Universitetet i Bergen

Scrum og offshoring

En studie av prosjekter som benytter Scrum kombinert med offshoring

Cato Haukeland

Masteroppgave i Informasjonsvitenskap

Universitetet i Bergen

Innlevert: 31. mai 2009

Veileder: Solveig Bjørnestad

Abstrakt

Smidige metoder er blitt mer og mer utbredt blant systemutviklingsprosjekter. Blant disse metodene har Scrum i senere tid fått mye oppmerksomhet. Parallelt med at smidige metoder har fått fotfeste, har offshoring blitt mer vanlig. Offshoring innebærer at man innenfor ett og samme prosjekt har medlemmer lokalisert i ulike land.

En del selskaper har de siste årene kombinert både smidige metoder og offshoring. I denne studien undersøkes 12 prosjekter hvor Scrum sammen med offshoring er benyttet. Målet med studien er å se hvordan disse prosjektene har valgt å implementere Scrum i et miljø hvor offshoring benyttes. I tillegg er det lagt fokus på utfordringer som dukker opp i slike prosjekter og hvordan de undersøkte prosjektene har adressert disse.

Resultatene fra studien indikerer at det er mulig å gjennomføre prosjekter hvor Scrum benyttes sammen med offshoring. For at denne typen prosjekter skal lykkes har de undersøkte prosjektene valgt å fokusere mye på opplæring og sosialisering i starten på prosjektene, samt lagt en tilpasset plan for kommunikasjon mellom lokasjonene.

Studien legger til rette for videre forskning på denne typen prosjekter hvor eksempelvis oppstartsform kan undersøkes nærmere for å avgjøre hvor mye det har å si for prosjektet som helhet.

Nøkkelord: Scrum, smidige metoder, offshoring

Forord

Som siste steg i en toårig mastergrad i Informasjonsvitenskap, representerer denne studien ett års arbeid.

Interessen for Scrum startet da ett av fagene i mastergraden presenterte Scrum sammen med andre smidige metoder. Scrum skilte seg straks ut og det ble klart at denne metoden måtte studeres nærmere. Spesielt interessant var det at metoden er relativt fersk i forhold til andre metoder. Eksempelvis finnes det langt færre studier innen Scrum enn XP (Extreme Programming).

Etter å ha falt ned på vinkling og fått tildelt veileder, startet arbeidet med å lese seg opp på Scrum, både fra bøker og tidligere studier innen emnet. Etter noen måneder ble muligheten for å samarbeide med Accenture introdusert. Jeg fikk benytte Accenture sitt internasjonale nettverk for å finne passende prosjekter som kunne undersøkes.

Dette har direkte ført til at prosjektene undersøkt i studien er internasjonale prosjekter hos multinasjonale selskaper de fleste har kjenneskap til. Muligheten til å få undersøkt hvordan prosjekter hos slike selskaper utføres var svært motiverende.

Det må derfor rettes en stor takk til Accenture som har gitt anledning til å få studert prosjekter og fått kontakte sentrale mennesker hos Accenture innen smidige metoder. En stor takk rettes også til min veileder Solveig Bjørnestad hos Institutt for informasjons- og medievitenskap, Universitetet i Bergen. Hennes interesse for oppgaven og smidige metoder generelt, samt svært god veiledning underveis, har vært uunnværlig i denne sammenheng.

Universitetet i Bergen
Bergen, 31. mai 2009

_____
Cato Haukeland

Innhold

Liste med figurer

Liste med tabeller

1 Innledning

Informasjonssystemer er blitt mer og mer vanlig både på arbeidsplasser og i private hjem. De fleste benytter seg av en eller annen form for informasjonssystem i løpet av en dag, direkte eller indirekte. Dette kan være når man betaler med kort i butikken eller bestiller flybilletter på internett. Det dukker stadig opp nye tjenester og vi tar dem stadig raskere i bruk. I takt med at teknologien og hva som er mulig hele tiden forandrer seg, må systemutviklere forsøke å henge med så godt det lar seg gjøre. Utviklingsmetoder innen systemutvikling er i forandring og nye metoder etableres for ikke bare å svare på endringer, men omfavne dem. Disse metodene kalles agile eller smidige metoder, og har tiltrukket seg både tilhengere og motstandere.

Da verden rundt oss er i forandring og systemer utvikles i langt større grad, er det vanskelig å holde tritt med konkurrenter. Det er i det hele vanskelig å forutse hvordan et prosjekt vil skride frem og hvordan systemet til slutt vil bli. For nye systemer vil ikke kravene være fullt definert og kjent før etter at brukere har tatt i bruk systemet (Watts, 1995).

En etter hvert vanlig smidig prosjektmetode er Scrum (Schwaber & Beedle, 2001). Scrum inneholder en rekke aktiviteter, roller og regler som skal hjelpe utviklere å skape systemer som brukerne trenger, raskt og effektivt. Blant mange selskaper som benytter Scrum finner man Accenture, et stort internasjonalt IT-selskap som i flere prosjekter har benyttet Scrum som utviklingsmetode.

Scrum er inspirert av beste praksiser hos selskaper som Fuji-Xerox, Honda, Canon og Toyota. Toyota oppnår eksempelvis fire ganger bedre produktivitet og tolv ganger bedre kvalitet enn sine konkurrenter i følge Sutherland (2007). Sutherland (2007) hevder at det første Scrum teamet oppnådde et såkalt hyperproduktivt nivå i 1993. Flere andre presenterer også positive erfaringer ved bruk av Scrum (Schwaber, 2004).

En annen tendens som har utviklet seg i de senere år er offshoring, hvor store deler av arbeidet utføres i lavkostland som eksempelvis India, Russland eller land i Øst-Europa (Sutherland et al., 2007). Dette skaper utfordringer i forhold til prosjektstyring da utviklerne sitter på ulike geografiske lokasjoner, gjerne i andre tidssoner.

Det er i dag ikke uvanlig blant norske selskaper å sette ut deler av prosjekter til selskaper i lavkostland som Polen, Estland, Latvia, India etc. I slike sammenhenger kan samarbeidet

fungere på ulike måter. En kan skille prosjektet som utføres i lavkostlandet ut fra hovedprosjektet og gjøre det så uavhengig som mulig. På den måten vil en ikke behøve å ha tett kommunikasjon og samarbeid underveis. Denne fremgangsmåten vil sannsynligvis ikke gi tilfredsstillende resultater fordi det innen smidig utvikling må tas høyde for at kravspesifikasjoner og behov endrer seg i løpet av utviklingen, da disse sjelden er fullt ut kjent på forhånd (Watts, 1995). Det er dermed naturlig å tenke seg at et tettere samarbeid mellom lokasjonene vil være mer hensiktsmessig.

Flere har også tatt i bruk Scrum i distribuerte prosjekter ved å benytte offshoring. Sutherland (2007) presenterer flere prosjekter som har benyttet Scrum sammen med offshoring. Det samme gjør Danait (2005) som presenterer flere erfaringer gjort i denne sammenheng, blant annet fordelen ved bruk av en flat organisasjonsstruktur. I tillegg hadde flere aktiviteter knyttet til sosiale relasjoner en positiv effekt. De fleste artikler er case studier og henter derfor kun erfaringer knyttet til ett prosjekt. Det er av den grunn interessant å se på flere prosjekter hvor tilnærmet samme fremgangsmåte er benyttet for å se om disse kan vise til lignende resultater. Samsvarende erfaringer knyttet til slike prosjekter vil kunne gi verdifull lærdom til hvordan Scrum prosjekter med offshoring bør drives.

1.1 Problemstilling

Denne studien forsøker å avdekke utfordringer knyttet til Scrum og offshoring. Dette gjøres ved å undersøke nærmere hvordan Scrum benyttes i prosjekter hvor offshoring samtidig benyttes. Det finnes allerede prosjekter hos Accenture hvor Scrum og offshoring benyttes sammen, og oppgaven ønsker å se nærmere på disse prosjektene for å finne hvilke problemstillinger som dukker opp ved slike prosjekter. Problemstillinger som flere av prosjektene har erfart kan gi indikasjoner på endringer eller elementer som bør legges til metoden Scrum.

Det er gjennomført en undersøkelse av Scrumprosjekter gjennomført av Accenture, hvor offshoring benyttes. Disse prosjektene er sammenlignet med andre tilsvarende prosjekter for å kartlegge forskjeller og eventuelle problemstillinger knyttet opp til offshoring av Scrumprosjekter. Da det ikke finnes en Scrummetode tilpasset offshoring, kan studiens resultater gi nyttige innspill til hvordan en bør sette opp Scrumprosjekter hvor offshoring skal benyttes, og kanskje legge grunnlag for andre studier.

Problemstillingen oppsummeres i følgende spørsmål:

Spørsmål 1:

- Hvordan har de aktuelle prosjektene implementert Scrum sammen med offshoring?

Spørsmål 2:

- Hva har vært utfordringene knyttet til bruk av Scrum sammen med offshoring, og hvordan har de aktuelle prosjektene møtt disse utfordringene?

1.2 Anvendelse

En studie som denne vil kunne anvendes av alle selskaper som driver eller skal drive med Scrum kombinert med offshoring. Forhåpentligvis vil gode råd hentet inn fra Accentures erfaringer med offshoring gis til slike selskaper. Resultatene vil gjenspeile de viktigste erfaringene de ulike prosjektene har gjort seg, samt hva de selv anser som viktige faktorer knyttet til suksess hos denne typen prosjekter.

Det presenteres prosjekter fra flere land rundt om i verden, og resultatene vil av den grunn ikke være knyttet til ett spesifikt land. Erfaringene kan derfor antas å være av mer universell grad, enn hva tilfellet hadde vært om alle prosjektene var styrt fra eksempelvis Norge.

1.3 Motivasjon

Både bruk av offshoring og bruk av Scrum er i vinden som aldri før, og flere selskaper ønsker å ta i bruk både Scrum og offshoring. Det finnes i dag ingen kjente artikler eller studier som tar for seg flere store og små internasjonale prosjekter som benytter Scrum og offshoring. Det er derfor interessant å se nærmere på nettopp dette. Kan man hente ut gevinster fra bruk av Scrum samtidig som en får gevinster som knyttes til offshoring? Eller er disse ikke kompatible og man vil tape både gevinstene fra smidig utvikling og offshoring? Det er selve kombinasjonen mellom Scrum og offshoring som gjør temaet motiverende og spennende.

Scrum fokuserer på tett og personlig kommunikasjon, mens dette er en klar utfordring ved bruk av offshoring. Å studere hvordan Accenture organiserer slike prosjekter vil kunne gi verdifull kunnskap om hvordan slike prosjekter kan organiseres og hvorvidt det er gevinster å hente på kombinasjonen Scrum og offshoring.

2 Teori

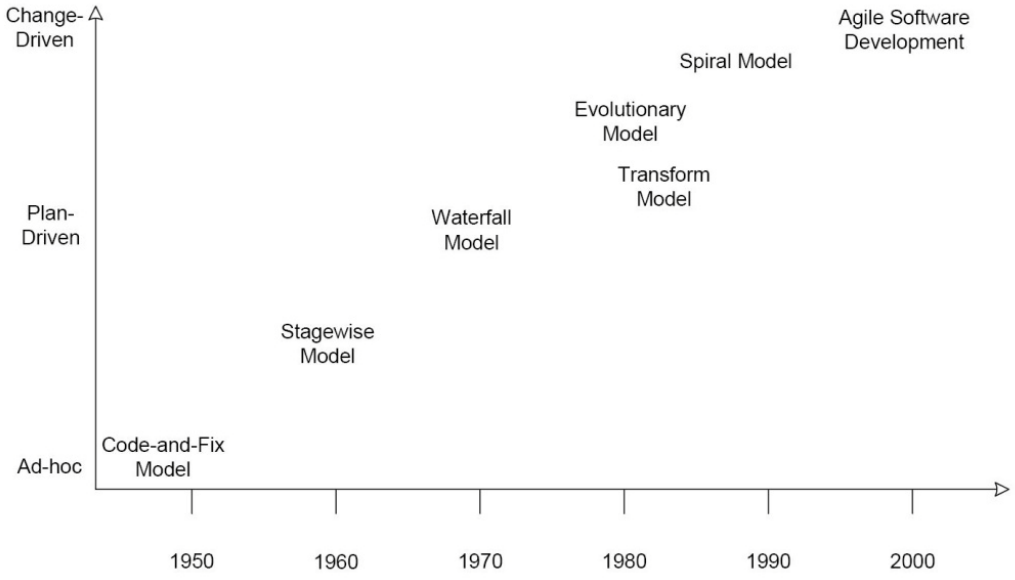
Det refereres ofte i dag til to hovedtyper av prosjektmetodologier innen systemutvikling. En har tradisjonelle metoder som i stor grad baseres på dokumentasjon og kravspesifikasjon tidlig i prosjektet og en har smidige metoder som i senere tid har fått god vind i seilene (Cohen et al., 2004). I tradisjonelle metoder starter arbeidet med dokumentering av et komplett sett av krav etterfulgt av arkitekturdesign, utvikling og inspeksjon/testing (Cohen et al., 2004).

Smidige metoder vektlegger i mindre grad dokumentasjon og tar i stor grad høyde for at krav vil endre seg i løpet av systemutviklingsperioden. Sentralt i smidige metoder står det agile manifest gjengitt under.

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>
> - **Individuals and interactions** *over* processes and tools
> - **Working software** *over* comprehensive documentation
> - **Customer collaboration** *over* contract negotiation
> - **Responding to change** *over* following a plan
>
> That is, while there is value in the items on the right, we value the items on the left more (Beck et al., 2001).

Tradisjonelle metoder blir her organisert under paraplyen tradisjonell metodologi og smidige metoder under smidig metodologi. Med metodologi menes de generelle verdiene knyttet til de ulike metodene, som også kan kalles metodelære. Begrepet metodologi kan være noe utvannet innen systemutvikling, og det velges her å benytte begrepet slik kildene selv gjør det. Med metodene menes derfor de som inneholder de konkrete, spesifiserte praksiser som benyttes i systemutviklingsprosjekter. Metodologi benyttes for å forklare fellestrekk mellom metodene.

2.1 Introduksjon til systemutviklingsmetodologi



Figur 1 Evolusjonen blant systemutviklingsmetoder (Salo, 2007)

Figur 1 gir en oversikt over hvordan utviklingen har vært blant systemutviklingsmetoder. Denne viser en mer finmasket utvikling enn kun tradisjonell og smidig metode. Den illustrerer at den tidligste fasen i stor grad var basert på ”ad-hoc” programmering, hvor programmereren utviklet og endret systemer uten noe definert kravspesifikasjon eller andre former for prosjektplaner. Denne typen utvikling hadde to steg, skriv kode og reparer feil i koden. Denne typen tilnærming førte til at etter mye kodefiksing var strukturen så dårlig at omfattende feilretting ble for dyrt. Dette førte til forståelsen og behovet for en designfase i starten av systemutviklingsprosessen (Boehm, 1986).

På 1970-tallet kom de plandrevede metodene som eksempelvis fossefallsmetoden. Disse er som navnet antyder, bygget opp rundt fastlagte planer tidlig i prosjektene. Systemkrav og brukskrav vil i slike prosjekter defineres helt i starten. I tillegg til Salo (2007) presenterer Avison og Fitzgerald (2003) en oversikt over metoder som kategoriserer det inn i pre-metodologiperioden, tidlig metodologiperioden, metodologiperioden og post-metodologiperioden som for øvrig er den de mener vi nå er inne i.

Pre-metodologiperioden

I denne perioden som hos Avison og Fitzgerald (2003) omhandler 60 og 70-tallet, ble applikasjoner utviklet og implementert uten eksplisitte eller formaliserte utviklings-metodologier. Fokuset låg på selve programmeringen samt problemløsing. Utviklerne

hadde i stor grad interesse og kunnskap om teknologien, men forstod sjelden forretnings- eller organisasjonskonteksten systemene skulle operere i (Avison & Fitzgerald, 2003).

Tidlig metodologiperioden

Slutten av 70-tallet og første del av 80-tallet var karakterisert av forsøk på å identifisere faser i utviklingsprosessen som en trodde ville forbedre utviklingsprosjektene. Systemutviklingsmetodologier ble i denne perioden først introdusert. En av de mest kjente blant disse er fossefallsmetoden eller SDLC[1] som den først ble kalt (Avison & Fitzgerald, 2003). Fossefallsmetoden vil senere presenteres sammen med andre tradisjonelle metoder.

Metodologiperioden

Metodologiperioden besvarte problemstillinger oppstått under den tidlige metodologi-perioden. Flere nye metodologier og tilnærminger så dagens lys, men fokuset var likevel det samme. Målet var å utvikle bedre produkter og forbedre utviklingsprosessen ved standardiserte metoder. Disse bestod av anbefalte faser, regler, teknikker, verktøy, dokumentasjon, ledelsesprinsipper og opplæring (Avison & Fitzgerald, 2003).

Post-metodologiperioden

Som navnet antyder er dette perioden hvor mange går vekk fra tidlige metoder da disse enten er for rigide eller ikke har gitt tilfredsstillende produktivitet. Avison og Fitzgerald (2003) hevder mange har skrinlagt metodologier helt og gått tilbake til å programmere i "prøv-og-feil-stil", grunnet mangelen på fleksibilitet i de tidligere metoder. De trekker videre frem at om ikke alle forkaster metodene, tilpasser de fleste metodene til å passe inn i typen prosjekter en gjennomfører.

Som nevnt opererer mange med to hovedtyper innen prosjektmetodologi for systemutvikling. Sett i lys av oversikten fra Avison & Fitzgerald (2003) kan tidlig metodologiperioden og metodologiperioden ses på som perioden for tradisjonelle metoder, mens post-metodologiperioden ses på som perioden for smidige metoder.

2.2 Tradisjonell metodologi

Tradisjonell metodologi handler i stor grad om en fastlagt plan for hele prosjektet, hvor målet utvilsomt er forutsigbarhet. Kunden spesifiserer en kravspesifikasjon for hva han vil ha, og utviklerne leverer akkurat hva kunden ber om. I de tradisjonelle metodene benyttes

[1] Systems Development Life Cycle

omfattende planlegging, kodifiserte prosesser og stort fokus på gjenbruk for å utvikle en så effektiv og forutsigbar aktivitet som mulig (Boehm, 2002).

Prosessen varierer noe fra metode til metode, men noen fellestrekk kan trekkes ut. Blant annet trekker Boehm (2002) frem at tradisjonelle metoder i stor grad er plandrevet, noe de også refereres til som. Med begrepet plandrevet menes at prosessen er dokumentert med spesifikke oppgaver og milepæler, samt produktutviklingsstrategier som kravspesifikasjon, design og arkitekturplaner (Boehm, 2002).

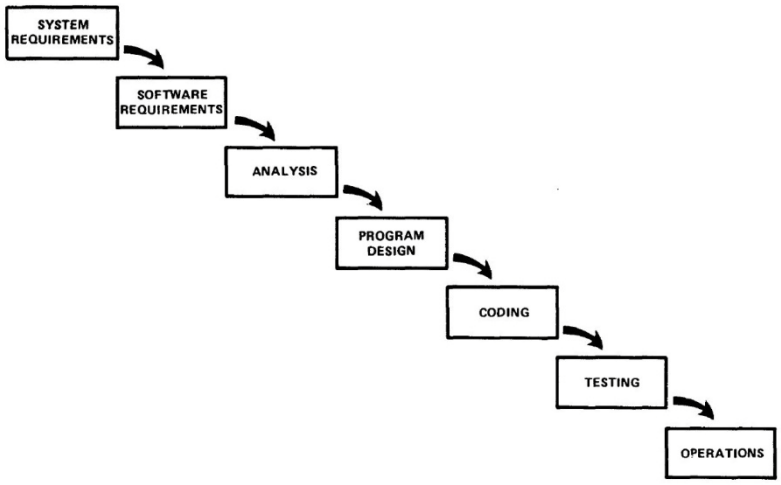
En generell karakteristikk av tradisjonelle metoder er at de består av 4 faser, planlegging, analyse, design og implementering (Dennis et al., 2004). De fleste metoder innen tradisjonell metodologi inneholder disse fasene i en eller annen form. I forhold til smidig metodologi er fokuset her sterkere på planlegging og analysedelen. Dette betyr ikke at dette ikke eksisterer innen smidig metodologi, men at den for tradisjonelle metoder er mer omfattende og rigide, samt at fokus i stor grad ligger på dokumentasjonen. Innen de smidige metodene er fokuset større på prosessen som fører til dokumentasjonen og ikke dokumentasjonen som sådan (Boehm, 2002).

2.3 Tradisjonelle metoder

Blant tradisjonelle metoder trekkes ofte fossefallsmetoden frem som et godt eksempel. Ved siden av fossefallsmetoden finnes også Spiralmodellen som eksempel på tradisjonelle metoder. Oppgaven velger her å kategorisere RUP som en tradisjonell metode, men erkjenner at den har trekk som gjør at enkelte kategoriserer den under smidige metoder da den ble utviklet på slutten av metodologiperioden. Derfor er det naturlig at den befinner seg i et grenseland mellom tradisjonelle og smidige metoder.

2.3.1 Fossefallsmodellen

Denne metoden er basert på fastlagte stadier hvor det meste av kravspesifikasjon og dokumentasjon utføres i første del av prosjektet (se Figur 2).

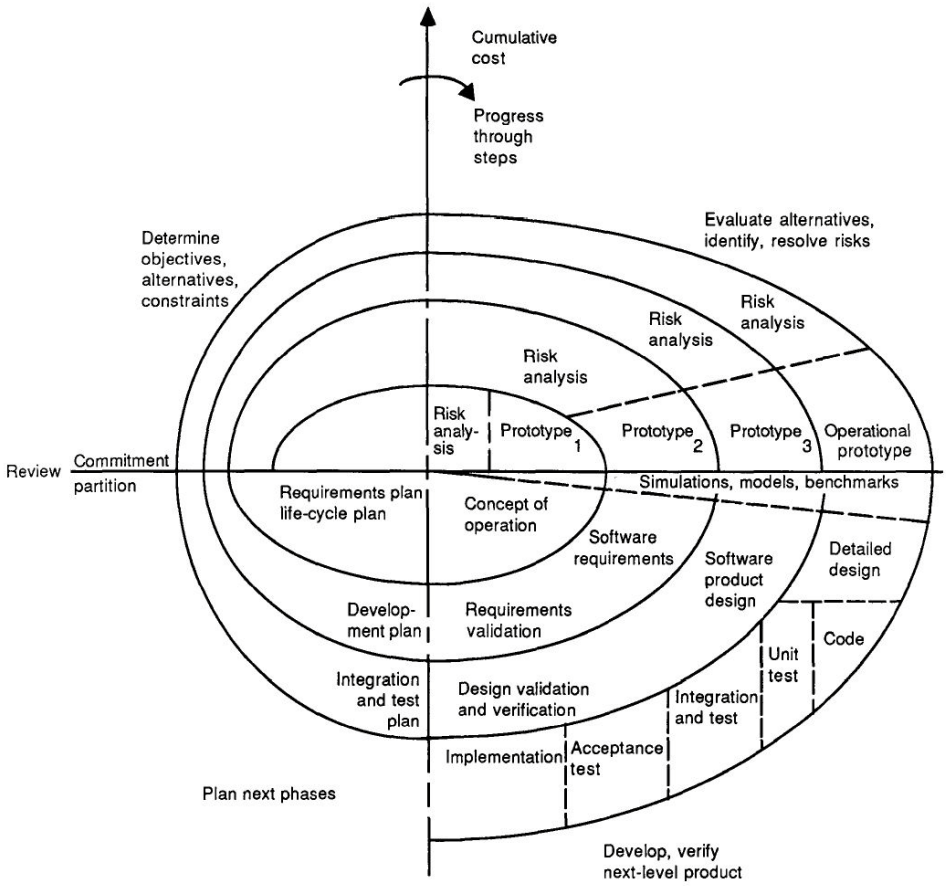


Figur 2 Fossefallsmodellen (Royce, 1987)

Fossefallsmodellen er mye brukt i systemutviklingsprosjekter og finnes i mange ulike former. Den oppgaven velger å ta utgangspunkt i, kommer fra Dr. Winston W. Royce publisert i 1987. Fossefallsmodellen har siden blitt modifisert for å ta høyde for endringer i blant annet brukerkravene, ved å kunne gå tilbake til tidligere stadier i modellen. Metoden kommer i mange ulike former, som vil gi ulike karakteristikker. Det tas her likevel kun høyde for den metoden presentert av Royce (1987) for å forenkle sammenligningen med andre metoder.

Tradisjonelle metoder baserer seg på tanken om at systemutvikling består av to hovedsteg, analyse og koding. Disse stegene finner en igjen i fossefallsmetoden, men med ytterligere steg før og etter. I utgangspunktet skal hvert steg i modellen fullføres før neste steg tar til, men i praksis er det naturlig at noen av stegene har en viss overlapping (Royce, 1987). Når det kommer til ”software requirements” gjøres de analysene ferdig før noe koding tar til (Cohen et al., 2004).

2.3.2 **Spiralmodellen**



Figur 3 Spiralmodellen (Boehm, 1986, s. 64)

Spiralmodellen er et resultat av erfaringene gjort med fossefallsmodellen i store statlige prosjekter. I motsetning til fossefallsmodellen, som er dokumentdrevet er spiralmodellen risikodrevet. Det er med andre ord et stort fokus på risikoanalyse ved hvert steg i modellen. Som navnet tilsier er modellen bygget opp rundt flere spiraler, hvor hver spiral tilsvarer et steg i prosessen (Boehm, 1986). Boehm (1986) hevder også at spiralmodellen er langt mer robust i forhold til tidligere modeller som for eksempel fossefallsmetoden.

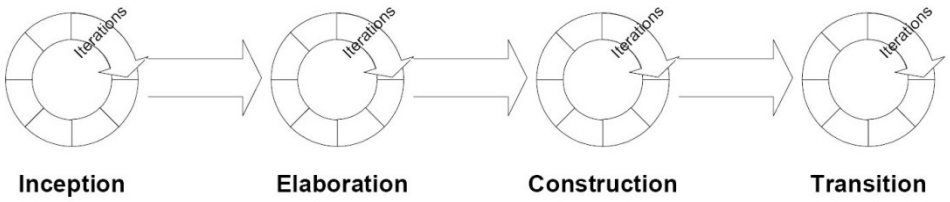
En starter en spiral med identifikasjon av innholdet i spiralen man er inne i, være seg systemkrav, systemdesign, testing etc. Neste steg er å evaluere det en tidligere har identifisert, som ulike begrensninger eller problemer som utgjør en risiko for prosjektet. En adresserer kilden til risiko ved å benytte verktøy eller teknikker som prototyping, simulering, referansekontroll, ytelsesmåling, brukerundersøkelser etc. (Boehm, 1986).

Modellen skal på en bedre måte håndtere endringer i kravspesifikasjonene underveis i forhold til fossefallsmodellen. Hver iterasjon i spiralmodellen er en egen fossefallsmodell

som starter med analyse, etterfulgt av design, implementering og til slutt testing (Cohen et al., 2004).

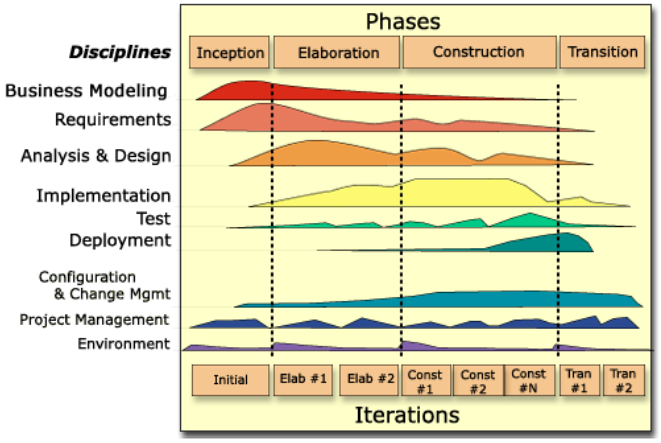
2.3.3 Rational Unified Process

Rational Unified Process (RUP) er en iterativ tilnærming til utvikling av objektorienterte systemer som oppfordrer til bruk av brukersenarioer i modellering av krav og i utviklingen av fundamentet til systemet (Abrahamsson et al., 2002).



Figur 4 RUP faser (Abrahamsson et al., 2002)

Levetiden til et RUP prosjekt er delt inn i fire faser kalt inspeksjon, utforming (elaboration), konstruksjon og overgang. Disse fasene er delt inn i iterasjoner hvor hver iterasjon skal resultere i et demonstrerbart system. Varigheten for hver iterasjon kan variere fra to uker til et halvt år (Abrahamsson et al., 2002). På tvers av de ulike fasene finnes ulike aktiviteter som dekker en eller flere faser. Disse er ”business engineering”, kravspesifikasjon, analyse og design, implementasjon, testing, levering, konfigurasjon og endringsstyring, prosjektledelse og ”enviroment” illustrert i Figur 5 (Kruchten, 2003).



Figur 5 Prosesstruktur – to dimensjoner (Kruchten, 2003)

Det er ikke gitt at alle definerer RUP som en tradisjonell metode. Abrahamsson et al. (2002) kategoriserer den under smidige metoder, mens Fowler (2005) mener den ikke

nødvendigvis hører hjemme under smidige metoder. Den er derfor her valgt under tradisjonelle metoder for å avgrense de smidige metodene til dem det er bred enighet om er smidige.

2.3.4 Problemstillinger knyttet til tradisjonelle metoder

Tradisjonelle metoder omfatter i denne oppgaven metoder innenfor tradisjonell metodologi eller pre-metodologiperioden og metodologiperioden slik Avison og Fitzgerald (2003) organiserer metodene. Enkelte av metodene er i stor grad lineære og forutsetter dermed at domenet knyttet til prosjektet er fullstendig kjent og at det ikke vil oppstå behov for endringer underveis. Royce (1987) skriver blant annet at han har stor tro på fossefallsmetoden men at implementasjonen både er risikofylt og inviterer til fiasko. Mye av dette fordi feil eller behov ikke oppdages før i testfasen, noe som fører til at prosjektet må helt tilbake til "software requirements"-fasen for å gjøre de nødvendige endringer.

Som en reaksjon på manglende struktur i de tradisjonelle metodene ble "Capability Maturity Model (CMM)" for programvare utviklet. Det er et rammeverk som beskriver nøkkelelementer for en effektiv systemutviklingsprosess. CMM beskriver en evolusjonær forbedringsvei fra ad hoc, umodne prosesser til modne, disiplinerte prosesser (Paulk, 1993). CMM benytter fem nivåer til å beskrive gode utviklings- og ledelsespraksiser. Den definerer 18 nøkkelprosesser og 52 mål for en organisasjon til å nå nivå fem. De fleste organisasjoners nivå er kaotisk (nivå én), mens kun fåtall har oppnådd nivå fem (Cohen et al., 2004).

Enkelte utøvere av tradisjonelle metoder opplevde på 1990-tallet det som vanskelig om ikke umulig, å definere alle krav og arkitektur for systemet i starten på prosjektet. Behovet hos brukerne og teknologien endret seg for raskt til at prosjekter innenfor tradisjonelle metoder klarte å henge med (Cohen et al., 2004). I tillegg blir kundene mindre og mindre i stand til å definere alle krav til et system på forhånd eller i starten av prosjektet, samtidig som de krever stadig mer av programvareleverandørene. Selv om spiralmodellen skal ta bedre høyde for endringer av blant annet brukskrav underveis, hevder flere utøvere av dette ikke er tilstrekkelig i den stadig utviklede forretningsverdenen (Cohen et al., 2004).

Smidige metoder, som diskuteres i neste seksjon, kom som en reaksjon på tunge, dokumentasjonsdrevne tradisjonelle metoder (Cohen et al., 2004). Det var og er i stor grad et behov for et alternativ til disse tradisjonelle metodene og det er her de smidige metodene kommer inn i bildet.

2.4 Smidige metoder

Smidige metoder blir også omtalt som agile[2] metoder. Oppgaven velger å benytte ordet smidige da det begrepet er det mest brukte på norsk. Denne seksjonen vil ta for seg litt bakgrunnshistorie knyttet til smidige metoder, samt det agile manifest, før en rask introduksjon til kjente smidige metoder vil bli gitt.

Så hva er å være smidig i denne sammenheng? Jim Highsmith uttaler at det å være smidig betyr at man er kapabel til å levere raskt, endre raskt og endre ofte (Cohen et al., 2004).

2.4.1 Historien

På 90-tallet fant en rekke brukere av tradisjonelle metoder utviklingstrinnene i metodene frustrerende og kanskje umulige. Både industri og teknologi var i for rask endring til at de klarte å holde tritt. Som et resultat utviklet flere konsulenter uavhengige metoder og praksiser for å møte endringer i industri og teknologi som de erfarte. Smidige metoder er ikke noe nytt, men baserer seg på velbrukte praksiser. Det er i stedet fokuset på verdiene som ligger til grunn, som er forskjellige fra tradisjonelle metoder (Cohen et al., 2004). Forskere ved DuPont Chemical's Advanced Research Facility kom etter forespørsel fra Ken Schwaber, frem til overaskende konklusjoner i forhold til hvorfor definerte prosesser forkynt av CMM ikke målbart leverte resultater (Cohen et al., 2004). Selv om CMM fokuserer på å omgjøre systemutvikling til gjentakende, definerte og forutsigbare prosesser, fant forskere ut at systemutviklingsprosjekter i stor grad var uforutsigbare og umulig å reprodusere av blant annet følgende årsaker:

- Prosessen er bare begynnelsen for å bli forstått.
- Prosessen er komplisert.
- Prosessen er i endring og uforutsigbar.

Disse konklusjonene fikk flere til å se nye veier etter løsningen på optimale prosjektmetoder. Ken Schwaber startet utviklingen av Scrum, hvor han oppdaget at for å være smidig, må en prosess akseptere endring i stedet for å fokusere på forutsigbarhet. Han og andre kom til at metoder som ville respondere på endring like raskt som endring fremkom var nødvendig (Cohen et al., 2004).

[2] Ordet agil gis følgende definisjon: adj. (mer -, mest -) lat: rask, smidig, i full vigør (Clue International Corporation – Ordbok for windows)

Det agile manifest

Tidlig i 2001 samlet 17 mennesker innen smidig systemutvikling seg for å diskutere nye systemutviklingsmetoder. Ut av dette kom Det agile manifest, som et alternativ til dokumentasjonsdrevne metoder. Det ble uttalt fra deltakerne at den smidige bevegelsen ikke var en anti-metode-bevegelse, men at det hele handlet om å gjenskape balanse.

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>
> - **Individuals and interactions** *over* processes and tools
> - **Working software** *over* comprehensive documentation
> - **Customer collaboration** *over* contract negotiation
> - **Responding to change** *over* following a plan
>
> That is, while there is value in the items on the right, we value the items on the left more (Beck et al., 2001).

Kort oppsummert handler det om å sette individer og interaksjon fremfor prosesser og verktøy, fungerende programvare fremfor omfattende dokumentasjon, samarbeid med kunde fremfor kontraktsforhandlinger og respondere på endring fremfor å følge en plan. Beck et al. (2001) hevder ikke at elementene på høyre side av manifestet er uviktige, men at de på venstre side er viktigere.

Glass (2001) er blant dem som har gjennomgått og kommentert manifestet. Han skriver han er enig i det første punktet om individer og interaksjon, hvor han skriver at programmerere og utviklingsteam i lang større grad påvirker kvaliteten til programvare enn prosesser og verktøy. Dette er ikke noen revolusjonerende uttalelse og de fleste innen tradisjonelle metoder vil være enige (Glass, 2001).

Vedrørende fungerende programvare fremfor omfattende dokumentasjon er Glass (2001) fremdeles enig med manifestet, men understreker likevel viktigheten av dokumentasjon i form av kravspesifikasjon og manualer. Innen tradisjonelle metoder har fokuset på dokumentasjon blitt for stort og for noen er det sett på som det primære fokus innen systemutvikling.

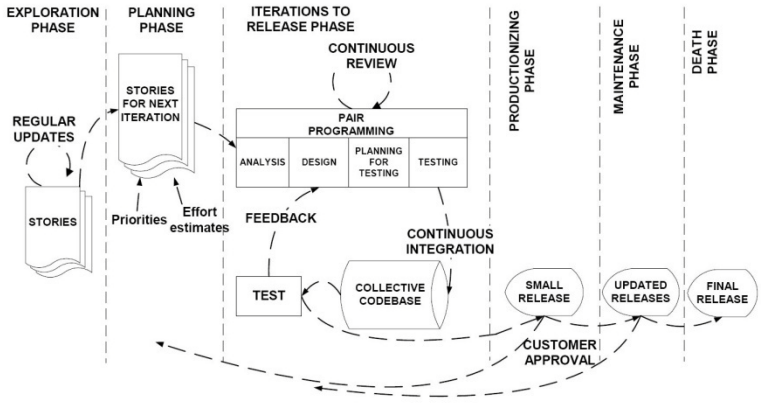
Glass (2001) hevder at samarbeid med kunde og kontraktsforhandlinger er like viktig, hvor kontrakter danner grunnlaget for samarbeidet med kunde. I en ideell verden ville kontrakter være unødvendig, men i den verden vi nå lever i, er det en forutsetning å ha formelle avtaler mellom parter.

Det siste punktet i det agile manifest er Glass (2001) ikke utelukkende enig i. Han trekker frem viktigheten av å respondere på endring underveis i systemutviklingsprosessen. Glass (2001) hevder også at endring i kravspesifikasjonene har vist seg å være en avgjørende faktor i systemutviklingsprosjekter som har mislyktes, uten at han presiserer hva som legges i begrepet mislykkes.

Det er også erfart at brukere sjelden på forhånd er klar over alle krav de ønsker et kommende system skal tilfredsstille. Disse to sistnevnte erfaringene passer heller dårlig sammen, men brukere har nå likevel kommet til at det ikke er til å unngå at det vil være behov for å gjøre endringer underveis i et systemutviklingsprosjekt. Planlegging er fremdeles en vital del i slike prosjekter for å oppnå suksess, men det må tas høyde for endringer underveis i prosjektet, samtidig som man må styre slik at disse ikke tar overhånd (Glass, 2001).

2.4.2 Extreme Programming (XP)

XP er den meste kjente og studerte av smidige metoder (Dybå & Dingsøyr, 2008). Den ble introdusert av blant andre Kent Beck i oktober 1998 og har vært gjengitt i en rekke artikler siden den gang (Cohen et al., 2004). Livssyklusen i XP består av 5 faser slik Figur 6 viser, ”Exploration”, ”Planning”, ”Iterations to Release”, ”Productionizing”, ”Maintenance and Death” (Abrahamsson et al., 2002).



Figur 6 XP – livssyklusmodell (Abrahamsson et al., 2002)

Metoden består av i hovedsak tolv regler her gjengitt på engelsk med norske kommentarer (Beck, 2000):

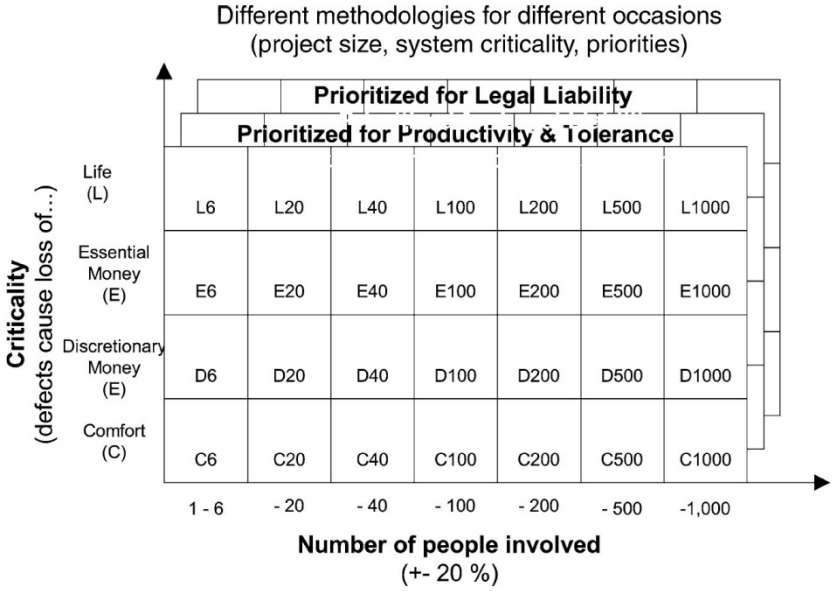
- *The planning game:* Ved start av hver iterasjon møtes kunder, ledere og utviklere for å estimere og prioritere krav for neste utgivelse. Kravene som etableres kalles brukerhistorier (”user stories”).
- *Small releases:* Den første versjonen av systemet settes i produksjon etter at man har fullført noen iterasjoner. Deretter settes fungerende versjon i produksjon i intervaller på alt fra noen dager til noen ukers mellomrom.
- *Metaphor:* Kunder, ledere og utviklere etablerer en metafor eller ett sett med metaforer for å skape en felles forståelse av systemet som skal utvikles.
- *Simple design:* Utviklere oppfordres til å holde designet så enkelt som mulig.
- *Testing:* En jobber etter prinsippet ”test først” ved hjelp av akseptanse-, enhets- og funksjonelle tester.
- *Refactoring:* Under uvikling bør designet omarbeides for å holde det så enkelt som mulig. Med dette menes at man endrer den interne strukturen uten å endre ekstern funksjonell adferd eller funksjonalitet. ”Refactoring” kan eksempelvis være å forbedre lesbarheten til kode eller forenkle kodestrukturen.
- *Pair programming:* To utviklere sitter på samme maskin og skriver all kode. Fungerer på den måten at én skriver koden mens den andre sitter ved siden av og observerer koden som blir skrevet. Det er vanlig at de to skifter rolle hyppig.
- *Continous integration:* Ny kode skal integreres med resten av systemet så ofte som mulig, som i denne sammenheng betyr flere ganger daglig. Alle funksjonelle tester må fremdeles passere etter integrering av ny kode. Viss ikke forkastes den nye koden.
- *Collective ownership:* Koden eies av alle utviklerne, og hvem som helst av utviklerne kan utføre endringer hvor som helst i koden ved behov. Koden deles med andre ord kollektivt. Dersom en feil oppstår, vil den første som vet hvordan dette kan fikses, sette i gang å reparere koden.
- *On-site customer:* Kunden jobber sammen med utviklingsteamet til enhver tid for å svare på spørsmål, utføre akseptansetester samt overvåke prosjektet for å se utviklingen har forventet progresjon.
- *40-hour weeks:* Overtid må unngås ved at man velger en edruelig mengde krav for hver iterasjon.
- *Open Workspace:* Utviklerne jobber i åpne landskap hvor alle er lett tilgjengelig for hverandre.

De tolv reglene hver for seg gir ikke resultater, men i kombinasjon gir de XP styrke som metode. Da reglene klart gir utrykk for at utviklingsgruppen må være samlokalisert setter dette begrensninger i forhold til størrelsen på gruppen. Av den grunn er anbefalt størrelse

mellom to og ti personer. Lengden på en iterasjon er blant de korteste innen smidige metoder og er på ca. 2 uker. XP har ingen støtte for distribuerte grupper da den fokuserer på fellskap og samlokalisering av grupper (Cohen et al., 2004).

2.4.3 "The Crystal Methods"

"The Crystal Methods" ble utviklet av Alistair Cockburn tidlig på 90-tallet. Han mente en av de største hindringene i produktutvikling var dårlig kommunikasjon, og utviklet derfor "The Crystal Methods" for å møte disse problemene. Ved å bytte ut mye skriftlig kommunikasjon med ansikt-til-ansikt kommunikasjon, vil sannsynligheten for er leverbart system øke (Cohen et al., 2004).

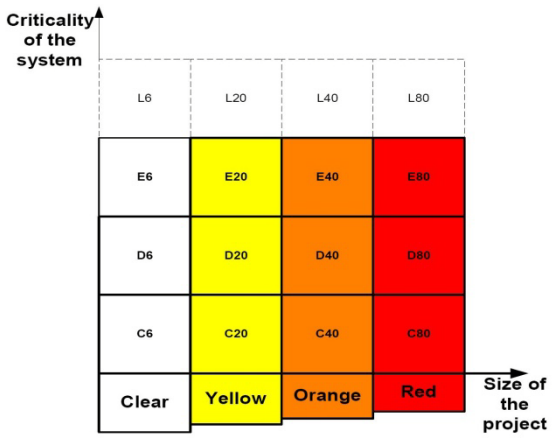


Figur 7 Rammeverk for Crystal Methods (Cockburn, 2003)

Som navnet tilsier består "The Crystal Methods" av flere metoder. Den mest smidige eller lette av metodene er "Crystal Clear", fulgt av "Crystal Yellow", "Crystal Orange" og "Crystal Red" (Cohen et al., 2004). Valget av metode er avhengig av antallet personer involvert i prosjektet, samt hvor kritisk systemet er. Figur 7 illustrerer hvordan rammeverket fungerer.

Den horisontale linjen i figuren illustrerer antall mennesker involvert i prosjektet. Antallet mennesker påvirker kompleksiteten til prosjektet og vil føre til en endring i forhold til valg av metode. Den vertikale linjen illustrerer hvor kritisk systemet er, noe som også påvirker

valg av metode. Ved å lese av krysspunktet mellom antall involverte og hvor kritisk prosjektet er, vil man kunne fastslå hvilken metode som er hensiktsmessig å benytte slik illustrert på Figur 8.



Figur 8 ”The Crystal Methods”: valg av metode (Abrahamsson et al., 2002).

”The Crystal Methods” kan håndtere prosjektgrupper av alle størrelser slik Figur 7 viser og en iterasjon kan vare opp til 4 måneder for veldig kritiske prosjekter. Metodene har også støtte for distribuerte grupper (Cohen et al., 2004). Felles for metodene er at de benytter inkrementell utvikling hvor hvert inkrement maksimalt har en lengde på fire måneder, men normalt mellom én og tre. Fokus er i hovedsak på kommunikasjon og samarbeid mellom mennesker, og metodene tillater inkludering av andre metoder og praksiser som XP og Scrum (Abrahamsson et al., 2002).

Cockburn (2003) skriver at artikkelen hvor han presenterer disse metodene, baserer seg på to tidligere resultater. Det ene er at siden personers problemer lett kan dominere resultatet av et prosjekt, må systemutviklingsmetoder ta høyde for personers karakteristikker. Spesielt det faktum at mennesker er forskjellig må tas høyde for, sammen med deres godvilje og evne til å kommunisere på en uformell måte. Det andre resultatet Cockburn (2003) baserer sin artikkel på, er at ulike prosjekter har ulike behov når det gjelder metoder.

2.4.4 Scrum

Scrum er en utviklingsmetode først beskrevet av Hirotaka Takeuchi and Ikujiro Nonaka (Takeuchi & Nonaka, 1986). Begrepet Scrum kommer fra rugby, hvor Scrum er en egen strategi for å få en ball som er ute av spill, inn i spillet igjen (Schwaber & Beedle, 2001). Scrum kan kategoriseres under smidige utviklingsmetoder sammen med blant annet XP, men den spesifiserer ikke hvordan selve utviklingen skal utføres. Den er designet for å

fungere sammen med eksisterende praksiser. Med andre ord er Scrum ment å støtte den overordnede delen av prosjektene, mens den gir mindre støtte for den spesifikke utviklingen, som eksempelvis hvordan koding skal foregå.

Tanken bak Scrum er at systemutvikling involverer flere miljøbestemte og tekniske variabler (eks. krav, tidsramme, resurser og teknologi) som sannsynligvis vil endre seg i løpet av prosessen. Dette gjør systemutvikling uforutsigbart og komplisert, med behov for fleksibilitet og mulighet til raskt å respondere på endringer (Abrahamsson et al., 2002).

Scrum er en metode som i utgangspunktet er ment for grupper som befinner seg på samme lokasjon og har ingen innebygget støtte for distribuerte grupper (Cohen et al., 2004). Dette fordi flere av elementene i Scrum er bygget opp rundt tett samarbeid og oppdateringsmøter hvor alle i gruppen deltar ansikt-til-ansikt.

Fordi Scrum vektlegges spesielt i studien, vil metoden presenteres i mer detalj i 2.5.

2.5 Scrum

Metoden Scrum består av retningslinjer, regler og ulike roller. Den er ment for å forbedre kvaliteten ved utvikling og maksimere avkastningsgrad (ROI) (Schwaber, 2004). I Scrum finnes en rekke praksiser for å etablere et miljø hvor produkter hurtig og inkrementelt kan bygges i komplekse miljøer (Schwaber & Beedle, 2002). Disse aktivitetene kan deles inn i ulike roller, regler og retningslinjer.

2.5.1 Scrum Master

Scrum Master er ansvarlig for suksessen til utviklingsgruppen. Han er ansvarlig for å sikre at Scrumverdier, metoder og regler blir overholdt (Schwaber & Beedle, 2002, s.31). Scrum Masteren fungerer som et bindeledd mellom utviklingsgruppen og ledelsen. Han representerer gruppen for ledelsen og ledelsen for gruppen. Alle aktivitetene i Scrum er han ansvarlig for å sette ut i livet. Han organiserer og setter opp de møter som er inkludert i Scrum, samt håndterer eventuelle problemer som dukker opp.

Schwaber (2004) skriver at Scrum Masteren er som en fårehund. Den har som sin viktigste oppgave å holde flokken samlet, samt beskytte den mot farer utenfra, som ulver. Det er også hans oppgave å sørge for at arbeidet går fremover. Eventuelle hindringer som måtte oppstå skal Scrum Masteren adressere og forsøke å løse, slik at gruppen igjen kan fortsette mot sitt mål.

Gruppeledere eller prosjektledere er ofte de som fyller rollen som Scrum Master. Det er likevel viktig å skille mellom en tradisjonell prosjektleder og en Scrum Master. En Scrum Master skal ikke ta beslutninger på vegne av gruppen når det eksempelvis gjelder delegering av oppgaver, da gruppen skal være selvorganiserende (se seksjon 2.5.6).

2.5.2 "Product backlog"

Det velges å benytte den engelske betegnelsen "product backlog" da det ikke er kjent en god norsk oversettelse, så for å unngå eventuell forvirring benyttes den engelske termen.

"Product backlog" er en liste med krav i prioritert rekkefølge som man ønsker utviklet. Listen inneholder både forretningskrav og tekniske krav som skal utvikles til et system. Schwaber & Beedle (2001) skriver at "Product backlog" representerer alt som de med interesse i produktet eller prosessen har tenkt ville være en god i idé å ha med i produktet. Alt som representerer arbeid skal inkluderes i listen. Innspillene kan komme fra mange ulike hold.

Supportavdelingen vil normalt melde inn feil i produktet som må rettes, markeds-avdelingen generer gjerne funksjonalitet og funksjoner, mens salgsavdelingen eksempelvis kan legge inn elementer i listen som vil forbedre konkurranseevnen til produktet. Dette er et eksempel på hvor mange interessenter et system kan ha, noe som igjen påvirker kompleksiteten slik vist i Figur 10. "Product backlog" er et levende dokument og skal til enhver tid illustrere den høyest prioriterte funksjonaliteten ønsket implementert.

2.5.3 Produkteier

Alle Scrum-prosjekter har en produkteier. Han er ansvarlig for "product backlog" og styrer prioriteringen av oppgaver i denne listen. Produkteier er også den eneste ansvarlige for "product backlog". Dette er for at utviklingsgruppen kun skal ha én person og ett dokument å forholde seg til når det gjelder "product backlog". Dersom noen ønsker oppgaver høyere eller lavere prioritert i "product backlog" må vedkommende overbevise produkteier før en slik endring kan utføres (Schwaber & Beedle, 2001). Produkteieren er også den som er offisielt ansvarlig for prosjektet.

2.5.4 "Sprint Backlog"

"Sprint Backlog" er en liste med arbeid utviklingsgruppen vil utføre i løpet av én iterasjon, ofte en kalendermåned. Det er utviklingsgruppen selv som bestemmer hvor mye arbeid som skal inkluderes i listen, men den må velge fra toppen av "Product Backlog". Det er

med andre ord ikke opp til utviklingsgruppen å definere hva slags arbeid den ønsker å utføre, kun hvor mye.

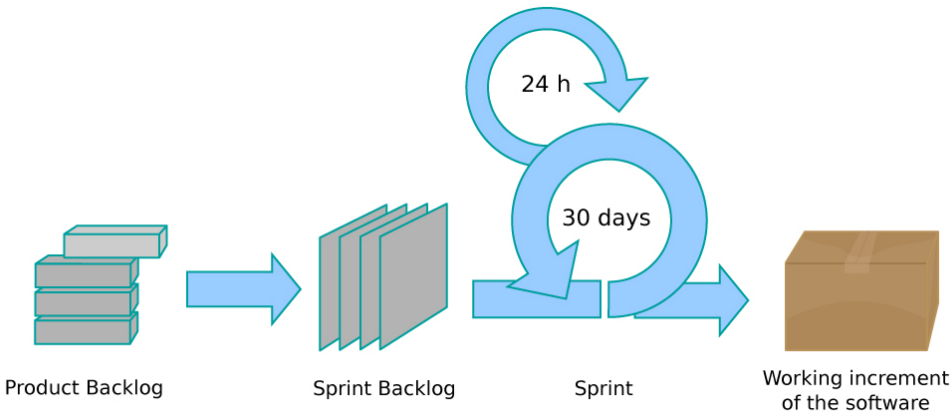
Gruppen velger den mengden arbeid den ser som sannsynlig at vil kunne gjennomføres på de 30 dagene. Etter denne perioden må utviklingsgruppen presentere et potensielt leveringsklart produkt for alle interessenter. Listen er organisert rundt hvor mange timer hver oppgave tar. Oppgavene er tildelt en utvikler som er ansvarlig for den aktuelle oppgaven. Denne tildelingen utføres av gruppen selv (Schwaber, 2004, s. 12).

2.5.5 Utviklingsgruppen

Utviklingsgruppen (”Scrum Team”) er selvorganiserende og bør bestå av ca. syv personer. Gruppen forplikter seg til å oppnå målet satt i forbindelse med utviklingen av prosjektet. Den har frie tøyler til å gjøre det den mener er riktig for å nå dette målet. Eneste begrensninger gitt er organisatoriske standarder og konvensjoner (Schwaber & Beedle, 2001, s. 36).

Det er viktig å være oppmerksom på dynamikken i gruppen. Hver person har styrker og svakheter, ulik bakgrunn og har erfaring og kompetanse oppnådd gjennom en unik utdannelse og arbeidshistorikk (Schwaber & Beedle, 2001, s. 36). Scrum er ment for å støtte hver person til å yte sitt beste i forhold til sine forutsetninger. Schwaber & Beedle (2001) anbefaler å benytte minst én svært erfaren utvikler i gruppen. Den erfarne vil veilede de uerfarne og dermed bidra til å heve kompetansen til de med minst erfaring og kompetanse.

2.5.6 Sprint



Figur 9 Illustrasjon av Scrum prosessen (Cohen et al., 2004)

En Sprint er en periode på normalt 30 dager hvor gruppen gjennomfører arbeidet det har lagt inn i "Sprint Backlog" slik illustrert i Figur 9. I denne perioden skal gruppen skjermes fra omverden for å fokusere på å løse oppgaven, som er å fullføre de oppgaver gruppen påtok seg ved utarbeidelse av "Sprint Backlog". Det er "Scrum Master" sin oppgave å sørge for at gruppen får arbeide uten innblanding slik tidligere beskrevet (Schwaber & Beedle, 2001).

I løpet av en Sprint avholdes det et "Daily Scrum Meeting". Et slikt møte arrangeres hver arbeidsdag av Scrum Master. Møtet er til for å skape god kommunikasjonsflyt gruppemedlemmene imellom og mot Scrum Master. På disse møtene blir alle stilt følgende tre spørsmål:

1 Hva har du gjort siden forrige møte?
2 Hva vil du gjøre frem til neste møte?
3 Hvilke hindringer står du ovenfor?

Det daglige Scrum-møtet skal kun vare i 15 minutter og hvert av medlemmene skal kun svare på de tre nevnte spørsmål. Eventuelle problemstillinger som kommer opp, skal adresseres på egne møter som avtales der og da. Normalt vil det være slik at Scrum Master og de / den de gjelder blir igjen etter møtet og diskuterer det som kom opp (Schwaber & Beedle, 2001).

2.5.7 Sprint planleggingsmøte

I forkant av hver Sprint arrangeres et planleggingsmøte, som består av to deler. I første del vil interessenter som kunder, brukere, ledelsen, produkteier og utviklingsgruppen avgjøre funksjonalitet og mål for neste Sprint. I andre del av planleggingsmøte deltar kun Scrum Master og utviklingsgruppen, hvor de tenker ut hvilke individuelle oppgaver som må utføres for å utvikle den ønskede funksjonaliteten. Gruppen vil så definere en "Sprint Backlog" som fastslår hva som er målet for den aktuelle Sprint.

2.5.8 Oppgaveestimering

I forbindelse med sprint planleggingsmøte må teamet estimere hvor mye tid hver enkelt oppgave, funksjon eller "user story" vil ta. Verken (Schwaber & Beedle, 2002) eller (Schwaber, 2004) skriver noe om hvordan denne oppgaveestimeringen skal foregå. Kun at det er en viktig del av Scrum at teamene selv estimerer hvor mye de kan ta på seg i løpet av en sprint.

I XP er denne oppgaveestimeringen mer forklart da XP omfatter mer detaljerte deler av systemutviklingsprosessen i forhold til hva Scrum gjør. Denne estimeringen er en så viktig del av forklaringen på hvordan teamene forplikter seg i fellesskap til å fullføre det de selv velger å ta på seg i løpet av en sprint.

Det finnes mange ulike teknikker innen oppgaveestimering. Planning poker er én av disse teknikkene som benyttes for å estimere hvor lang tid hver "user story" vil ta å utvikle. Estimering kan være en vanskelig oppgave og innen agile metoder løses utfordringer knyttet til estimering ved å velge den viktigste funksjonaliteten først (Haugen, 2006). Likevel er det behov for nøyaktig estimering og en teknikk som kan benyttes er som nevnt planning poker.

I planning poker forklarer kunden hver "user story" til teamet med utviklere. Disse diskuterer så arbeidet involvert slik at alle har nok informasjon til å kunne foreta en estimering av den aktuelle "user story". Hver uvikler estimerer oppgaven uavhengig og avslører estimeringen samtidig med de andre i teamet. Så vil utviklerne med høyest og lavest estimering redegjøre for sine estimeringer før teamet sammen blir enig om et estimat (Haugen, 2006). På denne måten forsikrer en seg om at alle utviklerne i teamet er delaktig i prosessen og at alles mening er hørt.

2.5.9 Sprint gjennomgang

Sprint gjennomgang ("Sprint Review") avholdes i slutten av hver Sprint. Denne gjennomgangen er som planleggingsmøtet delt opp i to deler. Den første delen vies til presentasjon av funksjonaliteten utviklet i løpet av den foregående Sprint. Denne delen varer 4 timer og det er utviklingsgruppen som står for presentasjonen. De øvrige deltakerne som brukere, ledelse, produkteier etc. har anledning til å komme med kommentarer (Schwaber & Beedle, 2001).

Den andre halvdelen av gjennomgangen refereres som regel til som "Sprint Retrospective Meeting". I denne delen er det kun utviklingsgruppen, Scrum Master og eventuelt produkteier som har anledning til å delta. I starten på dette møtet skal alle gruppemedlemmene besvare de to følgende spørsmål:

1 Hva gikk bra under forrige Sprint?
2 Hva bør forbedres i neste Sprint?

På den måten får gruppen anledning til å reflektere over egen arbeidsprosess og har muligheter til å gjøre nødvendig justeringer for å forbedre prosessen.

2.6 Empirisk prosesskontroll

Komplekse problemer er de som oppfører seg uforutsigbart, og det er derfor viktig å implementere empirisk prosesskontroll. Det er tre elementer som holder oppe en slik implementasjon og de er *synlighet, inspeksjon*, og *tilpasning*. Med synlighet menes at de aspekter ved prosessen som påvirker resultatet må være synlige for de som kontrollerer prosessen (Schwaber, 2004, s. 3).

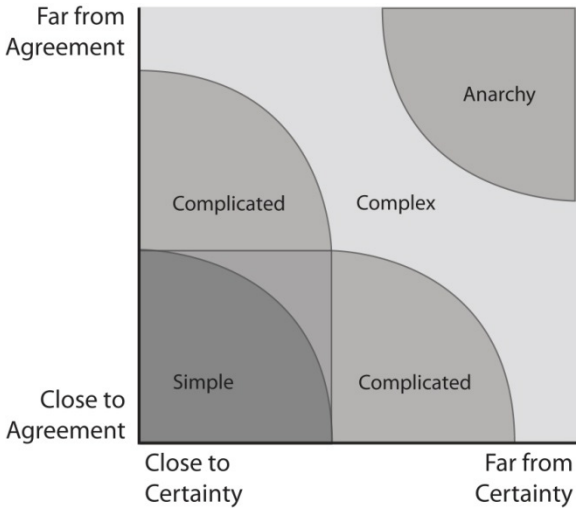
I Scrum vil en slik synlighet eksempelvis være hvorvidt en oppgave er utført. I tillegg til dette må også det som er synlig være sant. En må derfor være enig i hva som ligger i eksempelvis begrepet *utført* (Schwaber, 2004, s. 3).

De ulike sidene ved prosessen må inspiseres hyppig nok til at uakseptable variasjoner i prosessen kan avdekkes og dekkes av prosesskontrollens andre element *inspeksjon.* Det siste elementet i kontrollen er *tilpasning.* I situasjoner hvor en inspeksjon avdekker at en eller flere aspekter av prosessen er utenfor akseptable grenser, må prosessen justeres så raskt som mulig for å minimere ytterligere avvik (Schwaber, 2004, s. 4)

2.7 Kompleks programvareutvikling

Schwaber (2004) illustrerer kompleksitet i form av tre dimensjoner hvor den første er *krav*. Krav er en viktig del av all programvareutvikling og det finnes enkle programvarekrav, men ofte er det ikke slik. I mange situasjoner er mange interessenter involvert, med ulike behov som i tilegg vil forandre seg over tid og er vanskelig å artikulere.

Videre trekker Schwaber (2004) frem *teknologi* som en dimensjon, hvor han skriver at enkel teknologi nok finnes, men er sjelden brukt i programvareutvikling. Ofte må ulike typer teknologi tas i bruk for å støtte forretningskrav.



Figur 10 Kompleksitetsbedømmelsesgraf (Schwaber, 2004)

Den vertikale aksen i Figur 10 illustrerer graden av kompleksitet i forhold til *krav*, og den horisontale illustrerer kompleksitet i forhold til *teknologi*. Skjæringspunktet mellom den vertikale linjen fra prosjektets plassering på x-aksen og den horisontale linjen fra prosjektets plassering på y-aksen definerer det totale kompleksitetsnivået for prosjektet (Schwaber, 2004).

Den tredje dimensjonen er *menneskene* som utvikler programvaren. De har ulik kompetanse, intelligens, erfaring, synsvinkler, holdning og fordommer. Schwaber (2004) hevder at når denne dimensjonen legges til, går kompleksiteten gjennom taket. Han hevder at ved sammenslåing av disse tre dimensjonene fant det siste ”enkle” prosjektet sted en gang i 1969.

En kan med andre ord si at alle prosjekter i dag anses for å være komplekse systemer hvor et rammeverk å forholde seg til er viktig.

2.8 Andre smidige metoder

Ved siden av metodene som er nevnt finnes det andre metoder som ”Feature Driven Development”, ”Dynamic Systems Development Method” og ”Adaptive Software Development” som noen eksempler (Abrahamsson et al., 2002). Oppgaven velger ikke å utdype disse nærmere grunnet oppgavens omfang og fokus.

2.9 Oppsummering: smidige og tradisjonelle metoder

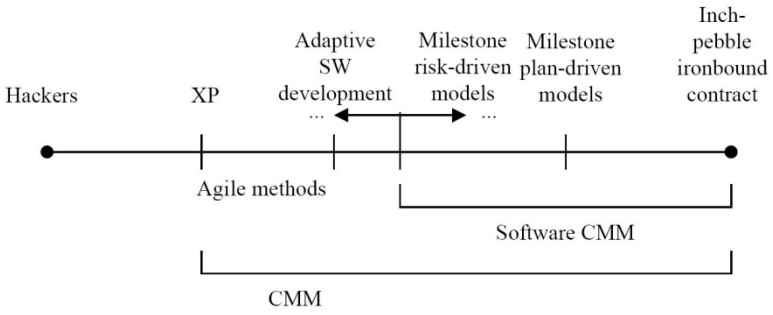
Nå som de mest relevante metodene fra smidige og tradisjonelle metoder er presentert, er det naturlig å ta en oppsummering, samt se på hovedforskjellene mellom disse. Nerur et al. (2005) presenterer en oversikt over hovedtrekkene som skiller smidige og tradisjonelle metoder gjengitt i Tabell 1, som her er oversatt til norsk.

	Tradisjonell	Smidig
Fundamentale antagelser	Systemer er fullstedig spesifiserbare, forutsigbare, og kan bygges gjennom nøye og omfattende planlegging.	Kvalitets- og tilpasningsegnet programvare kan utvikles av små grupper, hvor prinsipper om løpende designforbedring og testing basert på hyppige tilbakemeldinger blir benyttet.
Kontroll	Prosessfokusert	Menneskefokusert
Ledelsesstil	Ordre og kontroll	Lederskap og samarbeid
Kunnskapshåndtering	Eksplisitt	Taktisk
Rolletildeling	Individuelle – oppfordrer til spesialisering	Selvorganiserende grupper – oppmuntrer til rolleutveksling
Kunders rolle	Formell	Uformell
Prosjektsyklus	Viktig	Kritisk
Utviklingsmodell	Livssyklusmodell (Fossefallsmetoden, Spiralmetoden eller lignende)	Evolusjonær leveringsmodell
Ønsket organisatorisk form/struktur	Mekanistisk (byråkratisk med høy grad av formalisering)	Organisk (fleksibel og deltakende hvor samarbeid oppmuntres)
Teknologi	Ingen restriksjoner	Favoriserer objektorientert teknologi

Tabell 1 Tradisjonelle vs. smidige metoder (Nerur et al., 2005)

Tabellen viser hovedtrekkene for tradisjonelle og smidige metoder, hvor smidige metoder lett kan tenkes å passe mindre prosjekter og bedrifter da de er utviklet for enkel implementasjon. Smidige metoder håndterer endring ved å lene seg på mennesker og deres kreativitet i stedet for prosesser (Nerur et al., 2005). I mindre bedrifter kan det være mer gjennomførbart enn å fokusere på omfattende prosesser.

RUP er inkludert i tabellen da den er gjengitt i sin helhet fra Nerur et al. (2005). I denne studien er RUP plassert under tradisjonelle metoder, selv om den nok kan tilpasses både tradisjonelle og smidige metoder.



Figur 11 "The Planning Spectrum" (Boehm, 2002, s. 65).

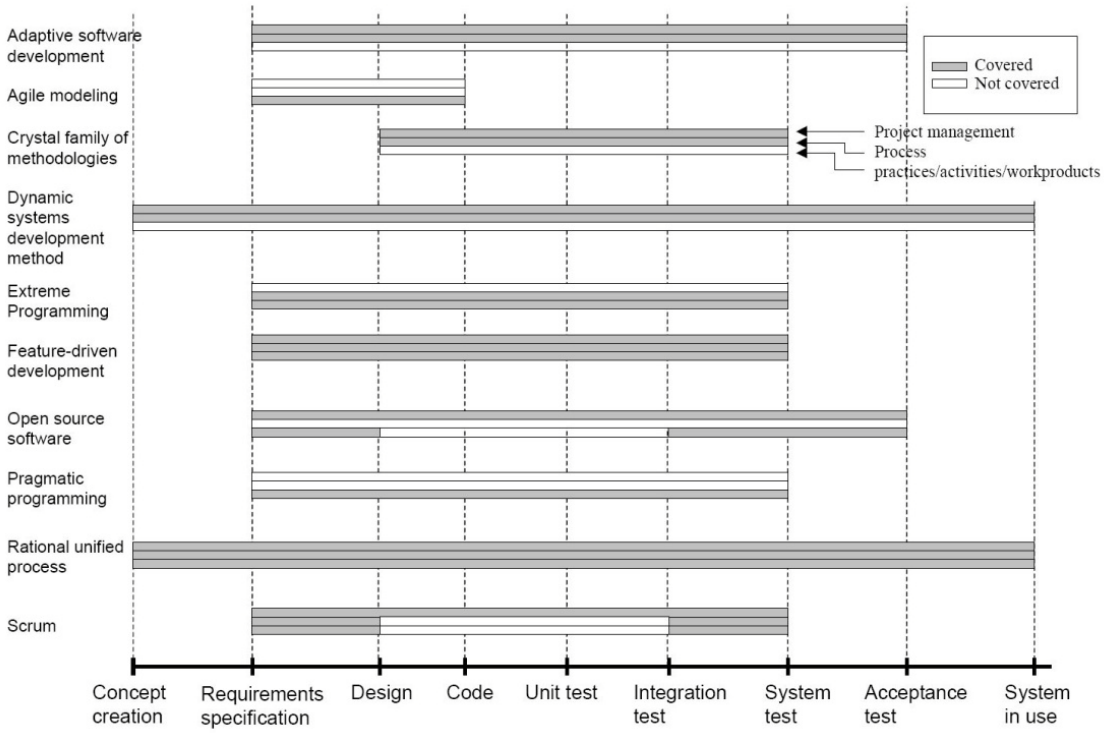
Boehm (2002) presenterer en oversikt i sin artikkel ”Get Ready for Agile Methods, with Care” over ulike metoder i forhold til hvor drevet av planlegging de er. Hackere preger venstresiden av figuren hvor de udisiplinerte metodene hvor planlegging er lavt prioritert befinner seg. Tilsvarende til høyre finner man metodene hvor planlegging og disiplin i stor grad er gjennomgående.

Metode	Nøkkelpunkter	Spesielle funksjoner	Identifiserte svakheter
Crystal	Familie av metoder. Hver har de samme underliggende kjerneverdier og prinsipper. Teknikker, regler, verktøy og standarder kan variere.	Metodiske designprinsipper. Mulighet for å velge den best egnede metode basert på prosjektets størrelse og kritikalitet.	For tidlig å identifisere svakheter. Metode under undervikling da kun to av fire foreslåtte metoder eksisterer.
XP	Kundedrevet utvikling, små grupper, daglige utgivelser.	Refakturering – fortløpende redesign av systemet for å forbedre ytelse og mottagelighet for endring.	Mens individuelle praksiser er passende for mange situasjoner, blir oversiktsbilder og ledelsespraksiser gitt liten oppmerksomhet.
RUP	Komplett programvare-utviklingsmodell inkludert verktøystøtte. Aktivitetsdrevet rolletildeling. Kan benyttes innen både tradisjonelle og smidige metoder.	Forretningsmodeller. Støtte av en familie med verktøy.	RUP har ingen begrensninger i forhold til bruksområde. En beskrivelse for hvordan skreddersy, spesielt i forhold til endrede behov mangler.
Scrum	Uavhengige, små, selvorganiserende utviklingsgrupper, 30 dagers sykluser.	Tvinger igjennom et paradigmeskifte fra "definert og gjentakbart" til fokus på utvikling av nye produkter.	Mens Scrum i detalj spesifiserer hvordan håndtere 30 dagers sykluser, gis det ingen detaljer for hvordan implementere og gjennomføre akseptansetester.

Tabell 2 Generelle kjennetegn hos smidige metoder (Abrahamsson et al., 2002)

Tabell 2 gir en kort oppsummering av de smidige metodene gjennomgått i oppgaven, samt en kort oversikt over metodenes svakheter. En ser blant annet av tabellen at Scrum ikke gir detaljert informasjon i forhold til implementering og akseptansetesting. Dette er relevant for oppgavens problemstilling da svakheter knyttet til offshoring, kan være generelle svakheter i metoden som forsterkes ved distribuert utvikling.

Figur 12 gir en oversikt over hvilke deler av utviklingsprosessen ulike metoder støtter.



Figur 12 Oversikt over hvilke deler av systemutviklingsprosessen metodene støtter (Abrahamsson et al., 2002).

2.10 **Open Source**

I tillegg tradisjonelle og smidige metoder er det relevant å trekke inn Open Source. Open Source er en betegnelse brukt om både Open Source programvare (OSS) og Open Source utvikling. I hjertet av Open Source og Fri programvare bevegelsen er prinsipper om at kildekoden til programvare skal være tilgjengelig slik at modifisering av programvaren er mulig (Feller & Fitzgerald, 2002).

Selve betegnelsen Open Source er det ingen som eier, til det er betegnelsen for generell. I de fleste sammenhenger, også her brukes begrepet Open Source om programvare distribuert under vilkår som overholder kravene i Open Source Definition (OSD). OSD er et dokument som er vedlikeholdt av The Open Source Initiative (OSI). En non-profit organisasjon startet for å fremme Open Source. Det er viktig å presisere at OSD i seg selv ikke er en lisens (Feller & Fitzgerald, 2002).

Fri og Open Source programvare referer til programvare som distribueres under vilkår som tillater brukeren å bruke programvaren, modifisere og redistribuere den til hvem en måtte ønske. Dette kan de gjøre uten å måtte betale utviklerne bak programvaren noen

form for avgift. Det finnes mange lisenser innenfor Open Source, men alle må imøtese disse essensielle kriteriene (Ockman et al., 1999):

- Retten til å lage kopier av programvaren, samt distribuere kopier
- Rett til å ha tilgang til programvarens kildekode, en nødvendighet for å kunne endre den.
- Retten til å forbedre programvaren.

Denne listen er utvidet i definisjonen av Open Source som beskriver innholdet i en gyldig Open Source lisens. Det mest vanlige lisensen som kvalifiserer som Open Source er GNU General Public License (Ockman et al., 1999).

Definisjonene for begrepene *Open Source* og *fri* programvare blir håndtert og overvåket av henholdsvis OSI og Free Software Foundation (FSI). Det er i følge (Feller, 2007) svært lite som skiller Open Source og fri programvare fra hverandre, og valget av lisensform baserer seg i hovedsak på ideologiske og ikke funksjonelle grunner.

I praktiske sammenhenger kan man med andre ord se på fri og Open Source som svært like. OSD dokumentet presenterer en del vesentlige punkter programvare distribuert under betegnelsen OSS må imøtekomme:

- **Fri redistribusjon.** Lisensen må ikke hindre noen fra å selge eller gi vekk programvaren, i sin originale eller modifiserte form. Lisensen kan ikke kreve noen form for avgift i den sammenheng.
- **Kildekode.** Programvaren må inkludere kildekoden i både opprinnelig så vel som i kompilert form.
- **Deriverte arbeider.** Lisensen må tillate modifikasjoner og deriverte arbeider. Den må videre tillate at den så blir distribuerte videre under samme lisens som den opprinnelige programvaren.
- **Integritet for forfatters kildekode.** Lisensen kan kun begrense distribusjon av modifisert kildekode, kun dersom lisensen tillater distribusjon av tilleggsmoduler som kan modifisere programvaren i ettertid hos bruker.
- **Ingen diskriminering av personer og grupper.**
- **Ingen diskriminering i forhold til hvem som kan ta i bruk programvaren.**
- **Lisensdistribusjon.** Rettighetene knyttet til programvaren må anvendes for alle som programvaren er distribuert til, uten at det skal være nødvendig å anskaffe en tilleggslisens. Dette punktet skal hindre andre fra å lukke Open Source programvare slik man ikke lengre kan modifisere og redistribuere programvaren.
- **Lisensen må ikke være spesifikk til et produkt.**
- **Lisensen må ikke være til hinder for annen programvare.**

Listen er gjengitt i forkortet utgave fra Feller & Fitzgerald (2002) hvor mer detaljert informasjon knyttet til hvert punkt er presentert.

I sammenheng med denne studien er det mest interessant å se på Open Source som utviklingsmetode og ikke som lisensform. Spesielt er det faktum at de aller fleste Open Source prosjekter er distribuerte svært interessant. Utviklerne, testerne og andre er ikke samlet på én lokasjon og har dermed ikke mulighet til å kommunisere ansikt-til-ansikt, slik det er stort fokus på i smidige metoder.

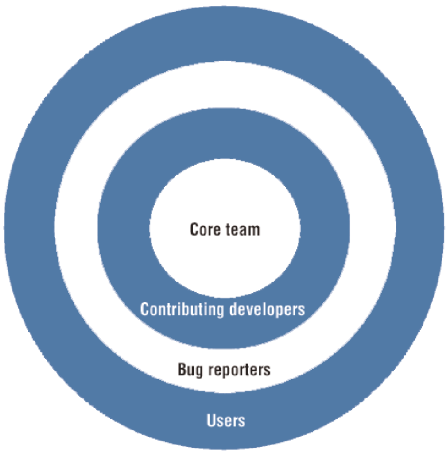
Medlemmene i Open Source-prosjekter er ofte spredt over hele verden, og knyttes sammen av internett. Metoden går hverken under betegnelsen smidig eller tradisjonell metode. Den er likevel interessant da den innehar noen fellestrekk med distribuert Scrum.

I tradisjonelle metoder har man typisk et sett av steg som gjerne er: planlegging, analyse, design og implementasjon. I Open Source utvikling er dette organisert på en annen måte. Planlegging, analyse og design fasen gjennomføres av prosjektets grunnlegger og er ikke en del av den generelle delen av Open Source utvikling (Feller & Fitzgerald, 2002).

Det gås ikke nærmere inn på bakgrunn, motivasjon eller historien bak Open Source da det ikke er relevant i denne sammenheng. Det interessante er å se på de sider ved Open Source som omhandler det å håndtere distribuert utvikling, da dette er noe som vil kunne være relevant for distribuert utvikling innen smidige metoder.

2.10.1 **Kvalitetshåndtering**

Det er flere faktorer innen Open Source som bidrar til kvalitet på det som utvikles. En av forutsetningene for god kvalitet er det å ha en stor gruppe til å utvikle kode hurtig, feilsøke effektivt og bygge ny funksjonalitet. Dette er også noe studier har vist å være den viktigste faktoren for suksess i Open Source prosjekter (Aberdour, 2007).



Figur 13 Løkmodellen (Aberdour, 2007)

Løkmodellen (Figur 13) illustrer hvordan et bærekraftig Open Source felleskap er satt sammen. Den består av en liten gruppe kjerneutviklere, samt et økende antall utviklere som bidrar mer eller mindre. De ytterste ringene i modellen består av flere feilmeldere og brukere som det er størst antall av (Aberdour, 2007).

Modularitet trekkes også frem som en viktig faktor til suksess i Open Source prosjekter. Å dele store funksjoner opp i mindre deler, gjør at man kan delegere uavhengige deler til enkeltpersoner for utvikling. Dette skaper kontroll, oversikt og sammen med høy eierskap til det man utvikler, er dette med på å bidra til suksess for prosjektet. I en undersøkelse gjort blant 100 Open Source prosjekter, ble det funnet en sammenheng mellom grad av modularitet og kvalitet (Aberdour, 2007).

Prosessen i Open Source utvikling er parallell og ikke lineær, noe som er en av de viktigste karakteristikkene av Open Source utvikling (Feller & Fitzgerald, 2002). Dette forutsetter høy grad av modularitet, med løse koblinger mellom modulene. På denne måten kan mange utvikle ulike deler av programvaren samtidig.

Prosjekter innen både Open Source og fri programvare er avhengig av mekanismer for sosial kontroll for å forhindre at prosjektet blir for komplekst (Walt, 2007). Kjernegruppen i disse prosjektene er de som styrer prosjektet. De avgjør for eksempel hva som skal inkluderes i fremtidige leveranser. Delegering av oppgaver skjer i mindre utstrekning, da en av karakteristikkene ved slike prosjekter nettopp er mangel på delegering. Perifere utviklere påtar seg oppgaver de selv ønsker å påta seg (Feller, 2007).

I Open Source prosjekter ser man også en høy grad av spontant arbeid. Dette være seg utvikling gjort uten en detaljert plan på forhånd og hvor koordinering foregår i etterkant

(Aberdour, 2007). Dette skiller seg ut fra tradisjonelle prosjekter, men har visse likhetstrekk med smidige metoder hvor det er fokus på mindre planlegging i forkant. Til slike fremgangsmåter fokuserer Open Source mye på ”peer review” og testing i etterkant for å sikre høy kvalitet.

2.10.2 **Prosjektverktøy**

Suksess i Open Source prosjekter er også basert på å ha en organisert tilnærming med sofistikerte hjelpemidler for å støtte distribuert kommunikasjon og samarbeid. Også verktøy for testing, kodehåndtering og versjonering er viktig for Open Source prosjekter. Det er verdt å merke seg at for testing benyttes ikke alltid hjelpemidler, men slike prosjekter er avhengig av en stor gruppe testere for å sikre at de fleste feil fanges opp. ”Given enough eyeballs, all bugs are shallow” (Raymond, 2001). Med dette etter hvert nokså kjente sitatet mens det at dersom en har mange nok betatestere og utviklere vil nesten ethvert problem bli fanget opp raskt og en løsning vil være åpenlys for en eller annen.

Versjonering som tidligere er nevnt, er også en av de viktigste karakteristikkene ved Open Source utvikling. Gitt den ”globale” lokaliteten blant utviklere som sjelden møtes ansikt-til-ansikt, er verktøy for konfigurasjonshåndtering kritisk for OSS. Både tilbakemeldinger og bidrag kan komme når fra hvor som helst, når som helst, og skulle slik blitt håndtert manuelt ville det krevd svært mange ressurser. Det mest vanlige verktøyet for konfigurasjonshåndtering er CVS (Concurrent Versions System), som i seg selv er et Open Source produkt (Feller & Fitzgerald, 2002).

2.10.3 **Kommunikasjon**

Den geografiske avstanden mellom medlemmer i et Open Source prosjekt skaper utfordringer for gruppen på flere måter. Denne avstanden kan føre til både dårlig og mangelfull kommunikasjon, problemer med produkt- og prosesstyring, koordineringsvansker, samt lav effektivitet og kunnskap blant medlemmene. Disse utfordringene er spesielt viktig i systemutviklingsprosjekter hvor kommunikasjon og koordinasjon er av de mest avgjørende faktorer for suksess.

En studie gjennomført av Annabi (2005) hvor gruppelæring var tema, viser at forutsetningen for læring i en gruppe, blant annet er delte mentale modeller. I XP omtales dette som metaforer og sørger for at oppgaver tolkes likt av alle medlemmene av gruppen. Regler for kommunikasjon og samhandling fører til at individuell læring kan overføres

gruppen som igjen vil føre til endring av gruppens adferd. Studien konkluderer med at gruppelæring finner sted i situasjoner hvor det er stor grad av kommunikasjon. I denne studien ble dette undersøkt ved å se på aktivitet i et diskusjonsforum som ble benyttet av prosjektet. Det er derfor først når det en person har utført integreres i gruppen og produktprosessen, at læring finner sted på gruppenivå og det bygges opp felles mentale modeller.

2.10.4 Open Source og smidige metoder

Det er likhetstrekk mellom Open Source utvikling og distribuerte smidige prosjekter. Blant annet det faktum at man ikke befinner seg i nærheten av hverandre sett i et geografisk perspektiv. For øvrig er det naturlig at noen tilsvarende kommunikasjonskanaler benyttes, som wiki, forum, e-post etc. For øvrig er det også vanskelig å trekke direkte sammenligninger mellom disse to prosjektmetodene da Open Source prosjekter normalt har svært mange mennesker involvert i utvikling i en eller annen form, sett i forhold til systemets størrelse. Det ser likevel ut til at distribuerte utgaver av smidige metoder kan trekke på erfaringer hentet fra Open Source prosjekter.

3 Relatert arbeid

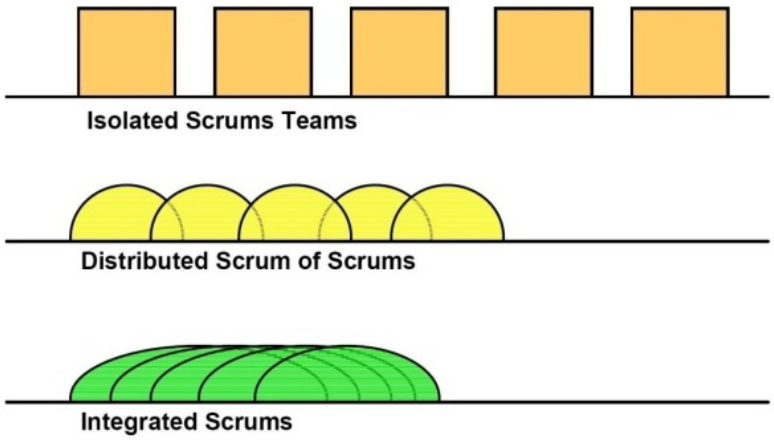
Scrum som metode er i stor grad inspirert av beste praksiser hos kjente selskaper som Toyota og Honda. Dette er med andre ord en metode som på mange måter er skapt i felten basert på hva som har fungert best (Sutherland & Schwaber, 2007).

Det er ikke gjort mange empiriske studier på Scrum. Dybå og Dingsøyr (2008) fant kun én slik studie gjort av Scrum. Studien de refererer til omhandler Scrum og dens virkning på overtid og kundetilfredshet. Det finnes derimot langt flere "lessons learned" og case studier, som presenterer verdifulle erfaringer knyttet til Scrum.

Når det gjelder Scrum og offshoring, er det først i de siste par årene at det har dukket opp noen artikler. De fleste artiklene er case studier, hvor ett prosjekt undersøkes og erfaringene knyttet til dette presenteres. Artikler som tar for seg flere lignende prosjekter for å så sammenligne erfaringene, finnes det få av.

3.1 Distribuert Scrum

Av de artikler som finnes innen Scrum og offshoring, står arbeid av Schwaber og Sutherland sentralt. Sutherlandet et al. (2007) trekker frem eksempler fra et selskap kalt SirsiDynix med en avdeling i Provo, Utah, USA og en i St. Petersburg, Russland. De benytter Scrum på begge avdelingene og disse er inkorporert med hverandre i såkalt "Distributed Scrum of Scrums" illustrert i Figur 14.



Figur 14 Strategier for distribuerte Scrum-grupper (Sutherland et al., 2007)

Figur 14 viser ulike strategier for distribuerte Scrum team. Man kan velge å benytte isolerte Scrum team, som arbeider helt uavhengig av hverandre. Disse er spredt over ulike lokaliteter og offshore teamet trenger ikke engang benytte Scrum.

Distribuert "Scrum of Scrums" er en annen strategi der man har isolerte Scrum-grupper på ulike lokasjoner, som integreres ved en Scrum of Scrums. I denne strategien møtes "Scrum Masters" for de ulike gruppene hyppig for å koordinere arbeidet mellom de ulike teamene. De danner dermed et eget Scrum team som har som oppgave å koordinere de andre Scrum teamene. Medlemmene i de enkelte Scrum teamene i distribuert "Scrum of Scrums" befinner seg lokalisert på samme sted hvor de har mulighet for tett kontakt i form av ansikt-til-ansikt kommunikasjon.

Den siste varianten er "integrated Scrums" hvor Scrum-gruppene er fullstendig distribuert. Med det menes at utviklere innen samme team kan befinne seg på totalt ulike geografiske steder (Sutherland et al., 2007). For eksempel kan team 1 ha medlemmer i USA, Canada, Norge, Finland og India, mens team 2 har medlemmer i USA, Norge, India og Australia. Medlemmene innad har i denne varianten av Scrum, ingen mulighet til ansikt-til-ansikt kommunikasjon med samtlige medlemmer i sitt team. Disse teamene er avhengig av å kommunisere via andre kanaler, som internett, telefoni, videokonferanser og lignende.

"Integrated Scrums" koordineres på samme måte som den distribuerte varianten, ved at det opprettes et Scrum team som har som oppgave å koordinere arbeidet. I eksemplet Sutherland et al. (2007) viser til er dette Scrum teamet lokalisert på samme sted, men dette presenteres ikke som et krav til denne varianten. Det er derfor naturlig å anta at dette teamet også kan være spredt over geografiske avstander på samme måte som de andre teamene.

Disse tre strategiene omfatter ulike grad av knytning mellom ulike lokaliteter. Den isolerte strategien har svært liten knytning på tvers av lokalitetene, mens den distribuerte strategien har noe tettere knytning. Den integrerte strategien består av svært tett knytning mellom utviklerne på tvers av de geografiske avstandene, noe Figur 14 forsøker å illustrere.

Sutherland et al. (2007) fokuserer på at gruppene hos SirsiDynix bestod av erfarne og dyktige utviklere i forhold til smidig utvikling. Av den grunn er det ikke nødvendigvis tilfelle at andre grupper vil kunne oppnå lignende resultater. SirsiDynix oppnådde nemlig svært gode resultater, og i følge artikkelen var dette det mest produktive prosjektet noensinne dokumentert i 2005. Dette benytter artikkelen som et "proof of point" på at det er mulig å oppnå like høy produktivitet ved offshoring av Scrum.

De gode resultatene det refereres til hos SirsiDynix var i all hovedsak knyttet opp mot antall linjer javakode, som igjen var sammenlignet med et tidligere prosjekt hvor fossefallsmetoden var benyttet.

Artikkelen trekker frem beste praksiser som firmaet etablerte i løpet av sine distribuerte Scrumprosjekter. Disse beste praksiser var (1) daglige Scrumteammøter for alle utviklerne fra de ulike lokalitetene, (2) daglige møter for produkteier-gruppen, (3) automatiske "builds" hver time fra ett sentralt forråd, (4) Ingen forskjell mellom utviklere på ulike lokaliteter fra samme team, (5) og sømløs integrasjon av XP praksiser som parprogrammering. Det fokuseres likevel ikke særlig på verken eventuelle svakheter ved Scrum i den forbindelse, eller på kvaliteten på produktene som ble utviklet av selskapet, samt hvorvidt utviklerne og brukerne var fornøyd med produktene. Dette gjør at det fremdeles er interessant å se nærmere på problemstillingen presentert i 1.1.

Cohn og Ford (2003) forteller om egne erfaringer med distribuerte Scrum-grupper. Prosjektene erfaringene er hentet fra er prosjektet de selv har vært involvert i, og hvor de ledet implementasjonen av metoden Scrum. Deres inntrykk er at Scrumprosjekter som inkluderer distribuerte grupper må skje i grupper som har jobbet minst 3 måneder med Scrum. Kulturelle og politiske forskjeller må også adresseres før utviklere starter på et prosjekt. De trekker også frem at for at et distribuert Scrumprosjekt skal heve sjansen for suksess betraktelig, må utviklerne møtes og tilbringe de første ukene sammen.

Jensen og Zilmer (2003) beskriver sine erfaringer fra en utviklingsgruppe i Danmark som benytter Scrum opp mot hovedkontoret i USA. Erfaringene de har gjort seg har vært svært positive, og de trekker frem Scrums fokus på kommunikasjon som ett av nøkkelpunktene til suksess. Ved å ha daglig kommunikasjon ved bruk av telefon, videokonferanse, e-post eller lynmeldinger[3] (instant messaging), sørget de for å ha en tett nok dialog med kontoret i USA til at problemer og misforståelser ble fanget opp og løst tidlig.

Et annet viktig poeng som trekkes frem, er behovet for å møtes fysisk ansikt-til-ansikt regelmessig for at teknologiske hjelpemidler i forhold til kommunikasjon skal en positiv effekt. I eksemplet fra Jensen og Zilmer (2003) kom utviklere og ledere fra ulike lokasjoner sammen annenhver måned. Disse møtene ble benyttet til planlegging av videre fremdrift, samt til å forsterke relasjonene de imellom.

[3] Lynmelding eller dirketemeldinger, referert til som instant messaging (im) på engelsk. Med lynmeldinger i denne studien, menes en type kommunikasjon over nett som foregår i tilnærmet sanntid og hvor klienter som eksempelvis Windows Live Messenger benyttes.

3.2 Prising og samarbeid med kunde

De fleste kunder er vant til fastpris på systemutviklingsprosjekter. De godkjenner en kravspesifikasjon som utviklerne legger til grunn for utvikling av systemet. Hvordan dette fungerer med Scrum er mindre kjent. I Scrum er en av hovedtankene at krav og behov ikke er fullt ut kjent på forhånd. Dette skaper naturligvis utfordringer knyttet til tilbudsprosessen mot kunde. Enhver bedrift som tar i bruk en metode må ta høyde for denne problematikken, da det hjelper lite hvor god metode man benytter internt om en ikke får solgt noen prosjekter.

Schwaber (2004) trekker frem problemstillinger tilknyttet fastpris og fast leveransedato. En bedrift tar kontakt med Ken Schwaber for å forhøre seg om hvordan Scrum adresserer nettopp denne problematikken, hvorpå svaret er at det gjør den ikke. Scrum sitt prinsipp er ”the art of the possible” og ikke ”you give me what I paid for, when you said that you’d deliver it” (Schwaber, 2004). Det er med andre ord fokus på å levere et så godt system som mulig i forhold til hva kunden trenger og ikke fokusere i like stor grad på kravspesifikasjon, kostnader og leveransetider.

Denne uttalelsen viser at selskaper som lever av å selge prosjekter i form av datasystemer, har en utfordring ved å bruke Scrum i forhold til å bruke tradisjonell metode. Unntakene er naturligvis kunder med kjennskap og tiltro til Scrum og som av den grunn ønsker å benytte den metoden.

Selskapet Schwaber (2004) presenterer valgte å adressere denne problematikken ved å benytte Scrum i tilbudsfasen. I denne fasen ville selskapet normalt presentere kostnaden, en dato, kvalifikasjoner, tidligere prosjekter, utviklingsmetodologi og en plan. Planen er en demonstrasjon av arbeidet som skal utføres så vel som bemanning og kostnadsestimering.

Ved å benytte Scrum til de så langt kjente kravene fra kunden i denne tilbudsfasen, kan disse inkluderes i ”product backlog” og dermed inkluderes i Scrum metoden. ”Product backlog” blir så benyttet for å vise kunden at en har forstått kravene, samt prioriteringen av disse. De mest verdifulle kravene vil da bli klassifisert som viktige mens de minst viktige får en lav prioritet. Selskapet som utvikler systemet vil da fokusere på å gjøre om krav til forretningsfunksjonalitet, mens kunden enkelt kan endre på de mindre prioriterte kravene uten nevneverdige problemer underveis i prosjektet (Schwaber, 2004).

Schwaber (2004) konkluderer med at selskapet som skal utvikle systemet vil forklare kunden at de fokuserer på å implementere de krav som gir høyest forretningsverdi, ved å følge en 80/20 modell. Den innebærer at kunden skal kunne hente ut 80 prosent

prosjektverdi fra 20 prosent funksjonalitet. Han forklarer dette med at de lavest prioriterte krav ofte er unødvendig krimskrams. Dermed kan kunden om ønskelig avslutte prosjektet når han føler de ønskede krav er oppnådd selv om det gjenstår flere i henhold til tilbudet. Dette medfører riktignok en avgift, men er likevel mer kostnadsbesparende enn å implementere den resterende funksjonalitet.

Mann og Maurer (2005) sin empiriske studie på Scrum viser en økning av kundetilfredshet ved bruk av metoden. I to år studerte de et selskap som innførte Scrum som metode i sine utviklingsprosjekter. De hentet inn data fra tre ulike kilder: logg av arbeidstid, spørreskjema og observasjoner gjort på arbeidsplassen. De benyttet altså både kvalitative og kvantitative data i studien. Fra timeregistreringssystemet kunne de hente ut kvantitative data på hvor lenge utviklerne arbeidet, mens observasjonene gav data av kvalitativ karakter.

Resultatene fra denne studien ble først synlig på slutten av prosjektet, og virker i så måte ikke inn på tilbudsprosessen så lenge kunden ikke har tidligere erfaringer fra Scrum (Mann & Maurer, 2005). Det gir likevel et argument som kan benyttes i en tilbudsprosess for å overbevise kunden om at Scrum er metoden som best kan omgjøre kundens behov til forretningsverdi.

Hole og Moe (2008) presenterer tre prosjekter hvor Scrum er forsøkt implementert i ”Global Software Development” (GSD). Gruppene var uerfarne i Scrum-sammenheng, men resultatene gav likevel verdifull informasjon. Prosjektene som beskrives finner sted i India og et østeuropeisk land. Målet med alle prosjektene var å implementere ett Scrum-prosjekt mellom Norge og eksempelvis India.

Tradisjonell GSD fokuserer i stor grad på ”ordre-og-kontroll”, formell kommunikasjon og er vanligvis implementert ved bruk av en mekanisk (byråkratisk med høy grad av formalitet) organisatorisk struktur. Smidig utvikling fokuserer på ”lederskap-og-samarbeid”, uformell kommunikasjon og ønsket om en organisk organisatorisk form (Hole & Moe, 2008). Det er med andre ord tydelige utfordringer knyttet til å implementere Scrum i et tradisjonelt GSD miljø.

Ingen av gruppene lyktes med å implementere et Scrumprosjekt på tvers av landene. De endte alle i større eller mindre grad opp med å dele opp rollen til offshore-gruppen, ved å gi dem ansvaret for spesifikke moduler. Hole og Moe (2008) konkluderer likevel med at situasjonen for disse prosjektene kunne vært annerledes om gruppene var mer modne og om offshore-gruppene hadde fått opplæring i Scrum, noe de i dette tilfellet ikke hadde fått.

En del erfaringer er det verdt å merke seg fra Hole og Moe (2008). Blant annet ble det erfart at kommunikasjonen fungerte bedre ved å tilegne en av de lokale utviklerne i offshore-gruppen til Scrum Master. Dette gav muligheten for å implementere en "Scrum of Scrums", samt at de bedret kommunikasjonen betraktelig ved at man kun behøvde å kommunisere med én person hos offshore-gruppen. I tillegg ble det vektlagt at kontinuerlig kommunikasjon var svært viktig for å lykkes med distribuert Scrum. Det mest effektive kommunikasjonsmidlet viste seg å være chat, da den minsket problemer knyttet til språkbarrierer en opplevde ved bruk av telefon og videokonferanser. Utviklerne følte det var lettere å forholde seg til skriftlig kommunikasjon, samtidig som dette var mer tidseffektivt og enkelt å sette opp, i motsetning til en videokonferanse.

Artikkelen konkluderer ellers med at ingen av gruppene klarte å oppnå gjensidig tilpasning, et viktig element for å lykkes med distribuert Scrum. En høy grad av tillit er nødvendig for å redusere direkte overvåking, kontroll og standardisering, noe som er viktig for å kunne oppnå gjensidig tilpasning.

3.3 Enigheter

De ulike studiene trekker frem ulike erfaringer knyttet til prosjekter hvor Scrum og offshoring benyttes. Det er likevel noen erfaringer som går igjen, og som av den grunn er verdt å legge merke til.

Blant annet nevner flere av studiene erfaring hos utviklerne som et viktig element. Hole & Moe (2008), Cohn & Ford (2003) og Sutherland et al. (2007) trekker alle frem at de som skal delta i prosjekter med Scrum sammen med offhoring bør være erfarne. Både erfarne som utviklere, men også erfaren med bruk av Scrum metoden. En kan kanskje si at det er en selvfølge at det går bedre med erfarne utviklere enn nyutdannede. Det at disse studiene trekker frem disse erfaringene som en av de viktigste erfaringene gjort i sine studier, sier noe om hvor viktig akkurat dette er i denne typen prosjekter.

Ved siden av å ha erfarne utviklere nevner flere av artiklene viktigheten av regelmessige møter ansikt-til-ansikt, slik som blant annet Karsten & Cannizzo (2007) gjør. Det er ikke noe bred enighet om hvor i prosessen disse møtene bør foregå, men at i løpet av denne typen prosjekter bør utviklerne møtes med jevne mellomrom for å bygge sosiale relasjoner og utveksle kunnskap seg imellom.

Videre trekkes kommunikasjon frem som et viktig element i denne typen prosjekter. Karsten & Cannizo (2007) trekker frem høy båndbredde i forbindelse med kommunikasjon, noe som kan tolkes dit hen at videokonferanser og videosamtaler er

viktig i slike prosjekter. Ellers trekkes det frem ulike erfaringer knyttet til kommunikasjon. Noen har funnet det mest hensiktsmessig å kanalisere kommunikasjon på tvers av lokasjoner gjennom få personer, mens andre prosjekter har lagt opp til bred kommunikasjon på tvers av alle utviklere og alle lokasjoner. Det er ikke mulig å trekke ut en felles konklusjon i forhold til hvordan kommunikasjon bør organiseres i slike prosjekter, men at det er en svært viktig faktor for å lykkes er nokså sikkert.

4 Metode

Metode kan ses på som et redskap eller et verktøy i forbindelse med forskning. Et slikt redskap eller verktøy benyttes for å få svar på konkrete spørsmål, eller tilegne ny kunnskap innenfor et felt (Larsen, 2007). Det er vanlig å kategorisere metode i en av to hovedkategorier – kvalitative og kvantitative metoder. Kvantitative metoder håndterer og/eller resulterer i kvantitative data, og som ordet tilsier er dette data som er målbare. Slike data kalles gjerne harddata (Larsen, 2007). Disse kan komme fra ja/nei spørsmål, spørsmål hvor det finnes et sett med svaralternativer eller hvor svarene i seg selv er tall. I tillegg kan det også for eksempel være data fra weblogger eller data generert av programmer. Resultatene fra slike undersøkelser er tall og kan presenteres på ulike måter som eksempelvis i form av grafer.

Kvalitative metoder gir kvalitative data, som sier noe om kvalitative (ikke-tallfestbare) egenskaper (Larsen, 2007). I motsetning til dataene i kvantitative metoder kalles disse dataene ofte myk-data.

Larsen (2007) gir en god oversikt over de viktigste forskjellene mellom de to metodene, basert på blant annet (Hellevik, 2002) gjengitt i Tabell 3.

	Kvantitative metoder	Kvalitative metoder
Problemstilling	Spørsmål og hypoteser	Spørsmål og temabeskrivelser
Enheter og variabler	Bredde: få opplysninger om mange enheter	Dybde: mange opplysninger om få enheter
Innsamlingsmetoder	Systematisk og strukturert (Eksempelvis faste spørsmål i samme rekkefølge via et spørreskjema)	Ustrukturert (eller mindre strukturert) eks. Uformelle intervjuer
Presentasjon av data	Tall i form av tabeller og figurer	Illustrasjoner ved sitater
Arbeidsform	Forholdsvis liten fleksibilitet. Arbeider med fasene nokså adskilt.	Stor fleksibilitet. Fasene er ikke så adskilte, det er ofte slik at utviklingen av problemstillingen, datainnsamlingen og analysen foregår til dels samtidig og i flere runder.
Nytte	Kan generalisere	Kan normalt ikke generalisere, men undersøkelsen kan ha overførbarhet.
Type informasjon	Kan sammenligne. Systematisk klassifisering og opptelling av noen utvalgte egenskaper.	Helhet og fullstendighet, ønsker å se mønster i helheten av egenskaper.

Tabell 3 Forskjellene mellom kvantitative og kvalitative metoder (Larsen, 2007)

4.1 Eksplorerende, beskrivende og forklarende undersøkelse

Kvalitative studier kan basere seg på flere ulike underliggende filosofiske antagelser (Myers, 1997).

Hellevik (2002) beskriver et sett med ulike undersøkelser hvor den mest vanlige i situasjoner hvor man mangler et godt begrepsskjema er *eksplorerende* undersøkelse. Her er målsetningen å komme frem til en mer presis problemstilling, som så kan undersøkes i en senere studie.

Beskrivende undersøkelse benyttes når problemstillingen er mer presist formulert, og man kan foreta en grundigere og mer systematisk studie av enhetene. En slik undersøkelse kan gå ut på å gi en så nøyaktig beskrivelse av ulike trekk ved enhetene som mulig (Hellevik, 2002).

Den siste av disse er *forklarende* undersøkelse hvor målet er å påvise årsaker til mønstre en finner. En slik undersøkelse er vanlig når det foreligger omfattende kunnskaper om emnet (Hellevik, 2002). Forklarende undersøkelse baserer seg på antagelsen om at vår kunnskap om virkeligheten er en sosial konstruksjon av menneskelige faktorer. Ulike faktorer som språk, bevissthet og delte antagelser gir oss tilgang til virkeligheten (Myers, 1997).

4.2 Ekstensivt vs. Intensivt

Valg av kvalitativ eller kvantitativ metode legger føringer for valg av ekstensivt eller intensivt undersøkelsesopplegg. I en ekstensiv undersøkelse vil mange enheter undersøkes med få variabler. Man ønsker ved et slikt opplegg å se på ett eller noen få aspekter ved svært mange enheter. Hellevik (2002) trekker frem et eksempel hvor en ønsker å se på hvem som tar initiativ til å få saker behandlet i et kommunestyre. I dette eksemplet ville variabelen vært hvem som tok initiativ til å få saken behandlet og enhetene ville vært de aktuelle sakene.

I et intensivt undersøkelsesopplegg, som er mest vanlig i kvalitative studier, velger man å gå i dybden ved å undersøke mange variabler hos få enheter. En ville eksempelvis forholde seg til én enhet hvor mange variabler ble undersøkt, noe som kalles en case studie. Case studie er den mest vanlige formen for kvalitativ undersøkelse i informasjonssystemer (Myers, 1997).

Skulle man kombinere ekstensivt sammen med kvalitativ metode, sier det seg selv at det fort kan bli en omfattende studie. Kombinasjonene ekstensiv – kvantitativ og intensiv – kvalitativ innehar begge lignende størrelsesomfang. Førstnevnte undersøker man mange, men går ikke så dypt i hvert objekt, mens den andre går dypt i hvert objekt men undersøker ikke så mange. Dermed vil en kombinasjon av ekstensiv og kvalitativ innebære at man skal undersøke mye og mange, dersom det er mennesker det dreier seg om. Det er

også årsaken til at man har naturlige kombinasjoner mellom disse undersøkelsesoppleggene og metodene.

4.3 Etnografisk studie

Etnografisk undersøkelse kommer fra sosial og kulturell antropologi hvor en etnograf tilbringer svært mye tid ute i felten. Etnografer fordyper seg i livene til de menneskene de studerer og søker å plassere fenomenet som studeres i deres sosiale og kulturelle kontekst (Myers, 1997).

Ved å utføre en etnografisk studie i en organisasjon må forskerne observere deres omgivelser uten fordommer eller tidligere antagelser. De deltar i organisasjonen til de som blir observert og deltar i samtaler, er med i møter, leser dokumenter osv. Etnografi er en systematisk tilnærming som fører til empirisk validerbare konklusjoner (Sharp & Robinson, 2004).

4.4 Observasjon og utspørring

Det finnes mange former for observasjon i forbindelse med forskning. Hellevik (2002) deler observasjon inn i to hovedkategorier: feltstudier og laboratorieundersøkelser. I feltstudier kan observatøren innta mange roller. Forskeren kan velge å innta en deltakende eller ikke-deltakende rolle. Innenfor deltakende roller kan forskeren velge enten å være passiv eller aktiv i sin deltakende observasjon. I aktiv deltakende observasjon spiller forskeren en aktiv rolle i forhold til hva som skal skje innenfor det sosiale systemet som undersøkes (Hellevik, 2002).

I passiv deltakende observasjon, vil forskeren ikke ha en aktiv rolle i forhold til hva som skal skje. Forskeren forsøker på ingen måte å skjule for de som observeres at de er deltakende i en studie (Hellevik, 2002).

Når forskeren velger å innta en ikke-deltakende rolle, vil han ikke delta blant aktørene som studeres, men iaktta dem fra en avstand. En slik type undersøkelse vil ofte medføre at de som observeres, ikke er klar over at de blir observert.

Laboratorieundersøkelser utføres i svært kontrollerte og kunstige omgivelser. En vil ved slike undersøkelser isolere enhetene fra sine naturlige omgivelser, slik at forskeren har mulighet til å kontrollere omgivelser som han tror kan påvirke resultatet. Dette kan eksempelvis være hvordan rommet er møblert, eller muligheten for observatører til å sitte bak et enveisspeil for ikke å påvirke enhetene (Hellevik, 2002).

I tillegg eller i stedet for observasjon, kan en studie utføres ved hjelp av utspørring. Utspørring kan foregå ved hjelp av et spørreskjema, noe som er svært vanlig innenfor kvantitative undersøkelser. I en kvalitativ undersøkelse slik oppgaven benytter, er intervju en vanlig form for utspørring.

Intervjuer kan organiseres på ulike måter. Normalt snakker en om strukturerte, ustrukturerte og semi-strukturerte intervjuer. I strukturerte intervjuer følger intervjuer et fast skjema, fastlagt på forhånd og det er ikke rom for improvisasjon. Slike intervjuer trenger nødvendigvis ikke holdes av forskeren (Myers & Newman, 2006). I ustrukturerte og semi-strukturerte intervjuer følger man ikke et fastlagt skjema, men har gjerne forberedt en del spørsmål på forhånd. Intervjueren er som regel forskeren. Dette er den mest vanlige formen for intervju innenfor informasjonssystemer (Myers & Newman, 2006).

5 Forskningsdesign

Oppgaven har som mål å kartlegge hvordan Scrum fungerer i forbindelse med offshoring av Scrum prosjekter. Den baserer seg på intervjuer av sentrale personer i Accenture som har hatt roller i Scrum prosjekter hvor offshoring har vært benyttet.

Det ses på som mest hensiktmessig å utføre en kvalitativ studie da de er få enheter som undersøkes, samt at relevant forskning på området er begrenset. Av den grunn er domenet mindre kjent og en kvalitativ tilnærming vil i tillegg kunne gi innsikt og svar på spørsmål som ikke er formulert på forhånd.

5.1 Intervjuguide

Intervjuguiden (se vedlegg 1) som intervjuene er basert på, er formulert med tanke på å besvare de to spørsmålene fra problemstillingen:

- Hvordan har de aktuelle prosjektene implementert Scrum sammen med offshoring?
- Hva har vært utfordringene knyttet til bruk av Scrum sammen med offshoring, og hvordan har de aktuelle prosjektene møtt disse utfordringene?

Det er ikke mulig å gjennomføre observasjoner i de aktuelle prosjektene, da disse ikke er utført i Norge. Dataene vil derfor i sin helhet komme fra telefon og ansikt-til-ansikt intervjuer med sentrale personer knyttet til prosjektene. Disse vil inneha roller som Scrum Master, Produkteier eller veiledere. På denne måten er det mulig å få et bredt og sammensatt bilde av hvordan prosjektene fungerer for ulike parter. Deres oppfatninger og tolkning av prosjektene vil bli dataene denne studien baserer sine konklusjoner på. Disse erfaringene vil bli satt opp mot hverandre i et forsøk på å trekke ut felles erfaringer som vil kunne gi en indikasjon på generelle problemstillinger knyttet til Scrum-prosjekter hvor offshoring benyttes.

Det benyttes semi-strukturerte intervjuer, som tilsier at det ikke vil følges en fastsatt plan for intervjuene. Det vil derimot planlegges noen sentrale spørsmål som vil bli stilt, mens resten av intervjuet vil basere seg på en dialog rundt utfordringene knyttet til Scrum og offshoring. Faste spørsmål som tas opp er først generelle faktabaserte spørsmål knyttet til prosjektet, som varighet, omfang, antall mennesker involvert og lignende. Videre vil spørsmål knyttet til opplæring av ressursene innen metode og domene for prosjektet, samt hvordan den generelle oppstarten av prosjektet er organisert. Videre vil spørsmål knyttet til

kommunikasjon og sosiale relasjoner stå sentralt. Det vil i tillegg stilles spørsmål for hvorvidt de anser prosjektet å være vellykket, noe som baseres på kriterier presentert i 5.2.

Av mer generelle spørsmål vil det tas opp hvilke utfordringer de ulike prosjektene har møtt og hvordan disse har blitt adressert. Om prosjektene benytter andre metoder enn Scrum ses også på som relevant, i tillegg til om det er praksiser hos de ulike metodene som ikke benyttet.

Prosjektene har svært ulike forutsetninger slik at det vil være svært vanskelig å følge en fastlagt spørsmålsplan for alle intervjuene. Med dette menes at noen prosjekter kan ha lang erfaring med offshoring men ikke smidige metoder, mens andre kan har brukt smidige metoder i lang tid og har nylig tatt i bruk offshoring. Det er derfor forsøkt å holde intervjuene så åpne som mulig for å forsøke å fange opp disse forskjellene mellom prosjektene.

5.2 Undersøkelsesopplegg

I valget mellom ekstensivt eller intensivt undersøkelsesopplegg er det i denne sammenheng valgt et intensivt undersøkelsesopplegg. Det er fem personer intervjuet noe som tilsier at undersøkelsen ikke er av en ekstensiv karakter. I intensive undersøkelser går man mer i dybden hos hvert objekt enn ved ekstensive undersøkelser. Dette er fordelaktig for oppgaven da den søker forståelse for hvordan Scrum er implementert sammen med offshoring. Helt konkrete spørsmål er ikke definert på forhånd og det ses derfor på som mest fordelaktig å bruke mer tid pr. objekt, slik at en lettere kan fange opp problemstilling som ikke var tenkt ut på forhånd.

Videre er det valgt en kvalitativ metode, som er det mest vanlige å velge sammen med et intensivt undersøkelsesopplegg. Dette vil hjelpe med å oppdage ukjente problemstillinger som muligens kan være utgangspunkt for andre studier.

I tillegg til spørsmålene formulert i problemstillingen er det også avgjørende å hente inn data for hvorvidt de ulike prosjektene har vært en suksess eller ei. Dette vil basere seg på uttalelser fra intervjupersonen på hvorvidt de mener prosjektet er vellykket, samt om kunden er fornøyd med produktet. Om prosjektene er levert i henhold til tidsfrister og innenfor de kostnadsrammer satt av kunden vil også være faktorer knyttet til suksess.

Studien vil i hovedsak ligge mellom en *forklarende* undersøkelse og *eksplorerende* undersøkelse på grunn av et delvis manglende begrepsskjema. Det å inkludere en

eksplorerende undersøkelse vil hjelpe til med å oppdage nye problemstillinger som ikke er kjent på forhånd.

5.3 Oversikt over prosjektene

Figur 15 viser en grafisk fremstilling av den geografiske spredningen mellom prosjektene involvert i denne studien. Som denne illustrerer spenner flere av prosjektene seg over svært store avstander. De fleste prosjektene spenner kun over to lokaliteter, mens noen av prosjektene har opptil fire lokaliteter. Parentesene representerer hvilke prosjekter som er lokalisert hvor. Dette er videreført i resten av studien. Alle tall i parentes etter ordet prosjekt eller navnet på et prosjekt referer til prosjektnummeret. Disse numrene spesifiseres i de neste avsnittene. Altså betyr eksempelvis tallene i parentes bak England, at prosjekt 8 og 9 blant annet er lokalisert der. Prosjektene er nummerert for å spare plass når det refereres til prosjektene.

Figur 15 Oversikt over steder involvert i prosjektene (prosjektnummeret i parentes)

5.4 Fem prosjekter ved Accenture, Riga (1-5)

Kaspars Balodis og Andris Vanags har vært involvert i fire fullførte og ett pågående prosjekt hvor Scrum og offshoring har vært og blir benyttet. Kaspars Balodis er NetCentric CG Lead hos Accenture i Riga og innehar tittelen Manager. Han har innehatt rollen som produkteier i flere av prosjektene, men også vært den som har solgt inn flere av prosjektene

til kunder. Andris Vanags er senior systemanalytiker og er Scrum Master for alle prosjektene.

Hos den belgiske logistikkorganisasjonen har Accenture så langt vært involvert i fem prosjekter hvor Scrum har vært benyttet sammen med offshoring. Prosjektene innebærer en nettbasert applikasjon for håndtering av logistikk. Applikasjonen blir brukt av blant annet distribusjonskontorer og postsentraler. Prosjektene inkluderte mellom fire til seks medlemmer, hvor flesteparten befant seg i Riga. Logistikkorganisasjonen har tidligere benyttet Scrum og er godt kjent med metoden. Vanags er Scrum Master i de nevnte fem prosjektene, hvorpå ett av prosjektene p.t. ikke er avsluttet.

Figur 16 Geografisk plassering for prosjekt 1-5

Prosjekt hos stor logistikkorganisasjon, Belgia (1)

Det første prosjektet var et pilotprosjekt, som hadde som mål å bygge kompetanse, lære om verktøy, teknikker og metode. Prosjektet bestod av seks medlemmer hvor to av dem var sertifiserte Scrum Mastere, men de fire resterende var ny i forhold til Scrum og fikk intern opplæring. Prosjektet hadde en varighet på seks måneder.

Prosjekt hos stor logistikkorganisasjon, Belgia (2)

Det andre prosjektet utført hos logistikkorganisasjonen var et lavt prioritert vedlikeholdsprosjekt, som var fullt distribuert og varte i tre måneder. Prosjektet ble startet opp like etter det første prosjektet var avsluttet. Prosjektet bestod av totalt syv medlemmer hvor fire var offshore og tre onshore.

Prosjekt hos stor logistikkorganisasjon, Belgia (3-4)

Det tredje og fjerde prosjektet startet utviklingen av deler av den nettbaserte applikasjonen. Disse prosjektene inneholdt tre utviklere hvor alle tre var offshore i Latvia, mens produkteier var onshore i Belgia. Begge prosjektene hadde en varighet på ca. fire måneder

Prosjekt hos stor logistikkorganisasjon, Belgia (5)

Det femte og siste prosjektet Andris Vanags har hatt en rolle i pågår fremdeles og har pr. januar 2009 pågått i fem måneder med fem utviklere. Prosjektet omfatter en modul i den nettbaserte applikasjonen.

5.5 To prosjekter ved Accenture, USA (6-7)

John Wilson arbeider ved Atlanta- og Chicago-kontoret til Accenture i USA. Han er manager og har lang erfaring innen smidig metode og deltatt som veileder (”coach”) i en rekke prosjekter hvor smidige metoder har vært brukt. I de siste to prosjektene han har deltatt i, benyttes Scrum sammen med offshoring. Det første prosjektet er lokalisert på østkysten av USA og har en offshore gruppe i Mumbai, India. Det andre prosjektet er lokalisert i USA, Amsterdam, Singapore og India.

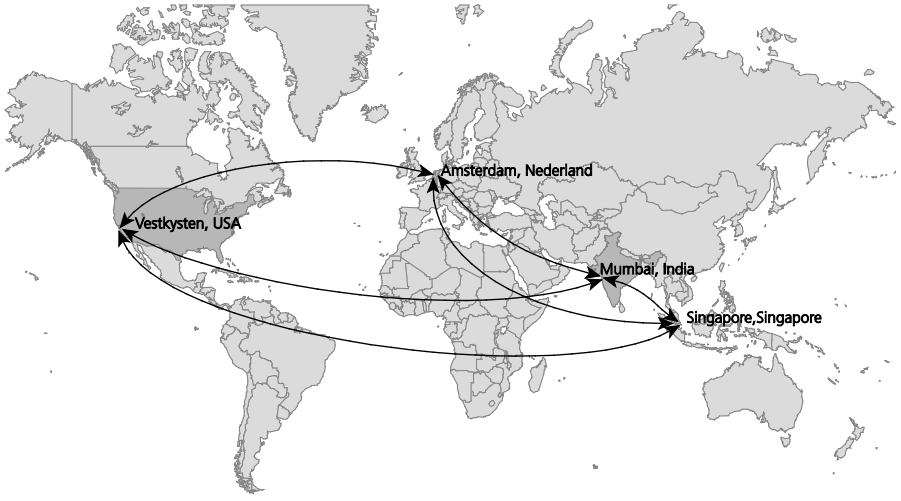
Prosjekt hos finansinstitusjon (6)

Wilson ble hentet inn av et stort finanskonsern i USA for rådgivning vedrørende innføring av Scrum i deres prosjekter. Selskapet har planlagt å bytte ut et 15 år gammelt system som driver 2 500 kontorer i Nord Amerika og påvirker rundt 12 000 mennesker hver dag. Det gamle systemet kjører på en stormaskin mens det nye systemet skal baseres på klient-server arkitektur. Prosjektet er planlagt å vare ca. tre år og består av flere faser. Prosjektet er ca. ett år gammelt (p.t. januar 2009). Prosjektets geografiske spredning er illustrert i Figur 17.

Figur 17 Geografisk plassering for prosjekt 6

Prosjekt hos teknologiselskap innen telekom (7)

Det andre prosjektet Wilson er involvert i, er et middels stort prosjekt hos et betydelig teknologiselskap i USA. Prosjektet inneholder rundt 35 personer fordelt på fem Scrum team som hver består av mellom fem og åtte personer. Prosjektet er å utvide et salgssystem som skal støtte selgere i sitt daglige virke. Systemet skal håndtere klienter, kontrakter, kommunikasjon etc. Systemene til firmaet selges videre til blant annet IBM, som igjen inkluderer det i sine systemer. Prosjektet har så langt var i ett år og er planlagt å vare ett år til.



Figur 18 Geografisk plassering for prosjekt 7

Prosjektet er satt opp til å kjøre en fullt distribuert utgave av Scrum hvor to team er lokalisert på vestkysten av USA, ett team er fullt distribuert mellom USA og India, det

fjerde teamet er splittet mellom USA, Amsterdam og India, mens det siste teamet er splittet mellom USA, Singapore og India. Det var ikke lagt opp til samling av teamene i starten av prosjektet så teammedlemmene lokalisert i de ulike delene av verden hadde aldri tidligere hilst på hverandre.

5.6 STS prosjekt i England og India (8)

Figur 19 Geografisk plassing for prosjekt 8

Kunden er ledende innen finansielle tjenester og blant USAs ti største finansinstitusjoner. De tilbyr et bredt spekter av finansielle tjenester og løsninger på tvers av Amerika, Europa og Asia. STS er en pakke med systemer for å gjennomføre handel på tvers av investeringsbanker. Det er et virksomhetskritisk bakenforliggende system som skal byttes ut, noe som setter store krav til kvaliteten til det nye systemet som er under utvikling. STS omfatter funksjonelle områder som regnskap, oppgjør, dokumentasjon og etableringstjenester.

Prosjektet styres fra England, mens størsteparten av utviklingen foregår i India. Rakesh Jaiswal fungerer som produkteier for prosjektet. Prosjektet har fem personer i England og 45 i Mumbai, India. Det er et fremdeles pågående prosjekt og har så langt var ca. ett og et halvt år.

5.7 Fire prosjekter ved Accenture sitt leveransesenter i India (9-12)

De siste prosjektene studert er fire prosjekter hos leveransesenteret til Accenture i India. Dr. Raj Mokashi har vært involvert i disse prosjektene som prosjektleder. Prosjektene han har vært involvert i som er omtalt i studien, har foregått på bortimot samme tid. DART Amazon Next, England, USA og India (9)

Figur 20 Geografisk plassering for prosjekt 9

For et internasjonalt finanskonsern leverer Accenture sitt leveransesenter i India (IDC) et Java-prosjekt. IDC deltar sammen med to prosjektmedlemmer i England og to i USA, hvor prosjektet blir styrt fra. I India befinner det seg 22 utviklere som arbeider på prosjektet. Det er organisert slik at de i India står for design, implementasjon og test, mens planlegging, arkitektur og akseptansetester drives fra England og USA. Prosjektet er p.t. pågående og har så langt vart i ett år, noe som betyr at de allerede har høstet mange verdifulle erfaringer.

Elektronisk bestillingssystem, Nederland og India (10)

For et ledende europeisk flyselskap utvikler IDC et elektronisk bestillingssystem for flyreiser basert på SOA (Service Oriented Architecture). Prosjektet er svært komplisert men innehar mange likhetstrekk med DART Amazon prosjektet. Bestillingssystemet skal benyttes av brukere, agenter og ansatte i selskapet. Det skal også fasilitere bestilling av hotell og leiebil i tillegg til flyreiser som er ens primære produkt. Prosjektet har 22 utviklere offshore i India og 12 onshore i Nederland.

Figur 21 Geografisk plassering for prosjekt 10

Applikasjon for én av verdens ledende mobilprodusenter (11)

Figur 22 Geografisk plassering for prosjekt 11

Det tredje prosjektet deler mange likheter med de to tidligere omtalte prosjektene. Dette prosjektet hadde som oppgave å utvikle en office-kommunikator for mobiltelefoner som benytter Symbian som operativsystem. Ni utviklere var offshore (India) mens to var onshore (Finland) i dette aktuelle prosjektet.

Applikasjon for en av verdens største bilprodusenter (12)

For en av de aller største bilprodusentene i verden er det under utvikling en applikasjon til bruk for forhandlerne av bilmerket. Systemet skal støtte blant annet salg og økonomi. Prosjektet har ni utviklere offshore i India mens fem er onshore i USA.

Figur 23 Geografisk plassering for prosjekt 12

5.8 Intervjuanalyse

I analysen av intervjuene i studien vil oppgaven i stor grad trekke ut elementer som omhandler de spørsmål definert i intervjuguiden. Da det er til en viss grad er gjennomført et *eksplorerende* undersøkelsesopplegg vil det forsøkes å ta høyde for problemstillinger som ikke er definert på forhånd. For å kunne fange opp disse elementene er listen over problemstillinger under nyttig som utgangspunkt i analysen av intervjuene.

Det er ikke lagt opp til en fast liste med nøkkelord som oppgaven vil ta utgangspunkt i. Intervjuobjektene i studien er fra ulike kulturer hvor begrep og formuleringer er svært ulike. Det ses derfor på som risikofylt å definere ord og begreper på forhånd som skal benyttes for å trekke ut informasjon fra intervjuene. I stedet vil begreper knyttet til punktene under bli plukket ut for å forsøke å dekke over eventuelle språklige og kulturelle forskjeller blant intervjupersonene.

I transkriberingen av intervjuene er det oversatt fra engelsk til norsk, slik at de kan benyttes som sitater i studien, hvor det er hensiktsmessig. Ved å oversette dem, antas det å være lettere å sammenligne intervjuene seg imellom da begreper og formulering vil være mer uniform enn i sin opprinnelige form. Det naturligvis en risiko for at formuleringer endrer

seg i en oversettelsesfase, men det ses på som mindre sannsynlig at selve budskapet i svarene vil endre seg vesentlig.

Eksempel på deler av transkribering av to intervjuer er vedlagt (vedlegg 2) hvor en ser spørsmål og svar som er gitt. Disse svar ble så gjennomsøkt etter begreper og ord som passet inn med problemstillingene presentert under. Hvor det ble funnet slike begreper ble disse merket (på papir) slik at en lett kunne finne tilbake til disse når en skulle sammenligne informasjon fra de ulike intervjuene.

Problemstillinger som er aktuelle for studien:

- Hvordan prosjektene etableres (Oppstartsfasen) (nøkkelord: i starten av prosjektet, oppstartsfasen etc.)
- Informasjon om involverte utviklere (Generell erfaring, opplæring, domenekunnskap og kunnskap om Scrum).
- Varighet, størrelse og kompleksitet på prosjektet.
- Bruk av metoder (Hvilke metoder benyttes i tillegg til Scrum? Hva praksiser er eventuelt utelatt? Hvilke nye praksiser er inkludert?).
- Problemer og utfordringer underveis knyttet til offshoring (Interne problemstillinger ikke relevant).
- Kommunikasjon mellom geografiske lokaliteter (Valg av kommunikasjonskanaler, erfaringer knyttet til bruk av disse).
- Er prosjektet en suksess (Karakteriserer intervjuobjektet prosjektet som vellykket? Er kunden fornøyd? Er systemet levert til avtalt tid eller er i rute?).
- Endringer gjennomført underveis.
- Tilbakemeldinger fra kunder, utviklere og brukere.

Disse problemstillingene eller elementene danner grunnlaget for analysen av prosjektene. Hva som er relevant for oppgaven vil i stor grad styres av disse punktene. Det er naturlig at noen intervjuer går mer i dybden på noen av punktene enn andre, og dette ses på som unngåelig. Det er naturlig at intervjuobjektet ikke nødvendigvis har svar på alle spørsmål som stilles. Da de gjennomføres flere intervjuer og disse omhandler en betydelig mengde prosjektet ses ikke dette på som et problem. Sannsynligheten for at alle problemstillinger vil dekkes av ett eller flere intervjuer ses på som sannsynlig.

Intervjuobjektene omtaler ofte prosjektene som en suksess eller ikke. Det alltid utfordrende å vite hva som til enhver tid ligger i begrepet suksess. Av den grunn vil det i denne studien legges noen føringer får hva som er suksess eller ikke. Et prosjekt defineres som suksess når det er levert til riktig tid og pris, tilbakemeldingene fra kunde er positive, samt at både

utviklere og prosjektledelsen selv karakteriserer prosjektet som en suksess. Kvaliteten på produktet er ikke medberegnet da det ikke er mulig å måle i denne studien.

6 Datainnsamling

Det vil i dette kapittelet beskrives hvordan datainnsamlingen for studien har foregått. Hvordan intervjuer har blitt holdt og hvordan prosjekter er valgt ut. I tillegg vil det presenteres kriterier for hva som er ekstrahert fra intervjuene.

Figur 24 viser datainnsamlingsprosessen, hvor prosjektutvelgelsen var første del av denne. Videre ble det etablert kontakt med relevante personer knyttet til prosjektene, før intervju ble holdt. Disse ble så transkribert før analysen ble foretatt på bakgrunn av de problemstillinger nevnt i forrige kapittel.

Figur 24 Datainnsamlingsprosess

6.1 Prosjektutvelgelse

Det er i denne studien intervjuet fem aktører som til sammen har vært involvert i tolv små og store prosjekter. For å finne frem til de aktuelle prosjektene er forfatters eget nettverk i Accenture benyttet for å finne prosjekter hvor Scrum og offshoring benyttes. I tillegg er skriftlige ressurser som egne internettfora for smidige prosjekter tatt i bruk.

Det har ikke vært behov for å foreta utvelgelse av hvilke prosjekter som skulle inkluderes i studien da alle prosjekter som er blitt gjort kjent, samt hvor det var mulig å få til et intervju, er inkludert i studien. Det har vært viktig at prosjektene studert har hatt base i ulike land, slik at resultatene er mer universelle enn om alle prosjektene var lokalisert i eksempelvis Norge og India.

6.2 Intervjuene

Intervjuene i studien ble satt opp ved at intervjuobjektene først ble kontaktet pr. e-post hvor informasjon om studien ble gitt. Deretter ble noen få faktaopplysninger forespurt som hva heter det aktuelle prosjektet, hvor mange utviklere deltar og hvilke land foregår prosjektet i. Da de mest grunnleggende faktaopplysningene knyttet til prosjektene var innhentet ble intervju avtalt. De ble her også forespurt hvilken intervjuform som var å

foretrekke for intervjuobjektet. Dette ble gjort for å sikre at intervjuobjektet var i en kjent og komfortabel setting.

Kaspars Balodis og Andris Vanags

Balodis og Vanags er involvert i prosjektene (1-5) som er lokalisert i Belgia og Latvia. Begge disse kommer fra Latvia og arbeidet med utgangspunkt fra Riga. Disse ble kontaktet via e-post med forespørsel om de kunne stille til et intervju. Det viste seg så at disse skulle til Oslo i forbindelse med et arrangement arrangert som arrangeres hvert år av Accenture. I den forbindelse med intervjuet planlagt å gjennomføre ansikt-til-ansikt. Intervjuet varte én time og 30 minutter og ble tatt opp ved hjelp av opptaksfunksjon på mobiltelefon. Intervjuet inkluderte både Balodis og Vanags hvor de hver svarte på de spørsmålene de hadde mest kunnskap og erfaring om.

John Wilson

Wilson ble kontaktet pr. e-post og et intervju ble avtalt. Da Wilson befant seg i dette tidsrommet i Chicago, USA, ble møtet avtalt å holde via telefon. Applikasjonen Skype[4] ble benyttet for å gjennomføre samtalen, som ble tatt opp ved hjelp av en opptaksapplikasjon[5] kompatibel med Skype. Intervjuet med Wilson varte én time og 45 minutter og omfattet begge prosjektene (6-7) han har vært involvert i.

Rakesh Jaiswal

Jaiswal ble i likhet med Wilson først kontaktet pr. e-post med korte spørsmål om prosjektet (8) før et intervju ble forespurt og avtalt. På grunn av geografisk avstand ble det også her avtalt å gjennomføre intervjuet ved hjelp av Skype med lydopptak av samtalen. Intervjuet hadde en varighet på ca. én time og 30 minutter.

Raj Mokashi

Mokashi ble også kontaktet pr. e-post og et intervju ble så avtalt. Da Mokashi er lokalisert i India var det også her naturlig å gjennomføre intervjuet ved hjelp av Skype. Denne samtalen ble i likhet med de foregående tatt opp. Intervjuet med Mokashi varte i ca. én time og 30 minutter. Han er involvert i prosjektene 9-13 som har India som offshore-lokalitet.

---

[4] Et dataprogram som brukes til IP-telefoni

[5] Skype Call Recorder

6.2.1 Intervjuspørsmål

Samtlige intervju var basert på den samme intervjuguiden (vedlegg 1) og de fikk alle således de samme spørsmålene. Intervjuene var av den semi-strukturerte sorten og oppfølgingsspørsmålene vil av den grunn naturlig nok variere. Målet med intervjuene var å skape en trygg og uformell dialog med intervjuobjektene slik at de følte seg komfortabel til å snakke om prosjektet. På denne måten vil de lettere trekke frem utfordringer og problemer så vel som positive erfaringer knyttet til prosjektene.

Lydopptakene gjort under intervjuene ble så gjennomgått og de alle meste er transkribert. Det som er utelatt er høflighetsfraser, introduksjon av intervjuet, samt avslutning.

6.2.2 Anonymisering

Intervjupersonene i studien er anonymisert ved at deres navn er byttet ut med fiktive navn. Når det gjelder prosjektene er disse anonymisert i den grad at det ikke oppgis navn på hvem som har prosjektet. Denne anonymiseringen er foretatt etter ønske fra intervjuobjektene.

7 Analyse

Dette kapittelet vil gjennomgå prosjektene basert på problemstillinger nevnt i avsnitt 5.8 Intervjuanalyse. Prosjektene presenteres her kategorisert etter de ulike intervjuobjektene, deretter prosjektene. Grunnen til denne kategoriseringen er at intervjuobjektene i stor grad refererte til flere prosjekter av gangen. I blant annet prosjektene (1-5), som var svært like ble erfaringer og justeringer ofte omtalt på tvers av prosjektene.

I neste kapittel vil prosjektene drøftes kategorisert etter spesifikke erfaringer prosjektene har erfart for å forsøke å hente ut generelle erfaringer på tvers av alle prosjektene.

7.1 Erfaringer fra prosjektene

7.1.1 Prosjekt hos stor logistikkorganisasjon, Belgia (1-5)

Utviklerne som befant seg i Riga, dro alle til Belgia for å starte opp det første prosjektet (1). Denne perioden var på rundt tre til fire uker før disse returnerte til Riga for å jobbe videre med prosjektet derfra. Tiden i starten av prosjektet ble benyttet til sosialisering, sette opp systemer som VPN systemer og ellers å klargjøre for et distribuert prosjekt. Selve prosjektet varte totalt i seks måneder og la grunnlaget for de neste prosjektene (2-5).

I starten på det andre prosjektet (2) reiste Scrum Master til Belgia for å starte opp prosjektet. De resterende offshore utviklerne i Riga ble ikke med onshore til Belgia. Prosjektet bestod av tre onshoremedlemmer og fire offshoremedlemmer, hvor Scrum Master var blant offshoremedlemmene. Vanags forteller at dette var et prosjekt som ble startet fordi klinten ikke var klar for det neste ”normale” prosjektet og ville derfor gjennomføre dette vedlikeholdsprosjektet i mellomtiden.

Prosjektet hadde store utfordringer med blant annet at det var lavt prioritert av oppdragsgiverne, samt av prosjektet havnet mellom to viktigere prosjekter. Medlemmene var dermed lite motiverte og prosjektet var i seg selv dårlig planlagt da tiden ikke tillot nøye planlegging i starten. I utgangspunktet arbeidet ikke utviklerne med flere enn ett prosjekt av gangen. Men det var alltid noe arbeid som hang igjen fra forrige prosjekt og noe planlegging i forbindelse med neste prosjekt.

Tredje og fjerde prosjektet (3,4) var bedre planlagt og er blitt karakterisert av gruppen og kunden som en suksess. Med dette menes de selv benytter ordet suksess, men i tillegg ble

disse prosjektene levert før eller til riktig tid, med gode tilbakemeldinger fra kunde. I disse prosjektene var kommunikasjonen mellom onshore og offshore god og utviklerne hadde fått bygget opp felles relasjoner i starten på prosjektet. De benyttet flere kommunikasjonskanaler hvor videokonferanser, e-post, lynmeldinger og diskusjonsforum var blant de mest sentrale. På denne måten fasiliterte de ulike kommunikasjonskanalene ulike kommunikasjonsbehov for utviklerne.

Det siste prosjektet (5) som fremdeles pågår er så langt i rute og har mottatt positive tilbakemeldinger fra både kunde og utviklere. Det har så langt ikke støtt på utfordringer knyttet til kommunikasjon og samhandling. Av de fem prosjektene (1-5) ble to av dem levert før planlagt leveringstidspunkt og de tre andre ble levert i henhold til leveringstidspunktet.

Generelle erfaringer

I de første prosjektene ble det forsøkt med en distribuert form for Scrum, se 3.1. Denne modellen viste seg for disse prosjektene å fungere dårlig med tanke på kommunikasjon. Det ble erfart fra Riga-kontoret at enkelte ble veldig beskjedne og passive i kommunikasjonssammenhenger med kontoret i Belgia. Dette kan skyldes språkbarrierer samt at videokonferanser kan oppfattes noe ubehagelig for noen. Disse som var beskjedne og stille under kommunikasjon mot Belgia var svært delaktige og positive under interne møter i Riga.

Det ble derfor lagt opp til en modell hvor det ble holdt lokale standup møter på hver lokasjon og en fra hvert kontor synkroniserte dette med det andre kontoret. Denne formen minner veldig om Scrum of Scrums selv om man ikke opprettet en egen Scrum-gruppe for å koordinere de to gruppene. Det ble fremdels benyttes videokonferanser og videosamtaler, men i mer uformelle sammenhenger, noe som viste seg å passe utviklerne fra Riga bedre.

Alle prosjektene benyttet Scrum sammen med praksiser fra XP. Balodis uttaler ”det ville ikke vært mulig å gjennomføre disse prosjektene uten praksiser fra XP”, noe Vanags er enig i. Videre trekker de to frem viktigheten av å ha en overordnet plan for prosjektet. At man arbeider med en smidig metode betyr ikke at man slipper å planlegge. De hadde best erfaringer med å planlegge godt på forhånd for så å starte prosjektet fra én lokasjon hvor man gjennomførte det de kaller en ”sprint zero”. Denne iterasjonen benyttes for å etablere sosiale relasjoner internt i gruppen samt planlegge den overordnede arkitekturen for systemet, samt forfatte krav og brukerhistorier. I løpet av denne iterasjonen må ”product

backlog" være klar og en må ha blitt enig om hvilke kommunikasjonskanaler som skal brukes i hvilke sammenhenger når deler av gruppen drar offshore.

Selve begrepet "sprint zero" er for studien ikke som et "offisielt" begrep innen Scrum og det antas å være et begrep som kommer fra Scrum-opplæringskurs. Det tolkes her som en sprint før de "vanlige" iterasjonene tar til. En sprint hvor de store linjene legges for prosjektet.

Etter at "sprint zero" er ferdig kan deler av gruppen dra offshore. Man gjennomfører så sprint 1 og 2 før man lager en lanseringsplan. I løpet av de første iterasjonene erfarte Balodis og Vanags at estimeringen ble bedre og man hadde et bedre bilde av når systemet ville bli ferdig. De to erfarte også at å starte med kortere iterasjoner i starten gav en positiv effekt på prosjektet. Det gjorde det lettere å estimere samt at man raskt fikk se resultater. Etter hvert kunne iterasjonene forlenges da gruppen var mer samkjørt og mer presis i sine estimater.

7.1.2 **Prosjekt hos finansinstitusjon (6)**

Selskapet involvert i prosjektet har hatt lang erfaring med offshore-utvikling, men da basert på tradisjonelle metoder. Utviklerne består av både nye og erfarne utviklere som har det til felles at de alle har liten kjennskap til smidige metoder. Prosjektet består nå av fem grupper hvor hver gruppe består av ca. to til tre designere, én til to testpersoner, én til to forretningsutviklere, seks til åtte programmerere og ellers analytikere. Hvert team består av totalt ca. 13-14 personer. Teamet er organisert slik at hver gruppe har en dedikert Scrum Master som ikke deltar i selve utviklingen, men innehar kun rollen som Scrum Master. Programmererne befinner seg i India, mens de resterende er onshore i USA. En teknisk leder er lokalisert onshore og en offshore, som fungerer som et bindeledd mellom de to gruppene innenfor samme Scrum team. Totalt består prosjektet av rundt 65 personer og kan dermed sies å være et nokså stort prosjekt.

Teammedlemmene onshore og offshore har hver dag fire timer overlapping hvor de er på jobb samtidig. For å sikre god kommunikasjon mellom menneskene i hvert Scrum team, er det satt opp dedikerte konferanserom hvor medlemmene arbeider hver dag. Disse rommene er utstyr med store flatskjermer slik at de offshore og onshore kan se hverandre til enhver tid innenfor de overlappende timene. Lyden på disse er i utgangspunktet avslått, men slås på når en på den andre siden signaliserer foran kamera at han ønsker å kommunisere med de andre. Lyden slås da på og man kan kommunisere fritt mellom de som er offshore og onshore.

Disse videokonferanserommene var satt opp for å gi Scrum teamene følelsen av å sitte sammen og oppfordre til lett og ledig kommunikasjon mellom individene i teamet. I dette rommet ble standupmøter, retrospektivmøter, planleggingsmøter m.m. holdt. I tillegg til videokonferanse ble også lynmeldinger, e-post, skjermdeling og telefon benyttet.

Wilson ble hentet inn tidlig i prosjektet for å veilede Scrum Mastere, teammedlemmer, ledere og andre involverte i hvordan gjennomføre prosjektet i henhold til smidige prinsipper. Det første Wilson sørget for var å innføre XP-praksiser i prosjektet. Hans tidligere erfaringer har vist at Scrum ikke fungerer uten andre metoder som eksempelvis XP. Dette for å kunne dekke de mer detaljerte områdene som Scrum ikke adresserer. XP har flere teknikker som hjelper på disse områdene som for eksempel par-programmering. Siste tilgjengelige oppdatering fra prosjektet forteller at det går svært bra og regnes å levere i henhold til tidsplanen.

7.1.3 Prosjekt hos teknologiselskap innen telekom (7)

Wilson ble hentet inn i prosjektet etter den fjerde sprinten. Det han da observerte var at daglige stand up møter ikke ble holdt konsekvent, samt at det ikke var satt opp dedikerte konferanserom eller gjort vurderinger i forhold til kommunikasjon med de andre teamene. Teamene i USA var ikke lokalisert på samme sted hver dag, noe som førte til vanskeligheter i forhold til det å skaffe seg videokonferanserom. Dette var helt nødvendig for å gjennomføre møter med de i Singapore, Amsterdam og India. Teamene hadde som et resultat av dette én time videokonferanse pr. dag.

I tillegg til kommunikasjonsutfordringer innad og mellom teamene var også kommunikasjonen mot kunden vanskelig. Det var vanskelig å få én å forholde seg til samt få kunden og ledelsen til å se verdien av å involvere seg mer i utviklingen. Dette var også ting som Wilson adresserte og ble i perioder selv en slags produkteier, før kunden selv erfarte verdien av å være involvert i utviklingen i større grad enn tidligere.

Det ble på tidspunktet Wilson ble hentet inn, ikke benyttet andre metoder sammen med Scrum. XP ble eksempelvis ikke benyttet, noe Wilson anbefalte prosjektet å implementere snarest. Dette er også nå implementert og har gitt et positivt resultat i forhold til effektiviteten. Spesielt handlet dette om å ha en metode å forholde seg til for de mer spesifikke delene av utviklingen. Scrum dekker som tidligere nevnt ikke alle deler ved systemutvikling og det er derfor naturlig å støtte seg på flere metoder. Prosjektet har nå (p.t. januar 2009) vart i ett år og er planlagt å vare ett år til. Det er gjort endringer i forhold

til lokasjoner til de amerikanske deltakerne som nå befinner seg på samme lokasjon hver dag.

Da systemer prosjektet utvikler enda ikke er lansert er det vanskelig å innhente informasjon vedrørende hvor vellykket prosjektet er. I følge Wilson er tilfredsheten hos lederne betydelig bedre etter at prosjektet startet å kjøre Scrum mer konsekvent, samt introduserte XP i utviklingen. Prosjektet har likevel allerede nå høstet erfaringer knyttet til bruk av XP i tillegg til Scrum, samt viktigheten av å være konsekvent i forhold til metode. Det å holde daglige møter og sikre at kommunikasjonen er god er eksempler på erfaringer de har gjort seg. De har også erfart bedre sosiale forhold mellom prosjektmedlemmene har påvirket prosjektet positivt og at dette burde vært adressert i starten av prosjektet.

7.1.4 STS prosjekt i England og India (8)

Prosjektet er organisert i fem Scrum team fordelt på fire Scrum Mastere. Én Scrum Master har ansvaret for to Scrum team, men de resterende har ansvar for hvert sitt team. Disse er dedikerte Scrum Mastere og deltar således ikke i selve utviklingen. De har som oppgave å sørge for at Scrum-prosessen blir overholdt, samt fjerne hindringer teamene møter underveis.

STS prosjektet lanserer deler av systemet hvert kvartal hvor det da går i produksjon og erstatter deler av det gamle systemet. Iterasjonene er på 15 dager, noe som ble valgt for å ha god kontroll på utviklingen underveis, slik at feil og andre problemstillinger raskt blir oppdaget og adressert. For hvert kvartal blir product backlog elementer valgt ut og kategorisert i kategoriene høy, middels og lav prioritet. I starten på kvartalet blir de med høy prioritet valgt først, før man i slutten av kvartalet adresserer de elementene med lav prioritet, samt feilrettinger oppdaget i løpet av kvartalet.

Fordelingen av disse oppgavene blir først gjort av produkteier sammen med alle Scrum Masterne, samt representanter fra kunden. Så vurderer hvert Scrum team selv hvor mye de kan ta på seg for hver iterasjon. I dette arbeidet benyttes teknikken ”planning poker” for å estimere hvor lang tid hvert backlog element vil ta. Jaiswal uttaler i denne sammenheng at ”det er interessant å se hvordan én utvikler tror en oppgave er svært komplisert, mens etter diskusjon med resten av gruppen innser han at oppgaven derimot er svært enkel”.

I dette prosjektet ble det i starten valgt å hente over åtte utviklere fra India til England for åtte ukers opplæring. I denne perioden ble de opplært i metoden Scrum samt andre praksiser som blir benyttet i prosjektet eksempelvis noen fra XP som par-programmering. I

tillegg til Scrum-spesifikk opplæring ble alle kjent med arkitekturen, produksjonsprosessen, leveranseverktøy og produktspesifikke elementer.

På spørsmål om hvorfor de valgte denne måten å starte opp prosjektet svarte Jaiswal følgende:

> Dette prosjektet er en av de fem mest kritiske og omfattende applikasjonene hos denne kunden, og det er derfor vanskeligere for utviklerne å forstå prosjektet. Spesielt leveranseprosessen og produksjonsmiljøet. Dataene i databasen er også svært sensitivt og databasen er antagelig den største i verden. Derfor ses opplæring på som svært viktig i dette prosjektet. Alle må ha en god forståelse av prosjektet.

Etter denne opplæringsfasen, returnerte fire av utviklerne tilbake til India hvor de ble Scrum Mastere for hvert sitt team. De fire resterende ble igjen i England for å jobbe med prosjektet derfra. I løpet av opplæringsfasen ble den første iterasjonen gjennomført hvor alle var samlet på samme lokasjon slik at kommunikasjonsforholdene var optimale. Denne iterasjonen var primært viet planlegging samt å legge de overordnede føringer for arkitekturen.

Som forventet for prosjektet var effektiviteten nokså lav i første periode av prosjektet. Estimeringene traff dårlig og prosjektet leverte kun 50 % av estimerte oppgaver den første tiden. Etter hvert ble alle i teamene mer vant til estimeringsteknikkene i tillegg til at domenet og prosjektet ble mer innarbeidet. I dag regner Rakesh Jaiswal estimeringene å være rundt 95 % korrekte.

Da systemene som utvikles er virksomhetskritiske ble det etter hvert innført ”test først” prinsipper hvor det først ble skrevet tester som deretter programmererne skulle skrive kode for å passere. Dette i tillegg til automatisert testing førte til at kvaliteten på leveransene økte merkbart. Denne endringen ble gjennomført et stykke ut i prosjektet på bakgrunn av tilbakemeldinger fra både utviklere, produkteier og kunden.

I forhold til kommunikasjon mellom gruppen onshore og de offshore er denne i følge Jaiswal god i forhold til at det kun benyttes videokonferanse én gang pr. måned. Til daglig benyttes telefon, lynmeldinger, e-post og diskusjonsforum. Jaiswal trekker frem opplæringsperioden som én av årsakene til at kommunikasjonen fungerer godt selv om ikke videokonferanse benyttes i større grad. At en del av gruppen onshore og offshore kjenner hverandre godt gjør kommunikasjonen lettere og mindre anstrengt.

Tilbakemeldingene fra oppdragsgiver som i utgangspunktet var skeptisk til bruk av smidige metoder for kritiske systemer er nå svært positiv til prosjektet og systemene som er

levert. Det var i følge Jaiswal noe frustrasjon i starten fra oppdragsgivers side over dårlig estimering av oppgaver og dermed mindre leveranser enn planlagt. De siste tilbakemeldingene gitt til prosjektet er positive, hvor det fremheves at estimering og kvalitet er over forventning.

7.1.5 DART Amazon Next, England, USA og India (9)

Prosjektet består av en stor andel uerfarne utviklere som ikke har erfaring med Scrum tidligere og heller ikke i en offshore setting. Flere av de fra India reiste i starten til USA hvor de første iterasjonene ble gjennomført. I løpet av denne perioden fikk inderne kjennskap til arkitekturen og domenekunnskap knyttet til finanssystemer. I tillegg fikk de anledning til å bli godt kjent med prosjektmedlemmene i USA og England, slik at tillit og felles respekt kunne etableres. Dette er vel så viktig som kunnskap om arkitekturen og domenet i følge Mokashi.

Etter de første iterasjonene reiste inderne tilbake til India hvor de så satte opp prosjektet der for å jobbe videre med det. I denne fasen ble også nye medlemmer inkludert og ble opplært av de som hadde vært onshore.

Det har senere blitt vanlig at både offshore og onshore medlemmer har reist mellom lokalitetene for utveksling av erfaring og generell sosial interaksjon. Primært er tilfellet det at onshore medlemmer drar offshore i korte perioder for å bidra med domenekunnskap, hjelp til testing, forklare prosesser eller bistå med spesifikke problemer.

Kommunikasjonen mellom onshore og offshore gjennomføres i dette og de øvrige prosjektene hos IDC via flere kanaler. Til standup møter og lignende benyttes telefonkonferanser hvor alle deltar. Det kan også forekomme interne standupmøter for de ulike lokaliteter før disse så koordineres i etterkant. Til annen kommunikasjon mellom enkelte medlemmer av prosjektet benyttes lynmeldinger, e-post, telefon og forum. For generell kommunikasjon benyttes en egen prosjekt-wiki og forum. Disse prosjektene har valgt ikke å benytte videokonferanse på grunn av dens kostnad og krav til infrastruktur.

I tillegg til nevnte kommunikasjonskanaler benyttes også WebEx, en løsning fra Cisco for webkonferanser hvor blant annet skrivebordsdeling er en vesentlig del. WebEx benyttes sammen med telefonkonferansene for å kunne kommunisere visuelt så vel som verbalt.

I likhet med de andre prosjektene oppgaven har undersøkt, benytter dette prosjektet XP-teknikker sammen med Scrum. De har kun valgt ut enkelte XP-teknikker som hyppige

iterasjoner, brukerhistorier og akseptansetester. Teknikker som par-programmering benyttes ikke i dette prosjektet.

Erfaringer høstet

Prosjektet har så langt høstet noen verdifulle erfaringer. Blant annet er opplæring på metode og prosess erfart som viktig i starten på denne typen prosjekter. I dette prosjektet fikk ikke utviklerne opplæring i Scrum og XP teknikker. Dette førte til en del frustrasjon rundt standupmøtene i starten som ble fryktelig lange og tok derfor mye av den tiden prosjektmedlemmene eksempelvis skulle bruke til programmering. Det ble tatt opp problemstillinger som få av medlemmene hadde interesse av og disse møtene ble derfor lite effektive. Etter hvert ble denne problematikken adressert og nå som teamene er innforstått med konseptet bak standupmøtene fungerer disse i langt bedre grad. Spesifikke problemstillinger tas opp i ettertid mellom dem det angår.

Den viktigste lærdommen prosjektet har gjort seg er knyttet til erfaringen hos prosjektmedlemmene. Prosjektet har opplevd problemer knyttet til uerfarne utviklere og kommunikasjon. Uerfarne utviklere har vist seg å være mer knyttet til designdokumenter enn de mer erfarne. Da det i denne typen prosjekter ikke finnes detaljdokumenter er man avhengig av at de riktige kontrollspørsmål stilles og at man sørger for å unngå misforståelser. Mokashi trekker frem blant annet at:

> Et vanlig problem i disse prosjektene, inkludert dette er problemer knytte til uerfarne utviklere. Det er behov for mer erfarne utviklere i denne typen prosjekter hvor smidige metoder sammen med offshoring benyttes. Man må derfor være nøye når teamene settes sammen for å sikre nok erfarne utviklere. Dette er noe vi har erfart i dette prosjektet. De uerfarne utviklerne trenger mye mer dokumentasjon enn erfarne, spesielt i slike prosjekter. Så når vi har mer erfarne utviklere trenger vi ikke dokumentere like mye, og kan arbeide på en mer smidig måte, samtidig som at vi ikke har like stort behov for kommunikasjon.

I senere prosjekter vil man i langt større grad benytte mer erfarne utviklere for å unngå denne problematikken. I smidige prosjekter som er samlokalisert antas ikke dette å være et like stort problem, og det vil derfor være naturlig å bruke uerfarne utviklere i slike prosjekter før de settes inn i offshore-prosjekter.

Dette prosjektet og de tre andre prosjektene hvor Mokashi har vært involvert er svært like i måten de er satt opp, samt erfaringer de har gjort seg. Det er av den grunn viet mest plass til det første prosjektet, mens de andre vil gis en kortere beskrivelse. Forskjeller mellom prosjektene vil nevnes for de øvrige. Intervjuet var også organisert slik at det første

prosjektet ble viet mest tid, mens de andre ble raskere gjennomgått, med informasjon om de erfaringene som skilte de prosjektene fra det første og hverandre.

7.1.6 Elektronisk bestillingssystem, Nederland og India (10)

I likhet med det forrige prosjektet ble flere utviklere fra India fraktet til Nederland for å delta i oppstarten av prosjektet. I denne perioden ble offshoreutviklerne kjent med arkitekturen og domenet til systemet. De første iterasjonene ble gjennomført i Nederland før offshoreutviklerne returnerte til India for å fortsette prosjektet der.

På samme måte som forrige prosjekt (9) var denne tiden verdifull i forhold til å etablere sosiale relasjoner mellom utviklere fra ulike geografiske lokaliteter. Disse relasjonene har positiv betydning i forhold til kommunikasjon når utviklerne befinner seg på ulike steder.

Kommunikasjonen mellom onshore og offshore var i dette prosjektet likt det forrige. Samme kommunikasjonskanaler og verktøy ble benyttet. Erfaringer knyttet til mangel på opplæring innen Scrum samt problemer med uerfarne utviklere var de samme her som hos prosjekt (9).

7.1.7 Applikasjon for én av verdens ledende mobilprodusenter (11)

I likhet med de to andre prosjektene (9,10) erfarte også dette prosjektet utfordringene knyttet til uerfarne utviklere, både når det gjelder utvikling generelt men også når det angår spesifikk Scrum-kompetanse. Mer omfattende opplæring innen Scrum i starten på prosjektet ville muligens kunnet bøte noe på dette problemet.

I tillegg ble det også erfart at kunden i dette prosjektet var av en oppfattelse at kravet og behovet for dokumentasjon var langt mindre enn hva det faktisk var. ”Noen kunder forstår ikke umiddelbart at de må gjøre en del dokumentasjon i denne typen prosjekter, men ikke så mye som i tradisjonelle prosjekter. Vi erfarte i dette prosjektet at det er behov for en del dokumentasjon når vi arbeider offshore.” sier Mokashi. Han sier videre at kunden i dette prosjektet trodde i starten av det var tilstrekkelig å beskrive over telefon hva de trengte. Etter en gjennomgang med kunden ble det enighet om at denne typen informasjon måtte dokumenteres skriftlig.

Prosjektet var ellers bygget opp på samme måte som de øvrige prosjektene (9,10) med oppstart onshore. Etablering av sosiale relasjoner, samt gjennomføre av de første iterasjonene, før prosjektet ble delt opp i onshore og offshore team.

7.1.8 Applikasjon for en av verdens største bilprodusenter (12)

For en av de aller største bilprodusentene i verden er det under utvikling en applikasjon til bruk for forhandlerne av bilmerket. Dette er et prosjekt som snart er fullført og som generelt anses som en suksess. Det er likevel gjort erfaringer knyttet til noe for optimistisk estimering slik noen av de andre prosjektene også har erfart. Utover dette er prosjektet nokså identisk med de andre hva gjelder oppstart, sammensetning og erfaringer gjort.

7.2 Oppsummering av erfaringene

Prosjektene undersøkt i denne studien spenner over hele verden, er av ulik størrelse og varighet, samt omfatter ulike fagområder. Mange av erfaringene gjort i de ulike prosjektene er allikevel svært like.

Tabell 4 viser en sammenligning av prosjektene med tanke på ulike elementer som oppstartsform, kommunikasjonsmetoder, opplæring etc. Denne er ment å gi en mer oversiktlig oversikt over likhetstrekk og forskjeller mellom prosjektene.

	1	2	3	4	5	6	7	8	9	10	11	12
Antall medlemmer	6	3	3	3	5	65	35	50	26	34	11	14
Antall land involvert	2	2	2	2	2	2	4	2	3	2	2	2
Varighet (ca. antall mnd.)	6	3	4	4	5	12	12	18	12	12	12	12
Opplæring gitt	X	X	X	X	X			X	X	X	X	X
Oppstart onshore	X	X	X	X	X			X	X	X	X	X
Erfarne utviklere (primært)				X	X	X	X					
Ansikt-til-ansikt kom.			X	X								
Videokonferanser (på ukentlig basis eller oftere)	X	X	X	X	X	X	X					
Reising mellom lokasjonene	X	X			X			X	X	X	X	X
Benytter XP i tillegg	X	X	X	X	X	X	X	X	X	X	X	X

Tabell 4 Sammenligning av prosjektene

Tabell 4 er kun ment for å gi en oversikt over prosjektene. Varigheten for prosjektene 9-12 er grove estimater, da intervjuobjektet kun refererte til ca. 1 år.

Radene hvor X er benyttet må også ses på som indikasjoner. For eksempel når det gjelder bruk av videokonferanser, ble dette benyttet i svært varierende grad og man kan derfor ikke direkte sammenligne prosjektene på bakgrunn av dette. Prosjekt 6 benytter for eksempel videokonferanser 4 timer hver dag, mens prosjekt 1 og 2 kun benyttet det noen

ganger i uken. Disse som benyttet videokonferanser i liten grad forsøkte å oppveie for dette ved å benytte andre kommunikasjonskanaler som telefon, e-post, forum, lynmeldinger etc. Disse er likevel vanskelig å føre inn i tabellen da prosjektene benyttet ulike kombinasjoner av disse, slik at man ikke kan foreta en direkte sammenligning.

Tabellen gir likevel en nyttig oversikt over prosjektene og viser hva de har vektlagt. Blant annet er det interessant å se at samtlige benytter praksiser fra XP, selv om de nødvendigvis ikke gjorde dette fra starten av prosjektet.

Kompetansen hos utviklerne i de ulike prosjektene har vært svært ulik. Noen prosjekter har hatt tilgang på mange erfarne utviklere, mens andre har startet sine prosjekter ved uerfarne utviklere. For de sistnevnte prosjektene har dette gitt verdifulle erfaringer. Blant annet er det erfart at mer erfarne utviklere håndterer og fungerer bedre i et offshore-miljø enn hva uerfarne gjør. Årsaken til dette slik de berørte prosjektene ser det, er at erfarne utviklere fanger opp problemstillinger og stiller de riktige spørsmålene, mens uerfarne ikke alltid har oversikt over prosjektet samt ikke klarer å få riktige avklaringer. Disse trenger mer oppfølging noe som er problematisk i et offshore miljø hvor kommunikasjon er mer utfordrende enn i et onshore-miljø.

Oppstarten for flere av prosjektene har noen likhetstrekk. Blant annet har flere av prosjektene valgt å sende offshore-utviklere onshore i oppstarten av prosjektet for opplæring og sosialisering med medlemmene som er onshore. Den eller de første iterasjonene utføres så onshore og benyttes til planlegging og oppbygging av arkitekturen. Når prosjektet er kommet i gang, drar offshore-utviklerne tilbake for å fortsette prosjektet der. Disse inkluderer så nye medlemmer av prosjektet som får opplæring av utviklerne som har vært onshore. Prosjektene som har benyttet denne metoden har sett på dette som et viktig element i prosjektet for å sikre god kommunikasjon mellom de ulike lokalitetene. I tillegg forsikrer man seg om at utviklerne offshore har en god forståelse av prosjektet og arkitekturen.

De kan tyde på at prosjektene som ikke valgte å sende utviklere onshore i starten, har opplevd større utfordringer underveis. Å si at disse var feilet på bakgrunn av dette er derimot vanskelig å ha dekning for da disse prosjektene ikke er fullført. De har også etter justeringer fått mer positive tilbakemeldinger fra kunden, samt opplevd bedre samhandling internt i prosjektene.

I tillegg til å samle utviklere på en lokasjon i starten er generell opplæring ansett som viktig hos alle prosjektene. Ikke alle har gjennomført dette i starten, men samtlige prosjekter har i

løpet av prosjektets levetid erfart viktigheten av grundig opplæring i starten. Både opplæring innenfor domenet til prosjektet samt prosjektmetode anses som svært viktig for prosjektets suksess.

Ett av prosjektene hos den belgiske logistikkorganisasjonen ble ikke regnet som en suksess, og mangel på planlegging og prioritet fra oppdragsgiver ses på som årsaken til dette. De andre prosjektene som i stor grad er regnet som vellykkede har alle før eller senere i prosjektets levetid fokusert på planlegging og involvering av oppdragsgiver. Det kan derfor antydes at planlegging og involvering av oppdragsgiver er en av faktorene for suksess i denne typen prosjekter.

Kommunikasjonsformene de ulike prosjektene har benyttet varierer noe. Noen har satt opp dedikerte videokonferanserom som utviklerne benytter daglig, mens andre prosjekter har valgt kun å gjennomføre videokonferanser med jevne mellomrom. Felles for alle prosjektene derimot er bruk av ulike kommunikasjonskanaler. Det å benytte riktig kommunikasjonskanal til riktig type kommunikasjon er sett på som viktig for alle prosjektene. De har blant annet alle tatt i bruk telefon, e-post og lynmeldinger for å nevne noen. Bruk av flere kommunikasjonskanaler kan derfor antydes å ha en positiv effekt på denne typen prosjekter.

8 Drøfting

Erfaringene fra prosjektene i denne studien vil i dette kapitelet diskuteres sammen med andre relevante studier på området. Ved å inkludere også andre studier vil oppgaven lettere kunne dra konklusjoner på bakgrunn av egne data, ved å se om disse samsvarer med hva andre studier har konkludert med.

På bakgrunn av resultatene fra intervjuene vil dette kapitelet deles inn i deler som samsvarer med de viktigste erfaringer identifisert i løpet av studien.

8.1 Oppstartsfasen

Oppstartsfasen for et hvilket som helst prosjekt er viktig. I tradisjonelle prosjekter benyttes denne fasen til nøye planlegging. I smidige prosjekter, spesielt de som inkluderer offshoring, er planlegging fremdeles svært viktig. Det andre prosjektet (2) hvor Andris Vanags var Scrum Master erfarte han nettopp problemene ved dårlig planlegging. Dette var et vedlikeholdsprosjekt som hadde lav prioritet hos kunden og ble derfor ikke planlagt i særlig grad. Det var naturligvis også andre årsaker til at dette prosjektet møtte store utfordringer, men i følge Vanags var mangel på planlegging en avgjørende faktor i denne sammenheng.

De øvrige prosjektene hos leveransesenter i Riga hvor Andris Vanags og Kaspars Balodis var involvert, var planlegging en viktig del av prosjektet. Oppstarten av det første prosjektet ble gjennomført onshore hvor planene for prosjektet ble lagt. Her ble offshore medlemmer av prosjektet hentet inn for å være med på oppstarten av prosjektet. Teknisk utstyr ble satt opp, plan for kommunikasjon med offshore team fastsatt, samt domenerelevant opplæring gitt. Denne perioden hvor alle var samlet på samme lokalitet gav også mulighet for sosialisering og etablering av tillit. Dette letter kommunikasjon senere i prosjektet hvor man jobber på ulike lokaliteter. Åpen og god kommunikasjon forutsetter at tillit er etablert (Hildenbrand et al., 2008).

STS prosjektet hvor Rakesh Jaiswal var involvert valgte også en lignende oppstartsfase hvor åtte utviklere ble hentet fra India (offshore) til England (onshore) for en felles oppstart av prosjektet. På samme måte ble denne perioden benyttet til sosialisering og etablering av tillit. I tillegg ble opplæring gitt på domenerelevant kunnskap, Scrum metoden og arkitekturen. Også i dette prosjektet trekkes denne formen for oppstart av prosjekter frem

som en viktig faktor til suksess, fordi den reduserer kommunikasjonsbarrierer som kan oppstå når man arbeider distribuert. De siste prosjektene presentert i oppgaven ved leveransesenteret i India benyttet også en lignende oppstartsform. Det ble også her fremhevet viktigheten av denne formen for oppstart.

Ett av prosjektene som ikke valgte å benytte denne formen for oppstart var det første prosjektet omtalt hvor Wilson var involvert. Detaljert informasjon om hvordan oppstarten var organisert for dette prosjektet er for oppgaven dessverre ikke kjent da Wilson ikke var involvert fra starten av i dette prosjektet. For å veie opp for mangel på sosialisering og tillit mellom lokalitetene ble ekstensiv bruk av videokonferanse benyttet for å skape en følelse av å være på samme lokalitet. I delen om kommunikasjon drøftes dette i mer detalj.

Sureshchandra & Shrinivasavadhani (2008) omtaler flere steg for adoptering av smidige metoder (Scrum og XP) i distribuerte omgivelser. I artikkelen nevnes det at i en oppstartsfase bør noen prosjektmedlemmer fra offshore teamene sendes onshore for å delta i oppstarten sammen med de som er onshore. Disse vil fungere som stedfortredere for de øvrige offshore medlemmene.

Vax & Michaud (2008) har gjennomført en case studie hvor offshore-utviklere ikke fikk anledning til å reise onshore i starten av prosjektet. De identifiserer dette som et behov og trekker frem viktigheten av å få anledning til å møtes ansikt-til-ansikt i starten av prosjektet. Gjennom å møtes ansikt-til-ansikt etableres også tillit samt at teamene bindes mer sammen (Ramesh et al., 2006).

I tillegg til sosialisering, samhold og tillit er oppstartsfasen også en viktig del av prosjektet for å sikre at alle involverte har den samme kunnskapen og forståelsen av systemet som skal utvikles. Med dette menes domenekunnskapen til systemet. Utvikler man et finanssystem er det naturlig med grunnlegende og relevant kunnskap innen finans. Finans blir i denne sammenhengen domenet til systemet. I tillegg er forståelse for arkitekturen viktig for alle teammedlemmer, noe de som ble intervjuet i denne studien bekrefter. Blant andre forteller Mokashi om at de anser det som viktig å starte opp prosjekter på måten de har gjort hvor forståelse av arkitekturen inngår som en del av opplæringen.

Flere av prosjektene i denne studien valgte å benytte tiden hvor offshore medlemmer befant seg onshore til å gjennomføre én eller flere iterasjoner. Prosjektene basert i Riga og Brussels gjennomførte det de selv kalte Sprint Zero hvor arkitektur og plan for systemet ble lagt. Andre prosjektet valgte å gjennomføre flere mindre iterasjoner i denne fasen for å

etablere gode rutiner og samarbeid blant medlemmene før offshore medlemmene returnerte til sin lokasjon.

8.2 Kommunikasjon

I tilknytning til alle prosjektene er det gjort erfaringer knyttet til kommunikasjon. I denne studien fokuseres det i hovedsak på kommunikasjonen mellom de ulike lokaliteter. I smidige metoder fokuseres det betydelig på kommunikasjon mellom teammedlemmene i et prosjekt. På grunn av mindre dokumentasjon og rigide prosesser er man avhengig av god kommunikasjon for å sikre god progresjon og levere et godt resultat.

Prosjektene i studien har lagt opp kommunikasjon på ulikt vis. Prosjektene basert i Riga og Brussels valgte primært å benytte videokonferanser, e-post, lynmeldinger og telefon. De erfarte i midlertidig at utviklerne offshore var beskjedne og reserverte i forbindelse med videokonferanser. Om dette skyldes kulturforskjeller eller språkbarrierer er vanskelig å vurdere. Dette resulterte i en omlegging av kommunikasjonsstrategien, hvor de ble holdt lokale møter før én person hos hver lokalitet samkjørte resultatene fra møtene. Så i stedet for å organisere prosjektet som en distribuert utgave av Scrum, ble en Scrum of Scrums modell benyttet.

Ved bruk av Scrum of Scrums bedret kommunikasjonen seg betraktelig og videokonferansene fungerte langt bedre enn tidligere. Balodis og Vanags trekker likevel frem at kommunikasjonen aldri var dårlig men at den ikke fungerte optimalt.

Det første prosjektet Wilson var involvert i valgte som nevnt tidligere ikke å samle alle utviklere onshore i starten av prosjektet. For å sikre god kommunikasjon basert på gjensidig tillit og respekt, ble det lagt opp til ekstensiv bruk av videokonferanserom. Fire timer hver dag bidro i dette prosjektet til at man klarte å etablere sosiale relasjoner mellom medlemmene slik at kommunikasjonen var åpen og uformell mellom de ulike lokalitetene.

Selv om lyden mesteparten av tiden var slått av, bidro videobildet av de andre medlemmene på den andre lokasjonen til å skape en følelse av at alle var samlet på ett sted. I tillegg var terskelen svært lav for å begynne å snakke sammen slik man gjør på en vanlig videokonferanse. Man trengte ikke å avtale tid for en videokonferanse og sette opp denne. Det var bare å vinke foran kameraet, så kunne man prate i vei.

Wilson fremhever at det å kunne se de andre utviklerne til enhver tid bidro sterkt til å skape relasjoner mellom medlemmene av prosjektet. Selv om lyden stort sett var av, fikk alle en følelse av tilstedeværelse av teamet på den andre siden av jorden.

Et annet prosjekt som valgte en svært ulik strategi enn de foregående, er STS prosjektet hvor Jaiswal har vært involvert. I dette prosjektet var det svært lite fokus på bruk av videokonferanser, så for å skape gode relasjoner ble det lagt opp til et omfattende program i starten av prosjektet hvor mange utviklere fra India var samlet onshore i England. I tillegg til at de første iterasjonene ble gjennomført her, ble fire av disse igjen når de øvrige returnerte til India. På den måten kunne man sikre god kommunikasjon da man hadde indere i England som både kjente kulturen og menneskene man skulle kommunisere med i India. I stedet for videokonferanser, ble blant annet telefon, lynmeldinger og e-post benyttet mye.

Disse erfaringene tyder på at det er flere måter å sikre god kommunikasjon i denne typen prosjekter. Prosjektene nevnt her karakteriserer seg alle som suksessfulle, men trekker også frem at kommunikasjon er en avgjørende faktor for at prosjektene skal bli en suksess.

En kan derfor vanskelig trekke frem én kommunikasjonskanal eller ett spesifikt sett med kommunikasjonskanaler som løsning på kommunikasjonsproblematikken i offshore prosjekter. Valg av kommunikasjonskanaler må stå i stil med de relasjoner utviklerne har seg imellom. Hvordan kulturen er på de ulike lokalitetene, samt språkproblemer og mennesketyper må tas i betraktning ved utforming av en kommunikasjonsstrategi.

Flere andre studier trekker frem kommunikasjon som særdeles viktig i distribuerte prosjekter slik som ved offshoring (Korkala & Abrahamsson, 2007). Kommunikasjon er nøkkelen til suksess i slike prosjekter. Åpen kommunikasjon er det mest kritiske elementet i bygging av suksessfulle distribuerte team. Alle involverte team må tilpasse seg til de andre og inngå kompromisser for å sikre åpen og god kommunikasjon, hvor nødvendige verktøy må tas i bruk (Therrien, 2008).

8.2.1 Ekstern kommunikasjon

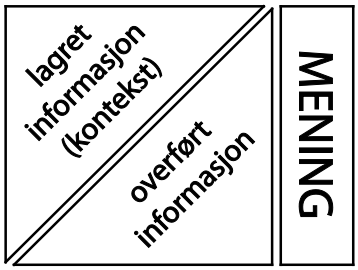
I tillegg til intern kommunikasjon mellom teamene er kommunikasjon ut mot produkteier og representanter for kunden også en viktig del av kommunikasjonsbehovet. Therrien (2008) nevner erfaringer gjort hos First American Corelogic hvor offshore temaet ikke hadde direkte kontakt med produkteier. Dette førte til dårlig kommunikasjon og mangelfull tillit i prosjektet som helhet.

Telekomprosjektet (7) hvor Wilson var involvert er et godt eksempel på utfordringer knyttet til kommunikasjon med personer utenfor selve utviklingen av systemet. Man opplevde også der utfordringer knyttet til problemer med avklaringer da man ikke hadde én å forholde seg til som hadde mulighet til å ta avklaringer. Wilson trekker ikke frem

mangel på tillit som et resultat av dette, men utelukker ikke en sammenheng mellom ekstern kommunikasjon og tillit.

8.2.2 Sosiale relasjoner

De fleste prosjektene la stor vekt på å skape sosiale relasjoner mellom utviklerne i starten av prosjektet. Slike relasjoner var viktig for å sikre uformell kommunikasjon senere i prosjektet. I tillegg til det å skape sosiale relasjoner benyttet mange av prosjektene oppstartsfasen til å samle utviklerne til en lokasjon for felles oppstart. Dette gav utviklerne lik kunnskap om arkitektur, domenet til prosjektet, verktøy, metode, samt andre forhold knyttet til prosjektet.



Figur 25 Lagret og overført informasjon (Hall, 1984)

Hall (1984) illustrerer tydelig med Figur 25 hvordan det å ha mye felles lagret informasjon gir behov for mindre overført informasjon. Øvre del av figuren illustrer at man har mye lagret informasjon mellom seg og de man kommuniserer med og behovet for overført informasjon for at noe skal gi mening er liten. I den andre enden av skalaen er mengden lagret informasjon liten og man må derfor overføre mer informasjon for å oppnå like stor grad av mening.

Dersom en ser på det Hall (1984) skriver opp mot prosjektene studert i denne studien kan man antyde følgende. Ved å skape felles relasjoner og gi utviklerne mye av den samme informasjonen i starten av prosjektet, vil man sikre at de har mye lagret informasjon. Av den grunn vil de senere i prosjektet sannsynligvis ha mindre behov for overført informasjon. Dette er naturligvis en fordel da det nettopp er utfordringer knyttet til kommunikasjon som er en av de viktigste faktorer i offshore prosjekter.

8.3 Metode

Samtlige av prosjektene valgte i tillegg til å benytte Scrum, å inkludere praksiser fra XP (se 2.4.2) i løpet av prosjektets varighet. I alle intervjuene ble det uttalt at dette var helt

essensielt for prosjektet. Det behøvde nødvendigvis ikke være XP, men det falt hos alle prosjektene som den mest naturlige metoden å inkludere praksiser fra. Scrum dekker som nevnt i teorikapittelet ikke alle områder innen utvikling, og det er derfor viktig å supplere med passende praksiser.

En annen interessant observasjon fra intervjuene er at alle implementerte Scrum så å si i sin helhet. Alle praksiser nevnt i teorikapittelet var inkludert i prosjektene, noe som gjør erfaringene mer gjeldende. Dersom kun deler av Scrum hadde vært benyttet ville det vært mye vanskeligere å generalisere erfaringene fra prosjektene.

Det er i prosjektene benyttet Scrum of Scrums og fullt distribuert Scrum. De første prosjektene (1-5) hadde litt problemer med fullt distribuert Scrum og la derfor om til Scrum of Scrums. Det tyder på at den fullt ut distribuerte utgaven stiller høyere krav til kommunikasjon mellom lokalitetene enn Scrum of Scrums. Også telekom prosjektet (7) valgte en fullt ut distribuert utgave av Scrum noe som sannsynligvis ikke var den beste løsningen for det prosjektet. Wilson som fungerte som veileder (coach) for prosjektet, erfarte nettopp at kommunikasjon mellom lokasjonene ikke fungerte optimalt.

I tillegg valgte dette prosjektet ikke å etablere sosiale relasjoner i starten ved å samles onshore. Etter modellen til Hall (1984) fører dette til behov for overføring av langt mer informasjon mellom lokalitetene, enn om det var lagret mer informasjon mellom utviklerne. Disse funnene kan i så måte tyde på at det er mer fornuftig å velge en Scrum of Scrums variant dersom man ikke har etablert sosiale relasjoner i starten av prosjektet, fremfor en fullt distribuert utgave.

8.4 Scrum og offshoring

Prosjektene i studien viser at Scrum og offshoring kan fungere godt om man gjør noen justering i forhold til hva man ellers ville ha gjort dersom det var et prosjekt hvor alle medlemmene var lokalisert på ett sted. Disse justeringene kan innbefatte blant annet det å planlegge hvordan kommunikasjon skal håndteres på tvers av lokasjoner. Hvordan teamene skal organiseres og koordineres. I tillegg har flere av prosjektene i denne studien valgt å starte opp prosjektet fra én lokasjon noe som tyder på å være positivt for prosjektene.

Ved å gjennomføre justeringene nevnt i denne studien, i tillegg til det man vanligvis gjør i planleggingsfasen av Scrum prosjekter, er det mye som tyder på at man kan gjennomføre vellykkede prosjekter hvor man benytter Scrum og offshoring sammen.

8.5 Validitet

I alle studier er spørsmål om validitet viktig å diskutere. Validitet refererer til datamaterialets gyldighet i forhold til de problemstillinger som belyses (Grønmo, 2004). Det er første og fremst det som betegnes som åpenbar validitet som her diskuteres. Åpenbar validitet baseres på trekk ved datainnsamlingen og datamaterialet som er åpenbare for forskeren selv og andre (Grønmo, 2004).

Prosjektene denne studien har undersøkt er knyttet opp til én eller to intervjuobjekter. Det er i all hovedsak kun foretatt ett intervju pr. intervjuobjekt da disse er travle mennesker som dessverre ikke har hatt anledning til å sette av mer tid.

Det faktum at resultatene stammer fra få personer knyttet til et prosjekt danner åpenbare problemstillinger knyttet til objektets oppfatning av prosjektet. Har vedkommende en oppfatning som deles av de andre i prosjektet? På en annen side er de som er intervjuet sentrale personer knyttet til prosjektene. De er personer som under hele prosjektet har vært tvunget til å foreta refleksjoner rundt kommunikasjonen i prosjektet, fremgangen, problemer underveis etc. I tillegg bør det tas med i betraktningen at det i denne studien er snakk om flere prosjekter knyttet til ulike intervjuobjekter som har erfart mye av det samme.

Det er også viktig å ta med i beregningen at det ikke er foretatt egne observasjoner i studiene og at resultatene i så måte kun baserer seg på observasjoner av andre som så igjen er videreformidlet.

Det er med andre ord ikke lett å vurdere validiteten til resultatene i studien, og den bør i hovedsak benyttes som grunnlag for øvrige studier, som kan konsentrere seg om å bekrefte eller avkrefte funn gjort i denne studien.

8.6 Begrensninger

Studien har sett på de overordnede elementene ved implementasjon av Scrum sammen med offshoring i systemutviklingsprosjekter. Det har med rammene lagt til grunn ikke vært mulig å studere prosjektene i detalj, derfor ei heller diskutere erfaringer knyttet til lokale problemstilling internt hos de ulike lokasjonene.

Det har heller vært oppgavens fokus å bringe frem de aller viktigste erfaringene prosjektene har gjort seg når det gjelder å binde sammen ett prosjekt over store avstander. De to hovederfaringene i så måte er oppstartsform og kommunikasjon. Disse er igjen knyttet opp

til sosiale relasjoner og erfaring blant utviklerne som synes å påvirke både behov og innholdet i kommunikasjonen mellom lokasjoner.

Det er som leser av studien viktig å ta inn over seg at studien ikke kan brukes som håndbok for opprettelse av Scrum prosjekter kombinert med offshoring. Andre studier og litteratur vil gi en grundigere innføring i Scrum og generelle erfaringer knyttet til metoden. Denne studien kan muligens fungere som et supplement til dette med erfaringer som kan være nyttige for denne typen prosjekter. Videre ses studien primært på som et utgangspunkt for videre forskning innen emnet.

9 Konklusjon

Smidige metoder har de siste årene fått mye oppmerksomhet. Forskning har i hovedsak vært fokusert på XP, mens færre har vært viet Scrum. Denne studien har studert tolv prosjekter ved å intervjue fem personer involvert i disse. Prosjektene er av ulike størrelser med alt fra tre til 65 medlemmer og med mellom to og fire ulike land involvert.

Felles for alle prosjektene er at Scrum er benyttet sammen med offshoring. Studien har sett på hvordan prosjektene har implementert Scrum og offshoring sammen, samt hvilke erfaringer de har gjort seg underveis.

Kapittelet oppsummerer de mest sentrale funn som er gjort i studien og ser på veien videre med tanke på andre forskningsoppgaver.

9.1 Starte fra én lokasjon

De mest sentrale erfaringene prosjektene har gjort seg er knyttet til prosjektoppstart og kommunikasjon mellom medlemmer på ulike lokasjoner. De aller fleste prosjektene har valgt å starte opp prosjektet på én lokalitet, for senere å utvide prosjektet til andre land. I denne utvidelsen blir utviklere som tidligere var hentet fra det aktuelle landet, sendt tilbake for å bidra i oppstarten i landet prosjektet utvides til.

To av prosjektene valgte ikke en slik oppstartsform og erfarte at det ville vært nyttig å starte opp prosjektet fra én lokasjon. For å demme opp for manglende sosiale relasjoner etablert i denne oppstartsfasen på én lokasjon ble ekstensiv bruk av videokonferanser benyttet. Spesielt gjaldt dette for ett av de to nevnte prosjektene. Ved å benytte videokonferanse inntil fire timer hver dag, klarte medlemmene i prosjektet å etablere sosiale relasjoner og danne felles kunnskap om prosjektet. Denne felles forståelsen av prosjektet lettet kommunikasjonsbehovet, og sørget for fremgang i prosjektet.

9.2 Kommunikasjon på tvers av lokasjoner

Kommunikasjon var også en sentral erfaring de andre prosjektene i studien gjorde seg. Samtlige erkjente at kommunikasjon på tvers av store avstander er en stor utfordring, og at man ikke vil kunne ha like tett kommunikasjon i denne typen prosjekter, sammenlignet med prosjekter samlet på én lokasjon. Løsningen på denne utfordringen ble det å benytte

videokonferanser, e-post, lynmeldinger, forum, wiki og lignende. På denne måten kunne ulike kommunikasjonsbehov bli tilfredstilt ved hjelp av ulike løsninger.

I tillegg til å gi utviklerne ulike kommunikasjonskanaler, ble det i starten gitt anledning for utviklerne å komme sammen på samme sted for å etablere sosiale relasjoner, samt få en felles forståelse av prosjektet. De intervjuede personene i studien tror dette kan ha hjulpet på kommunikasjonen senere i prosjektet. I tillegg skriver Hall (1984) at det å etablere felles informasjon mellom mennesker vil føre til at man senere trenger å overføre mindre informasjon, men likevel oppnå samme grad av mening, noe Figur 25 illustrerer.

9.3 Opplæring og erfaring

De fleste prosjektene valgte å gi opplæring i mer eller mindre grad i starten av prosjektet. De prosjektene som ikke gjorde dette, identifiserte det som et behov når spørsmål om opplæring ble stilt i intervjuet. I tillegg ble det identifisert behov for erfarne utviklere i denne typen prosjekter. I alle typer prosjekter vil de fleste enes om at det er behov for en viss andel erfarne utviklere. Det kan likevel tyde på at behovet er noe større i den typen prosjekter omfattet i denne studien.

9.4 Videre forskning

Det er interessant å se nærmere på noen av erfaringene funnet i studien. Blant annet kan det være interessant for videre forskning å studere hvorvidt oppstartformen for denne typen prosjekter påvirker behovet for kommunikasjon mellom lokasjonene i de senere fasene av prosjektet. Gjerne konkret om det å starte et prosjekt fra én lokasjon påvirker behovet for kommunikasjon senere i prosjektet og hvorvidt man kan kategorisere kommunikasjonen senere som mer effektiv enn hva ellers ville vært tilfelle. I tillegg vil det også kunne være nyttig å studere hvilke kommunikasjonskanaler eller sett med kommunikasjonskanaler som fungerer best i typen prosjekter undersøkt i studien.

Det er i hovedsak få studier som forsøker å bekrefte eller avkrefte spesifikke hypoteser i forbindelse med prosjekter som benytter Scrum og offshoring. De aller fleste er case studier som ser på alle erfaringene knyttet til ett prosjekt. Denne studien trekker noe vekk fra dette ved å undersøke flere prosjekter i en og samme studie. Det er derfor kanskje nå naturlig å ta utgangspunkt i funn gjort i denne og andre studier, og forsøke å avkrefte eller bekrefte dem. På den måten kan man nærmere seg en ”oppskrift” på hvordan denne typen prosjekter bør organiseres for å oppnå et best mulig resultat. Kanskje vil en Scrum metode tilpasset offshoring være et resultat av slik forskning.

10 Referanser

Aberdour, M. (2007). Achieving Quality in Open Source Software. *IEEE Softw., 24*(1), 58-64.

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods. [Report]. *Agile software development methods*, 107 pp.

Annabi, H. (2005). *Moving from individual contribution to group learning: the early years of the apache web server*. Syracuse University.

Avison, D. E., & Fitzgerald, G. (2003). Where now for development methodologies? *Commun. ACM, 46*(1), 78-82.

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

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). Manifesto for Agile Software Development. Retrieved 20.09, 2008, from www.agilemanifesto.org

Boehm, B. W. (1986). A spiral model of software development and enhancement. *SIGSOFT Softw. Eng. Notes, 11*(4), 14-24.

Boehm, B. W. (2002). Get Ready for Agile Methods, with Care. *Computer, 35*(1), 64-69.

Cockburn, A. (2003). *Series of dissertations submitted to the Faculty of Mathematics and Natural Sciences, University of Oslo*. Oslo: The Faculty : Unipub.

Cohen, D., Lindvall, M., & Costa, P. (2004). An introduction to agile methods. *Advances in Computers, Vol 62, 62*, 1-66.

Cohn, M., & Ford, D. (2003). Introducing an agile process to an organization. [Article]. *Computer, 36*(6), 74-+.

Danait, A. (2005). *Agile offshore techniques - a case study*. Paper presented at the Agile Conference, 2005. Proceedings.

Dennis, A., Wixom, B. H., & Tegarden, D. (2004). *Systems Analysis and Design with UML*: Wiley Text Books.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Inf. Softw. Technol., 50*(9-10), 833-859.

Feller, J. (2007). *Perspectives on free and open source software*. Cambridge, Mass.: MIT Press.

Feller, J., & Fitzgerald, B. (2002). *Understanding Open Source software development*. London: Addison-Wesley.

Fowler, M. (2005). The New Methodology. *Journal*. Retrieved from http://www.martinfowler.com/articles/newMethodology.html#rationalUnifiedProcess

Glass, R. L. (2001). Agile Versus Traditional: Make Love, Not War! *Cutter IT Journal, 14*(12), 12-18.

Grønmo, S. (2004). *Samfunnsvitenskapelige metoder*. Bergen: Fagbokforl.

Hall, E. T. (1984). *The dance of life the other dimension of time*. New York: Anchor Press.

Haugen, N. C. (2006). *An empirical study of using planning poker for user story estimation*. Paper presented at the Agile Conference, 2006.

Hellevik, O. (2002). *Forskningsmetode i sosiologi og statsvitenskap* (7. utg. ed.). Oslo: Universitetsforl.

Hildenbrand, T., Geisser, M., Kude, T., Bruch, D., & Acker, T. (2008). *Agile Methodologies for Distributed Collaborative Development of Enterprise Applications*. Paper presented at the Proceedings of the 2008 International Conference on Complex, Intelligent and Software Intensive Systems - Volume 00.

Hole, S., & Moe, N. B. (2008). A Case Study of Coordination in Distributed Agile Software Development. In *Software Process Improvement* (pp. 189-200).

Jensen, B., & Zilmer, A. (2003, 25-29 May). *Cross-continent development using Scrum and XP*. Paper presented at the Extreme Programming and Agile Processes in Software Engineering. 4th International Conference, XP 2003. Proceedings, Genova, Italy.

Karsten, P., & Cannizzo, F. (2007). The Creation of a Distributed Agile Team. In *Agile Processes in Software Engineering and Extreme Programming* (pp. 235-239).

Korkala, M., & Abrahamsson, P. (2007). *Communication in Distributed Agile Development: A Case Study*. Paper presented at the Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications.

Kruchten, P. (2003). *The Rational Unified Process : an introduction* (3rd ed.). Boston: Addison-Wesley.

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

Mann, C., & Maurer, F. (2005). *A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction*. Paper presented at the Proceedings of the Agile Development Conference.

Myers, M. D. (1997). Qualitative research in information systems. *MIS Q., 21*(2), 241-242.

Myers, M. D., & Newman, M. (2006). The qualitative interview in IS research: Examining the craft. *Journal*. Retrieved from https://studentportal.uib.no/dotlrn/classes/det-samfunnsvitenskapelige-fakultet/infomevi300/infomevi300-2008v/file-storage/download/44358549/sdarticle.pdf

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

Ockman, S., Stone, M., & DiBona, C. (1999). *Open sources : voices from the open source revolution*. Beijing: O'Reilly.

Paulk, M. C. (1993). *Key practices of the capability maturity model Version 1.1*. Pittsburgh, Pa.: Research Access for Software Engineering Institute.

Carnegie-Mellon University. Software Engineering, Institute.

Ramesh, B., Cao, L., Mohan, K., & Xu, P. (2006). Can distributed software development be agile? *Commun. ACM, 49*(10), 41-46.

Raymond, E. S. (2001). *The cathedral and the bazaar : musings on Linux and Open Source by an accidental revolutionary*. Beijing; Cambridge, Mass.: O'Reilly.

Royce, W. W. (1987). *Managing the development of large software systems: concepts and techniques*. Paper presented at the Proceedings of the 9th international conference on Software Engineering.

Salo, O. (2007). *Enabling Software Process Improvement in Agile Software Development Teams and Organisations*: Faculty of Science, University of Oulu.

Schwaber, K. (2004). *Agile project management with Scrum*. Redmond, Wash.: Microsoft Press.

Schwaber, K., & Beedle, M. (2001). *Agile Software Development with SCRUM*: Prentice Hall.

Schwaber, K., & Beedle, M. (2002). *Agile software development with Scrum*. Upper Saddle River, NJ: Prentice Hall.

Sharp, H., & Robinson, H. (2004). An ethnographic study of XP practice. *Empirical Software Engineering, 9*(4), 353-375.

Sureshchandra, K., & Shrinivasavadhani, J. (2008). *Adopting Agile in Distributed Development*. Paper presented at the Proceedings of the 2008 IEEE International Conference on Global Software Engineering.

Sutherland, J., & Schwaber, K. (2007). The Scrum Papers: Nuts, Bolts, and Origins of an Agile Process. *Journal*, 202. Retrieved from http://scrumtraininginstitute.com/home/stream_download/scrumpapers

Sutherland, J., Viktorov, A., Blount, J., & Puntikov, N. (2007). *Distributed Scrum: Agile Project Management with Outsourced Development*

Teams. Paper presented at the Hawaii International Conference on Software Systems. Retrieved 5.5.08, from http://jeffsutherland.com/scrum/SutherlandDistributedScrumHICSS2007FinalSubmission.pdf

Takeuchi, H., & Nonaka, I. (1986). The New New Product Development Game. [Harvard Business Review Article]. *Harvard Business Review*

Therrien, E. (2008). *Overcoming the Challenges of Building a Distributed Agile Organization*. Paper presented at the Proceedings of the Agile 2008.

Vax, M., & Michaud, S. (2008). *Distributed Agile: Growing a Practice Together*. Paper presented at the Proceedings of the Agile 2008.

Walt, S. (2007). *Free/open source software development*. Paper presented at the Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering.

Watts, S. H. (1995). *A Discipline for Software Engineering*: Addison-Wesley Longman Publishing Co., Inc.

Vedlegg 1: Intervjuguide

Generell informasjon om prosjektet

- Hva slags system som er/blir utviklet?
- Antall utviklere?
- Utviklernes bakgrunn / erfaring (offshoring, scrum, utvikling)
- Prosjektvarighet
- Erfaringer med Scrum (generelle)
- Hvordan er prosjektet organisert?
    - Lokaliteter?
    - Hvor befinner hvem seg?

Oppstart av prosjekt

- Implementasjon
    - Hvordan er Scrum implementert?
    - Hvordan er offshoring implementert?
- Beskriv oppstarten av prosjektet.
- Opplæring
    - Ble det gitt Scrum opplæring?
    - Ble det gitt domeneopplæring (opplæring innen domene systemet skal operere i)?
    - Ble det gitt annen opplæring?

Erfaring med Scrum og offshoring

- Erfaringer fra ulike Scrum-praksiser
    - Hvilke fungerer bra og hvilke fungerer dårlig
- Erfaringer med Scrum og offshoring
    - Hvilke erfaringer er gjort knyttet til implementasjon av Scrum og offshoring
        - Hvilken modell er benyttet?
    - Hvilke erfaringer knyttet til kommunikasjon med offshore-team er gjort?
    - Hvilke negative erfaringer / problemer underveis har prosjektet hatt?
- Hvilke endringer er evt. gjort underveis?
- Hvilke endringer er ønskelig å gjennomføre før neste prosjekt?
- Salg av prosjektet til kunde

    - Fastpris? Timebasert? Iterasjonsbasert?
    - Utfordringer knyttet til salg av prosjekt?
- Tilbakemeldinger fra kunde
    - Hvordan er tilbakemeldingene fra kunde, brukere etc.?
- Er prosjektet levert til avtalt tid og pris?

Vedlegg 2: Utdrag fra transkribering

Utdrag fra intervju med Rakesh Jaiswal

	SP: HVORDAN VAR PLANLEGGINGEN ORGANISERT? **SVAR**: DET ER PÅ PROSJEKTLEDELSENIVÅ AT ORGANISERING OG GENERELL PLANLEGGING FOR PROSJEKTET FOREGÅR. DE PLANLEGGER FOR HVA SOM SKAL UTVIKLES DET NESTE ÅRET. ETT ÅR ER DELT INN I FIRE STORE LEVERANSER SOM ER KVARTALSBASERT. SÅ BLIR HVERT KVARTAL DELT INN I TOLV ITERASJONER SOM HVER VARER 15 DAGER. FOR HVERT KVARTAL DEFINERER MAN DE VIKTIGSTE OPPGAVENE SOM LEGGES I DE FØRSTE ITERASJONENE. DE NESTE ITERASJONENE BENYTTES TIL LAVERE PRIORITERTE OPPGAVER ELLER Å FIKSE FEIL FRA DE FORRIGE ITERASJONENE. DETTER ER DET HØYNIVÅ PLANLEGGINGSSTRUKTUREN FOR PROSJEKTET. **SP**: HVEM UTFØRER DENNE PLANLEGGINGEN? **SVAR**: PLANNLEGGING OG ESTIMERING ER ANSVARET TIL SCRUM MASTERNE OG ”PRODUCT MANAGER” SOM HAR TETTESTE KONTAKT MED KUNDEN SOM HAR MEST KUNNSKAP OM HVILKE OPPGAVER SOM VIL GI MEST FORRETNINGSVERDI. **SP**: FIKK MEDLEMMENE AV PROSJEKTET NOEN FORM FOR

	OPPLÆRING? I SÅ FALL HVA SLAGS? **SVAR**: JA ALLE MEDLEMMER FIKK OPPLÆRING I SCRUM. HVILKE REGLER, TEKNIKKER OG METODER SOM FINNES I SCRUM OG HVORDAN MAN SKAL BENYTTE DEM. **SP**: HVORFOR VALGTE DERE Å STARTE OPP PROSJEKTET SLIK? **SVAR**: DETTE PROSJEKTET ER EN AV DE FEM MEST KRITISKE OG OMFATTENDE APPLIKASJONENE HOS DENNE KUNDEN, OG DET ER DERFOR VANSKELIGERE FOR UTVIKLERNE Å FORSTÅ PROSJEKTET. SPESIELT LEVERANSEPROSESSEN OG PRODUKSJONSMILJØET. DATAENE I DATABASEN ER OGSÅ SVÆRT SENSITIVT OG DATABASEN ER ANTAGELIG DEN STØRSTE I VERDEN. DERFOR SES OPPLÆRING PÅ SOM SVÆRT VIKTIG I DETTE PROSJEKTET. ALLE MÅ HA EN GOD FORSTÅELSE AV PROSJEKTET. **SP**: HVILKE UTFORDRINGER ELLER PROBLEMER HAR DUKKET OPP UNDERVEIS I PROSJEKTET? **SVAR**: I STARTEN HADDE VI EN DEL PROBLEMER MED ESTIMERING OG DA SPESIELT FORSTÅELSEN AV DE TEKNIKKENE VI BENYTTET. DETTE GIKK SEG TIL ETTER HVERT VED AT VI BRUKTE TID PÅ Å OPPLÆRE MEDLEMMENE I PRINSIPPENE BAK ESTIMERING AV OPPGAVER.

	SP: HVORDAN FUNGERER KOMMUNIKASJONEN MELLOM LOKASJONENE? **SVAR**: VI HAR EN DAGLIG STANDUP TELEFONKONFERANSE SOM ALLE DELTAR I. DET ER OGSÅ SATT OPP ET FORUM SOM KUNDEN DRIFTER. DET KAN VI FOR EKSEMPEL DISKUTERE DE ULIKE OPPGAVENE VI HOLDER PÅ MED. VIDEOKONFERANSE SKJER CA. ÉN GANG I MÅNEDEN. **SP**: HVORDAN FUNGERTE DISSE DAGLIGE SAMTALEN MED STAND UP? **SVAR**: DET ER NÅ I EN MER NATURLIG FORM ENN DET VAR I STARTEN. I STARTEN VAR IKKE UTVIKLERNE FLINK MED Å BESKRIVE HVA SOM VAR HINDRINGENE I FORBINDELSE MED EN OPPGAVE. DE UTRYKKER SEG NÅ MER NATURLIG OG KLARERE I FORHOLD TIL TIDLIGERE. **SP**: HVORDAN VAR OPPSTARTEN AV PROSJEKTET? **SVAR:** I STARTEN VAR 8 OFFSHORE UTVIKLERE ONSHORE I 2 MÅNEDER FOR Å FORSTÅ PROSJEKTETS ARKITEKTUR OG PROSJEKTETS PROSESS OG LEVERANSEVERKTØY OG

	METODOLOGI OG LEVERANSESYKLUSER. I TILLEGG FIKK DE OGSÅ MULIGHET FOR Å FÅ KJENNSKAP TIL PRODUKSJONSMILJØET. ETTER DENNE PERIODEN RETURNERTE FIRE AV DISSE TIL INDIA MENS DE FIRE RESTERENDE BLE IGJEN I ENGLAND SOM OFFSHORE LEAD. DE SOM RETURNERTE SATT OPP PROSJEKTET I INDIA. HVOR DE ETABLERTE SCRUM TEAMENE MED CA. 10 MEDLEMMER I HVERT TEAM.

Utdrag fra intervju med Mokashi

	SP: KAN DU FORTELLE OM OPPSTARTEN AV DETTE PROSJEKTET? **SVAR**: NOEN AV OFFSHORE UTVIKLERNE REISTE ONSHORE I STARTEN HVOR DE DELTAR I DE FØRSTE ITERASJONENE. DE FIKK I DENNE PERIODEN FORSTÅELSE AV MILJØET, FUNKSJONALITETEN TIL SYSTEMET, FORRETNINGSKRAVENE OG FORSTÅELSE AV ARKITEKTUREN. DET KOM SÅ TILBAKE TIL INDIA FOR Å SETTE OPP PROSJEKTET DERFRA. SETTE OPP ARKITEKTUR OG MILJØET FØR DE INKLUDERE FLERE MEDLEMMER INN I PROSJEKTET. DE OVERFØRER SÅ KUNNSKAP DE HAR OVER TIL DE NYE PROSJEKTMEDLEMMENE. DERETTER FORTSETTER ITERASJONENE FRA OFFSHORE. DEN VANLIGSTE TRENDEN ER Å STARTE OPP PROSJEKTER ONSHORE, HENTER OVER EN DEL MEDLEMMER, GIR DEM OPPLÆRING OG LAR DEM BLI KJENT MED HVERANDRE OG BYGGE SOSIALE RELASJONER, FØR DE RETURNERER OG ARBEIDET VIDERE FRA OFFSHORE. **SP**: ER DET NOEN REISER MELLOM LOKASJONENE

	UNDERVEIS I PROSJEKTET? **SVAR**: JA DET ER LITT REISING. NOEN GANGER ER DET FOR Å OVERFØRE KUNNSKAP, ANDRE GANGER FOR TESTING. ELLER FOR Å FORSTÅ PROSESSENE PÅ DE ULIKE LOKASJONENE ELLER ORDNE OPP I PROBLEMSTILLINGER SOM HAR DUKKET OPP. **SP**: HVORDAN ER KOMMUNIKASJONEN SATT OPP MELLOM LOKASJONENE? **SVAR**: DET ER DAGLIGE KONFERANSESAMTALER FOR STAND UP. ALLE I PROSJEKTET DELTAR I DISSE MØTENE. ALLE SOM ER ONSHORE OG ALLE SOM ER OFFSHORE. NOEN GANGER ER DET INTERNE STAND UP MØTER ONSHORE OG OFFSHORE, HVOR SÅ SCRUM MASTERNE KOORDINERER RESULTATET FRA DISSE PÅ TVERS AV LOKALITETENE. I DETTE PROSJEKTET ER DET ALLTID EN TELEFONSAMTALE MELLOM ONSHORE OG OFFSHORE DAGLIG. SÅ ER DET NOEN UKENTLIGE MØTER FOR PLANLEGGING. I DETTE PROSJEKTET HAR VI UKENTLIGE DEMONSTRASJONER FOR ONSHORE MEDLEMMER OG NOEN GANGER TIL KUNDEN. SELV OM DEMONSTRASJONENE KAN VÆRE HVER UKE ER SPRINTENE I DETTE PROSJEKTET 2 UKER LANGE. NOEN GANGER VIL MAN VISE NOE TIL KUNDEN I MIDTEN AV ET INTERSJON OG DA ER DET UPROBLEMATISK FOR OSS. DET ER GOD ØVELSE Å HYPPIG PRESENTERE DET MAN LAGER TIL ANDRE. **SP**: BENYTTES VIDEOKONFERANSER I DETTE ELLER DISSE PROSJEKTENE? **SVAR**: DISSE PROSJEKTENE BENYTTER

	VIDEOKONFERANSER I LITEN GRAD PÅ GRUNN AV UTFORDRINGER KNYTTET TIL KOSTNADER OG INFRASTRUKTUREN PÅKREVD FOR SLIKE KONFERANSER. DERFOR BENYTTES WEBEX MYE FOR Å VISE FREM TING TIL DEN ANDRE LOKASJONEN. ELLERS KAN ANDRE "DESKTOP SHARING" LØSNINGER BENYTTES SAMMEN MED TELEFONKONFERANSE. SP: HVORDAN PÅVIRKER DETTE KOMMUNIKASJONEN? SVAR: VEL, DETTE VIL IKKE VÆRE LIKE BRA SOM OM MAN VAR LOKALISERT PÅ SAMME STED. BYGGING AV SOSIALE RELASJONER OG TILLIT TAR LENGRE TID PÅ DENNE MÅTEN ENN ELLERS. DET ER DERFOR BEHOV FOR Å GJØRE UNNA NOEN AV DENNE DELEN I STARTEN AV PROSJEKTET HVOR MANGE OFFSHORE MEDLEMMER ER ONSHORE. PROSJEKTET ER KLAR OVER DENNE PROBLEMATIKKEN OG FORSØKER Å ADRESSERE DEN, VED Å LA MEDLEMMER REISE TIL DEN ANDRE LOKALITETEN MED JEVNE MELLOMROM. MAN MÅ JOBBE MED DETTE. FOR EKSEMPEL VED Å HA LENGRE INTRODUKSJONSSAMTALER, ELLER SAMTALER SOM IKKE ER ARBEIDSRELATERT FOR AT MEDLEMMENE SKAL BLI KOMFORTABLE MED HVERANDRE.