilpasser metoden til oss. Denne modellen oppdateres kontinuerlig.

Q1: Tror du det at dere er såpass små, gjør at dere kan bruke en agil metode bedre enn hvis dere hadde vært store?
A1:Tja, som eksempel brukte Agresso når jeg jobbet der samme metode og de var over 200 utviklere, så det er mulig å bruke agile metoder i store bedrifter.

Vedlegg C – Intervjuobjekt 3 (team 1 - serverprogramvare)

Q1: Atle Sørensen
A1: Representant fra team 1 (MSF for Agile) hos bedriften

Vi utvikler i utgangspunket server software, som man ikke har så ofte releaser på.

Q1: Er dette hyllevare eller er det skreddersøm til bedrifter?
A1: Det er for så vidt hyllevare, men med modifikasjoner til kunder.

Q1: Hva er din stilling?
A1: Jeg er utvikler og teamsjef for en utviklingsgruppe og ansvarlig for produktene.

Q1: Hva slags bakgrunn har deres utviklere?
A1: Master på alle sammen. Vi har noen i India som har bachelorbakgrunn, men mest mastere.

Q1: Hva slags metoder bruker dere?
A1: Tidligere brukte vi en litt basert på en software metode som hadde litt tilknytning til fossefallsmetoden. Men rett før nyttår gikk vi over til .NET 2.0 og visual studio team system. Og der er det innebygd støtte for diverse metoder. Og da valgte vi MSF for agile.

Q1: Så metoden ble egentlig valgt på bakgrunn av utviklingsverktøyet?
A1: Ja, blant de alternativene som fantes der følte vi at den var best for oss. Det er en blanding av fossefallsmetode og spiralmetode. Innenfor hvert segment eller fossefallsstep så kjører vi en spiral med arkitektur, development og testing. Når man har oppnådd det man ønsker går man videre til neste iterasjon. Istedenfor å kjøre fossefallsmodellen helt tradisjonelt.

Q1: Så dere har iterasjoner innenfor hvert steg av fossefallsstegene?
A1: En for hver iterasjon som det kalles.

Q1: Hvor mange utviklere er det som jobber innen ditt team på samme prosjekt?
A1: Vi er fire her på huset og fire i India. I tillegg kommer det inn en ny person som skal teste, han begynner nå til mandag.

Q1: Hva slags kunder er det dere har?
A1: Statoil, Hydro, PMX osv.

Q1: Har dere hatt noen utgifter med tanke på lisenser eller opplæring på metoden deres?
A1: Vi har selvfølgelig hatt utgifter med å innføre den, men lisensen følger med .NET. Den er bare en plug-in i .NET. Det fulgte med team system serveren. Og her når man har prosjekt må man velge en prosjekt ”template” som selve utviklingsprosessen skal foregå under. Der kan man kjøpe inn ulike plugins for ulike metoder. Men den som fulgte med team system var egentlig veldig grei.

Q1: Så dere kan i utgangspunktet bruker hvilken metode dere ønsker?
A1: Ja det følger med to default metoder. MSF for agile og MSF for CMMI. Det andre prosjektet på huset her har lagt inn en plugin som bruker SCRUM.

Q1: Dette er noe dere kjører fast på alle prosjekt? Dere har samme metode på alle prosjekt?
A1: Vi har samme metode for alle prosjekt som kommer inn under de produktene jeg jobber med. Så vi har valgt den metoden for alle prosjekter ja.

Q1: Er det på grunn av å slippe opplæringskostnader?
A1: Ja, da har man hver prosess har sin metode å rapportere bugs, forskjellige exit og entry pointer og forskjellig grad av dokumentasjon. Så hvis du velger en ny metode på et nytt prosjekt så må man forholde seg til så mange forskjellige prosesstandarder og det er lite hensiktsmessig.

Q1: Hvordan er det med dokumentasjon rundt metoden? Er det veldig mye formell A1: dokumentasjon eller er det mye som dere har tilpasset?
Metoden i seg selv er nok hvis man skal sammenligne med SCRUM, så valgte jeg den fordi det er en litt tyngre prosess, litt mer dokumentasjon og det kan vi tillate oss, siden vi har en lengre syklus for hver release. Og jeg syns også at dokumentasjon har vært et problem tidligere og da kan vi heller velge en metode som tvinger oss til å være litt flinkere med dokumentasjon.

Q1: Det er kanskje en av fordelene med metoden, at det er tilstrekkelig med dokumentasjon?
A1: Jeg syns det ja, det går an å modifisere disse prosessene, og jeg har gjort det. Jeg har lagt til egne items. For eksempel, Microsoft har definert hva en bug og hvilke tilstander disse kan ha. Og der har vi vært inne og endret slik at den passer vår metode. Det er ikke en ren MSF metode, men MSF er basisen og så har vi gjort en del endringer.

Q1: Hva er den største styrken til metoden?
A1: Fleksibiliteten i forbindelse med prosessen, man får veldig tidlig en god oversikt. Vi har en prosess hvor vi har design, implementasjon og test. Før du går til neste iterasjon. Når man er ferdig med en iterasjon så kan man komme med korrigeringer.

Q1: Produserer dere en prototype tidlig i prosessen da?
A1: Vi prøver å få opp et skall så tidlig som mulig. Ut ifra det gjør vi korreksjoner. Samtidig er den formell nok, uten at den er plagsom. Etter erfaring så blir mange store metoder for mye byråkrati.

Q1: Hvis dere hadde vært flere i teamet, hadde denne metoden hatt for lite dokumentasjon og administrasjon?
A1: Ja den hadde nok det, når vi er såpass små, kan vi ha daglig kontakt med alle i teamet. Det letter administrasjonen betydelig. En wild guess fra min side så vil kanskje metoden bli litt for liten når man begynner å nærme seg 50 personer.

Q1: Hva slags rolle spiller metoden i utviklingen? Er den et rammeverk eller følges den veldig rigid?
A1: I og med at prosessen i TFS er såpass knyttet opp mot alt annet i TFS. Tidligere kjørte vi visual studio for utvikling og visual source for bugs og dokumenter og excel ark for dokumentasjon. Nå kjører vi alt i samme system. Så gjennom utviklingsverktøyet kan man gå rett i metoden og se hvor er jeg i landskapet. Man kan få opp alle bugsene dine og fremtidige oppgaver. Dette er veldig raskt og enkelt, i tillegg er knytningen mot utviklingsverktøyet og

prosessen veldig tett. Dette er en veldig fordel, jeg husker hva slags ”Hiroshima” vi hadde her før. Informasjonen er lettere tilgjengelig.

Q1: Men med denne metoden er man nødt til å følge den rigid steg for steg?
A1: Nja, man kan legge det opp sånn. Derfor kan man tilpasse metoden til et større team, med rigide regler. Nå har ikke vi gjort det her, eller vi har en del regler sånn som at vi tvinger folk til å ved gjennomgåelse av kode må det være knyttet opp til en bug eller lignende. Dette er viktig for å dokumentere endringer. Så utviklingsverktøyet støtter opp om at metoden blir fulgt korrekt. Man kan huke av diverse features, slik at reglene blir enda mer rigide. Vi har for så vidt slike regler men det er mulig å omgå reglene våre, men det er noe jeg ikke har fortalt til så mange av utviklerne.

Q1: Oppsett av regler og lignende kommer vel litt an på utviklerne, om de har vært med på dette før og hvordan de arbeider?
A1: Nei, men det kommer an på hvordan man arbeider og. Slik som her på huset er det fordelaktig at vi sitter ved siden av hverandre. Mens hvis man er utenfor kontoret, må man følge mye mer opp. De har heller ikke mulighet til å sjekke inn og ut kode.

Q1: Så hver release er ca. 6 måneder? Og da følger dere en fossefallsmetode mellom hver release?
A1: Mellom hver release går vi igjennom et visst antall iterasjoner. De forskjellige delene design og lignende. Og så kjører vi den syklusen en gang, og da er vi ferdig med den iterasjonen. Etter dette kommer en intern release som man kan teste på. Og som man kan si hvor langt man er kommet. Mellom hver release har vi kanskje mellom tre og fem iterasjoner.

Q1: Er det noe du savner i metoden? Noe metoden kunne vært bedre på?
A1: Tja, det er litt sånn ubekvemmelige ting. Sånn som man skal sette opp en liste over personer som er med i teamet. Det er irrelevant for oss, det er overhead. Man skal lage visjoner for dette produktet. Ulempen er at har man først valgt en prosess må man følge denne prosessen. Man kan ikke velge en ny prosess underveis i utviklingen. Da må man bytte ut all koden og opprette alle prosjektene på nytt. Det er for så vidt ikke en feil i metoden, men i hvordan Microsoft har implementert disse metodene.

Q1: Når dere arbeider mot kunder, er de interessert i at dere følger en formell metode?
A1: Ja, noen etterspør det, noen vil kanskje kreve det. Vi har akkurat blitt kjøpt opp av kongsberggruppen, og de var veldig interessert i hvordan vi brukte metoder. Og vi fikk full score på den biten. Det forteler noe om hvor modne utviklerne er hvor god kontroll har du på kildekoden din, hvor raskt går det fra man melder fra en bug til den blir fikset, har man kontroll over de feil som finnes i systemet. Noen kunder etterspør det helt konkret, kanskje ikke absolutt hva slags metode vi bruker, men at vi har prosedyrer og retningslinjer for sånne ting. Men de spør ikke om man har brukt den eller den metoden, de spør om vi har en formell prosess.

Q1: Er dette en motivasjon for å bruke en såpass kjent og dokumentert metode fra Microsoft, i forhold til å lage en egen metode?
A1: Jeg mener at om jeg kaller det en metode eller hva slags navn metoden har syns ikke jeg er viktig. Jeg syns det viktigste er at man har en metode, selv om kanskje ikke den metoden er så god, enn at man ikke har det. Om man kaller den fossefallsmetoden, eller spiral eller hva som helst, det synes jeg er irrelevant.

Q1: Men er det relevant for kundene at metoden heter noe spesifikt eller er det kun prosessen de er ute etter?
A1: Det viktigste er at man har en prosess rundt det, og at man selv er klar over det. Så hadde noen kommet og sagt at bruk den eller den metoden syns jeg det hadde vært helt greit. Så lenge ikke det hadde blitt for store og tunge metoder.

Q1: Vi har fått inntrykk av når vi har snakket med andre utviklere som har kunder som ikke er så datakyndige. Og disse kundene er ofte lite interessert i prosessen rundt uviklingen.
A1: Ja de store har jo egen IT-avdeling som ofte går mer inn og ser på hvordan vi arbeider. De er opptatt at man kan dokumentere prosesser for feil og lignende.

Q1: Sitter dere mye sammen med kunder når dere utvikler?
A1: Vi får ikke noen kravspec, noen kunder har diverse krav. Da setter vi oss ned og diskuterer kravene og setter opp ei liste. Det er ikke alltid kundene helt skjønner hva de ønsker. Men det er sjeldent vi jobber direkte mot kunder når det gjelder utvikling. Vi ønsker at produktet skal være hyllevare. Da legger vi heller endringer som ad-ons slik at andre kunder ikke blir påvirket av endringene.

Jeg vil heller bruke en metode som ikke er kjent og passer meg enn en metode som ikke passer meg. Hva metoden heter bryr ikke meg. Dessuten har jeg inntrykk av at det er mange som har aversjon mot for eksempel XP og andre metoder, ettersom de har hørt forskjellige ting. Det er nok også forskjell på privat og offentlig sektor. Jeg snakket med noen av de i kongsberggruppen som kjøpte oss opp og leverte software til forsvaret, og der var de veldig opptatt av ISO sertifiseringer og formelle metoder. Vi har selv vært innom det men det blir for rigid prosess for oss og for mye arbeid så det er ikke aktuelt. Selv om man er sertifisert leverer man software med feil i, det er ikke noe å gjøre med det. Spørsmålet da er heller hva slags prosesser har man for å rette disse feilene.

Q1: Så det er viktig i metoden å ha en prosess for tilbakemelding av feil?
A1: Ja, nå rapporterer vi som regel flest feil selv, men vi får også tilbakemeldinger fra kunder, for tiden jobber vi med et system hvor kunder kan melde inn feil via internett. Og følge feilene og se hvordan feilrettingen går. Dette gjør vi slik at kunden skal føle seg litt mer integrert i prosessen.

Q1: Med tanke på design og slikt, tester dere dette ut mot kunder?
A1: Vi jobber mot en standard som er en offentlig standard. Vi har vært så heldig at det vi har utviklet, vi har vært så raskt ute med det at ofte har det blitt standarden i bransjen. I vår standard er alt definert, vi må følge det som står i den standarden og så finner vi en god måte for å implementere dette. Og så har vi snakket med andre firma som driver med utvikling av slik software, i tillegg har vi en offentlig server på nett som alle firma har tilgang til å teste på. I og med at vi har vært tidlig ute og vært såpass dyktige så følger de andre utviklingsfirmaene opp denne løsningen. I og med at kundene våre har vært med å lage den offentlige standarden har de på en måte sagt hva de vil ha på forhånd. Fordi det er sånn oljeindustrien funger, det er på en måte ikke så mange krumspring man kan gjøre. Mens de små kundene må som regel bare følge opp det de store firmaene gjør. De er fornøyd så lenge de kan samarbeide med de store firmaene.

Q1: Du sa tidligere at det er mange kunder som egentlig ikke vet hva de vil ha før de får det. Får dere mange tilbakemeldinger når dere har en ferdig release?
A1: Ja det skjer jo, hvis det er noe vi ser er fornuftig så legger vi det inn i hyllevaren til neste release. Men vi selger ikke et produkt og så går vi, mange av våre kontrakter går på at selskapene leier lisenser på våre produkt. Og vi har kontakt med dem hele tiden i lange tider.

Små selskaper som har fem ansatte, gjør ting og så har de det i hodet. Hvis det er noe de lurer på vet de hvem de skal snakke med. Sånn starter de fleste selskap. Men så kommer det flere og flere utviklere, det blir lengre og lengre siden man utviklet en del av produktet. Og så går det i glemmeboka. I små selskaper dokumenterer man lite og tester lite. Det er sammenheng mellom dokumentasjon og testing. Er det riktig eller er det galt når testen blir kjørt. Sjekk dokumentasjonen, men da finnes det ikke dokumentasjon. Men hvordan funker det da, det husker jeg ikke, da må vi bli flinkere til å dokumentere. Det er en syklus. Den prosessen tror jeg de fleste firma går igjennom før de begynner å bruke en formell prosess.

For eksempel kjøpte vi opp et firma for et år siden. De satt og rapporterte feil i notepad, dette funker veldig bra hvis man er en eller to personer, men før eller senere må det komme en formell prosess.

RUP er en veldig rigid metode, man innfører ikke den for fire fem personer.

Vedlegg D – Intervjuobjekt 3 (team 2 - klientprogramvare)

Q1: Atle Sørensen
A1: Representant fra team 2 (scrum) hos bedriften

Q1: Du i forhold til han andre, er du på et annet team?
A1: Ja, jeg er på et annet team. Hvor vi utvikler klientprogramvare for boreriggene. Så er jeg teamleader for et av produktene innen det og arkitekt for et annet. For det har vi valgt å bruke Scrum som metode.

Q1: Hvor mange er dere i ditt team?
A1: På Discovery-teamet er vi 5 her, 1,5 i Oslo og en 5-6 i India. Men på det produktet jeg jobber på nå er det bare meg og 1,5 i Oslo.

Q1: Dere utvikler programvare for klientene?
A1: Mest Windows-basert programvare, litt web-basert for å koble seg opp mot web-services, så vi gjør ingen databaser.

Q1: Du nevnte at dere bruker Scrum. Hva var grunnen til at dere valgte det, i motsetning til det andre teamet som har valgt en annen metode?
A1: Jeg har erfaring med Scrum fra forrige arbeidsplass, og implementert det der, og så hva de hadde her da jeg begynte her. Det var bare hips om happ, med en egen form for prosess/metode, med en del ferdige dokumenter som de fylte ut. Så da tenkte jeg å lansere Scrum for å få litt mer struktur og litt hyppigere lanseringer. Det var også fordi vi introduserte Team Foundation Server for et år siden, ca når jeg begynte her. Så var det først mitt team som kom inn på dette og det andre teamet kom i gang en måned etterpå.

Det var opp til oss hva slags metode vi ville bruke og han andre følte at Scrum var litt for løst med tanke på spesifikasjon og slikt. For de jobber veldig mye med folk i India, og det er det jo greit å ha ting litt mer spesifisert, men nå jobber jo vi også med folk i India og har ikke hatt noe problemer med det vi.

Q1: Hvor ofte har dere da en leveranse?
A1: Gjennomsnittlig har de sprintene vi har hatt nå vart i ca. 1,5 måned, på det produktet som jeg jobber på. Discovery Wells har hatt litt problemer med å komme med en release så ofte, så de har hatt noe lengre sprinter. Det har noe med at det er et mye større produkt, for det produktet jeg jobber på har vi ikke hatt en kunderelease på enda som vi har utviklet på i et år nå. Vi har jo review av produktet med funksjonalitet osv, internt og vi skal ha lansering nå rundt påske. Så vi er inne i siste sprint og våre folk er hos kunden og installerer i deres nettverk. Vi har kjørt en presentasjon, men produktet har ikke vært noe vi kan selge, det er bare en ekstragode til kunden.

Q1: Så dere har ikke mye arbeid mot kunden, ved at dere viser dem prototyper for eksempel?
A1: Nei, sånn som vi jobber nå så har vi produkteier som spesifiserer produktloggen, nå er jeg produkteier delvis på dette og en annen en som har erfaring fra salg og kjenner bransjen er dette og så er det han som er sjefen her (oss tre er produkteier på dette prosjektet). Han fra salg har hovedansvar med hvilke features som har høyest prioritet og sånn har vi gjort helt frem til nå og før vi starter en sprint tar vi et planleggingsmøte og samler teamet, ser på hva som de skal fokusere på i neste sprint og hva slags tilbakemelding vi har fått. Ikke

tilbakemelding fra kunde når vi har hatt lite kontakt med de nå, men regner med det kommer etter hvert.

Q1: Begynte dere ut fra en kravspek dere fikk eller er dette noe dere har funnet på selv?
A1: Det vi har gjort kommer ut i fra noen skjermbilder og noen sketsjer og sånn. I tillegg en visjon og en oversikt over oljerigger og plattformer. Det har vært det primære, så det har vært ganske løst med tanke på spesifikasjoner og det har vært veldig bra i forhold til den Scrum-prosessen. Der holder vi fokus på en måned av gangen, en måned går så presenterer vi og så finner vi ut at neste måned er det noe helt annet vi skal ha fokus på. Sånn har Scrum fungert veldig greit, for da har du noe ferdig noe i løpet av 1,5 måneder, og kan endre fokus.

Q1: Men, vil du da si det er fordelen, eller en av de sterkeste egenskapene til Scrum? Det at du får lever noe såpass kjapt.
A1: At du får levert kjapt, og vist det til kunde. Det er veldig viktig, fordi det er mange prosjekter som feiler pga det tar et halvt år til et år før kunden får se noe, og så var det ikke det som var det de skulle ha og da går det et nytt år for å gjøre endringer og da ender det med å bli kansellert pga endringer i behov.

En annen god egenskap med Scrum er at alle i teamet blir mye mer sammensmeltet. Ved at vi kjører et morgenmøte hvor alle rapporterer og går igjennom hva de skal gjøre og kan diskuterer hvis man har sittet fast på et problem i en dag. Kan da diskutere i plenum med en gang i stedet for å sitte på det, i forhold til når man har utviklere som er lite åpne som gjerne kan sitte på et problem i en uke uten å gå og spørre noen om hjelp. Man ”tvinger” utviklere til å snakke, for det er ikke alltid at utviklere er så utadvent.

Q1: Dette med at det var litt lite dokumentasjon, har det vært tilfredsstillende?
A1: Nja, det har vært svært lite dokumentasjon på akkurat dette prosjektet her pga vi har hatt mer fokus på å koke det ned og få fokus på funksjonalitet. Når vi nå er ferdig med versjon 1 vil vi notere ned hva vi har gjort, og spesifisere de nye tingene, men Scrum dikterer ingenting om dokumentasjon. Det er en struktur og vi har en del maler, templates som vi fyller ut, men vi heller litt over på en innføring av XP i tillegg til Scrum, for denne sier litt mer. Noe som fungerer kjempefint sammen, men XP har et litt mer negativt rykte pga navnet og mange tror kanskje det er ”cowboy-programmering”. Scrum har litt få regler, men fokuserer på å få ting bedre og bedre og bedre.

Q1: Utgifter med metoden har vært mest i form av opplæring?
A1: Vi har ingen sertifiserte scrum-masters her, men skulle gjerne ha hatt et slikt kurs i London. Vi har ikke hatt noen særlige utgifter med det for det har vært veldig greit og raskt å komme i gang med det. Som sagt er det ikke en svært komplisert greie og vi hadde ikke i firmaet en konkret utviklingsmetode i utgangspunktet så det som er vanskeligst er å få alle utviklerne til å oppdatere arbeidsoppgavene sine hver dag. Det er en del rapporter i scrum som blir veldig ubrukelige dersom ikke utviklerne oppdaterer disse. For eksempel hvis du kommer en mandag og ser det er 50 timer igjen på et prosjekt og ser etter to dager så er det fortsatt 50 timer så sender du en mail og plutselig er det 20 timer igjen. Det er en oppgave for meg som jobber sammen med folk i Oslo, for vi kan ikke ha morramøte sammen, så da blir det mange samtaler på skype.

Q1: Det er vel egentlig et krav i scrum, at man skal møtes ansikt til ansikt. Hvordan fungerer det å jobbe uten dette?
A1: Det er en utfordring og scrum krever også sterke individer. Mange utviklere gjør bare det blir fortalt og ikke mer, så må du purre fire fem ganger for å få gjort noe. Det er greit da å ha folk som kan tenke litt selv, ikke bare blir fortalt, for eksempel å komme frem til design og tekst og ikke spørre om alt dette. Det er viktig å ha et prosjekt basert på både folk og operasjoner.

Det andre teamet har jo valg MSN for Agile fordi de jobber mye mot India og det er ikke Scrum optimalisert for (avstandsprosjekter), men det er ingenting som stopper deg fra å gjøre det da.

Q1: Slik du ser det, er det viktig å tenke på hvem som er med i prosjektet når du velger metode?
A1: Ja, det er viktig. Jeg vil aldri i mitt liv jobbe med noe annet enn scrum, så hvis folk jeg jobber sammen ikke kan dette så får de lære seg det.

Q1: Når dere jobber med scrum, legger dere opp til at det er litt fritt eller må dere følge det slavisk?
A1: Det er veldig fritt, så å si et utgangspunkt bare. Det er noen regler vi må følge, men ikke slavisk. For eksempel Discovery Wells teamet kjører morgenmøte kun 3 ganger i uka og ikke hver dag, så har de et lengre møte på fredager pga kommunikasjon med India over skype tar litt lenger tid. Nå er jeg litt i mot hvordan de kjører møtene fordi de kjører de litt for lenge og diskuterer utenom det som skal inngå i en slik Daily-scrum. De har også problemer med å kjøre korte nok sprinter, så det tar alt for lang tid før det kommer ut en release. I mitt team da, så har vi ikke lenger daily scrum pga de andre sitter i Oslo, så jeg må purre litt på de for å oppdatere arbeidsoppgavene sine, men bortsett fra det så følger vi på mitt team scrum til punkt og prikke (så godt vi kan), spesielt i forhold til administrasjon av start og stopp på en sprint.

Det andre teamet går forbi deadline hele tida og klarer ikke gjøre ferdig en sprint, og da blir utviklerne demotivert som er det motsatte av hva vi vil med scrum og ha fokus på spirit slik at utviklerne ikke blir lei.

Det andre teamet dere snakket med har problemer med å få ferdig releaser de har, men det er mest pga det er et stort og komplekst system, men hadde de villet kunne de brukt scrum de også. Med scrum og at du har en release hver måned betyr ikke det at den skal til kunden, du kan gi den til kunden en gang i året for eksempel.

Q1: Er det noe du savner, noen mangler eller noe du mener Scrum kunne vært bedre på?
A1: Kunne hatt litt flere forskjellige steg i implementasjonen, kunne hatt litt flere regler. Legger gjerne opp til at man skal bruke XP eller andre former for retningslinjer for dokumentasjon og slikt. Kunne gjerne hatt en genial løsning på en ”scrum ad-on”/scrum til det neste steget.

Q1: Er det hyllevare det dere utvikler?
A1: Ja det vil jeg nesten påstå, for vårt system er det som er det letteste av alle våre systemer og installere. Det er tilgjengelig i to versjoner, den ene er klick once som er du publiserer via en webside med en link som kan klikke på og akseptere og så installerer den på maskinen og

så har vi en versjon som er vanlig windows installer og det eneste du må gjøre når du har installert er å koble opp mot en sitecom-server.

Q1: Føler du det er viktig for kundene at dere har en formell metode?
A1: Ja det har jeg lagt godt merke til og at store kunder er veldig interessert i hva slags prosess man bruker. Det viktigste er egentlig at du kan vise til en prosess og at det er en bra prosess, at det er noe som er kjent og at det ikke er noe du har snekret sammen selv. Selv om dette kan være bra og inneholder ting fra de kjente prosessene så har du ikke noe navn på det, og det er viktig at man har et navn som er kjent og etablert. Noen selskaper (store) krever også ISO-sertifisering men vi har ikke hatt noe problem med å ikke være det, når det stort sett går på pris og funksjonalitet.

Q1: Er pris en viktig faktor for kunden slik du ser det?
A1: Det er litt forskjell rundt forbi i verden på de forskjellige firmaene. I Norge for eksempel er det svært viktig med fastpris, mens for eksempel i Mexico er det dagrate som vil si at de betaler en viss sum for hver dag de bruker programvaren din for hver gang de driller etter olje. Så det er en mye mer gunstig form for avtale. Mye fleksibilitet her med sånne avtaler, men dersom vi utvikler noe som ikke kun er spesifikt for kunden kan det være det blir delfinansiering.

Q1: Å bruke en tradisjonell fossefallsmodell, hvordan hadde det fungert i deres prosjekt?
A1: Nei det hadde ikke fungert, man vet ikke nok på forhånd om hva man vil ha. Man starter med en visjon og da ville man i starten ikke kunne være i stand til å skrive spesifikasjon hvis man skulle jobbe på en slik måte. Hvis du skal lage noe du har lagd før eller noen andre har lagd før og vet for eksempel hvilke moduler etc som må med så kan være mulig å bruke en slik en tror jeg, men jeg ville uansett bruke scrum og hatt en lang sprint fordi man da har mye mer klart i starten og trenger ikke bruke tid på for eksempel kartlegging. Men man kan være i utgangspunktet mye flere folk i en stabil metode som fossefall, men du kan bruke scrum også med mange mennesker på et stort prosjekt skulle jeg tro, hvis du deler opp i mange moduler og flere team som jobber individuelt.

Vedlegg E – Intervjuobjekt 4

Q1: Atle Sørensen
Q2: Erlend Egeland
A1: Representant fra bedriften

Q1: Konkret på dette prosjektet, hva slags metode er brukt?
A1: Den metoden vi har styrt prosjektet etter er Scrum, men den sier mer om hvordan sørge for å styre fremdrift, men ikke så mye om hvordan du skal utvikle, teste osv.

Ikke så mye utvikling av ny kode, men tilpassning av et verktøy med egen kode og tilpasning grensesnitt mot andre systemer for å få dette til å kommunisere med dette innkjøpte systemet. I tillegg har det vært en del konfigurasjon.

Det er ikke et typisk rent IT-utviklingsprosjekt. Dere kommer vel tilbake til dette, men hva som har vært bakgrunn for valg av metode tar vi når dette kommer, men det er Scrum som har vært metode for dette prosjektet.

Q1: Er dette noe dere bruker generelt i hos dere?
A1: Ja, eller det blir brukt i to andre prosjekter per i dag som jeg vet om. Men det er ganske nytt for oss. Jeg begynte her i september i fjor og kjørte da dette prosjektet som ble det første prosjektet vi benyttet Scrum på.

Dette var fordi jeg hadde erfaring med dette fra tidligere arbeidsgiver.

Q1: Dere bruker i grunn litt forskjellige metoder da altså?
A1: Ja, litt, nå har ikke jeg vært i andre prosjekter her nede enn de jeg har styrt selv. Men eller så bruker de vel mer en metode som likner litt på RUP vil jeg tro. Som ikke nødvendigvis fyller alle kriterier, men iterativ utvikling tror jeg hvertfall vi kan si at det er det vi bruker.

Siden vi er et konsulentfirma og jobber etter kundens primisser er det opp til kunden å bestemme hvordan de lar oss som konsulenter jobber, men også hvor god støtte vi har i sine mottakerorganisasjoner som faktisk skal bruke systemet. Ulike metoder krever ulike typer involvering fra kundenes side. Det er ikke alltid de har verken tid, lyst eller kapasitet til å være ekstremt involvert eller engasjert i prosjektet.

Hvis du ikke kan kjøre et prosjekt der kunden deltar kontinuerlig vil man heller kjøre en del spesifisering i starten og heller kjøre noen brukertester underveis. Men dette prosjektet her hos AE har det vært veldig lite spesifisering på forhånd sånn at målet har blitt til underveis, og det passer jo egentlig veldig bra til Scrum. Det var egentlig også grunnen til at vi valgte metoden, pluss at AE har vært veldig flinke til å bidra underveis til for eksempel å ta beslutninger.

Q1: Dere vil sånn sett si da at det er kunden som bestemmer metoden dere bruker i et prosjekt?
A1: Ja, for i noen prosjekter som jeg kjenner til er tilfellet at for eksempel at ved et rederi vi jobber mot hvor de sitter fysisk i Grimstad og vi i Kristiansand, hvor det er kontakt over tlf og mail er det ikke likt som hos AE hvor vi sitter hos kunden. Så det er en forskjell i bruk av metode.

Q1: Har du noe inntrykk av at kunden vet noe om metode?
A1: Ja i de fleste tilfeller så er det en liten IT-avdeling eller noen med IT-kompetanse i mottaker siden som har en formening om hvordan du ønsker at prosjektet/utviklingen skal foregå.

Men jeg har opplevd at de fleste forventer seg i stor grad en fossefallsmetode, med spesifisering i forkant, utvikling og kommer tilbake 4mnd senere for å få produktet. Mange blir ofte overrasket over metoden siden det er en iterativ metodikk. Men de har som oftest en formening om hvordan et utviklingsprosjekt skal kjøres.

Q1: Hvor mange personer har det pleid å være på et prosjekt som dette?
A1: Ja her har vi nå 4 fra oss og 2.3 fra AE, mer eller mindre men ikke på heltid som jobber dette og møtes hver morgen for ”daily Scrum” så da jobber vi ganske tett (2 kontorer mellom hverandre med mye interaksjon).

Dette er kanskje litt spesielt for dette prosjektet, da jeg har hørt om andre prosjekter hos oss der kunden som oftest sitter hos seg og vi sitter hos oss så da jobber vi ikke så tett som vi gjør hos AE.

Q1: Slik som med Scrum, hva er da fordelene med dette?
A1: Fordelen er for det første er at når du spesifiserer en IT-leveranse kjenner du aldri alle forutsetningene og klarer ikke å formulere alle ønskene dine på papiret. Gjelder også forventninger og ønsker fra kunden, omgivelser, infrastruktur som endrer seg og derfor veldig vanskelig å spesifisere ting på forhånd. Det vi gjorde har var at vi hadde en viss formening og hadde en ramme for mye arbeid og investeringer de kunne tåle og ut fra det kjørte vi tre ukers iterasjoner, 4 ganger 3 uker og etter hver sprint (iterasjon) hadde vi en leveranse som ble testet med en gang og så justerte vi kravspeken etter dette. Så denne product backlog som er kravspek i scrum så vi ble endret eller omdefinert ganske hyppig. Helt klart et prosjekt som det var hensiktsmessig å ikke spesifisere alt på forhånd for da hadde vi bommet i forhold til hva de forventet seg.

Det gjorde det også veldig greit å bruke Scrum siden det var i stor grad konfigurering av et ferdig produkt og mindre utvikling fra bunnen av. Det gjorde at hvis du skal utvikle fra bunnen av bør du tenke litt mer igjennom arkitektur etc fra starten, mens her så var det allerede en arkitektur som var etablert og så fleksibel slik at man kunne bruke dette. Det var grensesnittet mot andre systemer som var utfordringen og ikke spesifisering av problemet på forhånd.

Q1: At dere fanger opp kravene underveis er dette den største fordelen?
A1: Ja det vil jeg si er en stor fordel, ja. Så er det jo også for oss som konsulentfirma en fordel at vi slipper å binde oss til å lage et sett funksjonalitet til en bestemt pris, at vi har en ramme for hvor mange timer vi kan bruke og kunden stoler på at vi vil klare å lage et produkt som er godt nok. Så er det litt mer opp til oss hva vi faktisk får til.

Da slipper vi krangling om en liten detalj som ikke har stor betydning som stod i kravspek men som ikke er gjort, selv om den stod i speken. Hvis ikke denne er gjort er jo det uheldig, men sånn som en mer tillitsbasert ordning er det for oss mer gunstig som konsulentselskap.

Q1: Kan det da være fare for at dere overskrider dersom dere støter på problemer?
A1: Ja, men du vil ikke overskride i tid, du vil heller da bare kutte bort funksjonalitet. Risikoen med dette er da at man kan risikere at prosjektet leverte så lite i løpet av den perioden det foregikk at man får en kunde som er missfornøyd. Men kutter enten funksjonalitet eller utvider scopet. Utvider vi scopet da kan vi enten ta det på egen regning som ikke er noe bra for oss eller be om mer penger for prosjektet noe som er uheldig for oppdragsgiver. Det er basert på tillit til at selskapet kan leve opp til det som er forventet, og hvis ikke det er tillitt hadde det nok ikke heller gått å ha et slikt prosjekt.

Q1: Slik som Scrum (metoden) har dere hatt noen utgifter på denne, for eksempel som opplæring?
A1: Ja, jeg hadde jo jobbet med Scrum-prosjekter og vært Scrum-master før, da jeg jobbet i Finn.no. Og jeg var opplært fra før og vi kjørte ikke noe kursing, eller noe annen opplæring enn at jeg holdt en introduksjon på 1-2 timer for prosjektdeltakerne og kunden om hvordan et Scrum-prosjekt skulle gjennomføres. Ut over dette gikk det seg til, man lærte mens prosjektet gikk og gled inn i rollene. Det som det var mest skepsis til i starten var at det ikke var en bestemt kravspesifikasjon annet enn en Excel liste som var product backlog, men det viste seg jo og at på dette prosjektet fungerte det bra. Hadde det vært et litt tungt it systemutviklingsprosjekt så ville vi nok lagt litt mer arbeid i product backlog før vi begynte første sprinten, men det var det ikke behov for her så da ble prosjektet selv en opplæring for deltakerne.

Q1: Så kunden vet da før prosjektet starter hva som er forventet av de, for eksempel i forbindelse med deltakelse?
A1: Ja jeg fortalte på introduksjonen hva som skal til for å kjøre et Scrum-prosjekt. Dvs hvilke aktiviteter som er med, møter som er med. Det var bare å presentere dette på kickoff og der og innsalget så understreket vi at det er viktig at prosjekteier eller en fra AE er med hver morgen på daily scrum. Det var de helt inneforstått med og det var det de trengte, så var det samme for prosjektdeltakere og kunde hvordan man lærte seg prosjektgangen.

Q1: Er dette avtalebasert, med at de skal stille seg så og så mye til rådighet?
A1: Det var ikke en formell avtale, så hvis det hadde gått galt, for eksempel hvis de ikke involverte seg så måtte jeg ha tatt det opp med min oppdragsgiver som var en styringsgruppe for prosjektet og sagt at nå får vi ikke involvering fra kunden og da klarer vi ikke å gjennomføre prosjektet. Men det ordnet seg av seg selv faktisk, uten problemer.

Dette har jeg fått høre i ettertid, men AE er jo vant til å kjøre prosjekter og være kunde og har da erfaring med hvordan prosjekter kjøres, men så kom dette med Scrum inn og da kunne man se at de våknet litt på kickoff hvor de forventet seg en vanlig presentasjon men så kom det noe helt nytt. Da var de ekstra positivt innstilt siden det var noe spennende, så det var en annen fordel vi hadde i dette prosjektet. Og de så da at det ikke tar så mye tid med det daglige møtet på 10 min. som kreves av dem, de tar like lang tid som en kaffe, lese avisen eller prate med naboen.

Q1: Vil du si det og er en fordel på et slik Scrum-prosjekt, at dere sitter her hos kunden? Det vil vel kanskje ikke fungere dersom dere sitter spredt?
A1: Ja, jeg vet at han ene som er med på prosjektet her har kjørt et scrum-prosjekt hvor kunden satt i Oslo med daily scrum over telefonkonferanse. Og da var det litt vanskelig og det er lettere for kunden å skyve bort møter og si det ikke passer når vi har satt opp konferansen. Så jeg tror de sleit litt med kundeinvolveringen når de ikke var på fysisk samme sted. Da må

man kanskje gjøre en litt mer formell avtale på forhånd, med at kunden er nødt til å stille opp, det er nok nødvendig.

For det er jo slik, det er jo ikke vi som leverandør av et system kan diktere at kunden må være til stede så og så mye, det er jo hvor mye de har tid til og kan være til stede. Hvis vi misstenker at det ikke er mulig å gjennomføre et så tett prosjekt så må vi kanskje heller bruke en annen metode, for vi ser jo at vi brukte i dette tilfellet veldig lite tid på planlegging, rapportering og spesifisering i forhold til utviklingstiden. Så det var et bra prosjekt som gikk på mindre tid, under budsjett og økt funksjonalitet i forhold til det opprinnelige, så kunden var fornøyd.

Q1: I et slikt Scrum-prosjekt, du produserer mye mindre dokumentasjon?
A1: Ja, det er jo product backlog som egentlig bare er en liste på høyt nivå over funksjonalitet, sprint backlog som er delt opp i aktiviteter og i tillegg som ikke er en del av scrum en status rapport som jeg avga til min oppdragsgiver og til min leder, det er for brukte timer sist uke og totalt slik at vi ikke gikk over budsjettrammen.

Q1: Når du sa du kjente til metoden, var det den eneste grunnen til at dere valgte den metoden eller var det andre spesifikke grunner til at dere valgte denne metoden?
A1: Hovedgrunnen var jo at vi så på det som et prosjekt der AE ikke helt visste hva de ville ha og det hadde blitt ganske tungt å spekke det i detalj på forhånd. Fordi det er veldig mange interessenter og et stort firma. En annen grunn var at dette var første store prosjektet i klarte og selge inn her hos AE, som er en viktig kunde på Sørlandet, og derfor hadde vi lyst til å gi et sterkt og godt inntrykk av oss slik at de ville bruke oss videre i fremtiden, for det har tidligere vært et annet firma som har vært hoffleverandør her av konsulenttjenester så når vi kunne få foten innenfor var det bra tror vi å komme noe som var nytt og friskt. Det fikk vi beskjed om underveis at de var veldig fornøyd med metoden, så det var litt flaks (og dyktighet) at vi klarte og velge riktig metode.

Q1: Det kan jo høres veldig risikabelt ut da hvis dere ikke var veldig kjent med metoden?
A1: Ja, men samtidig er det også (synes jeg) risikabelt å kjøre et fossefallsprosjekt hvor man i stor grad bommer på målet. Du vil aldri klare å spå hva kunde ønsker eller du vil kunne få levert.

Jeg vil si ulempen med tradisjonelle metoder er at du ikke har mulighet for å spå fremtiden. Men RUP er jo og en iterativ metode som kanskje er litt midt i mellom, men det er vel kanskje litt mer formelle krav til dokumentasjon og rapportering, så det å kunne fokusere på utviklingsoppgaver og det som ligger rundt tror jeg er bra. Scrum master har jo den andre siden i oppgave å fjerne hindringer for prosjektdeltakerne og teamet, og det er kanskje ikke så fremtredende i andre metodikker, for der kan du se at deltakerne blir mer forstyrret med andre ting som de egentlig ikke burde blitt forstyrret med.

Men jeg tror ikke Scrum ville passe til alle typer prosjekter, så det er noe man må se an, det må være rom for det og kunden må godta at ikke alt er spesifisert på forhånd. Det som ligger til grunn for å benytte scrum er at du ikke har helt klare og gode krav på forhånd, eller at du ikke ser det hensiktsmessig å bruke tid på å lage de kravene på forhånd, det er i grunn det viktigste.

Men og hvis du ser på den personlige utviklers kompetanse på utvikling så er det mer givende for den å være med på et SCRUM-prosjekt, for der får teammedlemmer mer ansvar enn i et tradisjonelt prosjekt hvor prosjekt leder har ansvar for planlegging, estimering, oppfølging mens i scrum så er det teamet selv som sier at vi skal klare det og det målet i neste sprint og vi trenger å gjøre de og de aktivitetene for å nå det målet og teamet tar også på seg oppgaver i løpet av sprinten hvor de fordeler oppgaver seg i mellom i stedet for at en prosjektleder skal stå og peke og fordele oppgaver. Det er mer sånn at våre konsulenter skal føle det mer meningsfylt og ikke så mye for kunden sin del.

Q1: Det setter da også mer krav til prosjektdeltakerne?
A1: Ja, det gjør det. En annen stor forskjell eller overgang er det å få forståelsen i teamet for at det er deres ansvar å levere og å gjøre oppgaven enn at det er prosjektlederens ansvar at ting blir gjort. Dette innebærer også at man må ha noen av deltakerne med stor erfaring slik at de er i stand til å se hva slags ting som er en nødvendighet for at teamet skal kunne klare å nå det og det målet. Men det gjelder også et hvert prosjekt med mange aktiviteter at man må ha noen med så stor erfaring at de kan klare å gjette hvor lang tid ting vil kunne ta. Så det er en fordel, absolutt, med erfarne deltakere men det er ikke kun behov for dette.

Q1: Har størrelsen på prosjektet noe å si på om man velger for eksempel Scrum som metode eller ikke?
A1: Ja det vil jeg absolutt si. Jeg har kun lest om i teorien at man deler opp et stort prosjekt i flere scrum-grupper og hvor scrum-masterne møtes i møter hvor de har en scrum over scrumene. Men det største prosjektet jeg har vært med på har vært i Finn.no der vi var ca 15 involverte totalt på et scrum-prosjekt. Så jeg har aldri prøvd det på noen større prosjekter enn det, så jeg er litt skeptisk til om det kan fungere men det kan jo godt hende, det vet jeg ikke som fakta.

Men for å si at størrelsen på kriteriet for at man kan bruke scrum eller ikke tør jeg ikke si, jeg måtte nesten ha forsøkt det i praksis, men det kan som sagt godt hende.

Q1: Hva vil du si er rollen til metoden? Skal den følges for eksempel til punkt og prikke eller er det mer som et rammeverk?
A1: Ja kanskje som et rammeverk, men det er samtidig strenge regler for hvordan dette daily scrum skal foregå med hvor mange spm man skal stille til hvert medlem og ikke gå utenfor i masse snakk. Det er også regler for dokumentene som product backlog, hvor strukturen er gitt så sånn sett er det jo ganske strengt. Men det er jo fritt i forhold til testing, kvalitet, dokumentasjon, det er helt opp til hvert enkelt prosjekt. Så sånn sett kan du si at scrum-metoden løper litt fra dette ansvaret. På en annen siden er det litt feil hvis en utviklingsmetode skal si hvordan alt skal foregå i detalj, for det er ikke likt i alle prosjekter og sammenhenger.

Det er ganske strengt hvordan du gjennomfører de få tingene du skal gjøre i scrum, men utover det er det for så vidt ganske fritt.

Q1: Hvordan takler dere da elementer som for eksempel testing?
A1: Jeg kan ta et eksempel fra et annet prosjekt hvor vi hadde en egen kvalitetssikringsavdeling som var med i de forskjellige prosjektene og drev med system og integrasjonstesting. Så der hadde vi at vi baserte oss enhetstester som utviklerne selv var ansvarlige for og så var det egne kvalitetstestere som var med og testet når det ble satt sammen. Der hadde vi et ganske etablert regime for hvordan alt skulle kvalitetssikres, og da var teamet satt sammen av noen utviklere og noen testere.

I dette prosjektet var det da mer opp til teamet selv å levere noe som hadde god kvalitet, pluss at vi hadde brukertester etter hver sprint og fikk på den måten en slags akseptansetest. Men det er ikke noe formelt om hvordan man skal dokumentere kode, verken i AE eller i vår bedrift. Vanligvis vil det være en standard som er avhengig av prosjektet, men i dette prosjektet er det ikke noe som sier noe formelt om det. Det er alt etter hvor opptatt kunden er av slike standarder, men vi har maler som vi pleier å bruke for dokumentasjon hvis kunden ikke har ønske om noe eget noe.

Q1: Når dere selger dere inn, slik som her, hva vil du si er det viktigste for kunden?
A1: De fleste er opptatt av hva det vil komme til å koste for å slippe den usikkerheten av å ikke vite prisen på produktet. Veldig mye er basert på tillitt til konsulentselskaper og at kunden har brukt de før og vet hva de er i stand til å levere. Så det som er viktig for kunden vil jeg si er stort sett: pris, prisramme, ramme og tillitt til utviklere.

For dette prosjektet hadde ikke AE brukt oss før, men IT-direktøren så at det var viktig å få inn mer enn et selskap som konsulenter og ville da prøve ut oss som selskap, på det prosjektet vi har kjørt hos dem i høst. Dette var et lite prosjekt i forhold til hvor mye penger som er i omløp i AE og er da sikkert en fin prøve for å se hva vår bedrift står for, så det var nok bakgrunnen til at vi kom innenfor, men nye prosjekter i fremtiden avhenger nok av tillitt til oss som selskap.

Q1: Ser du noen ulemper med metoder som for eksempel Scrum?
A1: Jeg vil si det må være litt mer retningslinjer for dokumentasjon, testing, kommentering osv.

Jeg har tidligere jobbet i et annet konsulentselskap som lagde sin egen versjon av RUP og gav det et eget navn, hvor de hadde planer, standarder etc for hvordan du skulle gjøre alt mulig. Dette er noe som i noen tilfeller kan være greit å støtte seg på, spesielt kanskje hvis du har lite erfaring selv eller har et team som har lite erfaring. Det blir da, tror jeg, lettere for uerfarne å komme inn å jobbe på prosjekter.

Faren med at man har for mye retningslinjer kan jo være at man produserer noe som faktisk ikke blir brukt, for eksempel et dokument eller en rapport som aldri blir lest er bortkastede timer.

Q1: Hva slags erfaring har utviklerne deres?
A1: Jeg har ikke vært her så lenge, men jeg har inntrykk at det er mye høye utdanninger hos oss, spesielt mastergrader fra HiA og selv har jeg fra NTNU. Men vi ansetter jo også nyutdannede mennesker.

Vi ser jo nå at det er mange sørlendinger som flytter fra Oslo til Kristiansand etter å ha jobbet noen år i store konsulentselskaper.

Q1: Mener du det er en fordel at man bruker metoder som er godt etablert?
A1: Fordelen med å bruke faglig baserte metoder er jo at de har litt i ryggen og bygger på noe, og at man ikke finner opp kruttet selv.

Vedlegg F – Intervjuobjekt 5

Q1: Atle Sørensen
Q2: Erlend Egeland
A1: Representant 1 fra norsk statlig etat
A2: Representant 2 fra norsk statlig etat

Q1: Vi bygger oppgaven på agile metoder, men dere går kanskje mer etter tradisjonelle metoder, slik som fossefall?
A1: Ja, jeg vil si det er mer tradisjonelt. På de store prosjektene så er det ganske som fossefall, hvor du lager en kravspek, banker inn faser, kanskje noen iterasjoner underveis, men det er nok sånn ja. Men på de endringene vi holder på med nå så går du gjerne flere runder, hvor det er små endringer som er ganske oversiktlige og greie. Da bruker vi mer en type RAD.

Q1: Jeg tenkte litt på hva slags metode dere bruker hos dere, med tanke på for eksempel hva slags steg er det dere går gjennom?
A2: Ja altså det er jo en faseinndeling, kort sagt. Vi har to forskjellige metoder. Når det gjelder utviklingsprosjekter så bruker vi et metodeverk som er utviklet av Oracle, som heter CDM. Der har du to varianter som er classic (en type fossefall) og fastback som er en RAD-sak. I det store prosjektet som var ferdig for ca 6 år siden så brukte vi classic. Etter det så har vi vært innom fastback i et par sånn utviklingsprosjekter (endringer). Men vi skiller mellom det vi kaller for prosjekt og det vi kaller for forvaltning. Forvaltning er kort sagt feilretting, vedlikehold og videreutvikling. Da har vi laget noe vi kaller for en forvaltningsmetode, den er rimelig fersk (dvs. kom på banen i fjor en gang), ble vedtatt i fjor. Denne heter: Rammeverk for (…)s systemforvaltningsprosess. Den heter for oss systemforvaltningsmetoden, og det er ikke dukket opp noen kallenavn, heftige forkortelser eller noe slikt enda, men det kommer nok.

Q1: Men den er direkte fra den metoden dere har fått fra Oracle?
A2: Nei dette er noe vi har kokt i sammen selv, men den er basert på diverse ting. Den er basert på først og fremst erfaring, dvs. hvordan vi gjør det og hvordan vi har tradisjon for å jobbe. Finnes også input fra Oracles metodeverk og kanskje også det som eksterne konsulentfirmaer har brakt inn. Men det foregikk et prosjekt som var satt sammen av mennesker fra forskjellige plasser i organisasjonen og sammen så kom vi frem til dette her rammeverket.

A1: Men det som kanskje er viktig nå er jo at vi har mange forskjellige typer systemer: stormaskinsystemer, UNIX-systemer og andre systemer. Så det er litt forskjellige metoder hos de som har jobbet med stormaskiner og vi har brukt våre metoder. Så det er forsøk på å få en felles forståelse av at sånn jobber vi. Så har vi hentet inn fagavdelingene når denne skulle lages, fordi en del av de tingene som går på analyse og design det krever at brukerne er med. Det er de som kommer med kravene til oss, så skal vi lage det de ber om og deretter skal de teste at vi har laget det vi ble bedt om.

Q1: Så kundene, hvis vi kan kalle dem det, har vært med på å utvikle denne metoden som dere nå bruker? Noe som kanskje ikke er så veldig vanlig.
A2: Ja jeg vil si de har det ja. Nei det er vel kanskje ikke så vanlig.

A1: Men dette er jo hovedsakelig fordi det er internt, for da kan vi gjøre det slik. For når vi kjørte et stort prosjekt for noen år siden så var det ikke sånn. Vi hadde ingen metode når vi

startet, prosjektet valgte verktøy, det valgte metode. Det sa noe om hvordan skal vi jobbe, og det er noe av dette vi har bygd videre på. Så var for at vi skulle få en felles forståelse av hvordan vi skulle jobbe. Så hvis noen sier at dette må inn i ”systemX” så setter fagavdelingene dette inn i ”systemX” og dermed vet vi at det er der vi har kravene våre.

A2: Men det ligger litt i det begrepet forvaltning her, for forvaltning er jo en greie som mange deltar i, både fra fag-siden, it-siden og kanskje litt driftssiden. For at vi skal få suksess med å anvende metoden så er vi nødt til å involvere alle i prosessen underveis. Slik som ”A1” sier, med at fagavdelingene stiller krav underveis, og prioriterer kravene som vi da programmerer, tester og effektuerer. Men underveis har jo både IT og fag roller, for eksempel i test der IT gjør vi systemtest mens fagsiden gjør akseptansetesten. Det er en prosess der både fag og it er inne i forskjellige steg/faser.

A1: Det er kanskje også noe som ikke er så vanlig, hvis du tenker på at du lager noe for en kunde. Det er ikke alltid du kjenner kunden så godt, men vi har jo den første torsdagen i hver måned hvor vi møtes og sier at alle endringer som nå er gjort de skal vi ha ut i produksjon. Da skal disse akseptansetestes, og det er jo noe fagavdelingene er kjent med at den datoen der har de satt av til testing av nye endringer. Det er jo også en del av metoden, for det er en del av måten vi jobber på.

Q1: Dette er jo da for mindre ting, rettelser etc. Den metoden var kun for mindre arbeid som forvaltning da altså?
A2. Ja, for når prosjekter er ferdig da har vi ikke prosjekt lenger, da har vi noe som kalles for linje. Linjeorganisasjonen er den organisasjonen slik den er når vi har normalsituasjon, og forvaltning er det vi driver på med når vi har normalsituasjonen. Som er feilretting, vedlikehold og videreutvikling. Det er jo på en måte hovedaktiviteten vår.

A1: Ja, men hvis du tenker på hva folk hovedsakelig jobber med nå så er de fleste i prosjekt. De er i ”prosjekt1”, ”prosjekt2”, ”prosjekt3” osv. De fleste er da egentlig i prosjekt og resultatene deres er rundt prosjekter som foregår.

A2: Det vi kaller en ”cutoff”, da tar vi med alle endringene som er klare og sender disse til produksjon.

A1: Da har vi et eget prosjekt på det, og alle de endringene som inngår der de blir med ut på samme måte som den vanlige feilrettingen, for å si det sånn. Hvis du får en feil i produksjon håndteres det slik som de planlagte endringene og så prioriterer vi i rekke med det.

A2: Begrepet prosjekt er vel kanskje ikke brukt heller helt konsekvent bestandig, men det er nå så.

Q1: Den metoden dere bruker, hva slags faser er det dere går gjennom der?
A2: Den har 6 faser, hvis vi kan kalle det det. Kartlegging, analyse, design, utvikling, test og produksjonsretting. Det er vanskelig og si hvordan overlapping kan foregå her, men det er klart det vil være mulig i noen tilfeller at designfasen starter før analysefasen er over. Men om vi retter med en gang, eller venter til neste gang vi skal rette feil før vi retter noe kommer litt an på, men vi gjør av og til selvretting i testfasen. Noen ganger ser vi at det er så stort, og det kommer litt an på sammenhenger med andre og avhengigheter mellom de som inngår i det vi kaller for en pakke. Vi snakker mye om pakker, som er en samling av endringer, feilretting etc. som skal gå i produksjon, og disse blir da samlet til denne første torsdagen i måneden og

der dette som da blir en pakke som da sendes av gårde til et testmiljø og når de ”medlemmene” i pakken har fått status OK blir den sendt til et miljø der det er drift som er ansvarlig for å teste den i det miljøet. Når dette også er OK blir det emigrert til produksjon.

Q1: Det er da slik som en versjonshåndtering slik man har i vanlig programvareproduksjon?
A1: Ja, for vi har jo månedlige releaser. Men med tanke på når komplette pakker og hvis vi da oppdager feil i akseptansetesten må vi vurdere om dette er en endring som må ut nå pga en tidsfrist eller om det er en for stor endring som man ikke rekker å få med. For eksempel man ikke bare fjerne en tabell og tro at resten fungerer, for det gjør det ikke, eller du kan ta ikke ta bort en databasepakke for det som kaller på denne er der fortsatt og da vil dette feile i testingen. Da må vi ta en vurdering på dette på bakgrunn av kompleksitet og hva er det enkleste for oss for å få ting til å gå i det normale.

Q1: Vil dere da si at dere går igjennom disse 6 fasene i denne modellen i løpet av en måned da?
A1: Ja altså det vi nevnte med kartlegging det er planlagte endringer der vi planlegger hva slags endringer vi skal gjøre neste år. Men i tillegg til dette dukker det opp feil, altså feilretting. Men skal du ha en ny funksjon eller lignende så planlegger vi det året før og at det skal være ferdig til den og den datoen. Men så kommer du plutselig over en produksjonsfeil, og det er klart at da ligger den planleggingen i den måneden du skal få det emigrert ut i produksjon. Analysen går inn i kartleggingen for her analyserer man hvor mye det for eksempel kommer til og koste slik at man kan planlegge endringen. Da kan man prioritere i forhold til om dette er en stor endring eller liten endring. Du har også en kobling mellom analyse og design, for når du sitter i design så ser du plutselig at denne endringen er mye større og da må du diskutere med fagavdelingene og spørre om det er sånn at denne endringen er viktigere enn den andre.

Q1: Fra kartlegging til produksjon, hvor lang tid vil dette ta? Er det noe normalt forløp på slikt?
A2: Det kommer veldig an på hva slags system vi snakker om. For hos oss finnes det for eksempel noen systemer som kjøres bare en gang i året (for eksempel systemQ), de vil da ha en fase som tar da et år. Andre systemer, da snakker vi om årsversjonssystemer, hvor stortinget kommer med endringer hvert eneste år, og disse skal da kjøres på et gitt tidspunkt. Da tar dette typisk en gitt tidsperiode hvert eneste år. Men for vår del som har den årssyklusen så kan jo fasene variere, men i og med vi har det vi kaller for månedlige migreringer så er arbeidet knyttet opp mot det.

Men vi kommer egentlig ikke gjennom det på en måned heller, fordi slik som ”A1” sa så begynner vi tidligere og vi prøver å leve opp til at vi har en prioritert liste over alle endringer som skal gjøres i løpet av året. Det er klart ved årets begynnelse og da er det det vi skal jobbe med det kommende året. Så begynner vi med de viktigste endringen og setter disse i produksjon, og for disse endringen vil syklusen var, tja, relativt kort. Men for de som ikke er så høyt prioritert så vil jo kanskje analysefasen komme på et senere tidspunkt og da vil kanskje tida også strekke seg ut. Så det er vanskelig og si hva slags tid det tar.

Q1: Det er mange som er veldig opptatt av at de kjører si 2 mnd, og da må de være ferdig til det tidspunktet. Slik er det ikke hos dere?
A1: nei, hos er kravet kvalitet og vi lar vær å emigrere ting ut i produksjon hvis vi ikke er sikre på at det er et sikkert og korrekt produkt.

Q1: Dere har vel kanskje da ikke den luksusen som andre har ved at de kan kutte funksjonalitet?
A1: Nja, vi kan si dersom det er kun endringsønsker at dette får dere ikke nå fordi det er viktigere at dere får det riktig enn at dere får det kjapt. Så det kan vi gjøre. Men så er det noen ting som hvor vi samarbeider med mange andre, den ”cutoffen” som er nå så har vi tre prosjekter som skal akseptansetestes. Hvis vi da sier at nei dette her er ikke ferdig da har vi et problem, for da ødelegger vi jo for to andre prosjekter. Da er det jo gitt frister og da må vi bare jobbe til det funker.

Q1: Alle har tidsfrister og krav, men dere har kanskje noen fagavdelinger som er nødt å ha ting på gitt tidspunkt?
A1: Ja det er noen lover og regler, for eksempel hvis riksrevisjonen skal ha en rapport og sier vi skal lage den, så må den være ferdig til den og den datoen. Hvordan vi ligger an spiller for de ikke noen rolle, for de skal jo ha denne rapporten. Så da kan du ikke kutte på verken funksjonalitet, lengde eller tid.

A2: Det er sånn opplagte ting, som når det for eksempel blir innført moms på transporttjenester, som var en ny sats. Og da fikk vi 1.7 et eller anna år tror jeg det var, som vi da får en helt absolutt frist å forholde oss til, og da må vi bare gjøre det.

Q1: Sånn i forbindelse med krav, får dere inn alt dere trenger av krav i forkant i form av en kravspesifikasjon?
A1: Vi har en slik ”groupware” løsning som heter Lotus Notes, så der skriver vi hva vi har gjort. Dersom det er noe vi lurer på angående et krav så sender vi det her til den brukeren det gjelder og så får vi svar tilbake når det er avklart. Samme gjør vi ved testing, da sender vi en sånn melding som vi spør om dette er klart eller kan du teste dette, så tester de det og så sender de tilbake om det var greit eller ikke.

Q1: Bruker dere noe form for prototyping eller testes det kun på helt ferdige systemer?
A1: Det kommer an på hvor stor endringen er, som for eksempel med momsendringen så er det kanskje bare 10 brev som skal endres og et skjermbilde. Da gjør du alle de endringene, og kjører du en integrasjonstest på skjermbildet, de brevene og så en databasepakke som brukeren ikke ser. Da gir de tilbakemelding på hva de kan se, som da er brev og skjermbilder men de gir ikke noe tilbakemelding på det de ikke ser som er databasepakken, men den setter vi som OK hvis det brukeren ser er OK. Så er det noen endringer som er bare en databasepakke og noen endringer som er veldig store, men flere moduler, der vi har gjerne flere folk som er involvert. Det er veldig avhengig av størrelsen på endringene vi gjør for hvordan vi jobber.

Q1: Hvor mange kan dere da være på et prosjekt med tanke på utviklere? Sånn gjennomsnittlig.
A2: Tja, hvertfall når bygde det MVA-systemet som vi jobber på nå så var vi vel på det meste ca 150 mennesker, mens gjennomsnittsbemanningen var, tja, rundt 100. Men på de gruppene som jobber hos oss nå så kan det være fra, tja, 2-3 stykker til ja.

Q1: Metoden dere bruker, kan man skalere den til å passe både større og mindre prosjekter? For eksempel om dere er 150 stk. eller 10 stk.?
A2: Om vi kan bruke den metoden hvis vi er 150 stykker det vet jeg ikke, men det er en politikk hos oss som sier at på større prosjekter skal det brukes Oracles metodeverk, da snakker vi om prosjekter. Men snakker vi om forvaltning, som er det daglige arbeidet, så men i det daglige arbeidet skal vi bruke denne forvaltningsmetoden. Vi i systemgruppe 5, som vi tilhører, vi er 17 mennesker som er stort sett utviklere, skal ikke si vi er noen eksperter på å bruke metoden, men vi har en prosess på gang som sier at vi skal ta den mer og mer i bruk. Så forholder vi oss til en fagside som, ja du kan si den ene delen kalles for fastsetting og den andre delen kalles for innkreving. Det er fordi at det er sånn at det noen som bestemmer hvor mye MVA som skal betales, og den andre biten som er innkreving sørger at pengene kommer inn. Og da er det en 5-6 fagpersoner på hver siden, så da er dette systemet rundt MVA bestå av en ca 10 stykker.

Q1: De kundene/brukerne som dere har, er det slik at de har god kjennskap til systemene dere lager og metoden dere bruker? For tradisjonelt sett har det vært litt problematikk med å få involvert brukere og la dem få forståelse for hva det er dere driver med?
A2: Ja de har vært med på å utvikle denne metoden.

A1: Ja det er helt klart. For det var et av problemene da vi jobbet med det prosjektet vi holder på å gjøre endringer på nå, at da hadde vi ikke noe felles forståelse og vi hadde ikke jobbet på den måten før, så det var en lærende prosess.

Q1: Brukerne hadde vært med å utvikle metoden. Er det sånn at dere kommer med forslag og brukerne kommer med forslag da, eller hvordan?
A2: Ja det er klart at de har jo god kompetanse rundt hva som skjer i kartleggingsfasen, og de har god kompetanse på testing, og det er klart det er først og fremst i de fasene de har kommet med innspill, til dels også analyse. Mens i design og utvikling var det vel vi som hadde mer og komme med. Vi har jo på en måte diskutert det gjennom oss alle sammen.

Denne metoden er jo på en måte en prosessbeskrivelse som sier at i den fasen så skal du gjøre de og de stegene, i neste fase skal du gjøre de og de stegene og så står det hvem som er ansvarlig for de forskjellige stegene. I tillegg til det, denne metoden ligger jo som en, ja på intranettet vårt, den ligger beskrevet der. Men i tillegg til å være en metode så tilbyr den også et sett med hjelpemidler, og disse hjelpemidlene er sånn typisk teststrategi, testplan, diverse standarder (programmeringsstandarder, utviklingsstandarder). For eksempel i kartleggingsfasen bruker vi noe som heter endringsforslag, og dette endringsforslaget skal ha en gitt form og da har du en mal for hvordan endringsforslaget skal se ut. Så da kan du elektronisk gå inn å plukke ut en masse forskjellige maler som hele organisasjonen kan bruke. Den store fordelen med det er at vi etter hvert er i ferd med å få et felles begrepsapparat for hele organisasjonen. Så uansett hvem vi snakker med, så vet de hva vi snakker om, for eksempel hvis jeg sier endringsforslag så vet alle hva jeg snakker om. Mens tidligere så snakket ikke vi på MVA-siden om endringsforslag, vi brukte kanskje begreper som ”aa” eller ”cr” (change request), mens vi nå er i ferd med å få et felles begrepsapparat. Når vi snakker om testing har vi definert at de hovedtestene vi har, eller testfasene som er systemtest, akseptansetest og produksjonstest, mens innenfor de testene kan du så klart ha stresstest, ytelsestest osv. Men dette er hovedfasene av testing, og tidligere var systemtest for noen noe helt annet enn det var for oss. Så dette er med på, rammeverket er med på å gi disse begrepene en felles forståelse. Jeg sier ikke vi er i mål med dette her nå, det er en prosess som tar lang

tid, for dette er en relativt stor organisasjon. Men jeg tror vi er på vei til å komme dit, og gjør at metoden er med på å skape mer effektivitet, mulighet for å, eller letter å jobbe på forskjellige systemer, og kanskje bidra til at kvaliteten på ting blir bedre.

Q1: Den malen, eller det dokumentet som beskriver metoden. Det er slik det er, må man følge den slavisk da kan man si?
A2: Nei, eller det er et interessant spørsmål, for da kommer vi inn på anvendelsen av metoden. For det vi har laget, vi har laget dette rammeverket, så har vi lagt det rammeverket ut på intranettet, så har vi i tillegg hatt en masse arbeidsgrupper som har jobba med forskjellige hjelpemidler som er knyttet til rammeverket. Men som A1 nevnte så er de systemene vi forvalter veldig forskjellige, tror vi har over 100 systemer, fra de helt små til de ganske store. Det er klart, trikset er, metoden vil bli brukt forskjellig og det er ulike deler av metoden som er mer eller mindre interessant for de forskjellige systemene. Da har vi et interessant begrep, eller sentralt begrep, som heter arbeidsplan og arbeidsplanen er hvert enkelt systems anvendelse av metoden. Så det vi på MVA-systemet har gjort, vi har gått gjennom aktivitetene i hver fase punkt for punkt, så har vi sagt at for vår del så er det disse punktene som er interessante å gjøre i kartlegging, disse punktene i analyse osv. Så har vi lagd et eget dokument som da beskriver mer i detalj hvordan vi bruker denne metoden, eller dette rammeverket. Eller customizing er et ”godt norsk ord” for hva vi har gjort. Det beskriver hvordan vi bruker metoden, samtidig som den er mer detaljert for den sier konkret hvilke leveranser vi skal ha, hva skal vi produsere i hver fase, hvem er ansvarlig for det, altså hvilken kontorenhet er ansvarlig for å gjøre det og det. Og det står jo ikke her, for den metoden er jo en generell metode for hele organisasjonen. Mens her står det hvilke roller som er ansvarlige for å utføre de forskjellige punktene underveis her. De som har lagd dette er i hovedsak tre grupper, det er systemgruppe 5 som er systemutviklingsavdelingen, fastsettelsessiden og så er det innkrevingssiden. Metoden sier at når du skal ta i bruk metoden så må du lage en arbeidsplan, og det er derfor vi har gjort akkurat dette, og den er ganske fersk så vi er på en måte i en sånn oppstartsfase. Den har med andre ord en del ting som man må gjøre, men det er til en viss grad opp til en selv hvordan man vil gjøre det.

Q1: Men hva vil du da si er styrkene ved å gjøre det på den måten? Hva er da fordelene med den metoden som dere nå da har valgt?
A2: Jeg syns det viktigste med metoden er at den er relativt godtatt, at man har enighet om den i hele organisasjonen, både i IT-sida og fagsida. At vi er enige om at, OK, sånn gjør vi det. Jeg syns som sagt det er det viktigste med tanke på størrelsen på organisasjonen, men det er sikkert andre som syns andre ting er viktigere.

Q2: Så du vil si det er liten forskjell på hvilken metode som brukes så lenge det er enighet om hvordan man bruker den og at den skal brukes?
A2: Ja det kan du i grunn godt si, ja.

A1: Ja altså dette her er jo ikke en revolusjonerende metode, det er jo en helt standard vanlig utviklingsmetode. Det er bare at vi har definert noen begreper som vi har felles forståelse for i organisasjonen, og for oss som jobba med metoder i prosjekter så var det en relativt grei overgang mens de som har jobba med en metode oppe i hodet sitt så har kommunikasjonen med oss blitt enklere. For å si det sånn så er det lettere å snakke med de nå enn det var før. For systemgruppa som er 140 ansatte som driver med utvikling av forskjellige typer applikasjoner så syns jeg det kan være greit å snakke om de samme tingene. Så nå har vi innen en fra Oslo som skal jobbe hos oss med et annet system enn personen vanligvis pleier, så det hjelper da med felles forståelse.

Q1: Når dere har en sånn metode og beskrivelse, vil det da være lett for nye mennesker å komme inn og forstå hvordan dere jobber?
A2: De ville fått det til selv om vi ikke hadde hatt metode, men det er enklere når du kan lese dokumentasjonen til metoden og forstå hvordan rollen min er.

Q1: Men når dere utvikler nye systemer, er metoden dere bruker da veldig forskjellig fra forvaltning?
A2: Nei, for Oracles CDM-metode har den samme faseinndelingen som forvaltningsmetoden. Riktig nok har de litt forskjellige navn, men den store forskjellen er vel rett og slett disse dokumentmalene som CDM legger opp til. CDM har vel en 1000 dokumentmaler som den legger opp til og en 400 leveranser som du kan, på en måte, produsere ved å bruke metoden. For eksempel laste ned maler du fyller inn, som skal godkjennes av fagsiden, som var det de begynte med, som gjorde at folk syns dette ble for mye og ble veldig komplisert. Men da gjorde vi på samme måte, vi satt oss også ned og lagde en arbeidsplan for CDM, så da har vi jo skjønt det at CDM krever jo også at du bruker en arbeidsplan og plukker ut de leveransene du faktisk skal levere. Med andre ord bare et skall. Det er veldig mange fellestrekk, men sånn layoutmessig og språkmessig og navnestandarder på dokumenter og sånn er veldig forskjellig. Hovedårsaken til at vi valgte CDM, var at det var en tjeneste fra Oracle.

A1: Så var det jo en helt grei metode, det var ikke noe galt med den. Den var ikke noe bedre eller dårligere enn andre. Den har jo også en kobling til de begrepene som ellers finnes i Oracle-porteføljen.

A2: Nå kjenner ikke jeg så veldig mange andre metoder, men det er klart at CDM, i og med den hadde disse malene, det var jo veldig greit da. Sånn at vi fikk produsert leveranser veldig fort, at ja okei jeg skal lage det her så da plukker jeg ut den og så lager jeg det. Jeg skal for eksempel lage en type liste, da henter jeg mal på den listen og fyller inn kravene, ikke sant. Skal den være detaljert og ikke overordnet, så henter du den malen, og sånn kan du holde på. Det er jo mulig at andre metoder tilbyr tilsvarende, men problemet er jo at den tilbydde så forferdelig mye, du trenger ikke levere alt, for eksempel grov, vanlig, detaljert og veldig detaljert av samme liste. Du trenger ikke alt, kanskje bare en før du hopper til neste steg.

A1: Det legges opp til at du skal gå gjennom steg og produsere ting på detaljnivå alt etter hvor langt du er kommet og hvor mye du vet. For eksempel i detaljering av arkitektur, så det er en grei måte å jobbe på hvis du skal jobbe iterativt.

A2: Så er det også en kobling mellom, i prinsippet så er det en kobling mellom, mellom CDM og Oracle sine utviklingsverktøy, men da må du være veldig flink til å på en måte gjøre det riktig i verktøyet og metoden.

Q1: Dere nevnte iterativt, hvordan går da leveransene? Får dere et krav og leverer om et år hvis det er fristen eller hvordan gjør dere det?
A1: Vi leverer kontinuerlig for å si det sånn, vi vil altså ha endringer ut så fort som mulig

Q2: Når dere starter på en endring, et prosjekt eller lignende, vet dere alltid hva det er dere skal gjøre eller står dere noen ganger og ”famler litt i blinde”?
A1: Du vet omtrent hva du skal gjøre, men du vet ikke helt hvordan du skal gjøre det tror jeg. Du får en beskrivelse av for eksempel: ”Dette skjermbildet virker ikke” eller ”Knappen jeg trykker på gjør ikke det den skulle/skal ha gjort”. Da finner du kanskje ut at det er noe feil, og

da må du først finne ut hvorfor blir denne feil før du kan rette feilen. Mens andre ganger kan det være veldig konkret, ved for eksempel at du skal endre teksten i et brev. Så det kommer veldig an på hva du skal gjøre. Og så er det jo større ting, og da må du kanskje ikke endre så mye men det er mye du må teste. Da vet du for eksempel bare at du skal endre tilgangskontrollen og da må du gjøre det du tror skal til for å løse det. Først må du bestemme hva slags løsning går vi for, og så må du prøve å gjennomføre det og så må du teste og se om virka det sånn som jeg hadde trodd. Så må du kanskje justere litt på måten å løse det på igjen. Så da har du ikke en konkret kravspek på forhånd, du sitter bare med et mål og kjører igjennom flere ganger til du når målet. Så har du andre store prosjekter der du skal for eksempel flytte et system fra en CyBase-løsning til en Oracle-løsning og da er i grunn kravspeken at du skal ha samme funksjonalitet i det nye systemet som du hadde i det gamle systemet. Så du må du altså lære deg det gamle systemet først.

Q1: Andre steder vet som regel ikke kunden hva de vil ha, men det er kanskje litt mer bevissthet hos dere og fagavdelingene?
A1: Ja, det vet jo stort sett hva de vil ha. Men det som vi har sagt at vi har forskjellige typer endringer. Noen som fagavdelingene ber om, som er tilleggsfunksjoner eller ny funksjonalitet. Men så har vi også ting som vi ser selv, for eksempel ytelsesproblematikk, teknisk ikke god nok løsning så her må vi så da kommer vi med krav selv som også skal da inn i den rettelsen eller endringen som gjøres.

A2: Så har du sånn type ting som det er ikke alltid fagavdelingene ser de ulike teknologiske mulighetene. Okei, vi skal ha standardbrev som skal sendes ut i de og de tilfellene. Ja det er greit, vi skal ha standardbrev, men i de brevene skal jeg kunne gå inn og skrive en tekst. Hvordan skal da dette gjøres? Det er jo veldig mange måter å gjøre det på, skal man bruke det ene eller det andre. Der kan jo vi kanskje bidra med å si at vi syns at det blir den greieste løsning i forhold til det systemet og om hvordan databasen er bygd osv. Der vil det jo være en dialog mellom oss og dem. Mens i andre tilfeller så vil det helt gitt, med jurister og lover og regler som vi må forholde oss til. Da skal det enkelt og greit fungere etter de lovene og reglene. Da har vi en dato, og fra den datoen skal det virke, uansett.

Q1: Savner dere noe i metoden eller er det noe som den kunne vært bedre på?
A2: Ja det er det sikkert, og den er jo veldig fersk som sagt. Vi har brukt et år ca på å komme i gang med den, lage arbeidsplanen vår. Det kommer til å bli revidert kontinuerlig, eller med jevne mellomrom, dette rammeverket. Litt vanskelig å svare på akkurat nå, men vi skal ta en evaluering på hvordan ting fungerer. Men det henger også sammen med at vi skal tilby et sett hjelpemidler som skal ligge på intranettet vårt, og jo flere hjelpemidler som blir ferdige og lagt opp der jo bedre vil jo folk oppleve at den servicen de får tilbud om å bruke er. Så det blir jo stadig vekk bedre.

Q1: Så metoden blir tilpasset etter hvordan dere jobber i praksis og ikke at dere tilpasser arbeidet etter hvordan "boka" sier det?
A2: Tja, jeg vet ikke helt. Jeg tror vel egentlig at prinsippene som er nedfelt her de er vel ikke så veldig avanserte, men de er i hvertfall at det er 6 faser, og fasene er definert, hovedaktivitetene skal utføres, det er definert et sett med roller som skal på plass og de ligger nok nokså fast. Men så kommer der kommer nye roller, eller endringer i rollebeskrivelse eller nye aktiviteter som ikke er definert så den blir videreutviklet og kanskje mer avansert etter hvert.

Det er en samling av egne erfaringer og CDM som er basisen. Men hun som jobbet med dette var ganske fersk og kan mye om dette. Vi har også sett på IEEE standarder når vi lagde det, andre etater så vi har prøvd å ikke hente kun interne erfaringer.

A1: Men det er jo sånn med en metode at man kan ikke drive og endre på dette hele tiden, for da blir forvirringen komplett. Først har folk jobba etter sin egen metode, så har de jobba etter en metode som er veldig lik bare ting har andre navn og så skal du venne deg til det. Så hvis du da skal gå inn å gjøre store endringer i det stadig så vil jo folk miste forståelsen. Da kan du heller jobbe med arbeidsplaner å gjøre tilpasninger etter hvert.

Q1: Mange har metode i teori, men det stemmer ikke med metode i praksis. Er det tilfelle her eller stemmer deres teori med praksis?
A2: Det er skissert et opplegg her med en del metodeansvarlige i forskjellige avdelinger, og de skal da ha et eget ansvar for å kunne metoden veldig godt. De skal da ta imot innspil om endringer osv. for så å møtes i det vi kaller et metodeforum for å gå gjennom revideringen.

Q1: Hva er da det viktig for dere for at dere skal få dette til å gli jevnt?
A2: Det er vel knyttet litt opp til hva du faktisk leverer, på basis av metoden så har jo vi en del rutiner som brukes i dette LotusNotes-baserte gruppevare-opplegget vårt. Og vi har rutine for hvordan du skal bruke dette, så da brukes metodeverket når du lager forskjellige forslag. Her har du da mange krav til hvordan du skal fylle ut diverse skjemaer, med tanke på status, hvor ting skal fylles ut, hva som skal være med osv. I analyse skal programmererne skrive sånn og sånn, test er det IT som har ansvaret for hvor man skal skrive hva man skal gjøre under en test og samme for fag ved akseptansetesting.

Det som blir resultatet av alt dette her er jo at du definerer en del steg som du skal innom. Hvis alle følger disse stegene da, så vil jo det til syvende og sist heve kvaliteten. Men det er klart at noen vil jo si at det funker jo, hoppe over et par steg og da går jo det utover kvaliteten. Til syvende og sist så koker det jo ned til at etterlevelse etter metoden er et ledelsesansvar, der du m motivere slik at du får med deg alle til å gjøre det slik det skal gjøres.

Q1: Dere har da ganske høye krav til dokumentasjon med tanke på mye jus osv?
A2: Ja det er veldig viktig.

Q1: Når dere jobber med fagavdelingene sitter dere ofte sammen med dem, på lotus notes eller bare hos dere?
A2: Bruker veldig mye arbeidsgrupper. Det er klart hvis det er en ny ting, så er det viktig at alle faktisk blir enige om hva som faktisk skal lages. Da må man kartlegge og analysere først, så teste ut litt om vi er på sporet og så lager vi ferdig et system som kan testes.

Vedlegg G – Intervjuguide (struktur)

Bakgrunn

1. Hva slags systemer utvikler deres bedrift?
2. Hva er visjonen/ målet for deres organisasjon?
3. Hva slags bakgrunn har utviklerne i deres organisasjon? (erfaring, utdanning)

Metoder

1. Hva slags metoder benytter dere per dags dato? Agile metoder, tradisjonelle metoder osv.
    a. Hvilken metode ble brukt på sist prosjekt?
    b. Hva gikk sist prosjekt ut på?
    c. Hvor mange utviklere jobbet på sist prosjekt?
    d. Hvem var kunde for siste prosjekt?
    e. Hva er utgiftene på metoden? (opplæring, kjøp av lisenser)
    f. Hva påvirker deres valg av metode i prosjekter, har dere en fast metode eller bytter dere fra prosjekt til prosjekt?

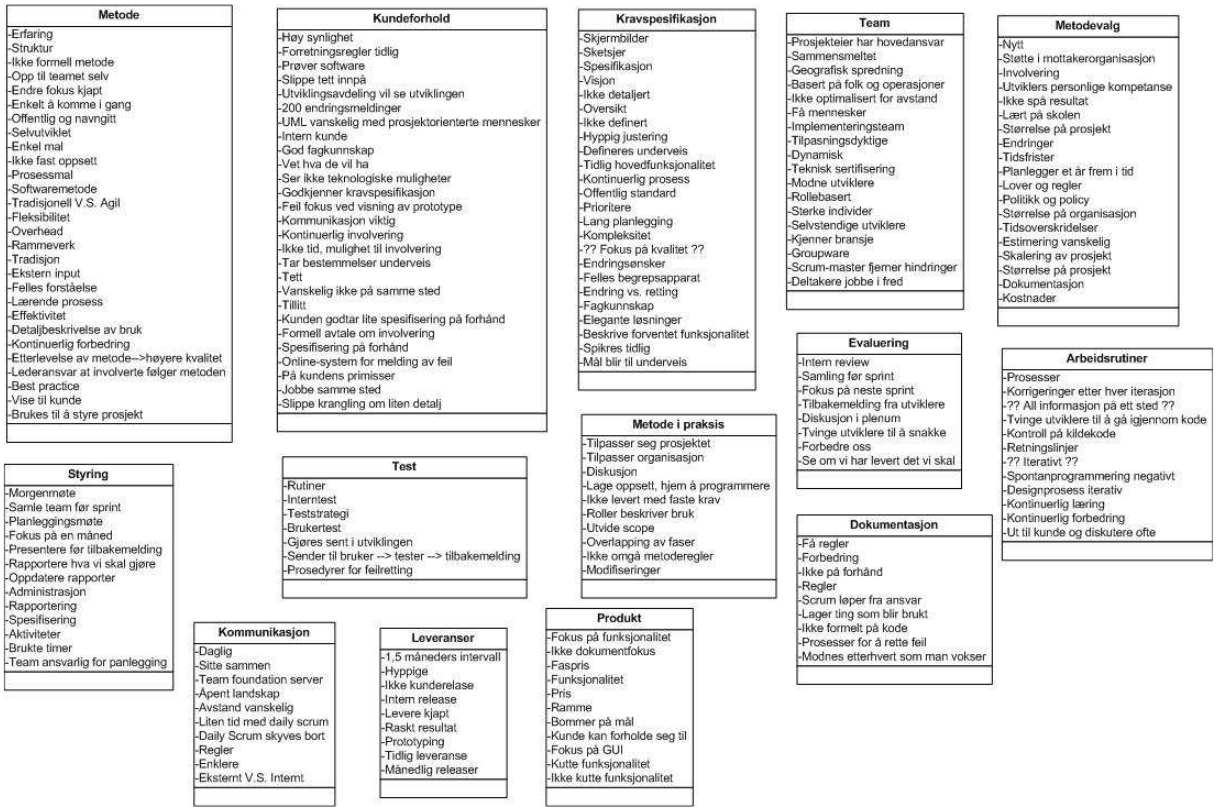
2. Hva er årsaken til at disse/denne blir benyttet?
    a. Hvilke egenskaper mener du er viktig i valg av metode?
    b. Hvilke egenskaper er viktig for å få en god utviklingsprosess?
    c. Hva er de største styrkene/ fordelene med deres metoder?
    d. Hvilken rolle har metoden i bedriften? (rammeverk, sertifisering, dokumentasjon)
    e. Hvilke forskjeller finnes det i formell metode og metode som brukes?

3. Hva kan være alternativer til disse/denne?
    a. Den metoden som ikke blir brukt, hva slags inntrykk/holdning har du til denne typen metode? (overordnet agil/ tradisjonell)
    b. Hvilke alternativer har dere eventuelt studert?
    c. Hva er ulempene med metoden dere bruker, noe du savner i metoden?

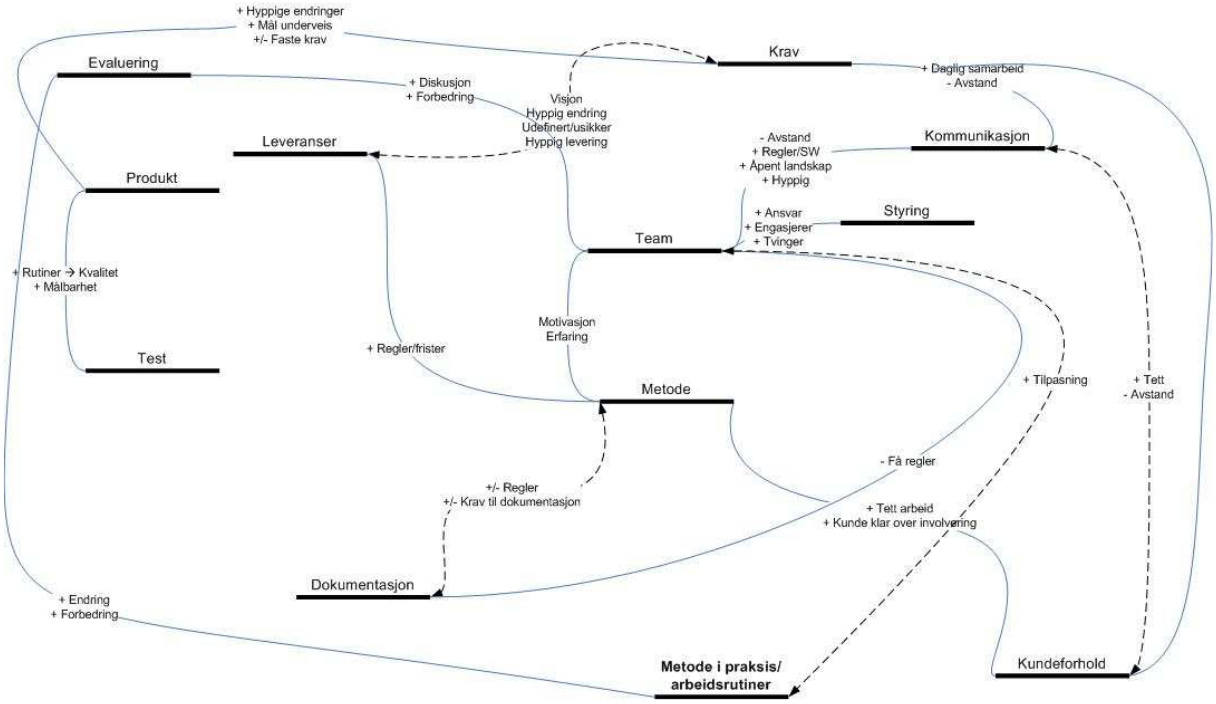
Kundeforhold og påvirkning

1. Hvilke/ hva slags kunder har dere? (bransjer, størrelse)
2. Hva slags momenter er viktige for kunder når dere inngår kontrakter?
3. Hva er kundenes oppfatning av metode? (viktig med en sertifisert og godkjent metode?)
4. Hvordan stiller dere dere til krav fra kunder om sertifiseringer og dokumentasjon? (har det blitt gjennomført program for sertifisering og opplæring)

Vedlegg H – Modell steg 1



Vedlegg I – Modell steg 2



Vedlegg J – Aksial koding