Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

MASTEROPPGAVE

Studieprogram/ spesialisering: **Industriell økonomi/ Prosjektledelse**	Vårsemesteret, 2012 Åpen
Forfatter: **Marina Haugvaldstad**	…… (signatur forfatter)
Fagansvarlig: **Eric Christian Brun** Veileder: **Eric Christian Brun**	
Tittel på masteroppgaven: **Vurdering av agile prosjektledelse metoder** Engelsk tittel: **Evaluation of agile project management methods**	
Studiepoeng: 30	
Emneord: Scrum XP DSDM Lean APF Kunden Team Kommunikasjon Usikkerhet	Sidetall: 89 + vedlegg/annet: 3 Stavanger, 11. juni 2012

SAMMENDRAG

I løpet av de siste årene har det blitt svært populært å benytte agile tilnærminger for å oppnå planlagte mål og gode resultater i et prosjektarbeid. Flere ulike prosjektledelse metoder kan bli brukt til disse formål, men ikke alle metoder kan garantere prosjektets suksess.

Bruken av den mest passende metoden vil bidra til økt sjanse for et vellykket prosjekt. Derfor har hovedfokuset i denne masteroppgaven vært å vurdere agile metoder som brukes i prosjektstyring og utarbeide et grunnlag for valg av den best passende prosjektledelse metoden.

Siden de fleste smidige metodene har blitt designet for utarbeidelse av IT-systemer, ble de mest populære prosjektledelse metodene innen programvareutviklingen tatt med i vurderingen. Scrum, Extreme Programming, Dynamic Systems Development Method, Lean Software Development og Adaptive Project Framework ble vurdert i forhold til utviklingsprosess, roller, team, kunden, kommunikasjon og usikkerhetshåndtering.

Den gjennomførte vurderingen har vist at alle de fem undersøkte metodene har kort utviklingsprosess, fokus på små team, meningsfull kundeinvolvering, effektiv kommunikasjon og usikkerhetshåndtering gjennom hele prosjektets livsløp. Det betyr at alle metoder har høy smidighetsgrad og kan bli benyttet i ulike agile prosjekter. Samtlige vurderinger har konkludert med at den største utfordringen som er relatert til valget av den mest passende metoden kan bli effektivt løst ved å ta i betraktning mål, spesifikasjoner og behov det konkrete prosjektet står overfor.

INNHOLDSFORTEGNELSE

SAMMENDRAG 2
INNHOLDSFORTEGNELSE 3
OVERSIKT OVER FIGURER 7
OVERSIKT OVER TABELLER 8
FORORD 9
DEL 1 10
INNLEDNING 10
1.1. BAKGRUNN 11
1.2. FORMÅL 12
1.3. OPPGAVENS OPPBYGNING 12
DEL 2 13
TEORETISKE ASPEKTER 13
2.1. HVA BETYR Å VÆRE ”AGILE”? 14
2.2. ”THE AGILE MANIFESTO” 14
2.3. KARAKTERISERING AV AGILE METODER 16
2.4. HVA KJENNETEGNER AGILE PROSJEKTER? 17
2.5. AGIL PROSJEKTLIVSSYKLUS 18
2.5.1. Iterativ prosjektmodell 18
2.5.2. Adaptiv prosjektmodell 19
2.6. ET SMIDIG TEAM 21
2.6.1. Selvmotivasjon og selvorganisering 21
2.6.2. Generaliserende spesialister. 22
2.6.3. Teamstørrelse 24
2.6.4. Ferdigheter 25

2.6.5. Teamsammensetning og roller 27
2.6.6. Prosjektlederens rolle 29
2.7. KOMMUNIKASJON 30
2.7.1. Effektiv kommunikasjon 30
2.7.2. Ansikt-til-ansikt kommunikasjon 31
2.7.3. Suksessfaktorer 32
2.7.4. Prosjektleder i kommunikasjonsprosess 33
2.7.5. Daglige møter 34
2.8. KUNDEN 35
2.9. USIKKERHET I AGILE PROSJEKTER 37
DEL 3 43
VURDERING AV EKSISTERENDE AGILE METODER 43
3.1. SCRUM 44
3.1.1. Prosess 44
3.1.2. Roller og ansvar 46
3.1.3. Team 47
3.1.4. Kunden 47
3.1.5. Kommunikasjon 48
3.1.6. Usikkerhetshåndtering 49
3.2. EXTREME PROGRAMMING (XP) 50
3.2.1. Prosess 51
3.2.2. Roller og ansvar 52
3.2.3. Team 53
3.2.4. Kunden 54
3.2.5. Kommunikasjon 55
3.2.6. Usikkerhetshåndtering 56

3.3. DSDM 57
3.3.1. Prosess 57
3.3.2. Roller og ansvar 58
3.3.3. Team 59
3.3.4. Kunden 60
3.3.5. Kommunikasjon 61
3.3.6. Usikkerhetshåndtering 62
3.4. LEAN SOFTWARE DEVELOPMENT 63
3.4.1. Prosess 63
3.4.2. Roller og ansvar 64
3.4.3. Team 65
3.4.4. Kunden 66
3.4.5. Kommunikasjon 67
3.4.6. Usikkerhetshåndtering 68
3.5. ADAPTIVE PROJECT FRAMEWORK 69
3.5.1. Prosess 69
3.5.2. Roller og ansvar 70
3.5.3. Team 71
3.5.4. Kunden 71
3.5.5. Kommunikasjon 72
3.5.6. Usikkerhetshåndtering 73
DEL 4 74
TOLKNING OG DISKUSJON 74
4.1. PROSESS 75
4.2. ROLLER OG ANSVAR 78
4.3. TEAM 80
4.4. KUNDEN 81

4.5. KOMMUNIKASJON 83
4.6. USIKKERHETSHÅNDTERING 85
4.7. OPPSUMMERING 86
DEL 5 88
KONKLUSJON 88
5.1. KONKLUSJON 89
REFERANSER 90

OVERSIKT OVER FIGURER

Figur 2.1. Iterativ prosjektmodell 19
Figur 2.2. Adaptiv prosjektmodell 20
Figur 2.3. Generaliserende spesialist 23
Figur 2.4. Et smidig team 28
Figur 2.5. Toveiskommunikasjon 30
Figur 2.6. Rikhetsskalaen av kommunikasjonskanaler 31
Figur 2.7. Kundesenteret utviklingsprosess 36
Figur 2.8. Risiko og muligheter 37
Figur 2.9. Usikkerhetstyper 38
Figur 2.10. Endring i sikkerhetsbilde 38
Figur 2.11. Konus av usikkerhet 39
Figur 2.12. Innsnevring av usikkerhetskonus 40
Figur 3.1. Scrum livssyklus 45
Figur 3.2. XP livssyklus 51
Figur 3.3. DSDM utviklingsprosess 57
Figur 3.4. Lean utviklingsprosess 63
Figur 3.5. APF livssyklusmodell 69
Figur 4.1. Teamstørrelse i smidige metoder 81

OVERSIKT OVER TABELLER

Tabell 2.1. Karakterisering av agile metoder 17

Tabell 2.2. Gradering av kompleksitet 41

Tabell 2.3. Gradering av usikkerhet 42

Tabell 4.1. Sammenligning av metoder i forhold til deres utviklingsprosess 77

Tabell 4.2. Sammenligning av metoder i forhold til roller og ansvar 79

Tabell 4.3. Sammenligning av metoder i forhold til teamets egenskaper 80

Tabell 4.4. Kundeinvolvering i smidige metoder 82

Tabell 4.5. Sammenligning av kommunikasjonsprosess i smidige metoder 83

Tabell 4.6. Smidige metoder i forhold til vurderingskriterier 86

FORORD

Denne oppgaven er det avsluttende arbeidet ved det 2-årige masterstudiet i Industriell Økonomi ved Universitetet i Stavanger. Masteroppgavens tema ble valgt ut fra både min spesialisering innen prosjektledelse og egen interesse for implementering av agile metoder. Arbeidet med oppgaven har gitt meg en verdifull mulighet til å fordype meg faglig innen prosjektstyring og utvidet min horisont i programvareutvikling.

Jeg vil benytte anledning til å takke alle som har bidratt med innspill og hjelp med denne oppgaven.

Først retter jeg en stor takk til min veileder Eric Christian Brun for god støtte, faglig sjenerøsitet og svært konstruktive veiledninger. Videre vil jeg takke min venn Roger Nyseth for korrekturlesing og nyttige tilbakemeldinger underveis. I tillegg ønsker jeg å takke mine foreldre i Russland som har lært meg å aldri gi opp, jeg er evig takknemlig for det.

Til slutt rettes den største takken til min kjære datter Viktoria for hennes forståelse og tålmodighet, nå er mamma tilbake for fullt!

Stavanger, 5.juni 2012

Marina Haugvaldstad

DEL 1

INNLEDNING

1.1. BAKGRUNN

Flere og flere industrier favoriserer prosjektarbeid som en problemløsende arbeidsform og benytter seg av mest mulig effektiv levering av produkter og tjenester. Effektiv prosjektledelse setter et sterkt fokus på tids-, kostnads- og ressursbesparelse, og valg av riktig metode kan være helt avgjørende for prosjektets suksess. Derfor anses valget av den beste og mest effektive metoden som en av de største utfordringene i prosjektstyring.

I løpet av de siste 60 årene har prosjektledelse metoder utviklet seg, og flere teoretikere har gjort forskjellige tilnærminger basert på ulike tenkemåter. Prosjektledelses teori skiller seg mellom tradisjonell og agil prosjektledelse, og ulike modeller fra begge kategorier er tilgjengelig for bruk.

I praksis viser det seg at agile metoder blir mer og mer foretrukket framfor tradisjonelle metoder. Grunnen til det er at agile metoder kan implementeres i komplekse og usikre prosjekter og har gode muligheter til å oppnå forhåndsbestemte mål, samt lykkes ved prosjektgjennomføring. Men fungerer de agile metodene like godt i alle prosjekter? Hvilke prosjekter passer de ulike metodene best til? Hva er mulighetene for suksess ved bruk av forskjellige smidige tilnærminger?

1.2. FORMÅL

Målet for denne masteravhandlingen er å se nærmere på hvordan agile prosjektledelse metoder fungerer i forhold til deres utviklingsprosess, rollefordeling, kundeinvolvering, kommunikasjon og usikkerhetshåndtering. Dette innebærer å se på fem agile prosjektledelse metoder innen programvareutvikling og disse er Scrum, Extreme Programming (XP), Dynamic Systems Development Method (DSDM), Lean Software Development (LSD) og Adaptive Project Framework (APF). På bakgrunn av disse undersøkelsene vil det bli gjennomført vurderinger av de overnevnte metodene og utarbeidet et grunnlag for mest passende metode i ulike agile prosjekter.

1.3. OPPGAVENS OPPBYGNING

Masteroppgaven består av fem deler. I den første delen tas det opp problemstillingen, målet og presentasjon av oppgavens sammensetning. Den andre delen inneholder teori som er relevant til problemstillingen og det som er felles for alle agile metoder. I tredje delen av oppgaven utføres selve vurderingen av fem agile prosjektledelse metoder. Hver metode vurderes i forhold til utviklingsprosess, livssyklusmodell, roller og ansvar, forhold til kunden, kommunikasjonsprosess og håndtering av usikkerhet. Videre blir metodene diskutert og sammenlignet i den fjerde delen. Siste del av oppgaven konkluderer til en løsning på problemstillingen og svarer på spørsmål om hvilke prosjekter som de vurderte metodene kan implementeres i på en best og mest effektiv måte.

DEL 2

TEORETISKE ASPEKTER

2.1. HVA BETYR Å VÆRE ”AGILE”?

I de siste årene har det blitt veldig populært å bruke agile metoder i prosjektledelse. Metodene tillater å oppnå en effektiv og mer fleksibel prosjektstyring. Men hva betyr det egentlig å være agile?

Begrepet ”agile” oversettes til ”å være smidig” på norsk og kommer opprinnelig av det engelske uttrykket ”agility”, som betyr å ha evnen til en hurtig, smidig og rask handling. En slik handling kan beskrives med en ferdighet til å bevege seg raskt og bestemt, og en evne til å endre retning, samt opprettholde balanse og likevekt.

Agile metoder er vagt definert og det finnes flere tilnærminger til definisjonen av de smidige metodene. Cockburn beskriver agile metoder som lettvektige og tilpasningsdyktige [1]. Highsmith sier at det å være agile innebærer å kunne levere raskt, endre raskt og ofte [2]. Agile metoder kan gjerne også bli beskrevet som en ”ny” tenkemåte i forhold til prosjektgjennomføring [3]. Det som er ”nytt” med de agile metodene er ikke den praksisen som de bruker, men deres anerkjennelse av mennesker som de viktigste driverne i prosjektets suksess, kombinert med et intenst fokus på effektivitet og manøvrerbarhet [4]. Praktikerne er enige i at det å være ”agile” innebærer mer enn å følge retningslinjer for å gjøre et prosjekt smidig. Ekte ”agility” er noe mer enn bare en samling av praksis, det er en sinnstilstand. Andrea Branca sier at ”andre prosesser kan se agile ut, men de vil ikke føles agile” [5].

2.2. ”THE AGILE MANIFESTO”

Agile metoder ble primært designet for systemutvikling som hadde behov for en bedre og mer effektiv utvikling av programvarer. Som en løsning på de unike utfordringene i programvare utviklingsprosjekter, begynte dataprogrammererne å bruke teknikker som lignet på agile tidlig på 1950-tallet. Senere ble disse teknikkene grovt anerkjent som en alternativ tilnærming til prosjektledelse. I 2001 ble det utarbeidet ”The Agile Manifesto” som hadde til hensikt å definere felles fokusområder for agil programvareutvikling.

”The Agile Manifesto” forsøker å beskrive allmenne prinsipper for en effektiv måloppnåelse ved bruk av agile metoder. I følge Wisocki gjelder manifestet alle agile metoder som kan anvendes i programvare-, produkt- og tjenesteutvikling [6].

Hensikten bak manifestet er å bli forberedt til å møte utfordringer i den moderne prosjektstyringen ved å være mer fleksibel og kundeorientert.

Manifestet setter fokus på følgende verdier [7]:

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

De sentrale verdiene er bygget på tolv hovedprinsipper som agilister følger opp [7]:

1. *Vår høyeste prioritet er å tilfredsstille kunden gjennom tidlige og kontinuerlige leveranser av programvare som har verdi.*
2. *Ønsk endringer i krav velkommen, selv sent i utviklingen. Smidige prosesser bruker endringer til å skape konkurransefortrinn for kunden.*
3. *Lever fungerende programvare hyppig, med et par ukers til et par måneders mellomrom. Jo oftere, desto bedre.*
4. *Forretningssiden og utviklerne må arbeide sammen daglig gjennom hele prosjektet.*
5. *Bygg prosjektet rundt motiverte personer. Gi dem miljøet og støtten de trenger, og stol på at de får jobben gjort.*
6. *Den mest effektive måten å formidle informasjon inn til og innad i et utviklingsteam, er å snakke ansikt til ansikt.*
7. *Fungerende programvare er det primære målet på fremdrift.*
8. *Smidige metoder fremmer bærekraftig programvareutvikling. Sponsorene, utviklerne og brukerne bør kunne opprettholde et jevnt tempo hele tiden.*
9. *Kontinuerlig fokus på fremragende teknisk kvalitet og godt design fremmer smidighet.*
10. *Enkelhet – kunsten å maksimere mengden arbeid som ikke blir gjort – er essensielt.*
11. *De beste arkitekturer, krav og design vokser frem fra selvstyrte team.*
12. *Med jevne mellomrom reflekterer teamet over hvordan det kan bli mer effektivt og så justerer det adferden sin deretter.*

2.3. KARAKTERISERING AV AGILE METODER

Selv om det finnes ingen allment akseptert definisjon, er det enighet om at agile metoder har felles egenskaper. Iterativ utvikling, omstillingsevne, fokus på samhandling og redusert byråkrati representerer felles kjennetegn som beskriver de smidige metodene.
De sentrale prinsippene gir mulighet til en raskere endringstilpasning basert på kundenes tilbakemeldinger og forutsetter en hurtig levering av et produkt. Smidige metoder er generelt en effektiv tilnærming til moderne prosjektledelse, men de krever betydelig disiplin og samordning på grunn av sin mangel på struktur.

Det hevdes at agile metoder er utviklet for å oppnå følgende [4]:

- Produksjonen av første prototype vil kunne gi en rask tilbakemelding og en levering av det endelige produktet på et tidlig tidspunkt
- Design av enkle løsninger som fører til færre forandringer
- Kontinuerlig forbedring av designkvalitet, noe som gjør neste iterasjon mindre kostnadskrevende
- Stadig testing for tidligst mulig eliminering av defekter

Cockburn verdsetter bruk av lette og tilstrekkelige regler med fokus på mennesket og kommunikasjonsfaktorer. Han definerer disse reglene som kjernen av agile programvareutviklings metoder og foreslår prinsipper som bringer ”agility”. Tilstedeværelse av følgende prinsipper i programvareutvikling styrker mulighetene for et vellykket prosjekt resultat [1]:

- Team på to til åtte personer
- Erfarne utviklere
- Kontinuerlige og raske tilbakemeldinger
- Korte intervaller som sikrer rask testing og endringstilpasning
- Helautomatiske regresjonstester som tillater kontinuerlig forbedring

I følge Boehm kjennetegnes agile metoder ved følgende karakteristika (Tabell 2.1):

Tabell 2.1. Karakterisering av agile metoder [8].

Faktorer	Karakteristika
Hovedmål	Rask verdi
Størrelse	Et lite team
Kravspesifikasjon	Rask endring
Kunder	Dedikerte, kunnskapsrike, samarbeidende, representative og maktfulle. Samlokalisering
Utviklere	Smidige, kunnskapsrike, samarbeidende. Samlokalisering
Arkitektur	For pågående krav
Refaktorisering	Billig

2.4. HVA KJENNETEGNER AGILE PROSJEKTER?

I følge Wicoski implementeres agile metoder i prosjekter som har et definert mål og en ukjent løsning på måloppnåelse. Prosjekter som bærer ”agility” karakteriseres av følgende særtrekk [6]:

- Mellomstore og store prosjekter
- Middels til høy usikkerhet
- Middels til høy risiko
- Stor grad av kompleksitet
- Forretningsmulighet som ikke ble utnyttet tidligere
- Meningsfylt kundeengasjement
- Et lite og samlokalisert team
- Toveis kommunikasjon
- Endringsorienterte

2.5. AGIL PROSJEKTLIVSSYKLUS

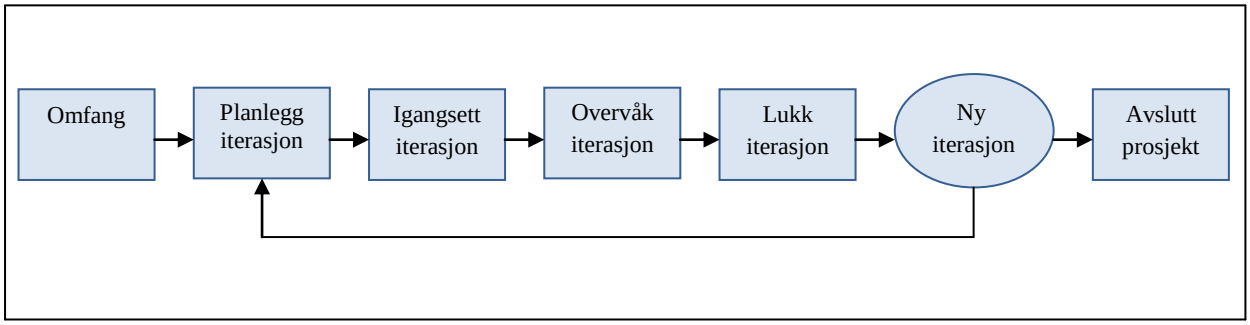
Agil prosjektlivssyklus er hovedsakelig representert av to typer modeller, iterative og adaptive. Begge modelltypene følger samme mønster, men skiller seg fra hverandre ved behov for kundeinvolvering, varierende grad av usikkerhet og forventninger om endringer underveis.

2.5.1. ITERATIV PROSJEKTMODELL [6]

Iterative modeller passer definitivt til prosjekter som gir muligheter til å lære og oppdage. Slike modeller implementeres i prosjekter der løsning på måloppnåelse er delvis kjent eller i prosjekter med forventninger om kommende endringer i omfanget. Det betyr at et slikt prosjekt bærer middels til høy usikkerhet om hva det skal levere (et produkt) og en liten usikkerhet om hvordan det skal oppnås (en prosess). I iterative prosjekter er det meste av løsningen kjent eller har blitt oppdaget, men der er det bare en del av detaljene som er uavklart eller vanskelige å definere. Siden det meste av løsningen er kjent, ligger det mindre risiko for store endringer ved prosjektgjennomføring. Det vil si at behovet for kundeinvolvering i prosjektet er redusert og ikke så stort som i adaptive modeller.

Iterative modeller gir mulighet til å få et sluttprodukt ved bruk av simuleringer og prototyper. Målet med en slik simulering eller en slik prototype er å vise kunden en midlertidig og kanskje ufullstendig løsning og få tilbakemelding om mulige forbedringer.

En iterativ modell består av en rekke prosessgrupper som gjentas sekvensielt med en feedback-løkke. En slik sekvensiell gjentakelse av plan- og gjennomføringsfasene utgjør en iterasjon som resulterer i en gradvis utarbeidelse av et ønskelig sluttprodukt (Figur 2.1). Iterasjonsprosessen gjentar seg til enten kundebehov er oppfylt eller budsjettet og tiden renner ut. Selv om det er liten risiko for forandringer, gir den iterative modellen mulighet til endringer mellom iterasjoner ved behov. Gjentakelse av iterasjoner på et tidlig tidspunkt i prosessen tillater å utvikle et sluttprodukt på en rask og rimelig måte.



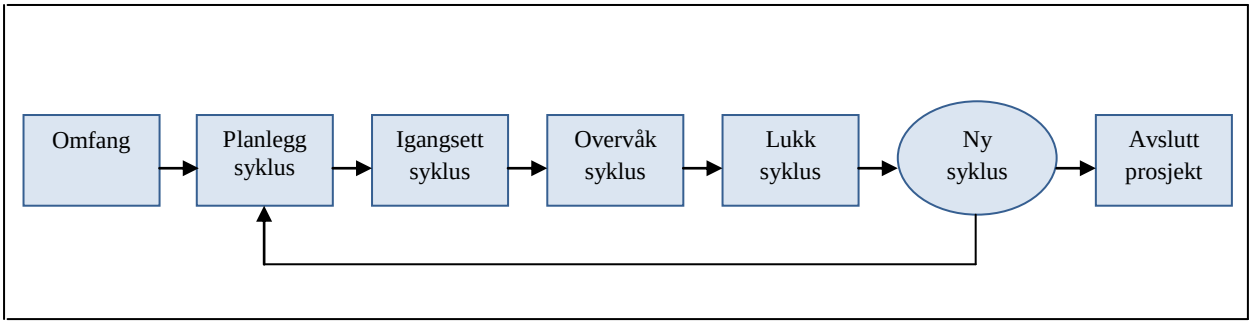
Figur 2.1. Iterativ prosjektmodell.

Iterative modeller tillater kunden å følge utviklingen og foreslå ønskelige forbedringer underveis. Modellene gir rom for omfangsendringer mellom iterasjoner og mulighet for tilpasning til endrede markeds- og forretningsforhold. Utfordringer med de iterative modellene er at de krever et aktivt kundeengasjement og har et større behov for samlokaliserte team. Siden den endelige løsningen ikke kan spesifiseres ved prosjektoppstart, kan det være problematisk å implementere midlertidige løsninger.

2.5.2. ADAPTIV PROSJEKTMODELL [6]

Adaptive modeller er hovedsakelig designet for programvare utviklingsprosjekter. Modellene passer til prosjekter med høy usikkerhet om produkt og middels til høy usikkerhet om prosess. I slike prosjekter er en liten del av løsningen definert ved prosjektets oppstart. Den ukjente delen av løsningen er representert ved manglende funksjonalitet. Der er kun hovedfunksjon i produktet som er kjent, men det ligger stor usikkerhet angående del-funksjoner og features. Den manglende løsningen kan også være en usikkerhet rund utviklingsprosess av det ønskelige sluttproduktet.

Jo mindre som er kjent om løsningen, jo mer risiko, usikkerhet og kompleksitet vil være til stede. For å dekke den usikkerheten må løsningen bli oppdaget. I en adaptiv modell vil dette skje gjennom en kontinuerlig endringsprosess fra en syklus til en annen. En adaptiv modell består av en rekke faser som blir gjentatt i sykluser, med en mulighet for tilbakemeldinger etter fullførelse av hver syklus. Figur 2.2 viser måten som modellen er konstruert på.



Figur 2.2. Adaptiv prosjektmodell.

Adaptive modeller er kjent av å ha samme mønster som en iterativ modell er bygd opp av. Modellene kjennetegnes også ved just-in-time planlegging. Hver syklus begynner fra en planfase der det utarbeides en detaljplan i tillegg til den grove og overordnede planen. Hver ny syklus lærer fra foregående syklus og gir en ny tilnærming til et ønskelig produkt. Etter det ble funnet en tilfredsstillende løsning fører siste fase av en syklus til prosjekt-avslutning. Adaptive og iterative modeller ser ganske like ut, men forskjellen kommer til å bli mer åpenbar innenfor hver prosessgruppe i en adaptiv modell.

Fordelene med en adaptiv modell er at den ikke er planfokusert. Dette hindrer forespørsler om omfangsendring og reduserer tidsbruk på ikke-verdiskapende planarbeid. Modellen avgir maksimal forretningsverdi innenfor gitte kostnads- og tidsrammer. På den andre siden avhenger modellen av et meningsfullt kundeengasjement. I en adaptiv modell kreves det mer kundeinvolvering enn i en iterativ modell, og uten kundens meningsfulle engasjement har den adaptive modellen liten sjanse til å lykkes. Modellen har stor sjanse for suksess når forhold til kunden er godt etablert, dvs. at prosjektdeltakere har tidligere god erfaring med adaptive metoder med samme kunde. En annen svakhet med modellen er at den ikke gir muligheten til å definere sluttprodukt ved prosjektoppstart.

2.6. ET SMIDIG TEAM

"Et team er en liten gruppe mennesker med komplementære egenskaper (faglige, mellommenneskelige, problemløsing), som er forpliktet av et felles formål, spesifikke resultatmål, og måter å jobbe sammen på, og som holder hverandre gjensidig ansvarlig for dette" [9].

Et smidig team har ikke en konkret definisjon, men kjennetegnes av et sett kritiske faktorer som bringer ”agility” til et team. For å virke smidig bør teamet være av en liten størrelse, bestå av generaliserende spesialister, være selvmotivert, selvorganisert, samlokalisert og være i stand til å håndtere endringer i prosjektet.

2.6.1. SELVMOTIVASJON OG SELVORGANISERG

Agilister setter et stort fokus på motivasjon av individer og selvorganisering i et team. Det femte prinsippet bak det smidige manifestet sier:

”Bygg prosjektet rundt motiverte personer.

Gi dem miljøet og støtten de trenger, og stol på at de får jobben gjort” [8].

Dette prinsippet fokuserer på motivasjon hos hvert enkelt teammedlem for å oppnå suksess i et prosjekt. Følgende sitat som ble sagt av general George Smith Patton jr. er et utrolig testament som gir makt til et selvmotivert og selvorganisert team:

”Fortell ikke mennesker hvordan de skal gjøre ting, fortell de hva de skal og la de overraske deg med sine resultater” [10].

Praktisk bruk av de agile metodene viser at teammedlemmer går langt utover oppfølging av en ordre og gjennomføring av det arbeidet som ble tildelt. Motiverte teammedlemmer forventer at de kan handle profesjonelt og bestemme selv hvilken vei de skal gå for å oppnå et godt resultat på hvert tidspunkt av prosjektet, eller i det minste forventer de å stille spørsmål som skal diskuteres med andre teamdeltakere, ledelsen og kunden. Men smidige metoder fokuserer ikke bare på at motiverte individer skal dukke opp.

Det er også viktig å skape et tilstrekkelig miljø for å bygge et motivert team. De smidige metodene verdsetter et miljø der den enkelte blir utfordret til å bygge sine evner og hvor motivasjonen er et forventet resultat av miljøet.

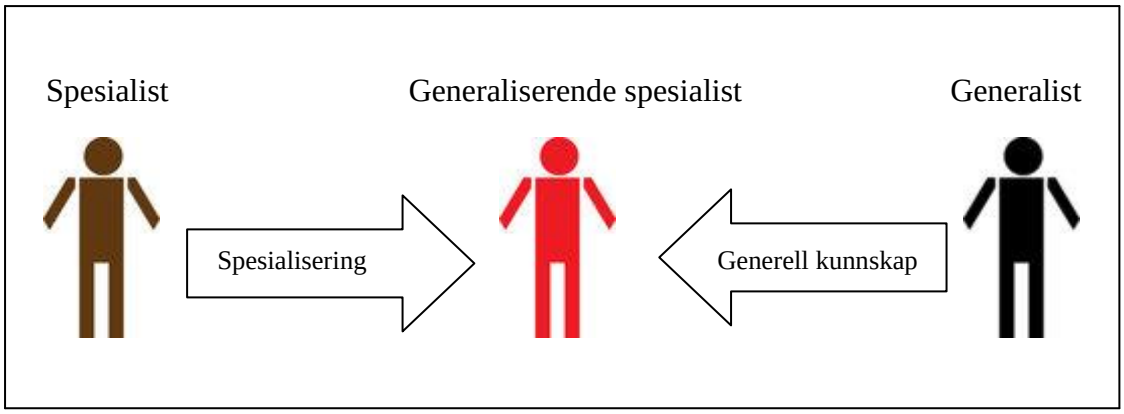
Agilister fokuserer på selvorganisering i et team som beskrives i det ellevte prinsippet. Filosofien bak prinsippet er lite bruk av kontrollmetoder og bygging av et team som har kunnskap, perspektiver, motivasjon og kompetanse. Selvorientert og selvorganisert team som får makt over sine handlinger og som er i stand til å påvirke beslutninger blir en god partner med ledelsen og kundene. Målet med det agile prinsippet er ikke bare å bygge et selvstyrt og motivert team, men det er også oppnåelse av teknisk dyktighet slik at teammedlemmer får muligheten til å holde de tekniske problemene under kontroll.

2.6.2. GENERALISERENDE SPESIALISTER [11]

Et agilt team er stort sett dannet av generaliserende spesialister (Figur 2.3).
En generaliserende spesialist er en person som:

- Har en eller flere tekniske spesialiseringer
- Har i det minste generell kunnskap om utvikling av programvarer
- Har generell kunnskap om den delen av virksomheten der den personen jobber
- Prøver aktivt å få nye ferdigheter

Generaliserende spesialister kalles ofte som håndverkere eller multidisiplinære utviklere. En generaliserende spesialist er mer enn bare en generalist. En generalist har generell og omfattende kunnskap og er ikke mester på noen som helst område. Tilsvarende er en generaliserende spesialist mer enn bare en spesialist som har et snevert fokus på et område. Det vil si at en generaliserende spesialist har generell kunnskap og har i tillegg en eller flere spesialiseringer. Dette gjør en generaliserende spesialist mer verdifull for et team fordi en slik person har flere styrker og dermed mindre svakheter.



Figur 2.3. Generaliserende spesialist

Det er flere grunner til hvorfor et agilt team bør bygges av generaliserende spesialister:

- Bedre kommunikasjon og samarbeid

Tilstedeværelse av generaliserende spesialister i et team tillater å oppnå bedre kommunikasjon og samarbeid. Generaliserende spesialister er kjent for å ha en god forståelse av hvordan alt passer sammen. De prøver stadig å øke deres forståelse gjennom verdsettelse av andre teammedlemmer. De er villige til å lytte til og arbeide med sine lagkamerater, fordi de vet at de vil sannsynligvis lære noe nytt.

- Mindre dokumentasjon

Et team med generaliserende spesialister vil sette mindre fokus på dokumenterbart arbeid. En generaliserende spesialist skriver hovedsakelig mindre dokumentasjon fordi det er et større utvalg av tilgjengelige alternativer.

- Forbedret fleksibilitet

Generalisering sammen med spesialisering hindrer beordring av oppgaver til enkeltpersoner, og dermed favoriserer teamets evne til dynamisk fordeling av oppgaver når nye omstendigheter oppstår.

- Mindre risiko

Den samlede risikoen i prosjektet reduseres gjennom dynamisk fordeling av oppgaver, læringsutbytte mellom teammedlemmer, gjennom økt kunnskap blant enkelte gruppemedlemmer og voksende ferdigheter hos teammedlemmene.

2.6.3. TEAMSTØRRELSE

Det har vært mye diskusjoner og debatter angående teamstørrelsen i agile prosjekter. Selv om de fleste agilister er enige om at et lite team er mer funksjonelt og produktivt i forhold til et større team, er det fortsatt en stor utfordring å definere den mest optimale teamstørrelsen.

Agile filosofer anbefaler et team på mellom 4 og 9 personer, men et agilt team kan også ha flere deltakere. Den anbefalte teamstørrelsen har større sjanser for å maksimere produktivitet. Den TSS Java Trends undersøkelsen som ble gjennomført i 2012, har vist at større team blir mer populære. 24 % av dem som deltok i undersøkelsen favoriserte et team på mellom 10 og 20 personer. Sammenlignet med tidligere resultater har antallet ”større team- favoritter” økt med 10 prosent siden 2010 [12].

Det er flere utfordringer med oppbygging og organisering av et stort team. Økende antall teamdeltakere fører til flere ulemper. I et stort team er det lett å miste fokus på arbeidet som utføres og bruke mye tid på unødvendig kommunikasjon. For å dra nytte av et team med et større antall deltakere, bør teamet splittes i flere sub-team. Scott Ambler anbefaler å dele et stort team i mindre grupper når teamstørrelsen er rundt 20 personer eller mer. Hvert sub-team bør ha ansvar for en eller flere deloppgaver, slik at hvert enkelt sub-team kan fungere som en liten agil gruppe som pålegges å levere sitt stykke arbeid på rett tid.

Antall teammedlemmer varierer også fra en agil metode til en annen. Tilsvarende varierer sammensetningen av teamet med metodene. Noen metoder kan kreve et team der deltakerne har bestemte roller, imens andre metoder kan være avhengige av et tverrfaglig team. Ambler antyder at antall medlemmer og teamsammensetningen bør justeres i forhold til prosjektets behov [11]. Selvfølgelig er teamstørrelsen svært avhengig av flere andre faktorer som prosjektets størrelse, kompleksitet, tilgjengelige ressurser, tidsrammer og budsjett.

2.6.4. FERDIGHETER [13]

De såkalte "myke ferdigheter" er en kritisk faktor for et agilt team. Disse ferdighetene omfatter fleksibilitet, tilpasningsevner og evner til å skape og vedlikeholde relasjoner, samt samhandle mellom ulike nivåer og funksjoner i en organisasjon. Disse trekkene er ofte referert til gruppedynamikken, og det er vanskelig å vurdere de overnevnte ferdighetene i et hvert team. I et smidig miljø er verdien av mellommenneskelige ferdigheter kraftig forsterket. Agile prosjekter har en tendens til å forfølge flere stier samtidig, og derfor må et smidig team kunne utvikle et nettverk av alle stier for å fremme suksess. Det er mange som har kompetanse til å delta i et nettverk, men det er svært få mennesker som har en ferdighet for å kunne opprette nettverket. Alle trenger ikke å være nettverksskapere i et smidig prosjekt, men der må være minst en som kan påta seg denne rollen. Det kan være prosjektleder eller teknisk leder, men ideelt sett noen på kjerneteamet.

Dette kan virke innlysende, men behovet for teknisk know-how er også forsterket i det agile miljøet. Derfor legges det stor vekt på bred teknisk kompetanse i et smidig team. For å opprettholde sin evne å reagere raskt på kontinuerlige endringer i teknologisk utvikling, bør smidige team være generelt mindre. Færre personer i teamet betyr at teammedlemmene må kunne bære flere hatter eller med andre ord må de være kompetente på flere ulike områder. Alle relevante kompetanseområder i et prosjekt må være dekket, men det er ikke rom for mye overlapping. For eksempel hvis en person er ansvarlig for et visst funksjonelt bidrag i teamet, så må den personen være i stand til å bære ansvaret på det området. Andre teammedlemmer er tilgjengelig for samarbeid, men det er den ansvarlige personen som må kunne foreta den endelige beslutningen. Med andre ord, er det vanskelig for en nybegynner å spille en hovedrolle i et agilt team. Dette betyr ikke at teamet skal sammensettes av de beste ekspertene i hele organisasjonen. Det kan faktisk være skadelig. Altfor mange eksperter kan skape spenning i teamet, slik at de skal prøve å påta seg en lederrolle. I slike situasjoner vil veldefinerte roller og ansvar være en god taktikk for å redusere dette problemet.

På den annen side er et smidig team ikke nødvendigvis et bra sted for juniorer, men det er et flott sted for personer som har middels ferdigheter. Det smidige teamet er en god øvingsplass for personer som mestrer en spesifikk ferdighet og er klar for å få kompetanse innen andre områder. Blanding av eksperter og personer med middels ferdigheter vil skape en sunn balanse og gi en øvingsbase for å utvide individuelle perspektiver i et agilt team.

Høy usikkerhet i et smidig miljø produserer hyppige endringer i prosjektplanen. Derfor må teammedlemmer være fleksible nok til å håndtere disse endringene på en effektiv måte. Det blir ikke nok med solide tekniske ferdigheter til å finne en ny løsning i det endrede prosjektmiljøet. Personer med tilpassningsevner vil være verdifulle i et miljø som har en tendens til å endre seg ofte. Det tyder på at i et agilt miljø må det være en sammensetning av bred teknisk kompetanse og tilpasningsdyktighet. Kombinasjon av disse ferdighetene tillater teammedlemmer å bidra på en mer meningsfull og entusiastisk måte. I enkelte smidige prosjekter kan tilpasningsdyktighet være så viktig at teammedlemmer med sterk tilpasningsevne kan bli hevet over de som har overlegne tekniske ferdigheter.

I et agilt team legges det like stor vekt på å kunne arbeide selvstendig som å samarbeide med andre. Teammedlemmer må være i stand til å kunne vurdere arbeidssituasjon og bestemme selv hvilken jobbmodus de skal være i. Etter hvert som prosjektet skrider frem vil behovet for samarbeid og selvstendig arbeid endres flere ganger. I noen tilfeller kan teammedlemmene ha parallelle oppgaver og da kreves det å arbeide både selvstendig og sammen med andre. Det smidige miljøet legger stor vekt på kombinasjon av samarbeidsevner og evner til å jobbe selvstendig. Når et agilt team skal bygges opp blir det verdifullt å finne personer som har erfaring med et vellykket prosjekt. Disse personene har jobbet i det smidige miljøet og kan effektivt skifte mellom arbeidsmoduser. De er i stand til å arbeide med minimal veiledning i et ustrukturert miljø og kan levere de resultatene som er nødvendige for å holde prosjektet i bevegelse.

Videre følger en oppsummering av de hovedtrekkene som er nødvendige til å bygge et effektivt og smidig team:

- Myke ferdigheter er en suksessfaktor for et agilt team
- Tilstedeværelse av bred teknisk kompetanse er et ”must”
- Kombinasjon av eksperter og personer med middels ferdigheter
- Personer med en spesiell evne til å skape nettverk er verdifulle
- Det smidige teamet må være i stand til å håndtere endringer i prosjektet
- Agile gruppemedlemmer må ha evner til en kontinuerlig endring av jobbmoduser.
- Tilpasningsevne kan være viktigere enn teknisk kompetanse i enkle smidige team.

2.6.5. TEAMSAMMENSETNING OG ROLLER [11]

Sammensetning av et smidig team varierer fra en metode til en annen og hver av metodene har forskjellige roller. I praksis kan antall roller og teamstrukturen avvike noe i fra anbefalinger som er gitt av agilister. Grunnen til det er et hvert prosjekt er unikt og ulike tilnærminger blir tatt i vurderinger under teambygging. Selv om variasjoner i teamsammensetningen ofte finner plass, må de ikke være vesentlig annerledes slik at det kan settes den smidige adopsjonen i fare.

Noen roller er felles for alle agile metoder, men de har forskjellige navn. Hvert teammedlem kan ha en eller flere roller og enhver rolle kan ha fra null til flere mennesker til enhver tid i prosjektet. De typiske rollene i et agilt team er:

- Teamleder

Har ansvaret for teamet, ressursanskaffelse og beskytter teamet mot problemer. Rollen omfatter de myke ferdighetene, men ikke disse ferdighetene som planlegging, budsjettering eller produktutvikling.

- Teammedlem

Et teammedlem kan være en utvikler eller programmerer og er ansvarlig for utvikling og levering av et system. Dette omfatter modellering, programmering, testing og avlevering.

- Produkteier

En aktiv representant av interessenter som har ansvar for det prioriterte arbeidet og beslutningstaking.

- Interessent

En interessent kan være en direkte eller indirekte bruker, leder av brukere, senior manager, driftsmedarbeider, støttemedarbeider eller en annen person som er interessert i prosjektet. Den overnevnte teamstrukturen er hovedsakelig aktuelt for et lite team. Når prosjektstørrelsen er på rundt 20 personer eller flere, bør teamet deles i mindre sub-team .

Et stort agilt team krever noen ekstra roller:

- Arkitektur eier

Har ansvar for arkitektoniske beslutninger i et sub-team og er delvis ansvarlig for den endelige retningen i prosjektet.

- Integrator

En integrator er ansvarlig for bygging av det endelige systemet og bruker resultatene fra hvert enkelt sub-team.

Et agilt team jobber tett med en støttegruppe som består av:

- Tekniske eksperter

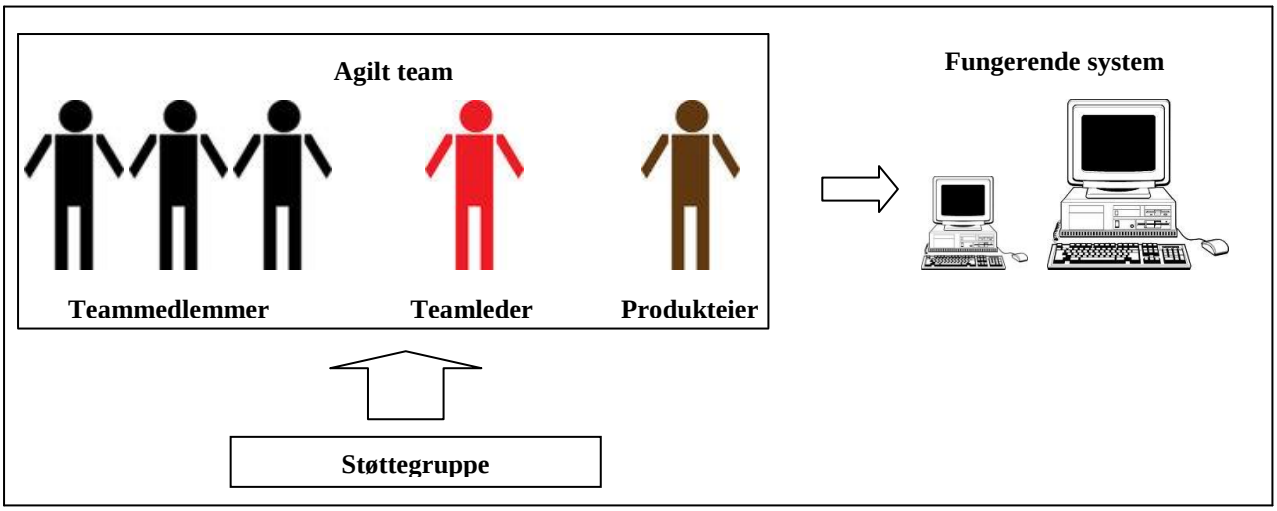
Ekspertene hjelper teamet å overvinne et vanskelig problem, for eksempel de kan hjelpe med design eller testing av database. Tekniske eksperter deler også sine ferdigheter med utviklere i teamet.

- Områdeeksperter

Representerer interessenter og gir en profesjonell hjelp på et område.

- Uavhengige testerne

Et agilt team samarbeider ofte parallelt med et uavhengig team. Et slikt team kreves hovedsakelig i meget komplekse prosjekter. Figur 2.4 illustrerer strukturen av et lite agilt team.



Figur 2.4. Et smidig team

2.6.6. PROSJEKTLEDERENS ROLLE [13]

Det er ingen enkel oppgave å styre et smidig prosjekt og å oppnå et vellykket resultat. Det kreves en spesiell kombinasjon av ulike ferdigheter og egenskaper. Et smidig prosjekt har et behov for en leder som kan bli mer engasjert i de hverdagslige aktiviteter og som er aktivt involvert i å fremme og opprettholde et prosjektmiljø. En agil prosjektleder må organisere teamet, opprettholde en gjensidig kommunikasjon og troverdighet med teamdeltakerne og drive prosjektet fremover. Prosjektlederen er ansvarlig for håndtering av kursendringer og må beholde prosjektets forretningsmål.

I et agilt prosjekt er prosjektlederen nødt til å etablere et godt samarbeid med kunden. Målet med et slikt samarbeid er å opprettholde et kunde-leverandør forhold og fokusere på en konstant kommunikasjon, regelmessige tilbakemeldinger og vinn-vinn løsninger. En agil prosjektleder fokuserer på fremdriften av hele prosessen og sikrer at funksjoner blir gjennomført i sin helhet i stedet for å legge vekt på oppfølging av individuelle oppgaver.

I et smidig prosjekt fungerer prosjektleder mer som en tilrettelegger og mindre som en formann. Prosjektlederen tildeler ikke arbeidsoppgaver, men skaper et arbeidsmiljø der teammedlemmer selv er i stand til å foreta vesentlige avgjørelser i stedet for å handle etter lederens ordre. Med andre ord fokuserer en agil prosjektleder på å motivere og lede fremfor å styre og kontrollere teamet.

Prosjektlederen kan ha en enorm innflytelse på arbeidskulturen ved sine handlinger og meninger. Derfor er det viktig med å etablere og opprettholde en sunn og åpen kommunikasjonskultur i prosjektet. Det må også legges stor vekt på samarbeid i prosjektet, både innad i teamet og mellom teammedlemmer og representanter fra omgivelser. Det må være en atmosfære av gjensidig respekt og en generell åpenhet i flyten av ideer, kunnskap og konstruktive kommentarer.

Behovet for en prosjektleder er ulikt i de smidige prosjektene. Noen agile metoder forutsetter bruken av en agil lederrolle i vesentlig grad, men i noen andre reduseres det behovet for tilstedeværelse av en prosjektleder.

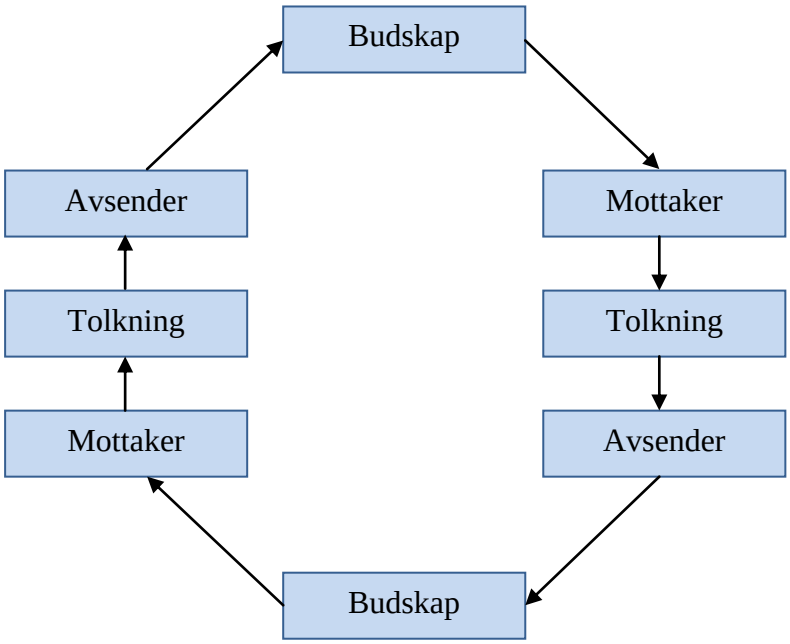
2.7. KOMMUNIKASJON

2.7.1. EFFEKTIV KOMMUNIKASJON

Agile metoder verdsetter en effektiv kommunikasjon som sikrer overføring av rett informasjon til rett person til rett tid. Effektiv kommunikasjon er avgjørende for å lykkes med et smidig prosjekt og kan defineres på følgende måter [4]:

- En utveksling av informasjon
- En handling eller forekomst av informasjonsoverføring
- En muntlig eller skriftlig melding
- En teknikk for å formidle ideer på en effektiv måte
- En prosess der betydninger er utvekslet mellom individer gjennom et felles system av symboler

I et agilt miljø som har en tendens til å ha hyppige endringer er det svært viktig å fokusere på toveiskommunikasjon som er illustrert i figur 2.5. Den type kommunikasjon kjennetegnes ved å ha en aktiv sender og en aktiv mottaker og gir mulighet til å oppklare misforståelser og feiltolkninger på en hurtig måte [14].



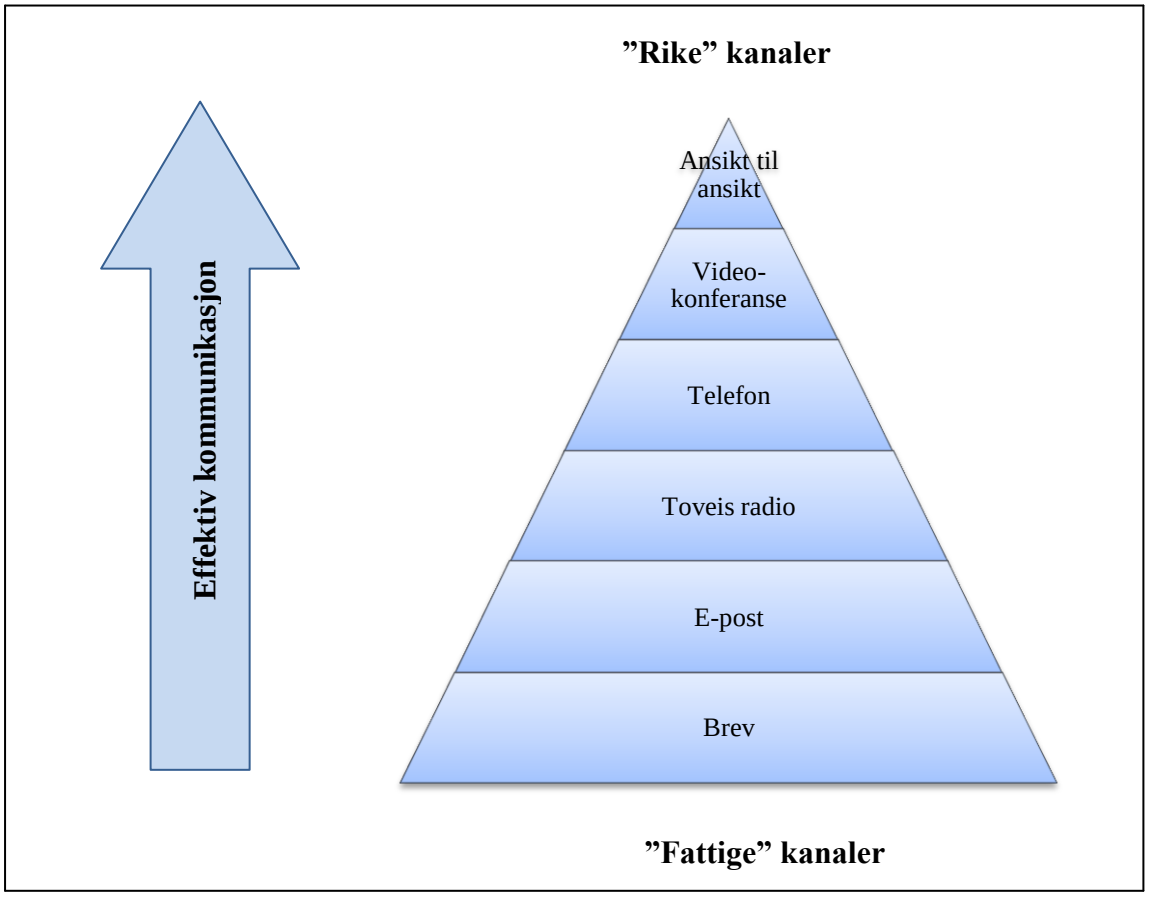
Figur 2.5. Toveiskommunikasjon

2.7.2. ANSIKT-TIL-ANSIKT KOMMUNIKASJON

I følge Cockburn er ansikt til ansikt kommunikasjon en av de mest effektive måter å kommunisere på og den type kommunikasjon som ligger øverst på rikhetsskalaen (Figur 2.6). Den ”rikeste” kanalen tillater å formidle komplekse budskap, dette gir en umiddelbar tilbakemelding og leder til en effektiv kommunikasjon i et smidig miljø [14].

Selv om agile metoder foretrekker ”ansikt-til-ansikt” overføring over enhver annen form av kommunikasjon, betyr det ikke at skriftlig kommunikasjon blir helt erstattet av personlige samtaler. Bruken av den skriftlige måten å kommunisere på reduseres i varierende grad, men beholder sin plass i agile prosjekter.

På tross av at de fleste agilistene favoriserer ansikt-til-ansikt kommunikasjonsform, har formen sine begrensninger og bør utvides med andre kommunikasjonstyper for oppnåelse av den mest effektive kommunikasjon.



Figur 2.6. Rikhetsskalaen av kommunikasjonskanaler.

2.7.3. SUKSESSFAKTORER [4]

Kommunikasjon er mest effektiv når prosjektdeltakere er villige til å jobbe sammen og gjør sitt beste for å skape en åpen og ærlig kommunikasjon. For å oppnå dette er det viktig å stole på informasjonen som mottas. En effektiv kommunikasjon mellom medarbeidere fører til en god læring og kunnskapsoverføring og den viser seg å være en betydelig suksessfaktor i programvare utviklingsprosjekter. Effektive kommunikatorer innser at målet er å dele informasjon og at denne informasjonsdeling er vanligvis en toveis kommunikasjon.

En annen viktig suksessfaktor er en evne til å velge og ta i bruk den mest passende form å kommunisere på. Det trengs også et positivt syn på dokumentasjon som kan komme i ulike former og være enten verdifull eller ikke. Derfor er det viktig å prioritere den informasjonen som har verdi og slik unngå tidsbruk på å bearbeide den unødvendige dokumentasjonen.

Det er også flere andre faktorer som påvirker kommunikasjon:

- Fysisk nærhet

Jo nærmere partene til hverandre desto større er mulighetene for kommunikasjon. I den ene enden av spekteret kan to personer bli lokalisert på samme rom og i den andre enden kan to mennesker befinne seg i forskjellige bygninger.

- Tidsnærhet

Samarbeid i ulike tidssoner kan påvirke kommunikasjonen. Jo mindre avstand det er i tid mellom to personer som jobber i et prosjekt, jo større troverdighet mellom dem og dermed større sjanser for å lykkes med prosjektet.

- Vennlighet

Evnen til å lytte på en annen person med god vilje og snakke uten ondskap er en viktig suksessfaktor. Vennlighet er den tilliten som to kommuniserende personer har med hverandre og det fellesskapet som de deler. Jo større grad av vennlighet desto større mengde av kvalitets informasjon vil bli vekslet og mindre vil være skjult.

I et tett samarbeid finnes det en mulighet for osmotisk kommunikasjon som kan defineres som overføring av indirekte informasjon gjennom å overhøre samtaler eller legge merke til ting som skjer rundt en kommunikator. Den osmotiske kommunikasjonen har sine fordeler og ulemper. Den kan ofte være gunstig hvis informasjonen som utveksles er verdifull eller kan være skadelig hvis informasjonen har form av falske rykter.

2.7.4. PROSJEKTLEDER I KOMMUNIKASJONSPROSESS

I et agilt miljø er prosjektleder ansvarlig for etablering av en åpen og effektiv kommunikasjonskultur i teamet. Kommunikasjonskulturen er svært avhengig av hvordan prosjektlederen engasjerer teammedlemmer og oppfordrer dem til å engasjere hverandre for å skape en god kommunikasjonsprosess. Dette omfatter de grunnleggende reglene i form av vanlig høflighet og profesjonell atferd som en leder bør følge opp [15]:

- Lytt mer til teammedlemmer

En prosjektleder som har gode evner til å lytte er verdifull for prosjektteamet. I en samhandlingsprosess gis det et inntrykk av hvordan kommunikasjonspartene reagerer på mottatt informasjon. Jo sterkere lytteevner en prosjektleder har, jo mer åpne, samarbeidsvillige og tillitsfulle blir teammedlemmer.

- Unngå gjetting på svar

Gjetting på svar er en vane som kan ha ødeleggende effekt på prosjektets funksjonalitet. Nemlig i programvare utviklingsprosjekter er det viktig å unngå beslutningstaking basert på svar som kan predikeres eller forutses. Det hindrer avlevering av en viktig funksjonalitetsdel i ubrukelig tilstand.

- Kommuniser mindre via e-post

Selv i et programvareutviklings miljø er det mest naturlig med en e-post kommunikasjon fordi utviklerne befinner seg foran en PC-skjerm, betraktes den som en meget dårlig kommunikasjonskultur. Prosjektlederen må sørge for tilstedeværelse av ansikt-til-ansikt kommunikasjon som tar mindre tid og garanterer en rask levering av tilbakemeldinger som ikke kan leveres i ASCII-tekst.

- Gi teammedlemmer nok tid

Et utviklingsteam er vant å jobbe hardt og under tidspress, og det kan føles unaturlig for deltakere å få et pusterom. Årsaken til det er at de kan bruke den gitte tiden til fordypning i ny praksis eller for vurdering av sin fremgang og innsats. Prosjektlederen må sørge for å forsyne teammedlemmer med slike pustepauser, ha realistiske forventninger, være tålmodig i perioder med høyt press og stole på at teammedlemmer gjør sitt beste.

2.7.5. DAGLIGE MØTER [15]

Møtefrekvens er proporsjonal med graden av usikkerhet. Vanligvis krever agile prosjekter daglige møter for å utveksle den viktige informasjonen mellom alle teamdeltakere og holde hverandre oppdatert om forandringer i prosjektet. Møtene er hovedsakelig korte og det avsettes begrenset tid til rapportering av resultater. De som deltar i møtene er generelt en del av flere prosjektgrupper, derfor må tiden begrenses til et minimum under møtene.

Et daglig møte har til formål å møtes ansikt-til-ansikt og rapportere om fremdrift i prosjektet. Det er viktig med å få en forståelse og unngå misforståelser som kan dukke opp under rapportering. Derfor er det viktig at rapportering har en visuell innvirkning. Møter kan gjøres mer effektive ved å identifisere obligatoriske og valgfrie deltakere, og la dem vite hva som forventes av dem slik at de kommer forberedt.

Daglige møter har både sine fordeler og ulemper. I det siste har det blitt satt mye fokus på de negative sidene og mange prosjektledere har blitt oppfordret til å kvitte seg med å kjøre daglige møter. Det er flere som mener at daglige møter gir dårlig start på dagen, oppfattes som avstraffelse, mangler empiriske bevis og er problematiske for et distribuert team [14]. Prosjektledere bør endre sin oppfatning om daglige møter slik at møtene kan bli en verdifull og smidig kommunikasjonskanal for prosjektinformasjon. Daglige møter må benyttes hvis det fungerer bra for prosjektteamet, men det er kanskje nyttig å undersøke om andre mekanismer som kan dekke kommunikasjonsbehovet i et smidig prosjekt.

2.8. KUNDEN

I en agil utvikling får kunden en sentral rolle, og hele utviklingsprosessen er orientert mot kundens ønsker og behov. Dette har sammenheng med at kunden er den som har bestilt produktet, og bestemmer således hva som blir produsert. Boehm og Turner beskriver en kunde som en sprek utøver som er samarbeidende, representativ, autorisert, engasjert og kunnskapsrik [16]. Det er viktig at kundens ferdigheter og kvalifikasjoner samsvarer de fleste eller alle av følgende attributter [15]:

- Har kjennskap til virksomheten
- Har forståelse for prosjektets mål
- Har nok tid til å jobbe med prosjektet
- Er respektert av virksomheten, brukerne og interessenter
- Er villig til å ta beslutninger
- Er villig til å inngå kompromisser
- Er ikke en perfeksjonist

Kunden kan ha ulike navn, men har en bestemt rolle i alle agile metoder. Graden av kundeinvolvering kan også variere fra metode til metode, men det kreves at kunden må bli involvert under hele prosjektets livssyklus. Det betyr at kunden skal til enhver tid være tilgjengelig for å utføre bestemte oppgaver [15]:

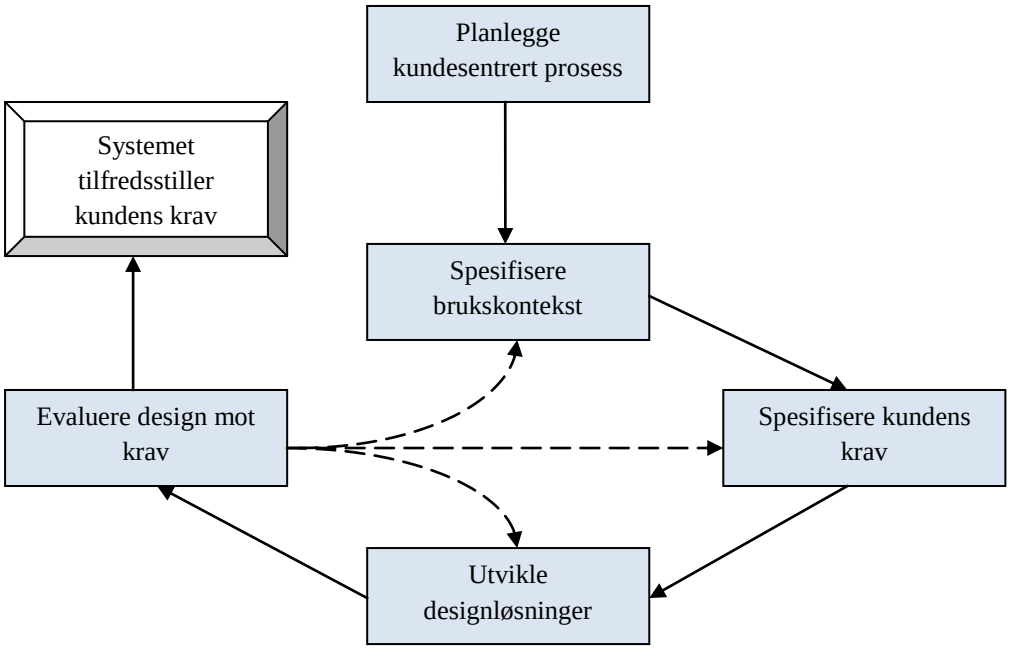
- Representere interessene til brukerne, interessenter og virksomheten
- Bistå i situasjoner der uenigheter oppstår
- Bidra til å drive en iterativ utviklingsprosess

En agil kunde kan være en person eller en gruppe. Hvis kunden er representert med bare en person, da får den personen liten beslutningsmyndighet, men må være mer effektiv for å få ulike aktører til å samarbeide og ta de viktigste avgjørelsene. Hvis det er snakk om en gruppe av personer, da får gruppen mer myndighet, men behøver ikke å ta alle beslutninger uten å konsultere ledelse. Uansett om det er en eller flere personer, må kunden kunne levere helhetlige og handlekraftige svar til enhver tid. Et godt samarbeid mellom kunden og utviklingsteamet kan betydelig øke suksessen i et agilt prosjekt. Kontinuerlig og meningsfull kundeinvolvering øker forståelsen for brukerens behov som fører til redusering av utviklingstiden. Kundedrevet planlegging sikrer valget av den riktige funksjonaliteten som er mest relevant til systemet som utvikles.

Det vil si at i smidige prosjekter arbeider utviklingsteamet bare med de funksjoner som er nødvendig til utviklingen av systemet. Tidsbesparelser basert på de riktige avgjørelsene som tas ved hjelp av kundedeltakelse øker verdien som leveres til brukerne. Kommunikasjon mellom kunden og teamet spiller en betydelig rolle for prosjektets suksess, og siden det er viktig informasjon som skal utveksles, favoriseres ansikt-til-ansikt kommunikasjonsformen mer enn andre former [16].

Tilstedeværelse av kunden i agile programvare utviklingsprosjekter tillater å få raske tilbakemeldinger og dermed kunne tilpasse seg til kundens nye ønsker og behov. Det også fører til mindre defekter i funksjonalitet fordi kunden er totalt involvert i prosjektet og styrer utviklingsprosessen. Kunden gir mulighet til utviklingsteamet å simulere raskere ved å avgi tilbakemeldinger etter hver iterasjon. Med andre ord å ha kunden til stede har stor betydning for å gjøre utviklingsprosess raskere, mer effektiv og mye billigere.

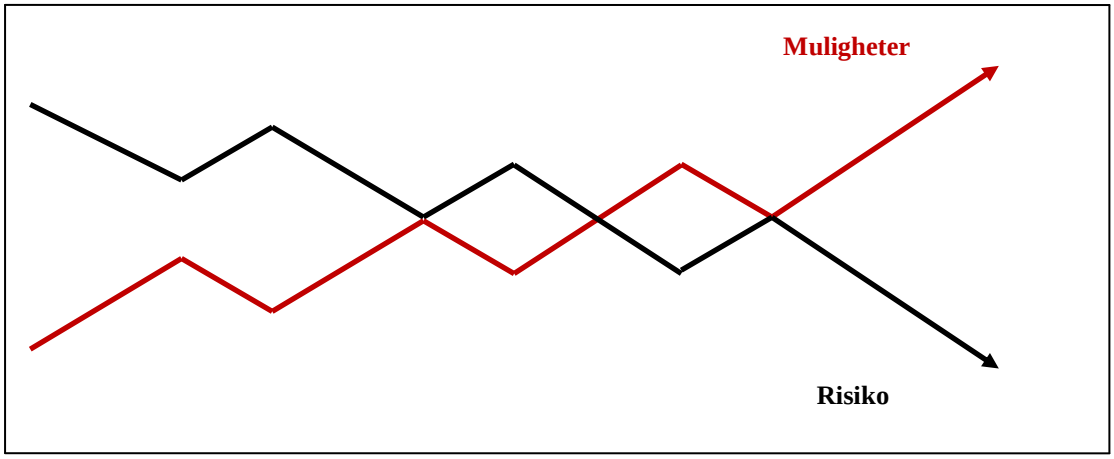
Det legges stor vekt på gjennomføring av kundesentrert utvikling (Figur 2.7) [17]. En slik utvikling oppnås ved å følge en iterativ prosess og evaluere og dokumentere brukbarheten av systemet. Utviklingsteamet må ha en god forståelse for kundens behov, derfor tilstedeværelse av kunden er en stor fordel for å finne ut om hvordan kunden vil benytte systemet. Designet av systemet må tilsvare kundens krav. Levering av prototyper til kunden tillater å evaluere oppfyllelse av kundens krav. Iterasjonsprosessen stopper når kravene er tilfredsstilt.



Figur 2.7. Kundesenteret utviklingsprosess.

2.9. USIKKERHET I AGILE PROSJEKTER

Usikkerhet defineres ofte som differansen mellom den informasjonen som er nødvendig for å ta en sikker beslutning og den informasjonen som er tilgjengelig på tidspunktet for beslutningen [18]. I følge definisjonen kan graden av usikkerhet betraktes som en funksjon av den mest nødvendige informasjonsmengden og tilgangen på informasjon. Mangel på informasjon er ikke eneste årsak til usikkerhet. I følge Andersen kan usikkerhet også anses som mangel på kunnskap og kontroll over et fremtidig forhold. [19]. Siden usikkerhet er knyttet til fremtiden, vil den føre til konsekvenser som kan være i form av enten risiko eller muligheter (Figur 2.8).

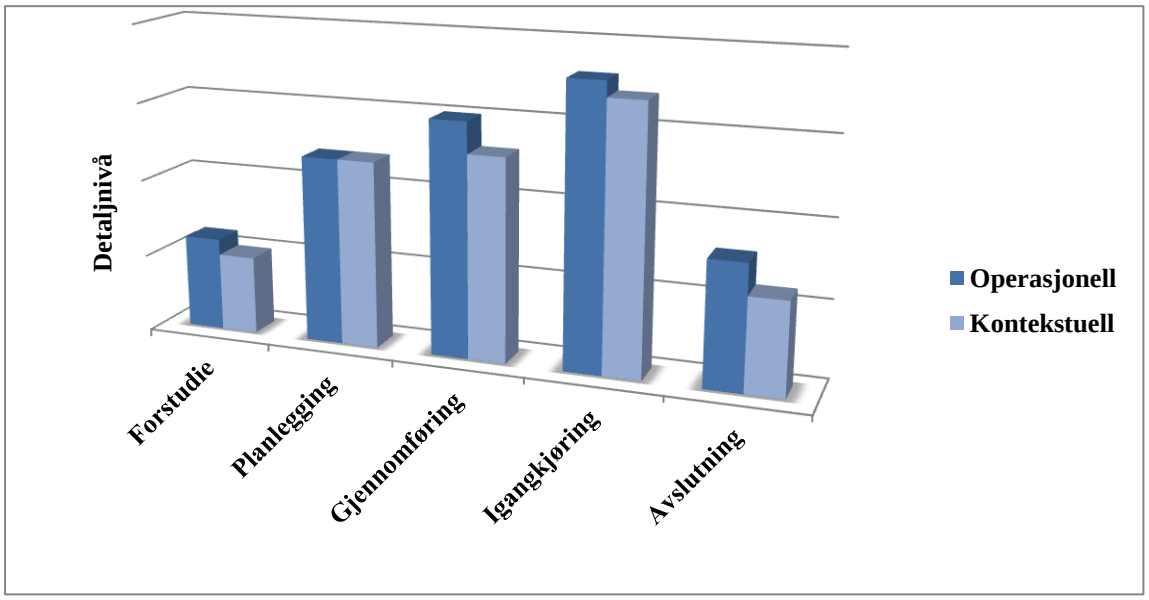


Figur 2.8. Risiko og muligheter.

Årsaker til usikkerhet kan være ulike og de kan anses som et fundament for kategorisering av usikkerhet. Usikkerhetskilder kan være interne eller eksterne. Med andre ord kan usikkerheten skapes i prosjektet eller komme fra omgivelser. Dette utgjør grunnlaget for usikkerhets-fordeling og da er det vanlig å skille mellom kontekstuell og operasjonell usikkerhet [20].

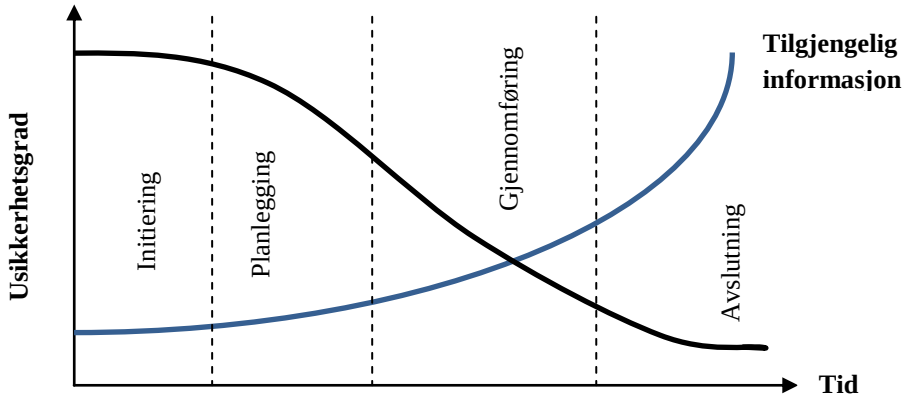
- Kontekstuell usikkerhet - usikkerhet knyttet til prosjektets omgivelser, naturen og prosjektets grunnbetingelser. Disse er helt eller for en stor del utenfor prosjektets kontroll.
- Operasjonell usikkerhet - usikkerhet knyttet til selve gjennomføringen av prosjektet og til de faktorer som prosjektet har en stor del av kontrollen over.

Siden kontekstuell usikkerhet er vanskelig å kontrollere, legges det stor vekt på å styre den operasjonelle usikkerheten som er relatert til interne forhold i prosjektet. Figur 2.9 representerer variasjon av usikkerhetstyper i ulike faser i prosjektets livssyklus.



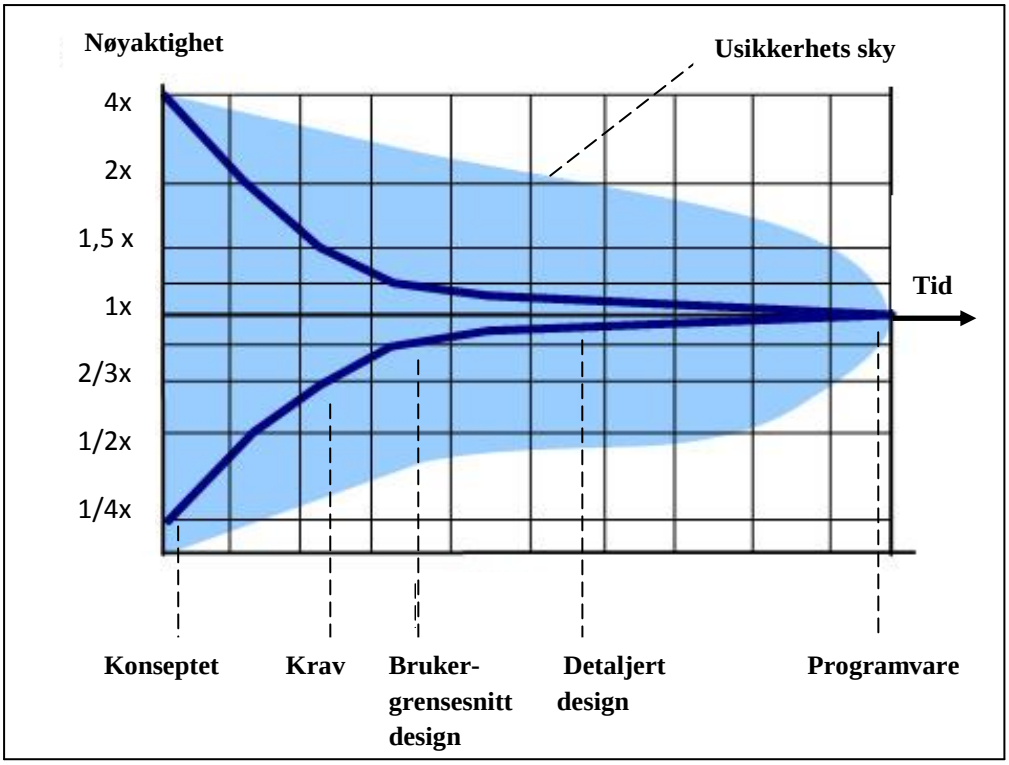
Figur 2.9. Usikkerhetstyper [21].

Usikkerheten har en tendens til å endre seg i løpet av prosjektets livssyklus (Figur 2.10). Tilgangen på informasjon ved prosjektsoppstart er liten og derfor er usikkerheten størst i den perioden. I smidige prosjekter hvor både det tekniske og det forretningsmessige miljøet endrer seg raskt, er løsninger, detaljer om spesifikke krav, prosjektplan og andre variabler uklare ved oppstart av prosjektet. Etter hvert når mer forskning blir gjort og en utviklingsvei blir tatt, avklares det som ikke var tydelig i begynnelsen av prosessen. Etter hver ny iterasjon avtar usikkerheten, og den går mot null når et fullstendig system er blitt produsert.



Figur 2.10. Endring i sikkerhetsbildet [22].

I den agile verden er dette fenomenet kjent som ”Cone of Uncertainty” eller konus av usikkerhet på norsk og er illustrert i figur 2.11. [23]. Begrepet brukes mest i programvareutvikling og beskriver endringer i nøyaktighet av estimering av en programvare. Figuren viser faser i et softwareprosjekt, og viser hvordan nøyaktighet øker etter hvert som prosjektet utvikler seg. Som figuren antyder oppstår det betydelig innsnevring av konusen i løpet av de første 20 - 30 % av den totale prosjekttiden. Det betyr at i denne perioden nøyaktigheten av estimering øker og usikkerheten avtar med lignende prosentandel.



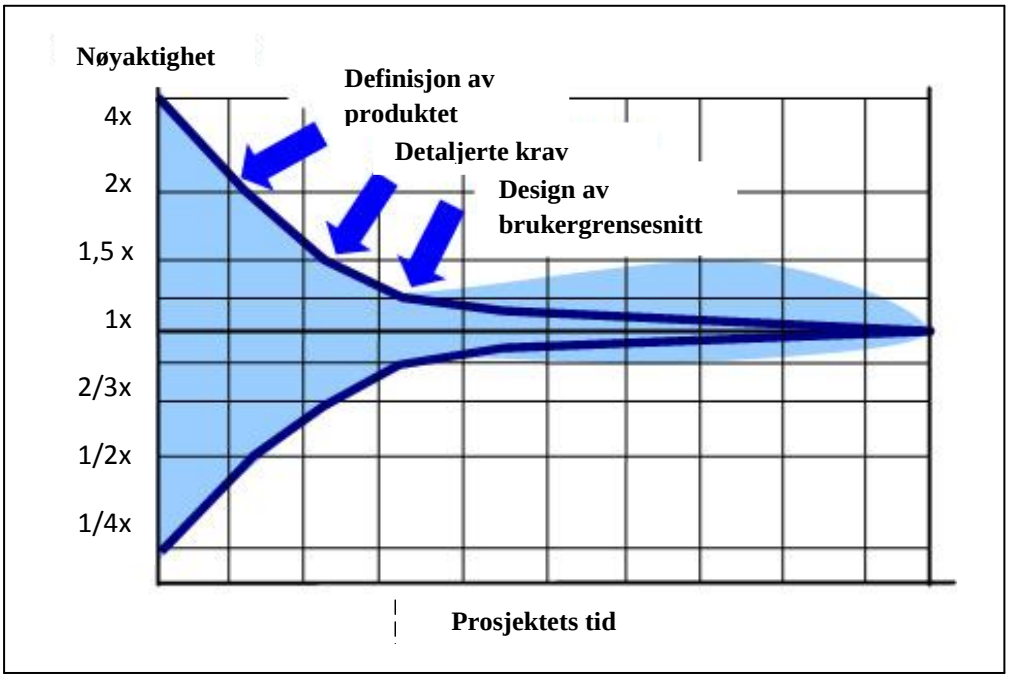
Figur 2.11. Konus av usikkerhet [24].

Den horisontale aksen viser milepæler som beskriver viktige delmål som blir oppnådd på et bestemt tidspunkt i et prosjekt. Den vertikale aksen representerer graden av feil som er funnet i estimater. Estimatene kan være for eksempel et kostestimat for et bestemt funksjonssett eller et estimat på innsats som vil være nødvendig for å levere det funksjonssettet. Estimater som beregnes svært tidlig i prosjektet kan ha en høy feilgrad. De estimatene som blir beregnet i konseptfasen kan være unøyaktige med en faktor på 4x på den høye siden eller 4x på den lave siden. Det totale intervallet mellom det høyeste estimatet og det laveste estimatet utgjør 16x.

Hvis prosjektet ikke blir gjennomført med hensyn til reduksjon av variasjoner i estimatene, forandrer usikkerheten sin form fra en konus til en sky som er vist i figur 2.12.
Årsaken til dette er at selve prosjektet klarer ikke å konvergere, eller med andre ord gir prosjektet ikke støtte til mer nøyaktige estimater.

Det er viktig at det tas hensyn til aktiviteter som hjelper å innsnevre konus av usikkerheten, og den mest effektive måten å redusere variasjonen i estimatene er å redusere variasjoner i selve prosjektet. Konusen blir smalere hvis og bare hvis det tas avgjørelser som eliminerer variasjoner i prosjektet.

Som figur 2.12 illustrerer, det er flere måter å redusere variasjonen i prosjektet på. Definisjon av produktet reduserer variasjonen til en viss grad. Ved å definere kravspesifikasjoner oppnås eliminering av variasjonen ytterligere. Designing av brukergrensesnittet bidrar til reduksjon av risikoen for variasjonen som følge av misforståtte krav. Hvis produktet ikke er definert eller hvis produktets definisjon blir omdefinert senere, vil da konusens grenser være bredere og estimering av nøyaktigheten vil bli dårligere.



Figur 2.12. Innsnevring av usikkerhetskonus [24].

Kompleksitet og usikkerhet betraktes som hovedfaktorer til fiasko for agile prosjekter. De to primære attributtene har en tendens til å påvirke hastighetslevering av forretningsverdien. For å kunne håndtere kompleksitet og usikkerhet, er det viktig å kjenne til deres viktigste drivere. Kompleksiteten i et prosjekt avhenger direkte av prosjektets struktur. I følge Todd Little kan prosjektets kompleksitet bli evaluert på grunnlag av flere faktorer som er representert i tabellen under. Kompleksiteten er gradert mellom 1 og 10, hvor 1 står for den minimale kompleksiteten og 10 beskriver den høyeste graden av kompleksitet [25].

Tabell 2.2. Gradering av kompleksitet.

Grad av kompleksitet					
	1	**3**	**5**	**7**	**10**
Teamstørrelse	1	5	15	40	100
Teamets lokalisering	Samme rom	Samme bygning	Kjøreavstand	Samme tidssone + / - 2 timer	Hele verden
Teamets kapasitet	Etablert team av eksperter	Nytt team av eksperter	Blandet team av eksperter og nybegynnere	Team med begrenset erfaring og noen få eksperter	Nytt team av nybegynnere
Kunnskapsgap	Utviklere kjenner så mye som ekspertbrukere	Utviklere kjenner området godt	Utviklere trenger hjelp	Utviklere må utvide kunnskapen	Utviklere har ingen anelse om området
Avhengigheter	Ingen	Begrenset	Moderat	Signifikant	Tett integrasjon med flere prosjekter

Tabellen ovenfor viser at teamstørrelse har stor betydning for prosjektets kompleksitet. Jo større teamet er, jo mer kompleks blir prosjektet. Lokalisering av teamet har også stor påvirkning på kompleksiteten. Hvis et team eller en vesentlig del av teamet befinner seg i forskjellige tidssoner, kan det føre til økning av prosjektets kompleksitet på grunn av svekkelse av kommunikasjon mellom teammedlemmer. Tilstedeværelse av eksperter i et team bidrar til reduksjon av kunnskapsgapet, og dermed minker det kompleksiteten. Avhengigheter av andre prosjekter vil generelt føre til økning av kompleksiteten.

Økning av usikkerhet i et prosjekt har flere ulike årsaker, men generelt avhenger prosjektusikkerhet av markedsforhold og prosjektbegrensninger. De viktigste faktorene som utgjør grunnlaget for prosjektets usikkerhet er representert i tabell nedenfor [25].

Tabell 2.3. Gradering av usikkerhet.

Grad av usikkerhet					
	1	**3**	**5**	**7**	**10**
Markedsusikkerhet	Markedet er kjent, kontraktsforpliktelse er definert	Små endringer i markedet er forventet	Initial gjetning av markedet	Betydelig usikkerhet i markedet	Nytt, ukjent og utestet marked
Teknisk usikkerhet	Forbedringer av eksisterende arkitektur	En viss sikkerhet	En viss usikkerhet	Krever inkrementell forskning	Ny teknologi og arkitektur, utforskning
Prosjektvarighet	1-4 uker	6 måneder	12 måneder	18 måneder	24 måneder
Avhengigheter, Fleksibilitet av scope	Veldefinerte kontraktsforpliktelser eller infrastruktur med publiserte grensesnitt	Flere grensesnitt, scope er ikke veldig fleksibel	Scope har en viss fleksibilitet	Noen publiserte grensesnitt, scope er svært fleksibel	Ingen publiserte grensesnitt

Ut fra tabell 2.3 kan man se at markedskjennskap spiller en betydelig rolle for prosjektets usikkerhet. Dersom markedsbehovet er ukjent, vil det bli mer krevende å styre prosjektet og usikkerheten blir høyere. Økning i antall kunder på markedet kan også virke forsterkende på usikkerheten. Teknisk usikkerhet kan forklares med bruken av ny teknologi. Det vil si at i utiklingsprosjekter der fokuset er satt på utvikling av nye produkter, brukes det vanligvis den nyeste teknologien, slik at disse prosjektene vil ha en høy grad av usikkerhet. Prosjektvarighet påvirker også usikkerheten. Jo lengre tid det tar fra oppstart til produktutgivelse, desto større sjanser er det for at prosjektet blir svekket av teknisk usikkerhet eller usikkerheten knyttet til markedet. Usikkerheten øker med prosjektets usikkerhet på samme måte som kompleksitet.

DEL 3

VURDERING AV EKSISTERENDE AGILE METODER

3.1. SCRUM

I en agil utvikling betraktes Scrum som den mest populære og utbredte metoden. Den kan beskrives som en smidig tilnærming eller et rammeverk som har til formål å utvikle komplekse informasjonssystemer. Scrum er en adaptiv, rask og selvorganiserende produktutviklings prosess, og selv om metoden er hovedsakelig utarbeidet for programvare-utvikling, kan den godt brukes til å håndtere alle slags prosjekter av en viss kompleksitet [26].

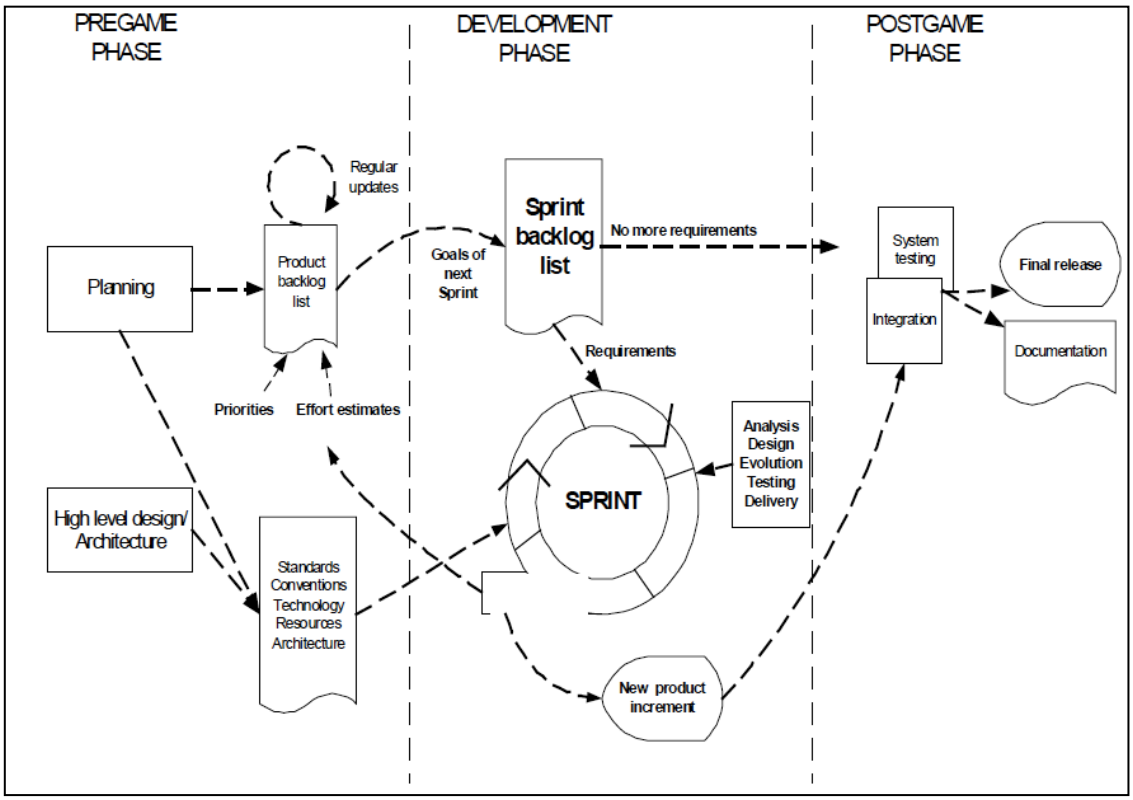
Hovedprinsippet bak Scrum er en empirisk tilnærming som baseres på synlighet, inspeksjon og tilpasning, og som utgjør grunnlaget for utviklingen av komplekse prosesser.
Scrum fokuserer på hvordan teamet bør fungere for å produsere fleksibilitet i et stadig skiftende miljø. Siden systemutvikling involverer flere tekniske variabler som for eksempel krav, tidsramme, ressurser og teknologi, vil disse variablene sannsynligvis endres i løpet av prosessen. Dette gjør utviklingsprosessen uforutsigbar, kompleks og vil kreve fleksibilitet for å kunne svare på endringene. Fleksibiliteten oppnås ved bruk av kontrollmekanismer, som også administrerer uforutsigbarhet og kontroll av risikoen. Dette resulterer til forbedringer av reaksjonsevner, fleksibilitet og pålitelighet.

3.1.1. PROSESS

Scrum betraktes som en adaptiv metode, men gjerne i grenselandet opp mot iterative.

Scrum livssyklus har tre hovedfaser som er en innledningsfase (pregame), utviklingsfase (development) og avslutningsfase (postgame) og er representert i figur 3.1. Den første fasen består av to underfaser, planleggingsfasen og arkitekturfasen. I planleggingsfasen etableres prosjektets overordnede mål, tas risikovurdering og defineres systemet, prosjektteam, verktøy og andre ressurser.

Det opprettes en funksjonsliste av forhåndsbestemte krav som kalles produktkø (Product Backlog) som blir stadig oppdatert. I den andre underfasen designes programvarearkitektur og utformingen av systemet skjer på et høyt nivå.



Figur 3.1. Scrum livssyklus [27].

I gjennomføringsfasen skjer utviklingen av programvare gjennom en serie med tidsbokser (iterasjoner) som heter Sprinter. En Sprint kan vare fra en uke til en måned, vanligvis 30 kalenderdager, og det kan være tre til åtte sprinter i en utviklingsprosess. Enhver Sprint leverer et nytt inkrement i form av en ny funksjonalitet eller en ny versjon av programvaren. De variablene som krav, tid, kvalitet, pris observeres og kontrolleres gjennom ulike Scrum praksiser i Sprinter.

Den tredje fasen eller post-game fasen er en avslutning av utviklingsprosessen. Hvis Product backlog inneholder ikke flere krav eller nye funksjoner som kan bli oppfunnet, så er systemet klar til utgivelse. Avslutning av utviklingsprosessen skjer gjennom testing av systemet og utgivelse av nødvendig dokumentasjon som har en betydning for oppsummering av prosessen.

3.1.2. ROLLER OG ANSVAR

Det er seks klart definerte roller i Scrum som har ulike oppgaver og formål [26]. Selve Scrum teamet har en veldig enkel struktur som innebærer tre roller som er Produkteier, Scrum Master og teammedlem. De tre andre rollene befinner seg i omgivelser og er knyttet til prosjektet. Disse er bruker, kunden og ledelse.

Scrum Master er ansvarlig for gjennomføring av prosjektet i henhold til Scrum regler, verdier og praksiser, og er ansvarlig for produktivitet i teamet og kvalitetssikring. Scrum Master kommuniserer med prosjektgruppen, samt med kunden og ledelsen.

Produkteier definerer utviklingsmål, funksjoner som skal utarbeides og deres prioriteringer. Produkteieren er offisielt ansvarlig for prosjektet, styring, kontroll og synliggjøring av Product backlog listen. Kandidat for denne rollen bestemmes av Scrum Master, kunden og ledelse. Produkteieren er typisk representert av kunden eller brukeren og kan være en del av teamet.

Teammedlemmer er selvorganiserende og har ansvar for utvikling og testing av prosjektets leveranse. Medlemmer har myndighet til å avgjøre nødvendige tiltak, og bestemmer selv hvordan måloppnåelse skal skje i hver sprint. Teammedlemmer er representert vanligvis av utviklere, designere og testere.

Kunden deltar i oppgaver som er relatert til Product backlog elementer i systemet. Kunden er den som har bestilt prosjektet.

Ledelse er ansvarlig for å forsyne prosjektet med ressurser. Den har også ansvar for mål- og kravsetting og endelig beslutningstaking.

Brukeren av programvaren er med andre ord sluttbrukeren som samarbeider med produkteieren om definering av de ønskede funksjonene.

Prosjektleder eksisterer faktisk ikke i Scrum, men rollen opprettholdes utenfor teamet og kan bli holdt av en interessent innen IT for å sikre god styring og godt resultat i teamet. En annen variant er at prosjektlederens ansvar blir fordelt mellom selve teamet og Scrum Master.

3.1.3. TEAM

Siden Scrum metodikken ikke tilbyr en komplett og detaljert beskrivelse av hvordan alt skal gjøres i prosjektet, er mye av dette overlatt til utviklingsteamet. Hensikten med dette er at teamet vil vite best hvordan oppståtte problemer blir løst. Med andre ord er teamet kjernen i Scrum.

Et Scrum team er selvorganiserende og kryss-funksjonelle. Teamet bestemmer selv på hvilken måte de skal organisere seg for å gjøre produktkø (Product backlog list) om til inkrementer av en leverbar funksjonalitet i hver Sprint. Utvikling av inkrementer og selvorganisering krever spesielle ferdigheter. Teamdeltakere må ha kunnskap om programmering, design av brukergrensesnitt, arkitektur, testing, kvalitetskontroll og forretningsanalyse. Hvert teammedlem fungerer som ekspert på sitt område og på den måten oppnår de synergien som forbedrer teamets effektivitet. Sammensetningen av teamet kan endres ved slutten av hver sprint. Dette har sine ulemper i form av produktivitetsreduksjon og må forebygges ved endring av teamet. Et Scrum team inneholder vanligvis en blanding av seniorer, mer erfarne medlemmer og juniorer. I kontrast, bringer Scrum teamet sammen ulike ferdigheter og nivåer.

Ideelt sett består team av sju medlemmer, pluss eller minus to personer. Teamstørrelse kan variere fra tre til ni personer. Hvis det er færre en fem teamdeltakere, blir det mindre interaksjon som kan redusere produktivitet i teamet. Videre kan det oppstå mangel på ferdigheter som kan føre til forsinkelse av leveranse eller fiasko i selve produktutviklingen. Hvis teamet skal bestå av flere enn ni medlemmer, skal det føre til problemer i koordinering som resulterer til økning av kompleksitet.

3.1.4. KUNDEN

I Scrum har kunden en betydelig rolle i utviklingsprosessen. Teamet er kundeorientert gjennom hele prosjektet. Kundeinvolvering skjer fra dag en av og kunden bestiller oppdraget. Selv om kundeinvolvering ikke har retningslinjer i Scrum og kundeoppgaver ikke er beskrevet i detalj, knyttes vanligvis kunden til oppgaver som er relatert til funksjonslisten (produktkø). Kunden knyttes til disse oppgavene, men utfører ikke de direkte. Grunnen til det er at kunden forblir utenfor prosjektet eller med andre ord befinner seg i omgivelser. Kundenes ønsker kan bli representert av produkteier som er direkte knyttet til Scrum-teamet.

Selv om kunden ikke har tilgang til utviklingsarbeidet, må kunden være tilgjengelig for utviklerne gjennom hele utviklingsprosessen. Korte iterasjoner i Scrum tillater å få hyppige tilbakemeldinger fra kunden. Etter hver sprint kommer kunden med tilbakemeldinger som utgjør en basis for neste sprint. Styring og prioritering gjør at kunden får oppfylt sine ønsker og får de viktigste funksjonene løst først.

3.1.5. KOMMUNIKASJON

Effektiv kommunikasjon står sentralt i Scrum. Fokusering på kommunikasjonen har en stor betydning for produktivitet i teamet. Det er mange hindringer for oppnåelse av en effektiv kommunikasjon, herunder kulturelle forskjeller, mellommenneskelige relasjoner og tids- og avstandsforskjeller mellom gruppemedlemmer. I Scrum legges ansvar for den effektive kommunikasjonen på Scrum Master. Scrum Masteren må sikre tilstedeværelsen av kommunikasjonshjelpemidler og sørge for at kommunikasjonen ikke blir forhindret. Scrum Masteren må også sørge for at kommunikasjonslinjene er åpne mellom teammedlemmene og mellom produkteier og teamet.

Kommunikasjonen i Scrum forbedres gjennom forpliktelser knyttet til ”Definition of Done” og bruk av tavler på daglige scrum møter.

I Scrum legges det stor vekt på en direkte dialog mellom teamdeltakerne. En slik dialog oppnås ved at teammedlemmer møtes regelmessig ansikt-til-ansikt. I Scrum blir slike møter kalt seremonier som består av sprintplanmøtet, det daglige scrum-møtet, sprintgjennomgang og sprint-retrospektivet.

I sprintplanleggingsmøtet bestemmer Scrum Master sammen med produkteier et sprint-mål og utarbeider og estimerer Sprint backlog. Møtet holdes for hver nye sprint, består av to deler og varer vanligvis i 8 timer. For et nytt team blir sprintplanleggingsmøtet avgjørende for selvorganiseringen.

De daglige scrum møtene har begrenset tid som ikke kan overstige 15 min. Scrum-teamet samles sammen for å utveksle informasjon om hva som ble oppnådd i de siste 24 timer. Møtet må ikke anses som statusrapportering til Scrum Master. Det er teammedlemmenes forpliktelser ovenfor kolleger. Gjennom disse møtene forbedres kommunikasjonen,

identifiseres og håndteres problemer som hindrer prosessen, tas raske beslutninger og forbedres ferdighetene hos hvert enkelt teammedlem.

På den siste dagen av sprint presenterer Scrum master sammen med teamet sprint-resultater til ledelse, brukere, kunden og produkteier i et uformelt fire timers møte. Sprintgjennomgangen har en stor betydning for det neste kommende sprintplanmøtet.

Etter hver sprint avholdes det et retrospektiv. I denne samlingen tas det en regelmessig vurdering av hva som fungerer og ikke fungerer. Hensiktet med sprint-retrospektivet er å finne ut samspillet mellom sprint og mennesker, relasjoner, prosesser og verktøy.

3.1.6. USIKKERHETSHÅNDTERING

Scrum bidrar til å håndtere forandringer, skiftende krav og usikkerhet. Spesielt i starten av et prosjekt er usikkerheten betydelig stor. Scrum håndterer usikkerheten ved å kjøre korte iterasjoner som tillater å avdekke usikkerheten etter hver iterasjon når løsningen blir mer fullstendig. Det som er virkelig viktig for usikkerhetshåndtering er at forbedringer iverksettes før neste sprint begynner. Med andre ord gir Scrum et godt grunnlag for forvandling av usikkerhet til sikkerhet på kortest mulig måte.

Uformell kommunikasjon spiller en betydelig rolle. Dette blir favorisert over formell dokumentasjon. I daglige møter brukes tavler som har vist seg å være et effektivt verktøy til å formidle informasjonen mellom teammedlemmer i et Scrum prosjekt. Tavlene brukes for å presentere brukerhistorier, daglige oppgaver og aktiviteter. Den visuelle kommunikasjonsformen fører til redusert kompleksitet og usikkerhet som er relatert til misforståelser.

Praksiser som genererer en selvorganiserende struktur virker også reduserende på risiko og usikkerhet. Synlighet i Scrum gjør det mulig å få vite om alle nødvendige tilpassninger som må tas i betraktning før neste iterasjon settes i gang.

Siden risiko og usikkerhet har stor påvirkning på produktsuksess, bør alle risikable og usikre faktorer prioriteres høyt. Hvis for eksempel utviklingsteamet er usikker på noen detaljer i brukergrensesnittdesignet, bør teamet teste og utforske disse elementene gjennom kundens tilbakemeldinger. Raske og hyppige tilbakemeldinger tillater å tilpasse endrede krav på en kjapp måte. Dette akselererer læring, håndterer usikkerhet og reduserer risikoen.

3.2. EXTREME PROGRAMMING (XP)

Metoden defineres som ekstrem og forutsetter at programmering gjennomføres på en spesiell og ikke ordinær måte [28]. I praksis har XP liten likhet med navnet sitt. I motsetning til å være ekstrem, virker den heller som en ganske forsiktig metode. XP har ikke bare programmering til hensikt, den omfatter hele systemutviklingsprosessen og derfor kan XP virkelig kalles ”forsiktig systemutviklingsmetode” [28].

Ekstrem programmering er en av flere populære agile metoder og har allerede visst seg å være svært vellykket i forskjellige bransjer over hele verden. Metoden er vellykket fordi den understreker både kundetilfredshet og teamarbeid. Kunder får levert programvaren når de trenger eller ønsker den i stedet for å få leveransen på et bestemt tidspunkt. I XP prosjekter består et team av likeverdige partnere som er presentert av ledere, utviklere og kunder.
De samarbeider tett i et selvorganiserende team som er svært produktivt på grunn av et enkelt, men effektivt XP-miljø.

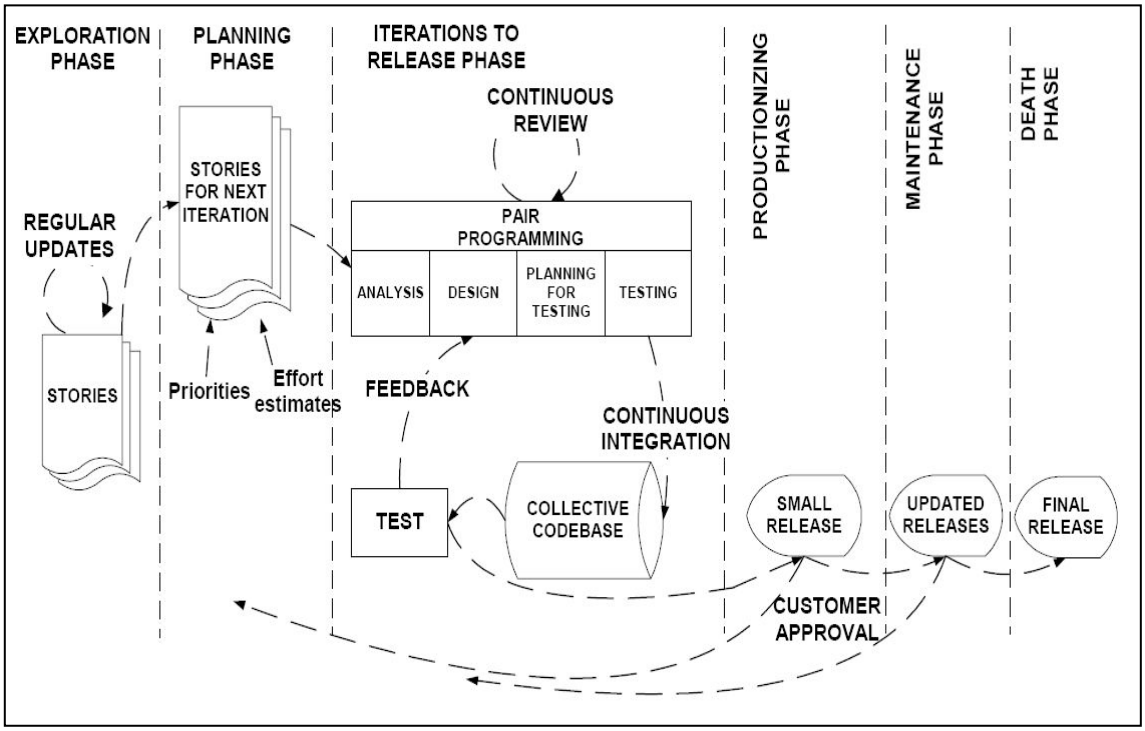
Ekstrem Programmering forbedrer softwareprosjekter på fem grunnleggende måter [29]:

- Enkelhet
- Kommunikasjon
- Tilbakemelding
- Respekt
- Mot

Det settes fokus på å holde designen enkel og ren. Små trinn i utviklingen maksimerer verdiskapninger og gjør prosessen av feilreduksjon mye enklere. Kommunikasjon og tilbakemeldinger er hovedverdiene i XP. Stadig kommunikasjon mellom kunder og programmerere er et must. Raske tilbakemeldinger tillater å gjennomføre endringer og levere systemet til kundene så tidlig som mulig. Teammedlemmers individuelle og unike bidrag er sterkt respektert. De overnevnte kjerneverdiene utgjør grunnlaget for en effektiv håndtering av endrede krav og ligger som basis for enkle XP-regler for planlegging, styring, design, koding og testing.

3.2.1. PROSESS

Livssyklus i XP prosessen består av seks faser: utforsking, planlegging, iterasjoner til frigjøring, produksjon, vedlikehold og avslutning (Figur 3.2).



Figur 3.2. XP livssyklus [30].

I den første fasen utforsker teamet alt om utviklingsprosessen og gjør seg kjent med teknologi og metoder som skal brukes i prosjektet. I den fasen beskriver kunden funksjonaliteten som er ønsket i den første utgivelsen av programvaren. Fasen kan vare fra noen uker til noen måneder avhengig av utviklingsteamets kjennskap til teknologien. Planleggingsfasen omfatter prioritering av de funksjonene som skal utvikles og estimeringsarbeid. Planlegging i den fasen tar vanligvis noen dager. Den neste fasen består av flere iterasjoner som gjennomføres i løpet av noen uker. Kunden bestemmer funksjonalitetsorden for hver iterasjon og tester funksjonene etter hver iterasjon. I produksjonsfasen testes systemet grundig før det blir avlevert til kunden. Vedlikeholdsfasen omfatter brukerstøtte som er nødvendig til å støtte kunden under utviklingsprosessen. Avslutningsfasen starter når kunden er tilfredstilt og har ingen flere ønsker eller krav til systemet. Prosessen kan også avsluttes på grunn av at det blir ikke mulig å oppnå målet som ble planlagt eller på grunn av problemer med finansieringen av utviklingsprosessen. Fasen avsluttes med utarbeidelse av den endelige dokumentasjonen.

3.2.2. ROLLER OG ANSVAR

Det er flere ulike roller i ekstrem programmering [31]. Rollene har forskjellige formål og oppgaver og tilsvarer tradisjonell organisering i IT-prosjekter. Den eneste unntakelsen er at kunden må være en aktiv deltaker i et XP-prosjekt.

Programmerer utarbeider tester, deltar i par-programmering og holder programkoden så enkelt som mulig.

Kunden skriver historier, utarbeider funksjonelle tester og bestemmer prioriteringsorden for kravene.

Tester har ansvar for å hjelpe kunden til å skrive og implementere funksjonstester. Testeren er også ansvarlig for å kjøre disse testene og vedlikeholde testeverktøyet.

Coach koordinerer, løser problemer i teamet og er ansvarlig for prosessen som helhet. Rollen forutsetter et godt kjennskap til XP. Coach overfører kunnskapen om XP fra andre prosjekter og veileder teamdeltakerne om prosessen.

Tracker er ansvarlig for oppfølging av prosjektet. Trackeren vurderer estimater som blir gjort av teamet og gir tilbakemelding om mulige forbedringer av fremtidige estimater. Trackeren vurderer også prosjektets fremgang i forhold til det som var planlagt.

Konsulent er et eksternt medlem som har spesifikk teknisk kunnskap og ansvarlig for veiledning av teamet.

Prosjektleder har en klart definert rolle i et XP-prosjekt. Prosjektlederen sørger for at prosjektdeltakerne utfører sitt arbeid til rett tid. Prosjektlederen tar nødvendige beslutninger, kommuniserer med prosjektteamet angående utviklingsprosess, vurderer situasjonen og finner løsninger til oppståtte problemer. I XP er det ikke behov for en tradisjonell lederrolle, men det er behov for en prosjektleder som kan koordinere, samt virke som en ekstern kommunikator og være en god problemløser.

3.2.3. TEAM

Scrum bidrar til å håndtere forandringer

XP metoden kjennetegnes ved å være veldig teamorientert. XP team er selvorganiserende og tverrfaglig, og teammedlemmer er ansvarlige for sin egen suksess. Dette betyr at teamet selv definerer suksess og lager planer for å oppnå den suksessen uten å involvere ledelse i prosessen. Grunnen til det er at teammedlemmer har den nødvendige kompetansen til å gjennomføre et selvstendig arbeid.

XP er også kjennetegnet ved å ha ”hele teamet”- praksis. En slik praksis innebærer et team som dekker alle ferdigheter som trengs for en suksessfull softwareutvikling. I praksis er et XP team sammensatt av eksperter (kunder), implementeringseksperter (programmerere) og kvalitetseksperter (testere). Teammedlemmer sitter sammen i et åpent landskap. Hele teamet arbeider sammen for å lage sine egne planer og levere et vellykket resultat.

XP metoden er også kjent for sin utviklingsteknikk, hvor to programmerere arbeider sammen ved et tastatur. Den måten å arbeide på kalles par-programmering. En slik programmering innebærer samarbeid mellom en driver og en navigator (observer), der driveren skriver inn koden og navigatoren går gjennom hver kodelinje som blir skrevet. Parene i et XP team er ikke festet og det anbefales at programmerere forsøker å bytte par så ofte som mulig. Dette gir mulighet for å bli bedre kjent med hele utviklingsprosessen.

XP metoden fungerer best i et lite og samlokalisert team. Den beste praksisen viser at XP teamet har stor suksess når det er sammensatt av seks programmerere, fire kunder, en tester og en prosjektleder. XP teamet kan ha færre eller flere enn tolv teamdeltakere, men da kan suksessen ikke bli garantert. Hvis teamet består av bare fem personer, fire programmerere og en produktsjef, kan det føre til overbelastning av teamet. På den andre siden, kan større team pådra seg ekstra kommunikasjon og redusere den individuelle produktiviteten.

Fulltids engasjement er sterkt favorisert i et XP team. Alle teammedlemmer må være lett tilgjengelig i løpet av hele prosjektets livssyklus og må gi full oppmerksomhet til prosjektet. Dette gjelder spesielt for kunder som blir ofte overrasket over engasjementsnivået som forventes av dem.

3.2.4. KUNDEN

I ekstrem programmering er det klare retningslinjer for kundeinvolvering. Kunden må være tilgjengelig gjennom hele utviklingsprosessen eller med andre ord må være til stede. Kunden kan ha en representant eller kan selv bistå i arbeidet. I XP er det viktig at kunden ikke bare skal hjelpe utviklingsteamet, men være en del av det. Grunnen til det er at kundeinvolvering trengs i alle faser i XP prosessen. Det er best å tildele én eller flere kunder til utviklingsteamet.

On-site kunden er ansvarlig for å definere programvaren som skal utvikles og avgjøre hva blir det mest verdifulle for interessenter. Kundenes viktigste aktivitet er planlegging som inneholder evangelisering av prosjektets visjon, identifikasjon av funksjoner og historier, risikostyring og utarbeidelse av en oppnåelig plan.

I utforskningsfasen må kunden utarbeide brukerhistorier eller behovskort (user stories) som vil brukes for utvikling av funksjoner videre i prosessen. Slike kort skrives gjerne sammen med utviklere. Behovskort inneholder ikke alle detaljer og derfor er programmerere sterkt avhengige av kundenes forklaringer. Under planleggingsmøter forhandler kunden et utvalg av brukerhistoriene som skal inngå i hver planlagte utgivelse. Kunden må også være tilgjengelig for å hjelpe til med funksjonell testing. Testdata må opprettes og resultater må beregnes eller verifiseres. Funksjonelle tester verifiserer at systemet er klart til å bli satt i produksjon. Det kan hende at systemet ikke vil passere alle funksjonelle krav. Da må kunden være klar for å gjennomgå test score og enten la systemet fortsette i produksjonen eller stoppe det.

Undersøkelser viser at å ha kunden til stede kan oppfattes av utviklingsteamet som forpliktelse til deres arbeid. På sin side kan kunden oppfatte par-programmering som en ganske støyende aktivitet og det kan også virke forstyrrende på arbeidet som utføres av kunden. I slike situasjoner trenger ikke kunden å sitte i samme rom som utviklerne, men må være tilgjengelig ved behov [32].

3.2.5. KOMMUNIKASJON

En av de kjerneverdiene i XP-programmering er kommunikasjon. Teamdeltakere kommuniserer med hverandre muntlig og det skjer vanligvis ansikt til ansikt.

XP metoden er kjent for programmering som foregår i par. I en slik par-programmering er kommunikasjonslinjene mellom utviklerne åpne. Utviklerne byttes i par ganske ofte og dette gir mulighet til å utveksle informasjonen og kunnskap, slik at alle programmerene får en sjanse til å jobbe med alle andre. På denne måten forbedres kommunikasjonen mellom teamdeltakere.

I begynnelsen av hver iterasjon skjer kommunikasjonen mellom teammedlemmer i planleggingsmøter. Teamet samles for å ta en retrospektiv kikk og planlegge ny iterasjon. Disse møtene tar vanligvis to til fire timer. I en XP-prosjekt, er kommunikasjonsreglene enkle, det vil si at alle kanaler er åpne til enhver tid. Programmerere kommuniserer med kunden og brukerne direkte, dette gjør kommunikasjonen mye lettere og fører til reduksjon av risikoen i prosjektet. Alle interessenter vet hva de kan forvente fra resten av teamet.

Stand-up møter har en viktig rolle i kommunikasjonsprosessen i et XP team. Teamet samles til daglige møter som tar vanligvis fem til ti minutter. Alle teammedlemmer står opp i en sirkel for å unngå lange diskusjoner. Hensikten med disse møtene er å løse problemer, fremme teamfokus og øke kommunikasjonen mellom alle teamdeltakere. Daglige korte stand-up møter med alle teammedlemmer er mer effektive enn lange og sjeldne møter med noen få utviklere til stede.

Kommunikasjon i XP har toveisform og er basert på et system av raske tilbakemeldinger. Kunden spør brukerne om deres ønsker og programmererne forklarer tekniske problemer og stille spørsmål om krav. Datamaskinen varsler programmerere av programfeil og testresultater. Raske tilbakemeldinger gjør kommunikasjonsprosessen mye enklere. Programmerere og kunder får tilbakemeldinger om utviklingen i prosjektet på en kort tid. Programmererne skriver enhetstester og får tilbakemelding hver gang disse testene kjøres. Kundene utarbeider nye historier og får tilbakemelding i form av estimater på disse.

Alle faser i et XP prosjekt krever kommunikasjon med kunden, helst ansikt til ansikt. Å ha en on-site kunde i teamet er en måte å åpne opp kommunikasjonslinjene mellom kunden og utviklingsteamet.

Uten on-site kunden er gapet mellom utviklerne og kunden for stort, og det er usannsynlig at god kommunikasjon noensinne vil oppstå. Selvfølgelig kan ikke tilstedeværelse av kunden garantere suksess for et prosjekt, men i det minste gir det sjansen for suksess. Kommunikasjonen kan kollapse hvis ego kommer i veien eller hvis programmerere blir isolert og tilbaketrukket. Til tross for disse potensielle problemene, tilbyr XP minst en potensiell løsning på kommunikasjonsproblemet som oppstår i mange utviklingsprosjekter.

3.2.6. USIKKERHETSHÅNDTERING

De fleste XP prosjektene er innovative og har stor grad av usikkerhet angående det som skal utvikles. Kunder krever stadige endringer og utviklingsprosessen må tilpasse de nye ønskene og skiftende krav. Det som gjør XP attraktiv er at metoden gir en tilstrekkelig løsning og kan håndtere hyppige endringer i utviklingsprosjekter ved hjelp av korte iterasjoner og kontinuerlig testing og integrasjon.

I XP utvikling er det viktig å ta risikoen i betraktning. En av risikoene som kan oppstå under gjennomføring av et XP prosjekt er at prosjektgruppen blir for stor. Dette kan føre til utvikling av uformelle kommunikasjonslinjer som kan resultere til misforståelser i prosjektet. Uformell kommunikasjon i seg selv er en effektiv og rask måte å kommunisere på og har mange fordeler, men i store grupper kan det føre til kaos og ukontrollerte situasjoner som kan resultere i tap av kommunikasjonstid. Ved å holde et team av anbefalt størrelse på mindre enn tolv personer unngås den type risiko og da kan usikkerheten som er relatert til kommunikasjon bli håndtert på den måten.

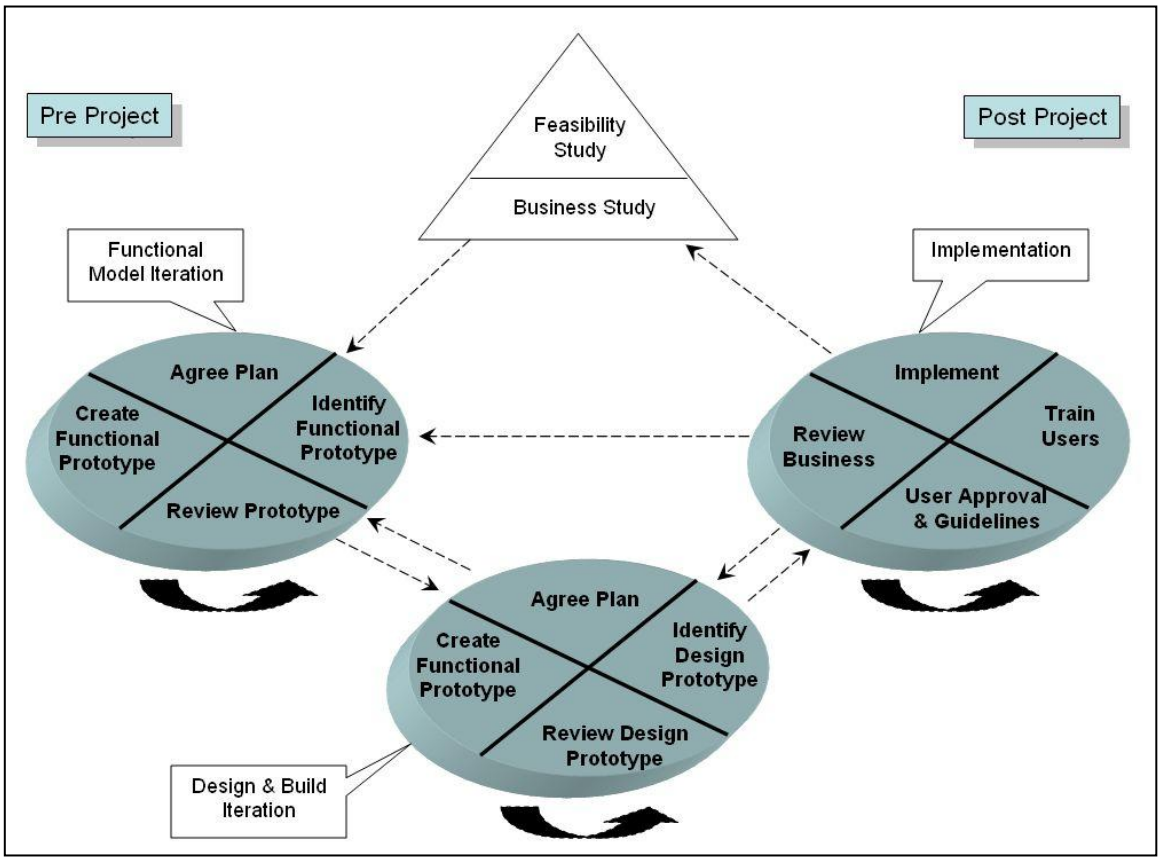
Usikkerheten kan skapes ikke bare på grunn av mangel på den nødvendige informasjonen som trengs for effektiv prosjektgjennomføring, men også på grunn av mangel på kunnskap. XP håndterer denne type usikkerhet ved å praktisere par-programmering, der de mindre erfarne programmererne kobles til personer med en bred og lang erfaring. De erfarne utviklerne blir veiledere for uerfarne teammedlemmer og på den måten overfører de sin kunnskap til de utviklene som er preget av mangel på kunnskap.

3.3. DSDM

Dynamic Systems Development Method eller DSDM er et rammeverk som har fokus på å levere forretningsløsninger raskt og effektivt. Den grunnleggende ideen bak DSDM er at i stedet for å fastsette produktets funksjonalitet i første omgang, fikses det først tid og ressurser og deretter justeres funksjonalitetsmengden.

3.3.1. PROSESS

DSDM prosess består av tre sekvensielle faser: forprosjekt, prosjekt livssyklus og post-prosjektfase (Figur 3.3) [33]. Prosjektfasen er den mest omfattende av de tre fasene og består av 5 stadier som danner en iterativ tilnærming til systemutvikling.



Figur 3.3. DSDM utviklingsprosess.

Forprosjektfasen begynner før prosjektets offisielle oppstart. I denne fasen tas avgjørelser som er nødvendige for å starte prosjektet.

Prosjektets livssyklus er den neste fasen i DSDM utviklingsprosess. Den fasen består av fem stadier som prosjektet må gå gjennom for å lage et system. De første to stadiene, Mulighetsstudie (Feasibility Study) og Forretningsstudie (Business Study) er sekvensielle faser som utfyller hverandre. I mulighetsstudien utforskes alle muligheter angående prosjektet. Med andre ord vurderer DSDM teamet mulige fremgangsmåter, risiko, tidsrammer og økonomiske aspekter. Mulighetsstudien varer ikke mer enn noen uker. Neste fase studerer de forretningsmessige sidene av prosjektet. Den fasen er like kort som mulighetsstudien. Videre utvikles systemet iterativt og inkrementelt og det skjer i de neste tre fasene. Løsningsstudien (Exploration) består av fire iterasjoner som legger vekt på identifisering, planlegging, produksjon og kontroll, og fokuserer på å lage det riktige systemet. I utviklingsfasen (Engineering) blir det ferdigtestede systemet produsert. Implementeringsfasen (Implementation) omfatter avlevering av systemet, opplæring av systembrukere og utarbeidelse av avsluttende dokumentasjon.

Post-prosjektfasen handler om vedlikehold, forbedringer og feilrettinger i henhold til DSDM prinsipper.

3.3.2. ROLLER OG ANSVAR

Metoden definerer flere sentrale roller som inngår i et DSDM team [33]:

Ambassadør-Bruker fungerer som en mellommann mellom kunden, brukere og utviklingsteamet.
Visjonær holder prosjektets kurs mot forretningsmål.
Rådgiver-Bruker: har kunnskap om virksomhetsområder som trenger å bli automatisert.
Det finnes også andre roller som er definert av DSDM:

Teknisk Koordinator: koordinerer de ulike tekniske aspekter av systemet og sikrer at de samhandler jevnt og riktig.
Sponsor bidrar med midler og ressurser.

Prosjektleder overvåker utvikling og prototyping.

Teamleder sørger for at teamet fungerer godt sammen.
Senior utvikler forvandler krav til kode som skal leveres.
Fasilitator har fokus på gruppeoppbygging, konflikthåndtering og prosesser.
Scribe er ansvarlig for å dokumentere beslutninger, diskusjoner og planer laget av teamet.

3.3.3. TEAM

DSDM metoden er kjennetegnet ved å ha et selvdrivende og fokusert team. DSDM teamet får myndighet til å fatte vedtak. Det settes grenser på hvor langt teamet kan gå i sine avgjørelser. Vanligvis kan teamet gjøre små avgjørelser som er for eksempel avklaring på kravbetydning i praksis, avklaring på hvorvidt midlertidige produkter er akseptabelt når det gjelder funksjonalitet og brukervennlighet, prioritering av krav og endring av detaljer i den tekniske løsningen.

DSDM metoden fungerer best i små og fokuserte team der roller og ansvar er klart definerte. Dette hjelper å avklare hvem som skal ta beslutninger og dermed unngås misforståelser og konfliktsituasjoner. DSDM teamet har vanligvis mellom fire og seks medlemmer og alle teammedlemmene blir tildelt beslutningsmyndighet. Det er viktig at ledelsen ikke blir involvert i tidkrevende beslutningsprosesser. Brukerne som er ganske godt kjent med problemet og er direkte involvert i teamarbeidet tar på seg ansvar for beslutninger. DSDM metoden lar utviklere og brukere jobbe tett sammen for å skape dynamikk i teamet, og det anbefales at de overnevnte teammedlemmene arbeider i par.

DSDM legger stor vekt på riktig sammensetning av teamet. Prosjektdeltakere har vanligvis forskjellig erfaring og ulike egenskaper. Derfor er det svært viktig å balansere ulikt fordelt kompetanse og ferdigheter i teamet. Teamet bør inneholde alle de nødvendige kjerneferdighetene, og det forventes at utviklere er multi-dyktige og har kunnskap om design, programmering og testing. Utviklerne må også være gode lagspillere som fokuserer på teamarbeid og ikke bare på teknologiske problemer.

De ideelle teammedlemmene er aktive, ansvarlige, erfarne, utdannede, de er også rike på nye ideer og har myndighet. Alle deltakere må kunne styre sitt arbeid på en god og effektiv måte.

Hvert teammedlem må også kunne arbeide på en kooperativ og samarbeidsvillig måte med alle andre medlemmer av teamet.

Disse egenskaper er veldig ønskelige, men er ikke et krav for et godt fungerende DSDM team. Grunnen til det er at metoden tillater å overføre kunnskap og lære nytt om prosessen gjennom de korte iterasjonene. Det fører til økning av teamets effektivitet i DSDM utviklingsprosessen.

3.3.4. KUNDEN

Kundeengasjement er den viktigste nøkkelen for å lykkes med DSDM i et prosjekt. I den metoden er utviklingen drevet av kundens tilbakemeldinger som utgjør et ”must” for oppnåelse av en effektiv løsning. I DSDM er kundene proaktive og deler sin arbeidsplass med utviklerne, som gjør det lettere å ta nøyaktige beslutninger. DSDM garanterer kvalitet ved å få kundene involvert tidlig i prosessen og holde dem involvert gjennom hele livssyklusen.

Aktiv kundeinvolvering i prosjektet fører til økning av motivasjon hos kundene. De vil ta et sterkere eierskap i løsningen gjennom at de hele tiden er med på å influere det endelige resultatet. Noen av brukerne må være tilstede under hele utviklingen for å sikre de nøyaktige tilbakemeldingene. Omfattende kundeengasjement kreves særlig når mulighetsstudiene repeterer seg. Andre fasene i DSDM livssyklus trenger også et betydelig samarbeid med kunden og det vil si at aktiv kundeengasjement er avgjørende i gjennom hele utviklingsprosessen.

Kundene som er involvert i utviklingen representerer brukerens interesser og har nok kunnskap og kompetanse som trengs til å gi systemet en riktig retning. Hvis kundemedvirkning skjer gjennom prototyping, må brukerne kunne verifisere at programvaren produserer de riktige løsninger uten å ha utdypning i de tekniske detaljene. Hvis brukeren ikke deltar i utviklingsprosjektet, må utviklerne gjøre antagelser om hva som er nødvendig for en effektiv løsning. Dette har sine ulemper i form av utsettelse av tidsrammer og økonomiske tap. Derfor har aktiv kundemedvirkning en avgjørende betydning for DSDM prosessen, slik at en kontinuerlig kundekontakt sikrer oppnåelse av de viktigste målene som har virkelig verdi for virksomheten. DSDM appellerer til kundene fordi den metoden ble skapt og utviklet av kundene.

3.3.5. KOMMUNIKASJON

Kommunikasjon og samarbeid mellom alle deltakerne er nødvendig for å oppnå et vellykket prosjektresultat. I DSDM er kommunikasjon en av de kjerneverdiene. For å oppnå en effektiv kommunikasjon mellom alle prosjektmedlemmer er det viktig at samlokalisering av teammedlemmer finner sted. Samtidig er det viktig at hvert enkelt teammedlem ikke har for mange mennesker å forholde seg til. DSDM favoriserer ansikt til ansikt kommunikasjon som skjer gjennom korte daglige møter og tett samarbeid mellom alle interessenter i prosjektet.

DSDM akselererer utviklingsprosessen gjennom forkortelse av kommunikasjonslinjene mellom brukere og IT ansatte i prosjektet, mellom analytikere og designere, mellom gruppemedlemmene og mellom alle interessentene i prosjektet. Det finnes ulike måter å oppnå dette på og mekanismer som forkorter kommunikasjonslinjer varierer fra en kanal til en annen, men hovedmålsettingen er å sørge for at barrierene mellom brukerne og utviklere ikke oppstår. En typisk hindring som ofte eksisterer er den språklige barrieren i dokumentasjoner som utarbeides for brukerne. Ved å ha brukerne i teamet, kan behovet for dokumenter elimineres. Det er mye lettere å få forståelse for prosessen gjennom å stille spørsmål eller på en visuell måte enn å lese et dokument som er skrevet på et faglig språk. Derfor vil samlokalisering av utviklerne og brukerne betydelig forkorte kommunikasjonslinjer mellom forretnings- og utviklingsgruppene i et DSDM prosjekt.

En annen hindring er at store prosjektteam bruker vanligvis formelle kommunikasjonsmetoder for å sikre at alle prosjektdeltakere er klar over hva som skjer. Produksjonen av dokumentene kan være tidkrevende og vil derfor forsinke overføring av informasjon fra avsender til mottaker. I DSDM foregår utviklingen i små team der teammedlemmer bruker uformelle kommunikasjonsmetoder som er mye raskere og mer effektiv. Et lite team vil vanligvis være mer kreativt og dermed vil teamets effektivitet bli økt. DSDM sikrer at de viktigste IT rollene er til stede gjennom hele prosjektet. Den spesifikke DSDM rolle Ambassadør Bruker anerkjenner betydningen av kommunikasjon mellom utviklingsteamet og brukerne i resten av organisasjonen.

3.3.6. USIKKERHETSHÅNDTERING

DSDM metoden har flere fordeler og de tyder på at DSDM prosjekter har en god sjanse for å lykkes, eller med andre ord unngår metoden usikkerhet som kan oppstå ved prosjektgjennomføring. Hvis høy usikkerhet oppstår i begynnelsen av et smidig prosjekt og der er mye som tyder på at prosjektet kan feile, må bruken av DSDM metoden bli vurdert. DSDM metoden bruker tidsbokser, workshops, inkrementell og iterativ levering som hindrer eller reduserer utviklingen av usikkerhet i systemutviklingsprosjekter.

Tidsbokser blir brukt i DSDM som en form for risikostyring. Planlegging av hver iterasjon foregår i en tidsboks med fast lengde som tillater å identifisere oppgave som lett kan strekke seg ut over tidsfristen. Teamet oppfordres til å levere et best mulig fungerende system innen tids- og ressursrammer. Funksjonaliteten som skal leveres justeres vanligvis til å passe innenfor den oppgitte tidsboksen. Tidsavgrensninger er ofte en primær driver i planlegging og bør ikke endres uten å vurdere prosjektets kritiske faktorer. Risikofaktorer for tapte tidsfrister kan være planleggings- eller gjennomføringsfeil i prosjektet, eller problem relatert teamarbeid. Bruken av tidsbokser i DSDM metoden gjør det mulig å utelate krav med lavere prioritet og dermed levere prioritert funksjonalitet til avtalt tid og kost.

For å redusere risikoen som er relatert til tidstap i DSDM prosjekter blir også følgende tiltak ofte vurdert:

- Omfangsreduksjon
- Tidsbegrensning
- Kostnadsøkning

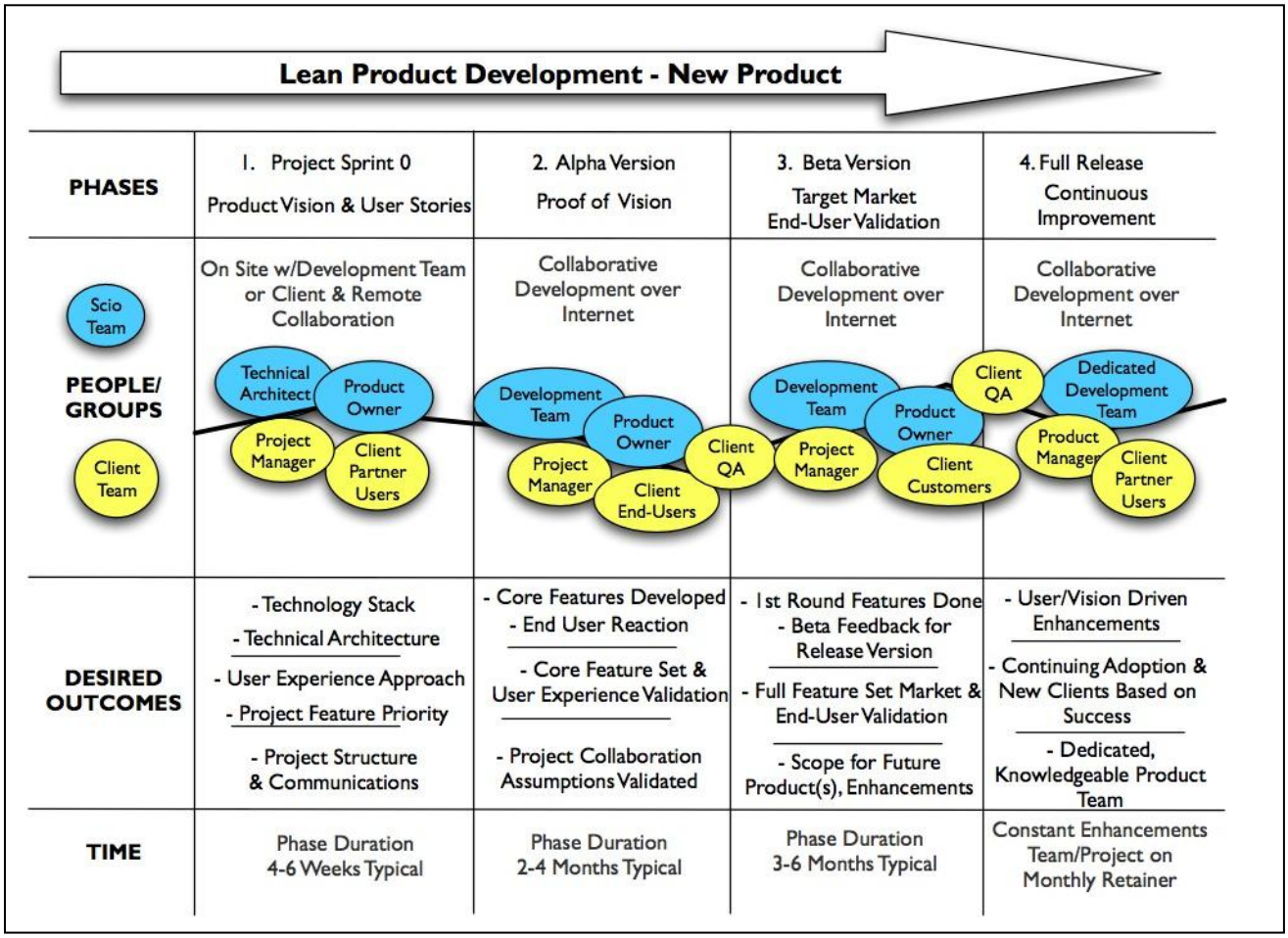
En typisk planleggingsfeil er en mangelfull arbeidsoppgave nedbrytning, noe som kan føre til undervurdering av den tiden som kreves for å utføre arbeidet. Teamrelaterte problemer kan omfatte problemer med intern kommunikasjon, mangel på erfaring, motivasjon eller nødvendig kryssfunksjonalitet.

3.4. LEAN SOFTWARE DEVELOPMENT

Lean Software Development (LSD) er anvendelse av Lean prinsipper til programvareutvikling. De prinsippene fokuserer på eliminering av faktorer som fører til sløsing av tid og ressurser, forbedring av læring og styrking av prosjektteamet. Lean favoriserer sene avgjørelser og raske leveringer. Basert på disse kjerneverdiene øker metoden effektiviteten og avgir mest mulig forretningsverdi.

3.4.1. PROSESS

Lean utvikling går gjennom fire faser, Sprint 0, Alfa Versjon, Beta Versjon og Frigjøring og er presentert i figur 3.4 [34].



Figur 3.4. Lean utviklingsprosess.

Den første fasen handler om kravkontroll, valg av teknologi og utarbeidelse av brukertilfeller. Utgivelse av denne fasen er et sett av tekniske spesifikasjoner og prioriterte brukertilfellene med estimater. Den andre fasen omfatter utvikling av det underliggende rammeverket og kjernefunksjonaliteten. I denne fasen verifiseres produktets visjon av sluttbrukere. Beta versjon utgir den første markedsversjonen av et produkt. Utgivelsen av den siste fasen er et produkt som er bygget på sluttbrukerens tilbakemeldinger.

3.4.2. ROLLER OG ANSVAR

Lean tilnærmingen definerer klare roller og ansvarsområder. Antall roller kan variere fra team til team, og teamet kan bli utvidet med ekstra roller fra omgivelser. Kjerneteamet i Lean har bestemte roller og disse er [35]:

Team Fasilitator fungerer som lærer eller veileder. Den personen oppfordrer teammedlemmer, gir forslag, retning og råd, men ikke gir løsninger.

Team Sponsor samarbeider tett om utvikling av prosjekt charter med Teamleder og Team Mester. Sponsoren tilbyr støtte, bistår teamet og oppmuntrer det til kreativ tenkning. Team Sponsoren er pålagt å dele Lean prinsipper, praksis og resultater med andre teammedlemmer for kontinuerlig forbedring og effektivisering.

Team Mester (Champion) bistår Teamleder i kommunikasjon mellom alle interesserte parter, deltar i framdriftsmøter og støtter teamet i utarbeidelse av kvartalsrapporter.

Teamleder styrer teamet, sørger for at teamet fungerer bra sammen, støtter teammedlemmer i å finne løsninger, hjelper teamet å bruke riktige Lean og Kaizen verktøy og sørger for måloppnåelse.

Ad Hoc fagekspert deltar på Lean hendelsen, gir innspill til nåværende og fremtidige prosesskart ved behov og gir tilbakemeldinger til teamet.

Teammedlem har en "kan gjøre" holdning og tar ansvar for å skape endringer, stiller spørsmål og kommer med forslag, lærer om Lean verktøy og teknikker, er åpen for nye ideer og arbeider sammen med andre medlemmer som et team.

3.4.3. TEAM

Et Lean team består vanligvis av ikke mindre enn fem, men ikke mer enn ni medlemmer. I tillegg til en teamleder anbefales det et team av syv medlemmer. To medlemmer som har kompetanse nær fokusområdet, to medlemmer som ikke kjenner hele prosessen, men er tettrelatert til prosessen, to medlemmer som er helt utenfor prosessen og ett medlem fra omgivelser, for eksempel kunde eller leverandør. Lean fokuserer på et tverrfaglig team som har alle nødvendige ferdigheter for en vellykket systemutvikling og inkluderer både kunden og IT spesialister.

Lean Software Development fokuserer på et selvorganiserende team som har myndighet til å velge det arbeidet som skal utføres. Det er viktig at hvert enkelt medlem får myndighet for å velge sitt eget arbeid. Teamet bestemmer hvordan arbeidet skal gjennomføres og velger den generelle strategien. Konkrete detaljer og relaterte arbeidsoppgaver planlegges just-in-time. Den langsiktige planlegging utføres av teamet i begynnelsen av prosjektet og deretter fullføres sammen med prosjektleder i løpet av prosjektets livssyklus. Lean teamet koordinerer sin innsats gjennom regelmessige møter. Selv om teamet påtar seg ansvar, tas de endelige avgjørelsene av prosjektlederen som samtidig ikke bør begrense teamet så mye og bør gi størst mulig myndighet som mulig.

Selvmotivasjon er en av de viktigste verdiene i et Lean team, men prosjektlederens medvirkning i motivering av teamet spiller fortsatt en stor rolle. Tilgang til kunden fungerer som indirekte årsak til motivasjon av teammedlemmer, slik at kundene hjelper å forstå bedre hensikten bak prosjektet. Teamet kan ikke bli motivert av direkte ordre om konkrete arbeidsoppgaver, derfor bør prosjektlederen sørge for ikke gå over sine grenser i tips og råd. Skeptikere bør heller ikke møte teamet. Det som er negativt med det er at negativt innstilte personer kan demotivere utviklerne.

Selvmotivasjon og selvorganisering i et Lean team fører til et bedre beslutningsgrunnlag, forbedring av kvaliteten av sluttprodukter og gir færre muligheter for feil. Teammedlemmer får større mulighet til å forbedre sin kompetanse og ferdigheter. Lean teammedlemmer har kontroll over sitt eget arbeid og det betyr at motivasjonen er høyere, noe som også resulterer i økt produktivitet. Lean teamet baseres på tillit og samarbeid og holder fokus på en evig søken etter sløsing og det å fjerne sløsing.

3.4.4. KUNDEN

De grunnleggende kjerneprinsippene i Lean Software Development handler om verdiskapning for kunden. Kunden står i fokus gjennom hele Lean livssyklus og prosessen tar utgangspunktet i kundens behov.

Kundeengasjement skjer gjennom hele utviklingsprosessen. Kunden involveres ved begynnelsen av prosjektet, slik at allerede i den første fasen oppstår det et stort behov for kundens uttalelser om deres ønsker og krav. Figuren 3.4. illustrerer kundens innvirkning i alle faser i Lean livssyklus. I den andre fasen som har fokus på utvikling av kjernefunksjonaliteten, endres behov for definisjon av kundens ønsker til deres tilbakemeldinger. Den tredje fasen i Lean utviklingsprosess er avhengig av kundens tilbakemeldinger om hele funksjonssettet og deres aksept for utgivelse av systemet. Kundeinvolvering i den siste fasen har til hensikt å fortsette med en kontinuerlig forbedring av det nyutviklede systemet.

Kunden spiller en betydelig rolle i et Lean team som er sammensatt av medlemmer med forskjellige ferdigheter, det vil si et tverrfaglig team. Kunden samarbeider tett med utviklerne og deltar på regelmessige møter. Et slikt tett samarbeid med kunden minsker behovet for omfattende dokumentasjon.

Kundeinvolvering tillater å få kontinuerlige tilbakemeldinger og derfor er samarbeid med kunden et krav i Lean systemutviklingen. Mangel på et nært samarbeid med kunden reduserer drastisk effektiviteten av Lean Software Development. Lean utviklingstilnærming skaffer et ideelt miljø for samarbeid mellom utviklingsteamet og kunden og oppfordrer kunden til å medvirke i et fellesskap gjennom definisjon av deres behov, hyppige tilbakemeldinger og daglige møter. Lean sørger for at forretnings- og IT enheter går tett sammen gjennom hele utviklingsprosessen.

3.4.5. KOMMUNIKASJON

Lean setter stort fokus på mennesker og kommunikasjon. Hovedprinsippet handler om en effektiv kommunikasjon mellom alle interessentgrupper. Hvis utviklerne kommuniserer effektivt, da er det mest sannsynlig at det skal levere et produkt av best mulig kvalitet og forretningsverdi.

Lean handler om å eliminere alt mulig avfall som fører til sløsing av tid og penger. Dårlig kommunikasjon er en av grunnene til bortkastet tid. Derfor er en effektiv kommunikasjon en kritisk suksessfaktor i Lean.

For å oppnå en god kommunikasjon må utviklere, programmerere, systemanalytikere og kunder/brukere samarbeide gjennom hele prosessen. Hyppige tilbakemeldinger fra kunder/brukere om krav, programprototyper, design spesifikasjoner er avgjørende for en god toveis kommunikasjonsprosess.

I Lean brukes det alle typer av kommunikasjon. Muntlig kommunikasjon er en foretrukket form, nemlig i uformelle situasjoner. Ved å kommunisere ansikt-til-ansikt skapes det tillit som fører til rask eliminering av avfall. Skriftlig kommunikasjon er den formelle formen å dele informasjon på. Den visuelle kommunikasjonen er nærmere knyttet til Lean prinsippene og skaper en plattform for enhetlig forståelse.

For å oppnå den mest effektive kommunikasjonen legger Lean-tilnærmingen vekt på følgende faktorer:

- Etablering av kommunikasjonskanaler
- Bygging av et bredt nettverk
- Fokus på effektivisering av møter (møter bør ikke være lenger enn 90 minutter og formålet bør være diskusjon av saker eller problemer)
- Fokus på minst mulig dokumentasjon
- Deling av kompetansen, verktøy og maler med andre teammedlemmer
- Måling av fremgangen

3.4.6. USIKKERHETSHÅNDTERING

I Lean settes det stort fokus på eliminering og reduksjon av faktorer som fører til ”sløsing”. Med sløsing menes alle aktiviteter som ikke tilfører noen verdi til kunden. Eksempler på dette kan være produksjon av ekstra funksjoner, defekter, unødvendig ventetid eller et høyt forbruk av ressurser. All sløsing eller ”avfall” er et resultat av mangel på den nødvendige informasjonen, eller med andre ord er et resultat av usikkerhet. Det betyr at eliminering og reduksjon av avfall er den direkte måten å håndtere usikkerheten på.

Kontinuerlige kravendringer fra kunder skaper stor usikkerhet rundt det endelige produktet. Lean håndterer den type usikkerheten ved hjelp av en fleksibel arkitektur, tester og overvåkingsteknikker som tillater å tilpasse seg endrede krav på en rask og lett måte. De to sentrale teknikkene som gjør endringene enkle er bygging av tester på ulike stadier i utviklingsprosessen og forbedring av design gjennom utskiftende funksjonalitet.

Det er mye taus kunnskap som kan gå tapt ved en skriftlig overføring av informasjon som kan resultere til økning av usikkerhet. Derfor anses kunnskapsoverføring mellom medlemmer i et lite, kryss-funksjonelt team som mest effektivt i forhold til overføring av informasjonen på papir. Lean Software Development prioriterer et samarbeidende team mer enn papirarbeid og på denne måten forbedres den effektive kommunikasjonen og reduseres usikkerheten relatert til informasjonstapet.

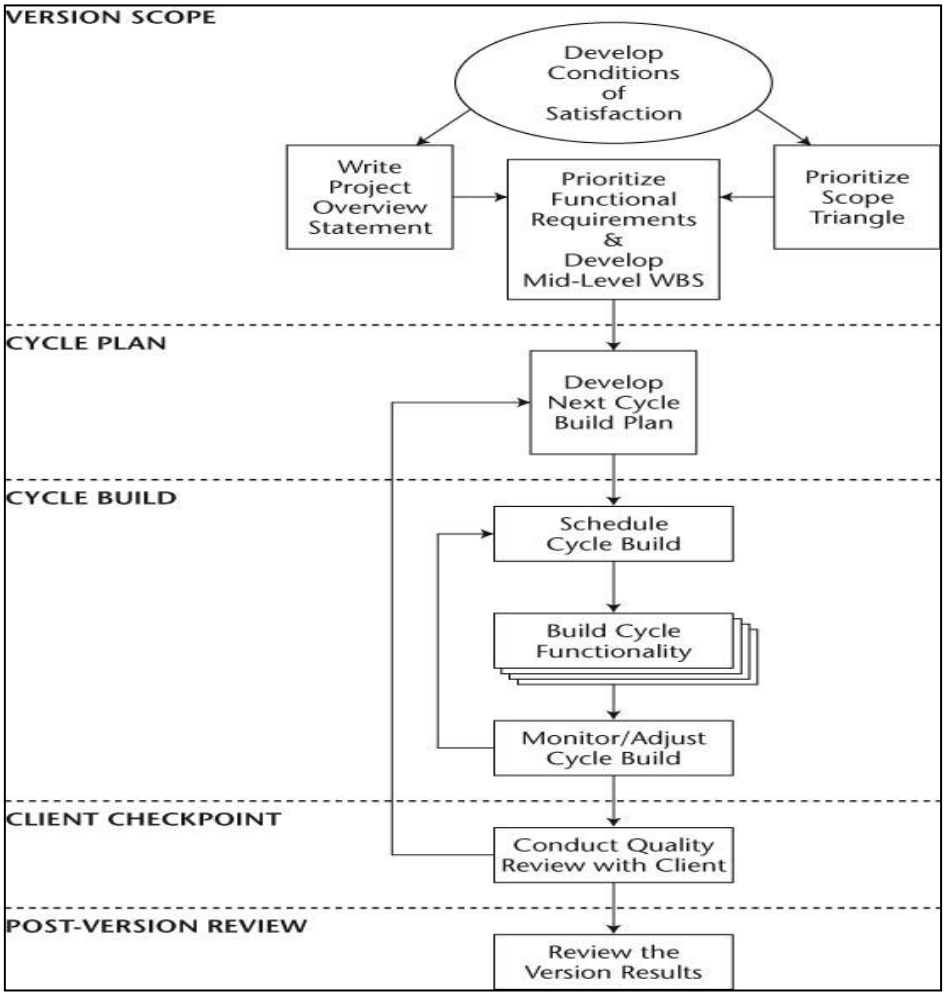
Lean handler om kontinuerlig forbedring som har stor påvirkning på usikkerhetshåndtering. Metoden bruker en forbedringsteknikk som kan anvendes fra en iterasjon til den andre. Forbedringen oppnås gjennom oppfølgning av ”Plan-do-check-act”-prinsippet som har til formål å kjøre et eksperiment og lage forbedringer basert på resultater. Kontinuerlig forbedring, effektiv kommunikasjon og eliminering av sløsing anses som de mest effektive Lean kjerneprinsippene som resulterer i reduksjon av usikkerheten gjennom hele prosjektets livssyklus.

3.5. ADAPTIVE PROJECT FRAMEWORK

Adaptive Prosjekt Framework (APF) er en smidig tilnærming til programvareutvikling som er designet for å levere maksimal forretningsverdi innenfor oppsatte tids- og budsjettrammer. APF er kjennetegnet ved å være kundefokusert og kundedrevet. Metoden tillater å levere raske og hyppige inkrementelle resultater og fokuserer på kontinuerlig introspeksjon og forandringer. Det som skiller APF fra andre utviklingsmetoder er at den tilnærmingen kan anvendes i alle typer prosjekter og ikke er begrenset til programvareutvikling

3.5.1. PROSESS

APF er en adaptiv utviklingsprosess som består av fem faser [6]. Fasene i APF prosessen er vist i figur nedenfor.



Figur 3.5. APF livssyklusmodell.

Den første fasen er en omfangsfase (Version Scope) som omfatter identifisering, prioritering og dokumentering av funksjonelle krav og utvikling av arbeidsnedbrytningsstruktur (Work Breakdown Structure - WBS). Fasen inneholder også utarbeidelse av betingelser for tilfredshet (Conditions of Satisfaction - COS), prosjektoversikt (Project Overview Statement – POS) og prioritering mellom tid, kostnader, kvalitet og omfang.

Den andre fasen (Cycle Plan) handler om utarbeidelse av plan for kommende syklus, valg av funksjonaliteter som skal utvikles, brytning av arbeidsoppgaver på et detaljert nivå og fordeling av disse oppgavene mellom teammedlemsgrupper.

I neste fase (Cycle Build) gjennomføres en detaljert planlegging og syklusen kan settes i gang. Den fasen gjelder også å oppfølge fremdrift innen syklus, monitorere risiko og rapportere prosjektstatus. Syklusen avsluttes når tiden er utløpt og det som ble ikke fullført i tiden overføres til neste syklus. Det er viktig i den fasen å registrere alle problemer i Issues Log og sette opp en liste over endringsforslag (Scope Bank).

I klientavsjekkingsfasen (Client Checkpoint) sjekkes nylig produserte funksjonaliteter mot det overordnede målet om maksimal forretningsverdi.

Den siste fasen er en ettervurdering(Post-Version Review). Den handler om evaluering av oppnådde resultater og et utvalg av de viktigste aspektene som skal brukes til forbedring i neste syklus.

3.5.2. ROLLER OG ANSVAR

I APF er sammensetning av teamet avhengig av forretningsområde og prosjekttype. Det vil si at roller og ansvar kan være ulike i et programvareutviklingsprosjekt, et produktutviklingsprosjekt og i et prosjekt innenfor forskning. De felles rollene i alle APF prosjekter er en co-manager for kundeteamet og en co-manager for utviklingsteamet [6].

Co-manager for utviklingsteamet administrerer utviklere og har en mer rådgivende rolle.

Co-manager for kundeteamet peker kundene i riktig retning og råder dem til å gjøre det beste valget blant en rekke alternativer.

Begge co-managerne har et likt ansvar for prosjektets suksess.

3.5.3. TEAM

Et APF team er sammensatt av to sub-team, et utviklingsteam og et kundeteam. I noen APF prosjekter med lav kompleksitet kan kundeteamet bestå av en person. I store og komplekse prosjekter har kundeteamet vanligvis flere medlemmer. Både i store og små prosjekter bør kundeteamet ha en person som har beslutningsmyndighet og kan fungere som co-manager for hele prosjektet.

Utviklingsteamet består av tekniske fagfolk og er ledet av en utviklings co-manager som samarbeider tett med co-manager for kundeteamet. Utviklingsteamet er ansvarlig for produksjon av leveransen. Hvis co-manager ikke får nok myndighet til å ta avgjørelser, blir prosjektet alvorlig funksjonshemmet. Derfor er tilstedeværelse av en selvstendig kundeleder en av suksessfaktorene i APF prosjekter.

Alle teammedlemmer har gode ferdigheter og kunnskap innen det fagområdet de jobber. Teammedlemmene er for det meste fagfolk på seniornivå som kan arbeide uten tilsyn. Prosjekter de gjennomfører er komplekse og har stor usikkerhetsgrad, derfor må teammedlemmene være kreative mennesker som kan finne en akseptabel løsning. De er selvstendige og vil være uavhengige, og dette kan gjøre dem mindre gode lagspillere. Det kan være ganske utfordrende for en co-manager å danne et godt miljø for et effektivt teamarbeid.

3.5.4. KUNDEN

Adaptive Project Framework er en ny måte å tenke kunderelasjon på. Det handler om best mulig forsyning og levering av den best mulige forretningsverdien til kunden. Det vil si at APF er en veldig klientfokusert metode. Kundene må være villige til å gå frem for å si sin tydelige mening. Deres forhold til prosjektteamet må være åpent. Kundene må også ha kunnskap om teknologien for å være likeverdige med utviklingsteamet. Det skapes stor synergi mellom to parter med helt ulike perspektiver på det samme prosjektet.

Klientfokusering setter kunden på første plass i prioriteringslisten. Det betyr at behovene til kunden må alltid komme først og deres interesser må være beskyttet. Den kjerneverdien som setter kunden i fokus tillater å maksimere forretningsverdien til kunden.

Den andre APF kjerneverdien handler om en kundedrevet prosess. Kunden får bestemme prosjektets retning. Kunden engasjeres i alle delene av prosjektet og i alle viktige aspekter, og får mulighet for å ta på seg prosjektlederens rolle og ansvar. Det skaper en følelse av eierskap og dermed kundens interesse og motivasjon øker betydelig. Derfor er meningsfull klient engasjement er en ”must” i alle APF prosjekter.

3.5.5. KOMMUNIKASJON

APF metoden legger stor vekt på en effektiv kommunikasjon mellom alle prosjektdeltakere. For å oppnå den mest effektive kommunikasjonen utveksler deltakerne informasjon mellom hverandre både på formelle og uformelle måter.

Den ønskelige effekten kan bli oppnådd ved kontinuerlig bruk av kommunikasjonsverktøy. For eksempel bør et APF prosjekt begynne med en strukturert samtale som kalles Conditions of Satisfaction (COS). Det kan være en kort, ansikt-til-ansikt samtale mellom en prosjektleder og en kunde eller det kan avsettes en uke til et formelt møte med flere deltakere. Det som er viktig med COS er at under den samtalen etableres et felles språk som skal brukes mellom kunden og teamet gjennom hele prosjektet. Samtalen resulterer til en fremforhandlet avtale som dokumenteres og som blir grunnlaget for problem- og konfliktløsning og beslutningstaking.

Den muntlige kommunikasjonsformen er sterkt favorisert i APF. Hvert teammedlem får mulighet til å delta i en åpen diskusjon med andre deltakere på daglige møter. Slike møter holdes hver morgen og varer fra 15 til 30 minutter. Daglige møter holdes for å avklare formål og oppdatere status for prosjektet. Problemløsning, beslutningstaking og konfliktløsning er ikke en del av møtene.

I APF er vanligvis teamet delt inn i to sub-team, et kundeteam og et utviklingsteam. Selv om sub-teamene jobber i samme prosjekt og har et felles mål, kan deres ønsker og synspunkter være forskjellige. Derfor er det viktig å etablere gode kommunikasjonslinjer mellom sub-teamene. Hvert sub-team holder vanligvis sine egne daglige møter, men teamene koordinerer arbeidet ved å utveksle informasjon mellom hverandre. Dette skjer ved mindre hyppige møter der bare to sub-teamledere deltar i en åpen diskusjon.

3.5.6. USIKKERHETSHÅNDTERING

APF er godt egnet for usikkerhetshåndtering og kan anvendes til de fleste prosjekter som er preget av høy kompleksitet og usikkerhet. APF leverer hyppige resultater som kan håndtere usikkerheten, og reduksjon av usikkerheten kan allerede skje i de første prosjektfasene.
I et APF prosjekt er Scope fleksibelt og planlegging foregår just-in-time. Korte iterasjoner tillater å levere resultater og få tilbakemeldinger fra kunden tidlig i prosjektet. Dette gir en god mulighet til å svare på raske endringer.

APF er utstyrt med mekanismer for læring og oppdagelse. Et eksempel på disse mekanismene er Swim Lanes som er egnet for å eksperimentere med mulige tilskudd til den løsningen som har blitt definert på et tidspunkt. Gjennom læring og oppdagelse tilskaffes mer av den manglende informasjonen som har en stor betydning for reduksjonen av usikkerheten.

APF metoden anses som høyt kundefokusert og kundedrevet. Kunden involveres i APF prosjekter i høy grad og driver hele prosessen. APF prosjekter setter fokus på små, samlokaliserte team der kunden tar en prosjektleder rolle. Ved å ha et eget kundeteam i prosjektet gjør det mulig å oppnå den mest effektive kommunikasjonen mellom kunden og utviklingsteamet og fylle opp informasjons- og kunnskapsgapet. Meningsfull kundeinvolvering tillater å fokusere på den endelige løsningen av høy forretningsverdi og minimere usikkerheten rundt det ønskede produktet.

DEL 4

TOLKNING OG DISKUSJON

4.1. PROSESS

En av de viktigste kjennetegn ved smidige metoder er tilpasningsdyktighet i forhold til de stadig endrede krav. Metodene håndterer endringer ved hjelp av livssyklusmodeller som kan ha en iterativ eller adaptiv tilnærming.

Scrum er en adaptiv utviklingsprosess som har en svært tilpasningsdyktig livssyklusmodell bestående av tre faser. Selve utviklingsprosessen går gjennom tre til åtte Sprinter som itererer til en fleksibel leveranse. Leveransen blir mer fullstendig etter hver Sprint som tar vanligvis tretti dager. Prosessen er godt egnet for å unngå utvikling av de produktene som kunden ikke vil ha. Fleksibiliteten forbedres gjennom en rekke kontrollmekanismer som brukes til å administrere uforutsigbarhet og kontrollere risiko. Scrum prosessen kontrollerer planlegging av produktutgivelse som gjør det mulig å levere produkter med høyere kvalitet og komme raskere frem til målet.

XP er en iterativ prosess bestående av seks faser der iterasjonene kan være ganske korte, fra noen timer til noen dager, men vanligvis tar en iterasjon en til tre uker. XP prosessen tillater å komme raskt med utviklingen. De korte iterasjonene støtter endringer underveis og dermed leverer en programvare av høy kvalitet. Programvaren har ofte en enkel design. Dette forklares med at produsert kode kontrolleres kontinuerlig med automatiske tester. Prosessen er godt egnet for kontinuerlig integrasjonstesting som kan kjøres flere ganger om dagen. XP prosessen er bygget opp slik at den kan tilpasse seg endringer faser og skiftende krav.

DSDM har en adaptiv livssyklusmodell som består av syv hvorav fem av dem er utviklingsfaser. Selve utviklingen foregår i iterasjoner og systemet leveres delvis i inkrementer i form av prototyper. Den inkrementelle prosessen resulterer til tidlig levering av systemet eller deler av systemet som trengs å bli levert tidligere enn resten av programvaren. Leveransen har en mer formell design og kan ikke kalles enkel. DSDM er tilpasningsdyktig og endringstoleranse er bygget inn i arkitekturen. Hyppige gjentakelser er bygget på respekt for prinsippet om reversibilitet. Avslutning av en iterasjon gir muligheten for å fortsette, stoppe eller gå tilbake. Prosessen er bygget opp slik at kvaliteten kan bli forbedret gjennom hele prosjektets livssyklus.

Lean Software Development er en iterativ prosess som identifiserer og eliminerer avfall gjennom kontinuerlig forbedring. Lean har en fleksibel arkitektur som er komponentbasert og serviceorientert. Design har et mønster som tillater å tilpasse seg endrede krav. Prosessen er bygget opp slik at risikoen blir isolert på et funksjonsnivå. Dette skjer ved å dele levering av funksjoner i flere trinn. Valg, planlegging, utvikling, testing og distribusjon av en funksjon må bli fullført før neste funksjon settes i gang. Lean har systematikk, ryddighet og en jevn flyt i prosessen som optimaliseres gjennom fjerning av ”sløsing”.

Adaptive Project Framework har en adaptiv livssyklusmodell som består av fem faser. APF prosessen gjør det mulig å levere resultater så tidlig som mulig. APF kan godt tilpasse seg endringer og dette kan gjøres til og med i første fasen. Det vil si at hvis det ligger behov for å endre Scope, er det mulighet for det i Version Scope fasen. Etter hver syklus avsjekkes leveransen mot kunden og hvis den ikke tilfredsstiller kravene, planlegges da neste syklus.

Sammenligning av metodene i henhold til deres livssyklus er vist i tabell 4.1. XP metoden har den korteste iterasjonslengden i forhold til andre metoder og er best egnet til en rask levering. Iterasjonene i XP kan være veldig korte og kan fullføres på noen minutter eller timer. Det som gjelder håndtering av endringer, viser tabellen at Scrum og APF kan raskere tilpasse seg til skiftende krav. Lean Software Development viser seg til å være best når det gjelder forbedringsprosessen som har en tendens til å være kontinuerlig. DSDM anses som den mest komplekse av alle metodene som ble vurdert i forhold til deres utviklingsprosess på grunn av sin formalitet. Metoden innebærer en progressiv utvikling av krav og det kreves full innsats på selve prosessen.

Tabell 4.1. Sammenligning av metoder i forhold til deres utviklingsprosess.

	Scrum	**XP**	**DSDM**	**LSD**	**APF**
Livssyklusmodell	Adaptiv modell	Iterativ	Adaptiv modell	Iterativ	Adaptiv modell
Utviklingsprosess	Iterativ og inkrementell	Iterativ	Iterativ og inkrementell	Iterativ	Iterativ
Antall faser	3	6	7	4	5
Iterasjonslengde	2-4 uker	Maksimum 1-2 uker (noen timer)	2-3 uker	2-4 uker	4 uker
Evner til å tilpasse endringer	Svært dyktig	Meget dyktig	Dyktig	Meget dyktig	Svært dyktig
Utviklingsformål	Fleksible og tilpasnings-dyktige løsninger	Raskere utvikling og under budsjett	Løsninger som oppfyller tidsfrister	Kontinuerlig forretnings-verdi gjennom reduksjon av feil og syklustider	Maksimering av forretnings-verdi gjennom justering av scope

4.2. ROLLER OG ANSVAR

I et smidig team påtar teammedlemmene seg et ansvar for en rolle ved behov. De vurderte metodene har forskjellige behov for ulike roller og rollesammensetningen varierer fra en metode til en annen. Tabell 4.2. viser rollesammensetningen i Scrum, XP, DSDM, Lean og APF og presenterer behovet for en prosjektleder i de overnevnte metodene.

Scrum kan ha flere teammedlemmer, men i utgangspunktet omfatter Scrum prosessen bare tre ulike roller. Hovedansvaret for prosessen bæres av Scrum Master som får delvis hjelp av Produkteier og Scrum Teamet. Ansvaret er klart definert og fordelt mellom de tre rollene. Det er viktig å nevne at i Scrum er det intet behov for en prosjektleder, siden teamet tar på seg en betydelig del av prosjektlederens rolle og at resten av ansvaret legges på Scrum Masteren.

I XP, DSDM og Lean varierer antall roller fra et prosjekt til et annet, og disse metodene har en annen tilnærming til prosjektlederens rolle enn Scrum metoden har. I XP-metoden er den rollen klart definert og prosjektlederen opptrer som Big Boss og har et fullstendig ansvar for prosjektets suksess. I DSDM og Lean Software Development har prosjektlederen en underordnet, en teamleder som delvis frigjør prosjektlederen fra ansvaret over teamet. DSDM rollefordeling trenger en mer detaljert organisasjonsstruktur. Det betyr ikke at DSDM prosjekter har en tradisjonell strukturform, men det vil si at det er et behov for en mer formell utbygging av organisasjonsstruktur enn i de andre fire metodene.

I APF er prosjektlederens rolle fordelt mellom to teamledere, en kundeleder og en utviklingsleder. En slik fordeling av ansvaret har sine fordeler og ulemper. Fordelen med to teamledere som har et likt ansvar er at utviklingen kan bli mer effektiv når en kundeleder bistår interesser for sitt team på samme måte som utviklingslederen gjør. Men hvis de to teamene får uenigheter, kan det resultere til et behov for en uavhengig person som vil finne en balanse mellom uenige teamledere. Det fører til svekkelse av kommunikasjonslinjer og kan resultere til tap av utviklingstiden.

Tabell 4.2. Sammenligning av metoder i forhold til roller og ansvar.

	Scrum	XP	DSDM	LSD	APF
Teammedlemmer	Scrum Master, Erfaren ingeniør, Junior ingeniør, QA Tester, Writer	Kunden, Programmerer, Tester, Tracker, Coach	Teamleder, Ambassadør-bruker, Rådgiver-bruker, Senior Utvikler, Utvikler, Scribe	Fasilitator, Champion, Teamleder	Programmerer, Tester, Utvikler
Prosjektmedlemmer	Scrum Master, Produkteier	Big Boss	Visjonær, Sponsor, Prosjektleder, Teknisk koordinator, Tilrettelegger	Prosjektleder Sponsor, Ad Hoc fagekspert	Kunde Co-manager, Utviklings Co-manager
Prosjektlederens rolle og ansvar	Ikke definert	Klart definert	Klart definert	Klart definert	Ikke definert
Ansvar for prosjektet	Fordelt mellom team og Scrum Master	Prosjektleder/ Teamleder	Prosjektleder/ Teamleder	Prosjektleder/ Teamleder	Fordelt mellom kundeleder og utviklingsleder

4.3. TEAM

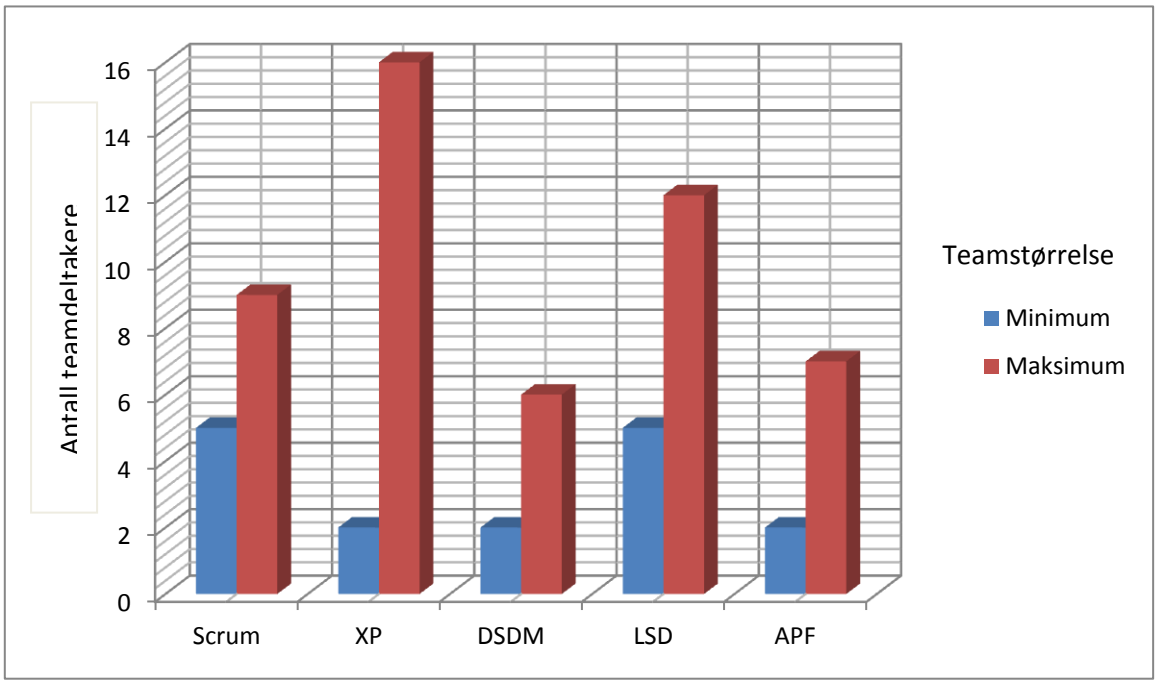
Teamstørrelse påvirker prosjektets kompleksitet og et stort team kan resultere til kompleksitetsøkning. Derfor anbefales det å ha et team av en liten størrelse i smidige prosjekter.

I figur 4.1 er metodene sammenlignet i forhold til teamstørrelse. Figuren viser at alle de vurderte metodene favoriserer små team. Scrum prosjekter trives best når teamet har syv deltakere, pluss minus to. XP, DSDM og Lean Software Development har ikke en fast regel som Scrum har, men disse metodene holder team også av en liten størrelse. Teamstørrelse i XP kan variere betydelig, men i praksis så viser det seg at et team på seks deltakere fungerer best for denne metoden. APF team har en helt annen oppbygning enn de overnevnte metodene og har sine fordeler og ulemper. Det er sammensatt av to sub-team med to til syv teammedlemmer i hvert sub-team.

Det som gjelder teamegenskaper, viser tabell 4.3 at Scrum har det meste uavhengige teamet som ikke har behov for en leder, og motivasjon, styring og organisering er teamets særtrekk. XP teamet har ganske lignende egenskaper, men forskjellen er at i et XP prosjekt foregår teamarbeid i par. DSDM teamet er også selvdrivende og selvmotiverende, men til en viss grad. Det vil si at teamet tar ikke alle beslutninger, men det får lov til å organisere seg og ta enkle avgjørelser. Et Lean team fokuserer på selvmotivasjon og hvert teammedlem kan ta beslutninger. I forhold til Scrum, XP, DSDM og Lean, kan et APF team ikke beskrives som selvdrivende. Myndigheten er fordelt mellom to teamledere der hver leder bærer ansvaret for sitt eget team.

Tabell 4.3 Sammenligning av metoder i forhold til teamets egenskaper.

	Scrum	**XP**	**DSDM**	**LSD**	**APF**
Antall team per prosjekt	1-4 eller flere	1	1-6	1-5	1-6
Team egenskaper	Selvorganiserende selvmotiverende, selvdrivende, kryssfunksjonelt	Selvorganiserende team der programmering utføres i par	Selvdrivende fokusert team med begrenset myndighet	Selvorganiserende selvmotiverende, myndighet til hvert medlem	Drives av to teamledere



Figur 4.1. Teamstørrelse i smidige metoder

4.4. KUNDEN

Smidige metoder favoriserer kundedeltakelse, men graden av kundeinvolvering varierer fra en metode til en annen. Tabell 4.4 beskriver kundeinvolveringen i de smidige metodene.

APF er den ledende metoden i henhold til kundeinvolvering. Kunden er ikke bare involvert i utviklingsprosessen, men har myndighet og tar de viktigste beslutningene. Siden APF har et team bestående av kundens representanter med sin egen kundeleder, tildeles metoden den høyeste graden av kundeinvolvering i forhold til andre smidige metoder. Lean Software Development kan også ha et kundeteam som blir direkte involvert i prosessen, men teamet får ikke samme myndighetsgrad som kundeteamet i APF metoden. Både DSDM og XP krever en kunde til stede (On-site customer), men DSDM er den metoden som er mest drevet av kundens tilbakemeldinger.

Scrum har den laveste graden av kundeinvolvering. Sammenlignet med andre metoder, har Scrum ingen retningslinjer for kundeengasjement. Det vil si at kunden blir ikke direkte involvert i utviklingsprosessen, men tar gjerne en plass utenfor prosjektet. Vanligvis har kunden i Scrum en representant som kan bli en del av teamet eller kan bli representert av Produkteier.

Tabell 4.4. Kundeinvolvering i smidige metoder

Smidige metoder	Kundedeltakelse
Scrum	Kunden har en representant
XP	Kunden har egen representant On-site kunde
DSDM	Kontinuerlig kundekontakt i form av tilbakemeldinger On-site kunde Proaktiv kunde
LSD	Kunden involveres i stor grad. Representanter eller kundeteam
APF	Kundedrevet utviklingsprosess Kundeteam har 50 % av myndighet

4.5. KOMMUNIKASJON

For å oppnå den mest effektive kommunikasjonen verdsetter de fleste agile metodene en enkel og direkte kommunikasjon der mediet i form av tale og kroppsspråk er en av de viktigste nøklene til prosjektets suksess. Ansikt-til-ansikt samtaler er favorisert i alle smidige metoder. Den kommunikasjonsformen utgjør en likhet mellom de vurderte metodene. Forskjellen mellom metodene er bruken av ulike kommunikasjonsteknikker gjennom hele prosjektet.

Tabell 4.5. Sammenligning av kommunikasjonsprosess i smidige metoder

	Mest brukt kommunikasjonsform	**Kommunikasjonsteknikker**
Scrum	Ansikt-til-ansikt	Sprintplanmøtet Daglige scrum-møter, Sprintgjennomgang Sprintretrospektivet Visuelle tavler Backlog grafen
XP	Ansikt-til-ansikt	Stand-up møter Planmøter Visuelle diagrammer Parprogrammering Prototyping
DSDM	Ansikt-til-ansikt Skriftlig kommunikasjon i form av dokumentasjon	Daglige møter Uformelle samtaler Hindring av barrierer Raske tilbakemeldinger Modellering
LSD	Ansikt-til-ansikt	Daglige møter Bruken av visuelle tavler Hyppige tilbakemeldinger Reduksjon av kommunikasjonsavfall

APF	Ansikt-til-ansikt	COS Daglige møter Klientavsjekking

Tabell 4.5 representerer likheter og forskjeller mellom typer av kommunikasjon som brukes i de vurderte metodene. Tabellen viser at daglige møter holdes i alle metodene. Møtene kan variere i henhold til varighetstid og antall deltakere, men felles for alle metodene er at de daglige møtene er korte og har et konkret formål.

For å oppnå den mest effektive kommunikasjonen brukes det også andre kommunikasjonsformer i tillegg til ansikt-til-ansikt samtaler. For eksempel i Scrum og Lean brukes det tavler som er et godt redskap for å få en visuell forståelse. I XP metoden foregår den visuelle kommunikasjonen ved bruk av store diagrammer. XP bruker også andre rike kommunikasjonsteknikker som parprogrammering og modellering. DSDM er den eneste metoden som bruker prototyping som et effektivt kommunikasjonsverktøy. Selv om metoden legger stor vekt på rike kommunikasjonsteknikker, anses DSDM som metoden med mest formell kommunikasjonsform. I forhold til andre metoder brukes det mye tung dokumentasjon i DSDM metoden. LSD har et sett av teknikker som hjelper å eliminere og redusere avfall og selve metoden legger stor vekt på oppnåelse av den mest effektive kommunikasjonen mellom alle interessegrupper.

Kommunikasjon mellom teamet og kunden kan ha ulike former. Tilbakemeldinger fra kunden er den mest populære kommunikasjonsteknikken i alle smidige metoder. Det foretrekkes å ha en kunde som kan være tilgjengelig gjennom hele prosjektet for å unngå misforståelser og akselerere kommunikasjonsprosessen. Det er vanlig å kommunisere med kunden via samtaler, ansikt-til-ansikt. I tillegg til den muntlige formen kan også godt brukes skriftlige former, e-post og brev som gir mer formell betydning og sikrer at for eksempel de fremforhandlede avtalene skal holdes.

4.6. USIKKERHETSHÅNDTERING

De fleste smidige prosjekter er preget av høy kompleksitet og usikkerhet. Jo mer komplekse prosjektene er, jo mer uforutsigbare og usikre blir de. Det er vanligvis en stor utfordring å håndtere usikkerhet i et komplekst prosjekt. Scrum, XP, DSDM, LSD og APF er de metodene som har gode teknikker som påvirker usikkerheten gjennom prosjektets livssyklus.

De vurderte metodene håndterer usikkerheten ved å kjøre korte iterasjoner og ved å tilpasse seg endringer på en rask måte. Usikkerheten avtar etter hver ny iterasjon når løsningen blir mer fullstendig. I Scrum håndteres usikkerheten ved iverksetting av forbedringer før hver ny Sprint, og usikkerheten forvandles til sikkerheten på kortest mulig måte. I XP kan iterasjonene ta bare noen minutter eller timer og hyppige endringer kan håndteres ved hjelp av kontinuerlig testing og integrasjon. DSDM metoden bruker tidsbokser med fast lengde som tillater å identifisere usikre oppgaver som lett kan strekke seg ut over tidsfristen. Lean håndterer usikkerheten ved hjelp av en fleksibel arkitektur, tester og overvåkningsteknikker som tillater å tilpasse seg endrede krav på en rask og lett måte. APF er utstyrt med Swim Lanes som reduserer usikkerheten ved å oppdage den manglede del av løsningen.

En effektiv kommunikasjon er et godt middel til å håndtere usikkerhet. En rask informasjons- og kunnskapsoverføring kan betydelig redusere usikkerheten ved å fylle opp gap som utgjør grunnlaget for usikkerheten. I de fleste metodene som blir vurdert settes det stor fokus på kommunikasjon. Informasjonsoverføring skjer vanligvis på en rask måte der prosjektdeltakerne kommuniserer ansikt-til-ansikt. Scrum, APF, Lean og XP har små team, favoriserer uformell kommunikasjon og tillater å oppnå en god informasjons- og kunnskapsoverføring gjennom hele utviklingsprosessen. DSDM metoden har mer utfordringer til å oppnå den effektive kommunikasjonen på grunn av flere teammedlemmer og fokus på dokumentasjon. Selv om formell kommunikasjon anses som mer troverdig og pålitelig, trengs det lengre tid for budskapsoverføring ved bruk av denne kommunikasjonsformen.

Usikkerheten kan bli forårsaket av flere faktorer. Derfor må i tillegg til effektiv kommunikasjon, korte iterasjoner og tilpasningsevner også vurderes prosjektvarighet, markeds- og teknisk stabilitet og mulige avhengigheter. Hvis alle faktorene kan bli tatt i betraktning, viser det seg at APF er bedre egnet til å håndtere usikkerheten i svært komplekse prosjekter enn andre metoder.

4.7. OPPSUMMERING

Vurdering av de fem metodene ble gjennomført med fokus på utviklingsprosess, roller, team, kunden, kommunikasjon og evner til usikkerhetshåndtering, og en kort oppsummering er vist i tabellen nedenfor.

Tabell 4.6. Smidige metoder i forhold til vurderingskriterier.

	Scrum	XP	DSDM	LSD	APF
Prosess	-Iterativ prosess -30 dagers syklus (Sprint) -Tilpasnings dyktig modell	-Enkel prosess med fokus på kommunikasjon -Korte iterasjoner -Hyppige utgivelse -Redesign	-Iterativ utvikling med fokus på levering til rett tid -Fokus på testing, kvalitet og produktivitet	-Iterativ prosess med fokus på kontinuerl. forbedring -Eliminering av avfall	Iterativ prosess med fokus på forretningsverdi maksimering på kort tid
Roller	-Få roller -Ikke behov for prosjektleder -Scrum Master og utviklingsteam deler ansvar	Få roller	-Flere brukers roller -Prosjekt lederens rolle er klart definert	Flere	-Flere -Kunden påtar seg prosjektledere ns rolle
Team	-Uavhengig, selvorganisere nde-Små team -Høy effektivitet og produktivitet blant utviklere	Små, selvorganiseren de team	-Små samarbeiden de team -Begrenset selvorganiser ing	Små tverrfaglig team	-Vanligvis små -Store team deles i to sub-team

Kunden	-Involveres i utviklings prosessen	-Onsite kunden -Kundens eierskap av funksjonsprioritering	Kontinuerlig kundeinvolvering	Involveres i stor grad	Kunden styrer prosessen
Kommunikasjon	Uformell	Uformell	Delvis formell	Uformell	Uformell
Håndtering av usikkerhet	Godt	Håndterer i en viss grad	Godt	Godt	Meget godt
Særtrekk	Fokus på forretningsverdi	Enkelhet	Fokus på forretningsbehov	Fokus på prosjektets avkastning (ROI)	Håndtering av kompleksitet og usikkerhet

DEL 5

KONKLUSJON

5.1. KONKLUSJON

Den gjennomførte vurderingen har vist at alle av de fem metodene bærer en betydelig grad av agility. Metodene kjennetegnes ved å ha kort utviklingsprosess, små team, meningsfull kundeinvolvering, effektiv kommunikasjon og er godt egnet for usikkerhetshåndtering gjennom hele prosjektets livsløp.

Valg av den mest passende prosjektledelse metoden kan være en utfordring og der er flere faktorer som bør komme i betraktning. Prosjektets behov anses som en av de viktigste valgskriteriene. Hvis for eksempel prosjektet er stort, har et behov for levering av høykvalitetsprodukt til rett tid, samt har behovet for mer omfattende kontroll, bør bruk av DSDM metoden bli vurdert. Scrum kan passe til prosjekter som har til formål å forbedre teamproduktivitet, maksimere forretningsverdien og levere fleksible og tilpasningsdyktige løsninger. Hvis kontinuerlig forbedring og prosjektets avkastning er de største forventningene, kan Lean metodikken være til hjelp. Hvis prosjektet er lite, svært dynamisk og har et behov for hyppige og raske utgivelser, samt kostnadsreduksjon, er XP en løsning som kan dekke disse behovene. Hvis prosjektet er av en høy kompleksitets- og usikkerhetsgrad og forretningsverdien må maksimeres innen korte tidsrammer, kan APF være den metoden som vil tilfredsstille behovene.

Den gjennomførte vurderingen gir ikke grunnlaget for rangering av metodene til beste og verste, men gir gjerne en basis for valg av best passende metode. Derfor er det ingen metoder som kan kalles gode eller dårlige, og valg av den best passende prosjektledelse metoden er i stor grad avhengig av prosjektets behov.

REFERANSER

[1] Cockburn, A., "Agile Software Development: The Cooperative Game", Addison-Wesley Professional, Boston, USA, 2007.

[2] Highsmith, J., Orr, K., Cockburn, A., "Extreme Programming", E-Business. Application Delivery, 2000.

[3] Svendsen, E., "Mot en agil tilnærming?", Den Norske Dataforening, 2011, [online]. <http://www.dataforeningen.no/mot-en-agil-tilnaerming.4959391-134233.html>. [Nedlastet 05/3-2012].

[4] Cockburn A., Highsmith J., "Agile software development: The business of innovation", IEEE Computer, No.9, 2001.

[5] Cockburn A., "Agile software development joins the 'would-be' crowd", Cutter IT Journal, No.1, 2002, [online]. < http://www.dataforeningen.no/kontakt-osspresse.134521.no.html>. [Nedlastet 08/3-2012].

[6] Wysocki, R., "Effective Project Management: Traditional, Agile, Extreme", John Wiley&Sons, Indianapolis, USA, 2009.

[7] Fowler, M., Highsmith J., "The Agile Manifesto", Software Development 9, No.8, 2001, [online]. <http://www.agilemanifesto.org/iso/no/>. [Nedlastet 13/3-2012].

[8] Boehm, B., "Get ready for the agile methods, with care", Computer 35, 2002, [online]. <http://sunset.usc.edu/events/2002/arr/Get%20Ready%20for%20Agiel%20Methods,%20with%20Care.pdf>. [Nedlastet 15/3-2012].

[9] Katzenbach, J.R., Smith, D.K., "The Wisdom of Teams: Creating the high-performance organization", McKinsey & Company, New York, USA, 2003.

[10] Patton, G. S., "War as I knew it", Houghton Mifflin Co., Boston, USA, 1947.

[11] Ambler, S., Lines, M., "Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise", IBM Press, Boston, USA, 2012.

[12] McKenzie , C., "When Managing Agile Teams, Size Really Does Matter", Enterprise Java Community, 2012, [online].

<http://www.theserverside.com/discussions/thread.tss?thread_id=63450>. [Nedlastet 16/3-2012].

[13] Chin, G., "Agile Project Management: How to Succeed in the Face of Changing Project Requirements", AMACOM Books, New York, USA, 2004.

[14] Cockburn, A., "Agile Software Development", Addison-Wesley, Boston, USA, 2002.

[15] Schuh, P., "Integrating Agile Development in the Real World", Charles River Media, Hingham, USA, 2004.

[16] Boehm, B., Turner, R., "Balancing Agility and Discipline: A Guide for the Perplexed", Addison-Wesley, 2003.

[17] ISO 9241-210: 2010, "Ergonomics of human-system interaction -- Part 210: Human-centred design for interactive systems", International Standard Organization, 2010.

[18] Husby, O., Kilde, H., Torp, O., "Usikkerhet som gevinst – Styring av usikkerhet i prosjekter", Norsk senter for prosjektledelse, 1999.

[19] Andersen E., "Prosjektledelse – et organisasjonsperspektiv", NKI forlaget, Oslo, 2005.

[20] Christensen, S., Kreiner, K., "Prosjektledelse under usikkerhet", Universitetsforlaget, Oslo, 1991.

[21] Rolstadås, A., Johansen, A., "From Protective to Offencive Project Management", PMI Global Congress, Malta, 2008.

[22] Gottschalk, P., Karlsen, J., "Prosjektledelse – fra initiering til gevinstrealisering", Universitetsforlaget, Oslo, 2008.

[23] Sheerin, L., "Software Estimation and the Cone of Uncertainty", Semantico, UK, 2009, [online]. <http://www.semantico.com/2009/06/software-estimation/>. [Nedlastet 19/03-2012].

[24] McConnell, S., "Software Estimation: Demystifying the Black Art", Microsoft Press, 2006.

[25] Little, T., "Context Adaptive Agility: Managing Complexity and Uncertainty," IEEE Software, 2005, Vol. 22, No. 2, 2005.

[26] Scrum, Wikipedia, [online]. <http://en.wikipedia.org/wiki/Scrum_(development)>. [Nedlastet 25/03-2012].

[27] Schwaber, K., "Scrum Development Process", OOPSLA Workshop on Business Object Design and Implementation, Springer–Verlag, 1995.

[28] Johansen, T., "Ekstrem Programmering - XP", Proxycom, [online]. <http://www.proxycom.no/faglig-forum/ekstrem-programmering.html>. [Nedlastet 1/04-2012].

[29] Extreme Programming, Wikipedia, [online]. <http://en.wikipedia.org/wiki/Extreme_programming>. [Nedlastet 2/04-2012].

[30] Beck, K., "Extreme programming explained: Embrace change, Addison-Wesley, 1999.

[31] Wells, D., "Extreme Programming: A gentle introduction", [online]. <http://www.extremeprogramming.org/>. [Nedlastet 5/04-2012].

[32] Koskela, J., Abrahamsson, P., "On-Site Customer in an XP Project: Empirical Results from a Case Study", EuroSPI, Trondheim, Norway, 2004, [online]. <http://agile.vtt.fi/docs/publications/2004/2004_eurospi_customer_in_an_xp_project.pdf>. [Nedlastet 7/04-2012].

[33] Dynamic Systems Development Method, Wikipedia, [online]. <http://en.wikipedia.org/wiki/Dynamic_systems_development_method>. [Nedlastet 14/04-2012].

[34] Dunham, M., "Lean Software Product Development in 4 Phases", Scio Consulting International Haut Tech, 2011, [online]. <http://blog.sciodev.com/2010/02/24/lean-software-product-development-in-4-phases/>. [Nedlastet 19/04-2012].

[35] Office of Lean Enterprise, "DEP Lean Kaizen Event Participant Roles and Responsibilities", Iowa Department of Management, [online]. <http://www.ct.gov/insidedep/lib/insidedep/lean/DEP_Lean_Kaizen_Event_Participant_Roles.pdf>. [Nedlastet 22/04-2012].