NTNU
Det skapende universitet

Smidig systemutviklingsmetode i praksis

Et case studie av Scrum

Kenneth Krogh

Master i datateknologi
Innlevert: juni 2014
Hovedveileder: Eric Monteiro, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Sammendrag

Smidige systemutviklingsmetoder, med Scrum i spissen, har utviklet seg til å bli de mest brukte metodene ved utvikling av nye systemer. Disse metodene har blitt presentert som en løsning på alle problemene med de tradisjonelle metodene, men i dag er det ofte slik at smidige metoder, som Scrum, ikke brukes helt som beskrevet i teorien. For å se nærmere på hvorfor det er slik ble det utført et fire uker langt case studie i et prosjekt som brukte Scrum for å se på hvilke deler som avviker fra teorien og hvorfor. Fra dette prosjektet ble det klart at alle deler fulgte teorien til en viss grad, men at bare halvparten av delene fulgte teorien i stor grad. De største avvikene gjaldt manglende informasjonsinnhenting i forbindelse med implementasjon av metoden, bruken av definerte roller, mangel på delte visjoner, samt bruken av sprint planning , sprint backlog og sprint evaluering. Prosjektet hadde et team som jobbet spesialisert hvor hver utvikler jobbet på faste deler i prosjektet, og en prosjekteier som gikk langt utover sine ansvarsområder og påtok seg arbeid som egentlig falt under rollen som scrummaster.

Avviket fra teorien var i dette prosjektet i stor grad som et resultat av at man tilpasset Scrum til situasjonen, men bruken av sprint backlog og sprint evaluering var ufrivillige avvik fra teorien som følge av at de ble brukt rituelt. Dette gjaldt særlig ved bruken av sprint evaluering hvor møtet ble utført bare for å ha gjort det. Informasjonen som kom frem i disse møtene ble ikke brukt videre, selv om teorien sier at disse møtene skal forbedre bruken av Scrum i prosjektet. Avhandlingen konkluderer med at Scrum bør brukes forskjellig avhengig av hvilken situasjon prosjektet er i. Størrelsen på prosjektet påvirker bruken av rollene i Scrum, spesialisering blant utviklerne og bruken av inkrementell utlevering, mens prosjektetgruppas egne grunnprinsipper bestemmer om resten av metoden skal brukes som i dag eller som et skall hvor innholdet i metoden i stor grad bestemmes av prosjektgruppa selv. Videre studier er nødvendig for å finspisse bruken av Scrum i de ulike situasjonene.

Abstract

Agile software development methods, with Scrum in particular, have over the last years become the most widely used medthods in the development of new systems. The agile methods have been presented as a solution to all the problems of the traditional methods, but today it is often the case that agile methods are not used completely as described in theory. To get a deeper understanding of why this is the case, a four-week case study of a project that used Scrum was performed to see which parts that differ from theory and why. From this project it was clear that all parts of Scrum followed the theory to a certain extent, but only half the parts followed the theory to a great extent. The largest deviations were related to lack of information retrieval associated with the implementation of the method, the use of defined roles, lack of shared vision, and the use of sprint planning, sprint backlog and sprint evaluation. The project had a development team where each developer was specialized and worked on fixed parts of the project, and a project owner who went far beyond her responsibilities and assumed work which actually was for the scrum master to perform.

The deviations from the theory were largely as a result of adapting Scrum to suit the situation in the project, but sprint backlog and sprint retrospective were involuntary deviating from the theory because of ritualistic use of these parts. This was particulary noticable for the sprint retrospective meetings which was held just to be able to be done with it. The information presented in these meetings was not used to improve the use of Scrum in the project, even though theory states that these meetings serve the purpose of improving Scrum in the current setting. This thesis concludes tthat Scrum should be used differently depending on the situation of the project. The size of the project affects the use of roles in Scrum, specialization among the developers and the use of incremental disclosure, while the fundamental ideas of the project group determines whether the rest of the method should be used like it is used today or as a shell in which the contents of the method is largely determined by the group itself. Further studies are needed to tweak the use of Scrum in the various situations.

Forord

Denne masteroppgaven ble utarbeidet våren 2014 som en avslutning på det femårige studiet som sivilingeniør i Datateknikk ved Norges teknisk-naturvitenskaplige universitet (NTNU). Hensikten med avhandlingen er å gi en dypere innsikt i hvordan den smidige utviklingsmetoden Scrum blir brukt i praksis, hvorfor den brukes som den gjør, og hvordan metoden kan og bør bli brukt videre i fremtiden.

Jeg vil rette en stor takk til min veileder fra NTNU, Eric Monteiro, for god veiledning gjennom konstruktive tilbakemeldinger i hele prosjektperioden. Takk til prosjektgruppa for muligheten for å utføre observasjon, samt en ekstra takk til Kristin Foosnæs fra NTNU Vitenskapsmuseet for verdifull innsikt i kundens side av prosjektet. Til slutt vil jeg takke alle diskusjonspartnere for deres innspill. Alle disse bidrag har vært uvurderlig for denne avhandlingen.

Kenneth Krogh
Trondheim, 9. juni 2014

Innholdsfortegnelse

Kapittel 1

Introduksjon

1.1 Motivasjon

Nye artikler i både *Computer World*[52, 53] og *Computer Weekly*[20] skriver om bruk av smidige utviklingsmetoder innen IT. Denne måten å utvikle nye produkter på blir stadig bedre kjent verden over, og publiseringen av disse artiklene viser at dette fenomenet nå begynner å skape en allmenn interesse. I det siste har Scrum også blitt større i akademisk sammenheng hvor stadig flere forsker på metoden [15]. IT-bransjen er kjent for utallige "hypes" og introdusering av nye ideer og kunnskap, uansett om det gjelder ny teknologi eller nye metoder. Flere nye ideer og kunnskap har blitt promotert som det neste store innen IT, som alle bedrifter burde ta i bruk. Dette gjelder for eksempel de smidige utviklingsmetodene [26]. For å forsvare seg mot en "hype" og ikke ukritisk ta i bruk disse nye ideene og kunnskapen som stadig oppstår, er det viktig å huske hvordan ting var før. Mange av ideene som presenteres har nye sider ved seg, men det er også ofte mange gamle, kjente sider ved seg. Det er ikke sikkert at alt er så nytt og revolusjonerende som presentert. Det at Scrum får publikasjoner både i nyhets- og akademisk sammenheng viser at det er en metode som setter spor, og det er nettopp dette som gjør Scrum interessant.

Systemutviklingsmetoder har gjennomgått en kontinuerlig utvikling siden de første datamaskinene kom på markedet. Fra å bruke de tradisjonelle utviklingsmetodene, som fossefallsmetoden [38], har man i dag i stor grad gått over til såkalte smidige utviklingsmetoder, for eksempel Scrum. Disse smidige utviklingsmetodene har stort fokus på kommunikasjon med kunden med tanke på forandring og fleksibilitet. Mens det i tidligere tradisjonelle utviklingsmetoder var brukerne som ble involvert i utviklingen har smidige

metoder valgt å involvere kunden. Å involvere kunden i større grad enn i de tradisjonelle metodene har vist seg å gi gode resultater for bedriftene som har tatt i bruk smidige utviklingsmetoder [16]. Det er nærliggende å tro at smidige utviklingsmetoder vil fortsette å bli tatt i bruk av stadig flere bedrifter [46], men at man i større grad må tilpasse metodene til de individuelle prosjektene for å oppnå ønskede resultater [25].

1.2 Forskningsspørsmål

Det oppstår alltid avvik mellom teori og bruk når det gjelder metoder og beskrivelse av prosesser og det er naivt å tro at dette er annerledes i forbindelse med smidige utviklingsmetoder. Selv om det er allment kjent at avvik oppstår er det mangel på forskning på dette området, spesielt i forhold til smidige metoder. Denne avhandlingen ser nærmere på avik mellom teori og bruk i smidige metoder og det overordnede forskningsspørsmålet er:

- Hvordan kan vi bedre forstå relasjonen mellom teori og bruk i smidige utviklingsmetoder?

Dette forskningsspørsmålet er for stort og lite spesifikt til å kunne komme med noen gode svar. Siden Scrum er den mest kjente smidige utviklingsmetoden er det denne metoden jeg vil forske på, i tillegg bryter jeg forskerspørsmålet opp i tre mer konkrete og reelle deler i forbindelse med denne rapporten:

1. Hvilke *avvik* oppstår ved bruk av Scrum som utviklingsmetode?
2. *Hvorfor* oppstår avvikene?
3. Hvilke *alternativer* finnes i bruken av Scrum?

Funnene i dette studiet blir analysert før jeg kommer med spesifikke forslag til hvordan Scrum bør brukes i fremtidige prosjekter.

1.3 Fremgangsmåte

For å finne svaret på disse spørsmålene ble det tatt utgangspunkt i eksisterende prosjekter, for å se hvordan Scrum ble brukt i praksis. Et eksempelstudie ("case study") i prosjektsettingen ble utført for å samle inn kvalitativ informasjon om bruken av metoden gjennom intervjuer og observasjoner.

Høsten 2013 ble det utført et mindre forstudie som forberedelse til hovedstudiet som ble utført våren 2014. De to studiene ble utført på samme måte, men på forskjellige prosjekter hvor begge brukte Scrum som utviklingsmetode. Selv om tema var likt var omfang, og særlig fokus på kundesiden, utvidet i hovedstudiet. Etter at forstudiet var avsluttet endte det med at den prosjektgruppen avviklet Scrum for en annen utviklingsmetode. Hovedpoenget med forstudiet var både å få innsikt i hvordan Scrum ble brukt og å få erfaring i å utføre informasjonsinnhenting i case studier. Rapporten har fokus på hovedstudiet, men drar også inn interessante aspekter fra forstudiet. Informasjonen som blir presentert i denne oppgaven er sterkt knyttet til den faktiske bruken av Scrum, men drar også inn andre faktorer som viste seg å ha en påvirkning for bruken av metoden. Basert på funnene presentert i rapporten kom jeg med forslag til forbedringer i Scrum, særlig med tanke på implementasjon og tidlig bruk av metoden. Håpet er at disse forbedringene skal gjøre at flere bedrifter og prosjekter tar i bruk Scrum på en god og forsvarlig måte, slik at metoden gir mest mulig tilbake til brukerne.

1.4 Oppbygning av rapporten

Denne avhandlingen er delt inn i fem kapitler. Første kapittel gir en introduksjon til temaet og problemstillingen, samt beskriver motivasjonen bak denne avhandlingen. Andre kapittel er et to-delt litteraturstudie, hvor første del beskriver den historiske utviklingen til systemutviklingsmetoder og hvordan Scrum i teorien skal brukes. Siste del av kapittel to beskriver eksterne forhold som potensielt kan påvirke bruken av utviklingsmetoder: kunderelasjoner, kunnskapsflyt og det å implementere en metode.

Tredje kapittel inneholder en beskrivelse av hvordan Scrum ble brukt i hovedstudiet utført våren 2014. Kapitlet begynner med en beskrivelse av metodene brukt for å innhente informasjon, før det fortsetter med bakgrunnsinformasjon om prosjektet og selve resultatene fra studiet. En analyse av disse resultatene blir presentert i kapittel fire, før en konklusjon blir trukket i siste kapittel sammen med forslag til videre studier.

Kapittel 2

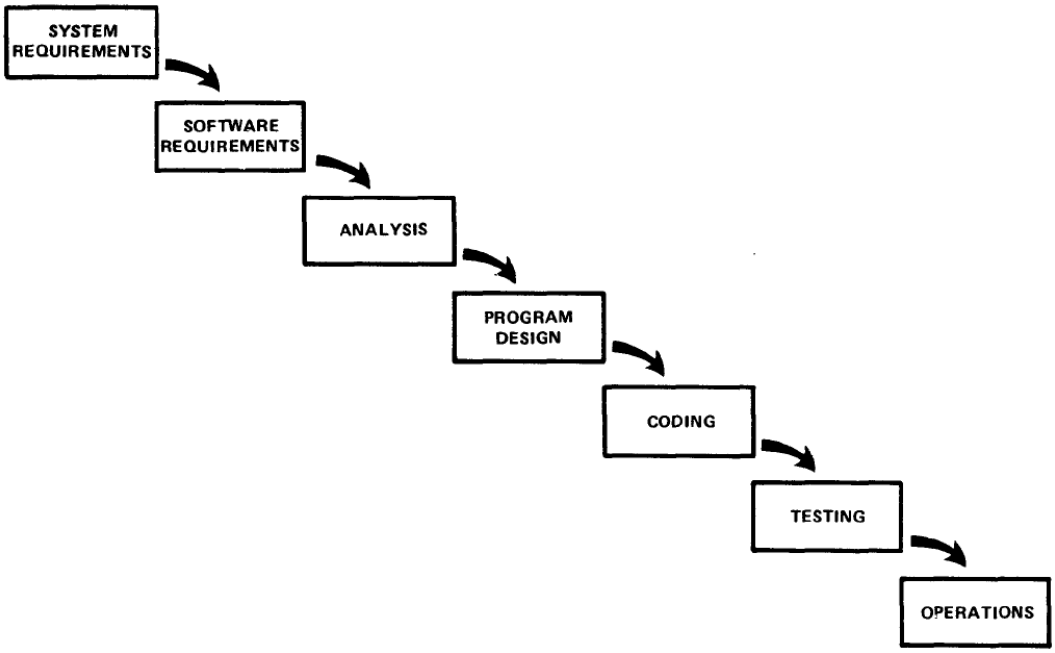
Litteraturstudie

2.1 Utviklingsmetoder

2.1.1 Historie

Fra den første programmerbare datamaskinen ble laget under andre verdenskrig har datamaskinen hatt en stor og kontinuerlig utvikling. I den første tiden ble programvare utviklet i to steg: *Analyse* og *Koding*, noe som fungerte veldig godt for små prosjekter. En studie av Winston Royce viste imidlertid at man var dømt til å feile hvis man holdt seg til disse to stegene i store prosjekter [38]. Dermed foreslo Royce å øke antall faser, slik at også større prosjekter kunne ha en mal å gå etter. Resultatet ble fossefallsmetoden, og en lang ferd mot de smidige systemutviklingsmetodene som vi kjenner i dag var i gang. Den opprinnelige fossefallsmetoden av Royce er vist i figur 2.1.

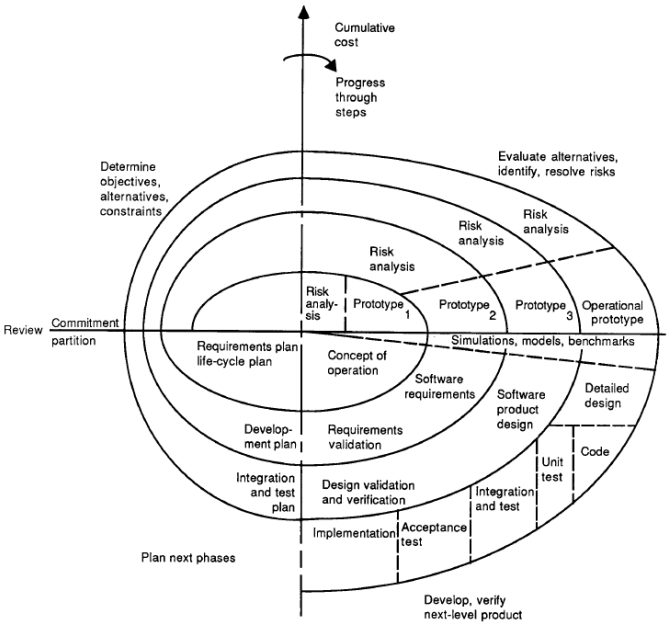
I motsetning til tidligere utvikling ble nå kundene, som gjerne var andre utviklere, involvert i prosessen og kravspesifikasjon ble formelt satt. Dette skulle også vise seg å introdusere nye problemer. I en rapport av Bell og Thayer [7] fra 1976 kommer det frem at store systemer har problemer i fasene *Kravspesifikasjon* og *Analyse* når de bruker fossefallsmetoden. Både eksterne og interne problemkilder identifiseres i artikkelen. Felles er at det store antallet deltakere og meninger, samt den lange tiden det tar å fullføre prosjektet, er roten til disse problemene. Dette førte til at man ønsket å redusere utviklingstiden. Man tok dermed i bruk programmeringsparadigmet *Structured programming* [13]. I en erfaringsrapport fra IBM [4] ble det konkludert at man ved å ta i bruk dette paradigmet reduserte utviklingstiden og økte kvaliteten på arbeidet.



Figur 2.1: Fossefallsmetoden til Royce [38, s. 2]

Etterhvert som større og mer kompliserte systemer ble etterspurt på 1980-tallet ble det klart at det opprinnelige fokuset på høy kvalitet, lav kostnad og differensiering ikke var nok til å utmerke seg i markedet. Dermed gikk bedriftene over til nye metoder som støttet fleksibilitet og lav utviklingstid. Disse metodene bygget på selvorganiserte team, overlappende utviklingsfaser og deling av kunnskap, noe som ga positive resultater [51]. I tillegg ble også *Spiral model* presentert som et nytt og bedre svar på de tradisjonelle metodene ved å løse problemene som oppstod på 80-tallet [10]. Denne modellen var risikodrevet med flere iterasjoner, se figur 2.2. Begge disse metodene var med å legge grunnlaget for smidige utviklingsmetoder.

Ved begynnelsen av 1990-tallet fortsatte utviklingsprosjekter og informasjonssystemer å øke i størrelse og kompleksitet. Blant annet fikk de som brukte *Spiral model* store problemer med å identifisere målsetninger og begrensinger til det nye systemet [9] siden mange av systemene hadde stadig flere endringer i kravspesifikasjon, funksjonalitet og innhold i løpet av utviklingstiden [19]. Som en løsning på disse problemene fikk smidige metoder se dagens lys. Essensielt er de smidige metodene forbedringer av, og utviklet som svar på, tidligere tradisjonelle metoder. Abrahamsson et al. [1] karakteriserer smidige metoder som inkrementelle, samarbeidsfokuserte,

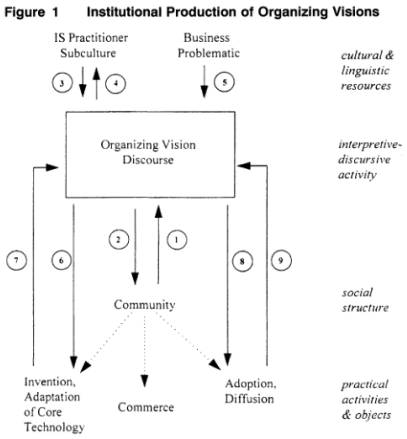


Figur 2.2: Boehm's Spiral model [10, s. 4]

fleksible og enkle å lære. Nettopp fleksibilitet er viktig i den forstand at forandring i planene vil oppstå og må dermed håndteres fortløpende [19, 23]. I siste halvdel av 1990 og de første årene etter 2000 ble en rekke smidige metoder presentert [16]: *Dynamic Systems Develop-ment Method* (DSDM), *eXtreme Programming* (XP), *Crystal*-metoder, *Feature Driven Development*, *Lean Development* og *Scrum*. Felles for disse er at de tar høyde for at endringer kommer til å skje, både i kravspesifikasjon og ønsker fra kunden. I begynnelsen ble metodene møtt med skepsis og blandet suksess [2, 21], men i dag er de smidige metodene (med *Scrum* i spissen) dominante som utviklingsmetoder [45, 54]. En mulig grunn til at *Scrum* utmerker seg er at den stadig har blitt utviklet videre [14, 36, 40, 42, 43] siden den ble introdusert av Ken Schwaber midt på 1990-tallet. Disse endringene var basert på erfaringer ved bruken av Scrum [12, 36, 40, 45, 48], med et ønske om å gjøre metoden bedre.

Scrum har utviklet seg til å bli brukt i mange forskjellige former. Relatert til 9 stegs modellen til Swanson og Ramiller [49] vist i figur 2.3 er Scrum nå i den 9. og siste fasen, *Diffusion*. Metoden har utviklet seg fra en ide i fase 1 til å komme i mange forskjellige former i fase 9. Alle disse forskjellige formene kom som et resultat av steg 8 *Adoption*, hvor man enten "mindful" (bevisst) eller "mindless" (ubevisst) [50] bestemte seg for å ta i bruk meto-

den på sin måte. Begge disse mulighetene kan ha gitt utslag til diffusjon og mange forskjellige retninger innenfor metoden. Ved mindful-diffusjon gikk man bevisst bort fra teori og hadde et mål ved å tilpasse Scrum til sin organisasjon. Mindless-diffusjon gjorde at man ubevisst gikk bort fra teorien, for eksempel som følge av at Scrum ble et rituale hvor prosesser gikk automatisk uten noe særlig med tanke bak valgene.



Figur 2.3: 9 stegs modellen til Swanson og Ramiller [49, s. 462]

Det å ta i bruk smidige metoder har i mange tilfeller vist seg å gi gode resultater. For eksempel klarte man ved å innføre Scrum i utviklingen av OSS (Open Source Software), å øke kommunikasjonen mellom utviklerne samt levere et produkt innen tidsfristen som var av høy kvalitet [29]. I dette tilfellet var det ikke mulig å innføre "ren" Scrum i prosjektet på grunn av geografiske, kulturelle og kommunikative problemer ved et OSS-prosjekt, derfor ble metoden tilpasset slik at den passet prosjektet best mulig. Slik er det også med mange av prosjektene i dag. Bedrifter og organisasjoner tar i bruk smidige metoder, tilpasser disse til sine egne prosjekter, og opplever gode resultater med dette [21, 39]. Om man spør om dette er riktig måte å bruke metodene på, vil svaret variere avhengig av hvem man spør. Til nå har man ikke klart å komme opp med en universell utviklingsmetode som passer alle bedrifter like bra. Frem til eventuelt en slik metode kommer, vil det være nødvendig å tilpasse de smidige metodene slik at organisasjonen får mest mulig tilbake ved å ta i bruk metoden. Med dette i bakhodet inviteres du som leser til resten av kapittelet, som inneholder informasjon om hvordan *Scrum* i teorien skal brukes.

2.1.2 Scrum

Siden Scrum er en utbredt metode er det mange meninger om hvordan Scrum i teorien bør brukes. I denne presentasjonen av Scrum er mye av informasjonen basert på tankene og ideene til to av de største bidragsyterne til metoden, Ken Schwaber og Jeff Sutherland. I tillegg er det tatt med elementer fra andre bidragsytere som for eksempel har mer detaljerte meninger angående spesifikke sider ved bruken av Scrum.

Roller

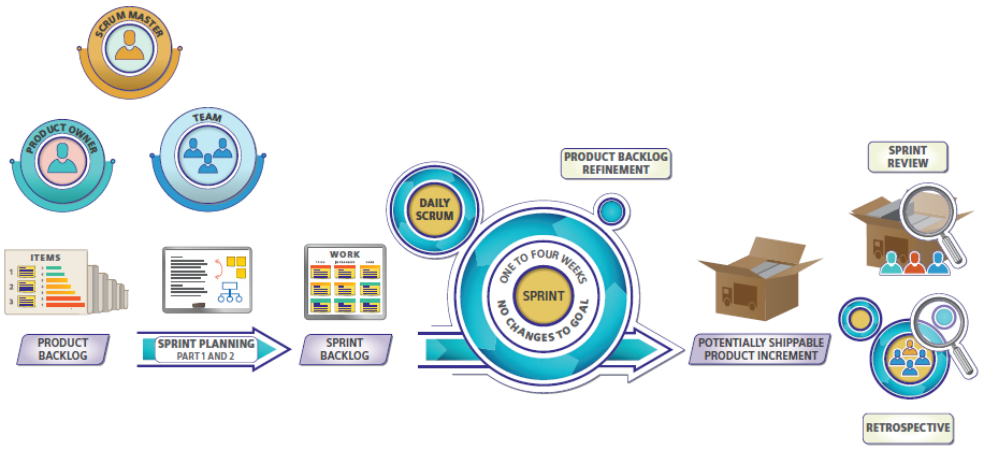
I *Scrum* opererer man med tre forskjellige roller [42]: produkteier (heretter PO), Scrummaster (SM) og teamet. PO er den som er ansvarlig for å representere de eksterne kildene som har interesse i prosjektet og dets resultat. Man kan si at PO er kundens representant i prosjektet. SM er ansvarlig for å sørge for at teamet følger Scrumprosessen. SM lærer bort hvordan Scrum skal brukes til alle involvert i et prosjekt, og sørge for at Scrum implementeres på en god måte som støtter organisasjonens kultur i tillegg til å levere de forventede fordelene med metoden. Teamet er ansvarlig for å utvikle funksjonaliteten og har ingen utpekt leder. Teamet har mye frihet og ansvar ved at det administrerer og organiserer seg selv. Kollektivt er medlemmene i teamet ansvarlig for prosjektet som helhet og for suksess i de mange iterasjonene som brukes i Scrum. Anbefalt størrelse på teamet er 3-9 medlemmer [42]. Ved bruk av Scrum er det viktig at alle medlemmene i prosjektet er enige om å bruke metoden fullt og helt. Når hele teamet er dedikerte og komfortable med det, vil utviklingen gå fort [12].

Erfaringer viser at selv om de fleste team klarer å takle mange problemer, er det viktig med en god SM som sørger for å holde teamet på rett spor [36]. SM må da holde seg til rollen sin ved å veilede og støtte medlemmene i teamet, slik at teamet får utført arbeidet [32]. Når det gjelder PO er denne rollen mer kontroversiell, og det er studier som argumenterer for at rollen som PO burde bli fjernet og la teamet ta denne jobben selv. Ved å gjøre dette vil utviklerne oppmuntres til å sette seg inn i kundens forretningsdomene og få en større følelse av eierskap for produktet, noe som kan føre til et bedre sluttresultat [8].

Prosess

Som flere andre smidige metoder er Scrum en risikodrevet, iterativ og inkrementell prosess: metoden utføres i sirkler, hvor produktet bygges litt etter litt for å minske risikoen knyttet til prosjektet. Scrum er basert på ideer fra Boehm's *Spiral model* og Takeuchi's bok *"The new new product develop-*

ment game" [2, 36]. I motsetning til mange andre smidige utviklingsmetoder har Scrum fokus på håndtering av selve utviklingsprosjektet i stedet for å foreslå fremgangsmåter for programvareutvikling. Det er slik at utviklerne selv bestemmer hvilke teknikker, metoder og fremgangsmåter som skal brukes i implementasjonsprosessen [2]. Dette gjør at Scrum lett kan brukes sammen med andre smidige metoder [41], som for eksempel *XP*. Figur 2.4 viser overordnet hvordan Scrum brukes i praksis, hvor man i en *sprint* selv kan bestemme hvordan selve utviklingen av programvare skal gjøres.



Figur 2.4: Prosessen i Scrum, hentet fra The Scrum Primer [14, s. 3]

Visjon

Selve utviklingsmetoden bygger på at kunden har en visjon (et mål) med det fremtidige produktet. Basert på denne visjonen har man noen tanker om hvordan produktet skal være og når man ønsker det utlevert. I Scrum skjer utleveringen inkrementelt med leveranse etter hver sprint er avsluttet. Dette gjør det lettere for teamet og kunden å se om produktet er i samsvar med det som ble spesifisert og dermed minsker risikoen for å levere et produkt som ikke tilfredsstiller kundens ønsker og behov. Visjonen blir da brutt ned i konkrete krav som stilles til produktet. I samarbeid med interessentene vil PO samle og legge disse (i prioritert rekkefølge) inn i en *product backlog*. Dette vil fungere som en spesifikasjon for hva interessentene ønsker av produktet [40].

Product backlog

En *product backlog* er et artifakt som hele tiden vil være oppdatert til å reflektere hva kunden ønsker ut av det ferdige produktet. I Scrum fungerer product backlog som en kravspesifikasjon som beskriver det kommende

produktet. Siden kravene til produktet ofte er i stadig forandring [43], må artifaktet til en hver tid være oppdatert til å gjenspeile disse kravene. Typisk sett samarbeider PO og utviklingsteamet med å oppdatere prioritet, samt legge til detaljer og estimater på oppgaver i product backlog. I figur 2.4 identifiseres dette som *product backlog refinement*. Oppgavene som ligger i en product backlog vil ligge i prioritert rekkefølge hvor det ofte er slik at høyt prioriterte oppgaver er veldefinerte og konkrete, mens de med lav prioritet ofte er vage og vanskeligere å definere. Man har dermed fokus på de meste prioriterte oppgavene, de oppgavene som skal utføres først. Selv om teamet ofte blir involvert i prosessen med å finjustere product backlog kan også PO når som helst oppdatere oppgaver som ligger i product backlog, uten å innvolvere teamet [44]. PO jobber tett med kunden, og vet best hvilke oppgaver som bør prioriteres og hva som må gjøres for å gjøre kunden fornøyd.

Sprint planning

Før hver *sprint* vil PO, SM og teamet samles for å holde et planleggingsmøte (*sprint planning*) hvor de bestemmer hvordan neste sprint skal være. Dette møtet er delt i to hvor hver del er satt til maksimalt 1 time per uke i sprinten [14]. Den første delen brukes til å sette mål for sprinten og velge hvilke deler av product backlog som vil bidra til dette målet. I den andre delen av planleggingsmøtet er det teamet som sammen bestemmer hvordan de prioriterte delene fra product backlog skal realiseres for å nå målet[1]. Dette resulterer i spesifikke oppgaver som skal fullføres i neste sprint. Alle disse oppgavene blir estimert (i forhold til planlagt tidsbruk) og satt inn i *sprint backlog*. Ut fra disse estimatene ser teamet om de eventuelt har for mye/lite å gjøre. I denne andre delen trenger ikke PO være til stede, men må være tilgjengelig for spørsmål [14] for eksempel hvis teamet ønsker å ta ut noen oppgaver fra sprinten.

Sprint backlog

En *sprint backlog* inneholder spesifikke oppgaver som teamet har som mål å fullføre i løpet av en sprint. Dette artifaktet synliggjør alt som må gjøres for å møte målene som teamet har satt, og viser hvordan man ligger ann i forhold til målene. Også dette artifaktet blir kontinuerlig oppdatert, slik at teamet vet hvilke oppgaver som er gjort, hvilke oppgaver som det nå jobbes med, og hvem som er ansvarlig for oppgavene. I motsetning til i product backlog har ikke PO noen mulighet for å gjøre endringer i sprint backlog: oppdateringer gjøres utelukkende av teamet selv. Siden en sprint backlog skal gi et nåværende bilde av hvordan man ligger ann i sprinten skal det

[1]Dette produktet er ofte en utvidelse av leveransen i forrige sprint

være lett å finne ut hvor mye arbeid som står igjen. En måte å se dette på er å lage et *burndown chart*, som viser grafisk hvordan man ligger ann i forhold til gjenstående arbeid i sprinten. Dette tallet skal inspiseres daglig for å se om man kan oppnå sprintmålet [44].

Sprint

Når planleggingsfasen er over vil prosjektet gå inn i en *sprint*, hvor selve utviklingsarbeidet utføres. En sprint ble opprinnelig foreslått å vare i 30 dager [40], men senere har flere argumentert for en moderasjon til opp til 1 måned (men gjerne under) [14, 36, 43]. Hver sprint skal resultere i et potensielt leverbart produkt for kunden. I løpet av en sprint jobber teamet selvstendig og bestemmer i stor grad selv arbeidsprosedyren, uten direkte påvirkning fra eksterne kilder. Et krav i sprinten er at teamet hver dag skal møtes for å oppdatere hverandre. Disse møtene, ofte kalt *standups*, skal holdes først på dagen og alle i teamet skal være med. Under møtet skal hvert teammedlem svare på tre spørsmål:

- Hva har du gjort siden forrige standup for å hjelpe teamet med å nå sprintmålet?
- Hva skal du gjøre frem til neste standup for å hjelpe teamet med å nå sprintmålet?
- Er det noen forstyrrelser som gjør at du og teamet kan få problemer med å nå sprintmålet?

Det er SM som er ansvarlig for at dette møtet blir holdt, foregår så effektivt som mulig og passe på at møtet ikke varer mer enn 15 minutter, uavhengig av teamstørrelse [43]. Derfor er det viktig at bare en person snakker om gangen, at man holder seg til disse tre spørsmålene, og at diskusjoner utsettes til etter møtet. Opprinnelig var standup åpen for folk utenfor teamet (blant annet PO) [40], men har senere blitt endret til at bare SM og teamet skal delta [43]. Denne presiseringen kom for å underbygge at det er teamet selv som bestemmer hvordan de jobber i sprinten.

Sprint evaluering

Etter hver endt sprint skal sprinten evalueres. Dette gjøres i to deler, både utad mot interessenter og innad i teamet. Den første delen er *sprint review*. Denne delen er foreslått til å vare 1 time per uke av sprinten [14]. Møtet brukes for at teamet skal presentere det som er gjort i løpet av sprinten, for PO og andre interessenter. Dette skal presenteres i kontekst av sprintmålet

og de delene av product backlog som var planlagt inkludert. Mesteparten av tiden går med til å presentere funksjonalitet og svare på spørsmål fra interessentene. Basert på dette møtet kan endringer på product backlog diskuteres og inkluderes i neste sprint. I dette møtet kan interessentene snakke fritt, med kommentarer og spørsmål til teamet.

Den andre delen av evalueringen, *Sprint retrospective*, er et møte mellom teamet og SM (PO kan være med om ønskelig). Lengden på møtet foreslås til 45 min per uke i sprinten [14]. Dette møtet har som formål å forbedre scrumprosessen innad i teamet for neste sprint ved at teamet identifiserer både ting som gikk bra, og ting som kan gjøres bedre neste gang. Deretter skal teamet bli enige om hva de kan gjøre for å implementere forbedringene i arbeidsdagen [43]. Dette møtet er en viktig arena hvor teamet selv kan evaluere arbeidet sitt, og oppfordres til å komme med forslag til forbedringer:

> "Retrospectives that don't result in change are sterile and frustrating" - Ken Schwaber [40, s. 139]

.

2.2 Kunderollen

I programvareutvikling har begrepet *kunde* gått gjennom noen endringer de siste tiårene. På tiden før fossefallsmetoden ble presentert var det ofte slik at de fremtidige brukerne av en programvare var andre utviklere [38], og det var disse som på ulikt vis ble inkludert i utviklingsfasen [22]. Man kan si at disse brukerne hadde samme rollen som *kunden* vi kjenner i dag. En stor forskjell er at kunden i dag har mindre teknisk kunnskap enn de tidligere brukerne. Dette kommer av at kundene i dag ofte ikke er utviklere: det er noen fra ledelsen som representerer kunden, med et godt øye til økonomisk gevinst og mindre teknisk kunnskap. I dag gjør dette at de som designer og utvikler systemet bruker kunden som en kilde til informasjon, som blir inkludert i fasene hvor *"[...] de negative konsekvensene av deres krav er så små som mulig"* [22, s. 14]. Denne involveringen er mer lik involveringen av *brukerne* i tidligere utviklingsmetoder enn den store *kunde*involveringen som smidige metoder promoterer.

Etterhvert har kundene blitt en stadig større del av programvareutviklingen. Man er avhengig av god kontakt med kunden både for å kunne skaffe seg prosjektet og å overbevise kunden om at man kan levere det ønskede produktet. Tradisjonelt sett ble kundene i stor grad involvert i begynnelsen av prosjektet før man startet selve utviklingen. Etterhvert som systemene vokste i størrelse viste det seg å være vanskelig å skjønne hva kunden *egentlig* ønsket. Et problem var at ferdigstilte produkter kunne vise seg å ikke tilfredsstille kundens behov.

Smidige metoder har et større fokus på samarbeid med kunden enn tidligere metoder. Kunden blir invitert til møter også i utviklingsfasene av løsningene for å forsikre seg at det ferdigstilte produktet er tilfredsstillende. I disse møtene har det vist seg at kundene ofte kommer med andre ønsker og kravspesifikasjoner enn de som ble identifisert i første omgang. Scrum løser dette ved å introdusere en egen rolle som bindeledd mellom kunde og utvikler. Ved å ha en PO som hele tiden representerer kunden mot utviklingsteamet vil man sørge for en kontinuerlig dialog mellom partene. Også bruken av *product backlog* er med å fremme kundens krav og ønsker. Det som kan være vanskelig med PO-rollen i Scrum er at kunden skal representeres gjennom en person, men kunden den representerer vil ofte være en større gruppe med ulike ønsker og meninger. Hvordan bestemmer PO hvilke ønsker som blir formidlet videre, som representerer hva kunden ønsker? Rapporter viser at dette kan gjøres gjennom demokratiske ideer, hvor alle får komme med sine meninger [22], men også dette har vist seg å kunne gi villedende resultater

[55]. Problemer har lett for å oppstå med store kunder, hvor kundens representanter mot PO ikke er de fremtidige brukerne av det ferdigstilte produktet. Om ikke kundens representanter har en god kontakt med de fremtidige brukerne av systemet kan man risikere at produktet ikke blir tatt i bruk selv om resultatet i utgangspunktet er godt i kunderepresentantenes øyne. Tross alt er det de fremtidige brukerne av systemet som bestemmer om produktet blir en suksess eller en fiasko.

Det finnes flere studier om at kunden spiller en svært viktig rolle i programvareutvikling. I et studie utført av Hoda et al. [24] kom det fram at dårlig kommunikasjon eller mangel på dedikasjon til prosjektet fra kundens side, kunne skape problemer med kravspesifikasjon og føre til mangel på produktivitet hos utviklingsbedriften. Også Korkala et al. [28] pekte på denne trenden gjennom et studie på utfordringer ved kundekommunikasjon. Gjennom dette studiet kom det frem at det også er andre utfordringer når det gjelder kommunikasjon med kunden, blant annet knyttet til hemmeligstemplet informasjon. Siden kunden ikke stolte fullt og helt på utviklingsbedriften holdt kunden bevisst tilbake informasjon, noe som gjorde at utviklingsarbeidet ble mer komplisert enn det unne ha blitt. Et annet problem som ble identifisert var at kunden var vant til å bruke en tradisjonell tilnærming når det gjaldt programvareutvikling, og dermed ikke hadde nok kunnskap om deres rolle ved bruk av smidige metoder. Dette gjorde at kunden ikke var like mye involvert i implementasjonsstadiet som utviklingsbedriften hadde ønsket og trengt. Nettopp kommunikasjon og samarbeid er nøkkelord for å effektivt kunne utføre smidig utvikling [6, 11].

Det er en utfordring i å få kunden til å skjønne hvor viktig det er med kommunikasjon mot utviklingsbedriften når man bruker smidige metoder. Mange kunder er vant til tradisjonelle metoder, og forstår ikke helt viktigheten av kunnskapsflyten mellom dem og utviklingsteamet. Det blir da opp til scrummasteren å sørge for at alle involverte i prosjektet (inkludert kunden) forstår hvordan Scrum skal brukes og hvorfor [40].

2.3 Kunnskapsflyt

Kommunikasjon og samarbeid, for eksempel mellom kunden og utviklingsteamet, er viktig for utvikling [11]. I lang tid har det vært vanskelig for den tradisjonelle kunden å ha en god kommunikasjon mot utviklerne [38], i stor grad på grunn av deres forskjell i bakgrunn, miljø og tekniske nivå. Scrum forsøker å forenkle denne kommunikasjonen gjennom å legge bedre til rette for kunnskapsflyt. Denne kunnskapsflyten realiseres i stor grad ved hjelp av PO-rollen og den stadig endrede product backlog, men også i evalueringsmøtene etter en sprint (sprint review). Det er derfor viktig at kunden kommer på disse møtene og stiller spørsmål direkte til utviklerne, slik at begge parter er enige om hva som eventuelt mangler og må gjøres i neste sprint. Tross alt er det kunden som betaler for produktet. Siden PO spiller en stor rolle i et utviklingsprosjekt er det viktig at kunnskapsflyten både mellom PO og kunde, og mellom PO og utviklingsteam er god.

Det er også viktig å tenke på kunnskapsflyten innad i utviklingsteamet. I Scrum bestemmer man selv utviklingsmetodene i en sprint. Det er for eksempel mulig å ta i bruk *pair programming* fra XP. Dette gjør at graden av kunnskapsflyt og deling kan variere noe avhengig av hvordan utviklingen gjøres. Det er blant annet vist at god kunnskapsflyt gjennom pair programming også kan bidra til å redusere feil i koden [47]. Scrum har stort fokus på samarbeid, også innad i teamet, og legger til rette for god kunnskapsflyt blant annet ved bruken av daglige møter (standups). Det er lettere å spørre hverandre om hjelp i Scrum enn i tradisjonelle plandrevne metoder [30]. I tillegg har man i forbindelse med product og sprint backlog bruk av tavler, lister og figurer som koordineringsmekanismer i utviklingsarbeidet. Disse finnes både i fysisk og digital form, hvor prosjektgruppa selv velger det alternativet som de foretrekker. Disse koordineringsmekanismene har som mål å gjøre det enklere for prosjektgruppa å fordele oppgaver samt vise hva de ulike oppgavene går ut på, i tillegg til å legge til rette for kunnskapsflyt innad i prosjektgruppa.

I begge disse typene med kunnskapsflyt er det et poeng at ikke alle trenger å vite alt, men nok til at oppgavene kan utføres uten problemer. Ved kunnskapsflyt mellom to individer vil individenes evne til å prosessere kunnskapen på en slik måte at den gir mening, bestemme om kunnskapen som deles er god eller ikke. Her kommer også taus og eksplisitt kunnskap inn i bildet [56]. Eksplisitt kunnskap er kunnskap som enkelt kan forklares, kategoriseres og representeres ved tall og logikk, og er relativt lett å gi mening til. Taus kunnskap er kunnskap det er veldig vanskelig å forklare og representere,

som for eksempel kunnskap om hvordan man skal sykle på en sykkel. Det er denne tause kunnskapen som i størst grad vil kunne gjøre kunnskapsdeling vanskelig og forstyrre kunnskapsflyten. Siden likhet i kultur, bakgrunn og erfaringer gjør deling av både taus og eksplisitt kunnskap enklere [56], er det nærliggende å tro at kunnskapsdeling innad i teamet er lettere enn for eksempel ut mot en kunde. Scrum har introdusert rollen som PO for å gjøre kunnskapsflyten mellom kunde og utvikler enklere. Dette kan realiseres siden individet med denne rollen har god innsikt i begge miljøer, og kan fungere som en tolk mellom miljøene. Som i de fleste andre situasjoner er man avhengig av at individet holder seg til rollen og dens oppgaver for å få fullt nytte av fordelene.

2.4 Implementasjon

Hva vil det si å *implementere*, eller ta i bruk, en smidig metode? Robey og Sahay [37] identifiserer fire spesifikke faser relatert til implementasjon av *informasjonssystemer*:

- Initiering
- Overgang
- Utrulling
- Spredning av kunnskap

Det er nærliggende å overføre disse fire fasene til implementasjon av nye utviklingsmetoder. Et initiativ må til for å spre ideen om smidige metoder. Uten en initiativtaker som setter i gang tankeprosessen vil implementasjon være utenfor rekkevidde. Ved bytte av utviklingsmetoder vil også en overgang være nødvendig, og denne må planlegges godt etter hvor man er i løpet i sin nåværende utviklingsmetode. En vurdering må gjøres om byttet kan introduseres gradvis for å lette overgangen [12]. Utrullingen av den smidige metoden vil være når gruppen tar i bruk den nye metoden i praksis, og henger tett sammen med overgangsfasen. Om overgangen er godt planlagt vil selve utrullingen bli enklere å gjennomføre, og hele teamet er klar for forandringen. Til slutt kommer fasen om spredning av kunnskap. Ved implementasjon av smidige metoder er dette kunnskapsdelingen innad i gruppa: hvordan metoden skal og bør utføres i praksis. Dette gjøres for at alle i gruppa skal være på samme bølgelengde og vite hva som forventes av dem. Til sammen vil disse fire fasene i stor grad bestemme hvor god implementasjonen av en smidig metode vil være.

Til syvende og sist er det opp til bedriften å utføre implementasjonen på en helhjertet og god måte, slik at den smidige metoden vil kunne gi positive resultater. En innovasjon som innføres basert på analyser og gode tanker bak valgene man gjør blir kalt for "Mindful innovation", mens innovasjoner som blir innført uten særlig med tanke på hvorfor man tar den i bruk blir kalt "Mindless innovation" [50]. Et viktig moment med mindfulness er at det kan skape motstand fra å hoppe på og ta i bruk nye innovasjoner uten å tenke over hva det vil gjøre med bedriften [50]. Det er viktig at alle involverte parter, ikke bare utviklerne, er klare og går inn for å ta i bruk den nye metoden. Om implementasjonen ikke gjøres helhjertet kan resultatet bli påvirket i negativ grad [12, 34, 37]. Det er verdt å påpeke at selv om implementeringen gjøres helhjertet, kan implementeringen gjøres i forskjellig grad. For eksempel kan daglige morgenmøter i Scrum ta en time og holdes med ledelsen og

prosjektgruppa, eller den kan ta 10. min og holdes innad i utviklergruppa. Erfaringer viser at det er viktig at bedriften implementerer smidige metoder slik at de passer best mulig til sitt formål [21, 31]. I tillegg er det viktig å tenke på at rutinene og kravene til en bedrift kan endre seg med tiden, slik at endringer på utviklingsmetoden kan være nødvendig også etter en vellykket implementasjon. Dette er viktig for å hindre at prosesser og rutiner innad i bedriften blir utdaterte og utføres som ritualer uten noen mening bak de [35].

Kapittel 3

Case studie: 14C

3.1 Metode

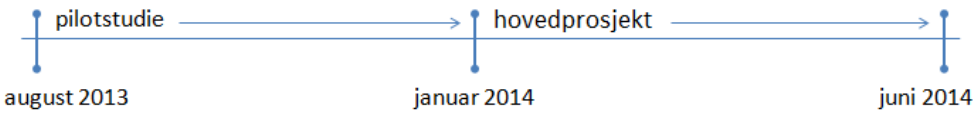
Tilnærming

I denne avhandlingen lå fokuset på å forstå og analysere arbeid i forbindelse med smidige metoder på et lokalt nivå. Et prosjekt ble undersøkt hvor det var viktig å innhente mye forskjellig type informasjon, slik at valg og handlinger kunne forklares i forhold til hverandre. Det var viktig å finne underliggende årsaker og motivasjonen bak valgene som ble gjort. Et kvantitativt studie ville ikke klart å plukke opp alle disse tingene, det ville ha kunnet innhentet målbare data uten å se på hva som ligger bak. Et kvalitativt studie ble utført i forbindelse med denne avhandlingen for å få en dypere forståelse enn man hadde gjort med et kvantitativt studie.

14C-prosjektet ble valgt både fordi det var et "typisk" Scrum-prosjekt og at jeg hadde kontakter som gjorde det enklere å selge seg inn til teamet. Prosjektet ble sett på som NTNUs flaggskip når det gjaldt bruken av Scrum, og hele teamet var klar på å bruke Scrum med de roller og aktiviteter dette innebærer. Dermed ble studiet utført i en naturlig og virkelig setting.

Før studiet av prosjektet startet, ble et pilotstudie utført på et annet prosjekt hvor de samme datainnsamlingsmetodene ble brukt (figur 3.1). Dette pilotstudiet ble brukt for å lære essensen bak de smidige metodene og for å lære metodikken rundt innhenting av data. Dette gikk både på å lære nyttige måter å innhente data, men også for å lære å forholde seg til gruppen som ble observert. Dette pilotstudiet pågikk fra august 2013 til desember 2013 hvor hensikten var å skaffe forskningserfaring og bli forberedt til hovedprosjektet som startet etterpå. Ved å få noe erfaring fra pilotstudiet ville det

bli lettere å utføre studiet på hovedprosjektet. Både med tanke på tid, men også med tanke på nyttige tips og triks som gjorde det enklere å innhente informasjonen. Pilotstudiet var også nyttig på den måten at det ga ideer til hva som kunne være lurt å undersøke nærmere. Det fungerte som "grounded theory", hvor man i løpet av pilotstudiet kom opp med interessante teorier som kunne utforskes videre i hovedstudiet.



Figur 3.1: Tidslinje for pilotstudiet og hovedprosjektet

I løpet av pilotstudiet ble mye tid brukt på å lese litteratur angående både forskningsmetoder og smidig utvikling. Siden dette var mitt første forskningsarbeid tok det et par uker før forskningen kom skikkelig i gang. De første ukene gikk mye tid med på å prøve ut metoder for innhenting av informasjon, men det viste seg at flere av metodene som ble forsøkt utført ikke ga gode resultater. Disse metodene ble dermed endret eller fjernet til hovedstudiet begynte. Dette gjaldt for eksempel bruken av notatbok i observasjonen. I pilotstudiet var notatboka med i alle observasjoner, mens notatboka i stor grad ble benyttet etter observasjonene i hovedstudiet. Dette ble gjort for å unngå å skremme observasjonsobjektene ved å ikke stikke meg ut som en forsker.

I pilotstudiet var det også første gang at jeg fikk innsikt i hvordan man i praksis jobbet med smidige metoder som Scrum. Dette gjorde at det tok noe tid før jeg kjente igjen kjente Scrum-prosesser og kunne begynne å sammenligne disse med den teoretiske bakgrunnen jeg hadde om Scrum. Det jeg lærte mest om angående forskningsmetoder i løpet av dette pilotstudiet var at selv om uformelle intervjuer var gode for å få mye informasjon om et tema, var også mer formelle spørsmål viktig for å få svar på detaljer innenfor disse temaene. Når det gjaldt bruken av smidige metoder ble jeg overrasket over at disse ikke ble brukt akkurat som teorien sa de skulle brukes. I pilotstudiet var det svært stort fokus på å bli ferdig så fort som mulig med møter som Standup og det var dermed ikke så stort fokus på det som ble sagt i disse møtene. Et godt eksempel var da et teammedlem kommenterte at "Nå tok standup bare 5 minutter, nå begynner vi å bli flinke!" Med tanke på den relativt korte observeringstiden i hovedprosjektet var pilotstudiet essensielt for å lære seg metodikk og bruken av Scrum, slik at tiden som ble brukt i hovedprosjektet ble brukt så godt som mulig. Også teoriene som ble op-

parbeidet i løpet av pilotstudiet kom til nytte ved at disse teoriene dannet utgangspunkt for forskningsspørsmålene i dette studiet.

Datainnsamling

Datainnsamlingsmetodene brukt var de samme både i pilotstudiet og i hovedstudiet, dog med forskjellig mengde av data i etterkant av studiene. Det ble utført et kvalitativt studie på fire uker i prosjektsettingen, med start 28. januar 2014. Ved første øyekast kan dette virke noe kort, men med en Sprint på to uker ble studiet utført over to hele sprinter. Dette ga et større grunnlag for generalisering av prosjektaktivitetene. *Case study* ble valgt som metode for studiet. Dette ble gjort for å kunne innhente kompleks informasjon, hvor prosjektet som helhet ble studert i stedet for bare å fokusere på enkeltfaktorer. Med et case study har man mulighet til å gå inn i situasjonen, og og gjøre undersøkelser fra innsiden av prosjektet. Typen case study som ble brukt i dette studiet var *explanatory case study* for å se på ulike underliggende faktorer som forklarte bruken av Scrum i prosjektet. Studiet hadde størst fokus på nåværende situasjoner og hendelser, men det ble også hentet informasjon angående forhistorien til prosjektet for å skape et mer komplett bilde av situasjonen.

I den tiden studiet pågikk ble jeg tildelt en arbeidsplass på det ene av to rom som prosjektteamet var delt på. Dette gjorde at det ble nødvendig å jevnlig bevege seg over i det andre rommet for å innhente informasjon fra hele teamet. Hovedmetodene brukt for innhenting av informasjon var observasjoner og ustrukturerte intervjuer. Disse metodene passet godt med et case study siden også disse har fokus på å skaffe så mye informasjon som mulig om instansene. Observasjonen ble utført som *complete observer*: bare observasjon, uten å ta aktiv del i prosjektet [33]. Med denne type observasjon var prosjektteamet klar over at de ble observert og, til en viss grad, hvorfor de ble observert. Teamet fikk vite at observasjonen gikk på bruken av Scrum, men fikk ikke noe informasjon om hva jeg ønsket å finne ut. Denne informasjonen ble holdt tilbake for at teamet skulle oppføre seg så naturlig som mulig.

Mange ustrukturerte intervjuer ble brukt i prosjektet, ved at et tema ble introdusert og intervjuobjektene fikk snakke fritt ut fra det. Dette ble gjort for å innhente masse informasjon som senere kunne analyseres. Disse ustrukturerte intervjuene ble utført som en uformell samtale i pauser og ved andre naturlige situasjoner. I tillegg ble alle personer tilknyttet prosjektet (PO, SM, teamet og kunderepresentanten) intervjuet i en mer formell setting.

Semi-strukturerte intervjuer på 15 minutter ble utført på tomannshånd på et møterom. Disse intervjuene var semi-strukturerte i den grad at jeg som observatør hadde noen forhåndsbestemte temaer som intervjuobjektene ble presentert for og som deretter fikk snakke fritt innen disse temaene. Dette ble gjort for å skaffe mer informasjon innen temaer det hadde vært vanskelig å få informasjon om tidligere, eller for å se om teamet hadde forskjellige meninger og oppfatninger innen ulike temaer. Intervjuspørsmålene varierte ut fra hvilken rolle intervjuobjektet hadde i prosjektet, og en fullstendig liste over disse spørsmålene finnes i vedlegg B. En oversikt over datainnsamlingen som ble gjort i forbindelse med studiet er vist i figur 3.1.

Type	**Varighet / antall**
Case study	4 kalenderuker
Observasjoner	19 dager
Semi-strukturert intevju	9
Ustukturert intervju	Daglig
Notater	88 håndskrevne A4-sider

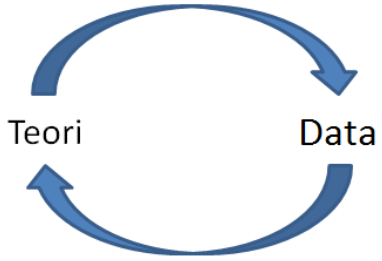
Tabell 3.1: Datainnsamling som ble utført i prosjektsettingen

På dette prosjektet startet jeg med blanke ark: jeg visste ingenting om prosjektet, bortsett fra at de brukte Scrum. Etterhvert som mer tid ble tilbrakt i prosjektet ble det stadig klarere både hva prosjektet gikk ut på og hvordan de praktiserte Scrum. Mye tid ble derfor brukt på å gå frem og tilbake i notater for å få et klarere bilde over hendelsene som oppsto. Siden dette studiet gikk over 4 uker er ikke hver minste detalj inkludert i denne rapporten. Temaene i denne avhandlingen er tatt med fordi de på en eller annen måte er interessante i forhold til Scrum-prosessen. Dette kan være hendelser som er forskjellig fra hvordan Scrum i teorien skal brukes, eller det kan være bekreftelser på hvordan Scrum virker. Felles er at de åpner for analyse og diskusjon ved bruken av Scrum.

Dataanalyse

Analyse av dataen som ble innhentet gjennom case studiet begynte med en gang case studiet begynte, og forsatte i flere måneder etter case studiets slutt. Dette gjaldt både for pilotstudiet og hovedstudiet. Når det gjelder pilotstudiet hadde jeg som forsker en forutintatt mening om at Scrum kun hadde positive sider. I løpet av pilotstudiet og hovedstudiet ble synet mitt gradvis farget både av prosjektmedlemmenes tanker og mine egne observasjoner. Ved slutten av hovedstudiet hadde jeg blitt mer kritisk til Scrum,

noe som gjorde at mitt generelle syn på metoden var annerledes enn ved begynnelsen av observasjonene. Jeg beveget meg stadig frem og tilbake mellom teori og observasjoner for å minske innvirkningen mitt syn hadde på observasjonen (figur 3.2).



Figur 3.2: Dataanalyse

Siden mitt syn på metoden ble gradvis forandret gikk jeg også frem og tilbake på notater for å få nye innspill og ideer fra tidligere stadier som hadde modnet og dermed var lettere å se og analysere. Å gå tilbake på tidligere notater var helt essensielt siden jeg i begynnelsen var relativt kunnskapsløs angående Scrum, og trengte mer teoretisk bakgrunnsstoff for å prosessere de innsamlede dataene. For eksempel hadde jeg i pilotstudiet mye teoretisk kunnskap om hvordan Scrum skulle brukes, men dette var første gang jeg så Scrum bli brukt i praksis. Dette gjorde at jeg mange ganger måtte gå tilbake på de tidligere observasjonene mine og gjøre nye analyser av disse ettersom jeg opparbeidet meg mer kunnskap. Dette gjorde det for eksempel lettere for meg å forstå at verktøyet Jira ble brukt til både product backlog og sprint backlog. Et annet eksempel var bruken av "groomingmøter" under pilotstudiet. Der gikk SM og PO sammen for å gå gjennom product backlog og "forberede den til neste sprint". Under det andre av disse "groomingmøtene" fant jeg ut at dette egentlig var et møte om å omprioritere og utdype oppgavene i product backlog, og ikke en forplanlegging av neste sprint som jeg fikk inntrykket av under første møte. Derfor gikk jeg tilbake på notatene mine fra første groomingmøte, og med ny kunnskap så jeg at også det møtet hadde gått på omprioritering og utdyping av product backlog.

Refleksjoner

For å vise på et dypere nivå hvordan dette studiet ble gjennomført evalueres studiet etter syv prinsipper presentert av Klein og Myers [27]. I dette delkapittelet har hvert av de syv prinsippene fått sitt eget avsnitt, slik at det er lett å finne igjen de ulike prinsippene.

1 *Hermeneutisk sirkel*: I dette studiet startet jeg uten forkunnskap og bygget opp forståelse etterhvert som studiet pågikk. Jeg gikk jevnlig tilbake til tidligere notater og funn med en nyfunnet forståelse som førte til at jeg fikk en ny forståelse av situasjonene og klarte å se dem i en større sammenheng. Det ble vekslet mellom å se på enkelthendelser og koble dem mot en større sammenheng. Dette førte til en større forståelse etterhvert som studiet pågikk, og endringer ble gjort på rapporten for å passe med den nye forståelsen.

2 *Kontekstualisering*: Senere i kapittelet blir det gitt en bakgrunn for prosjektet, for å sikre at leseren har samme utgangspunkt som jeg hadde da studiet ble utført. Ved å gjøre dette håper jeg å gjøre det lettere for leseren å se hvorfor jeg har trukket de konklusjonene jeg har gjort. Alle mennesker tar i mot og prosesserer informasjon på ulike måter som gjør at leserne kan være uenige i enkelte slutninger jeg trekker, men jeg håper at kontekstualiseringen av prosjektet vil gjøre det klarere hvorfor jeg har trukket de ulike slutningene.

3 *Interaksjon mellom forsker og subjekt*: Forståelsen av prosjektet ble økt gjennom observasjoner, intervjuer og diskusjoner i pauser. For å sikre at resultatene var troverdige og for å oppklare eventuelle feiltolkninger ble en runde med tilbakemeldinger utført. Her fikk både kunderepresentant og medlemmer i teamet mulighet til å si hva de mente om mine observasjoner. Noe av denne informasjonen førte til spesifiseringer i avhandlingen, mens andre gikk på uenigheter av hvordan Scrum ble brukt i prosjektet. Med et friskt og nytt syn på prosjektet som kom gjennom forskeren er det mulig at jeg la merke til ting som teamet ikke var klar over, og at de derfor var uenige i det som hadde blitt observert. I denne avhandlingen er informasjonen presentert slik forskeren opplevde bruken av Scrum, supplert av informasjon fra teamet for å finne bakomliggende begrunnelser for valgene.

4 *Abstraksjon og generalisering*: Alle utviklingsprosjekter er forskjellige på detaljnivå, noe som kan gjøre det vanskelig å få de samme resultatene om man forsøker å gjenskape studiet. Siden prosjektet kan kategoriseres som et typisk Scrum-prosjekt, og ved å ha høy gjennomsiktighet i hva jeg har gjort, vil resultatene i studiet være troverdige. Selv om detaljene angående bruken av Scrum an variere i svært stor grad fra prosjekt til prosjekt vil dette studiet allikevel være generaliserbart fordi de store linjene som blir trukket i konklusjonen hever seg over detaljnivået i studiet.

5 *Dialogisk argumentasjon*: Ved begynnelsen av studiet hadde jeg stor tro på Scrum som en god utviklingsmetode, noe som kan ha påvirket mitt

syn på funnene i en mer positiv retning. Etterhvert som studiet pågikk ble jeg mer og mer objektiv i forhold til funnene og hva de kunne bety. Mitt syn på Scrum ble i hovedsak endret på grunn av større innsikt i litteraturen og erfaringer med bruken av metoden, men også basert på prosjektteamets syn på Scrum. Etter dette studiet har jeg blitt mye mer kritisk til Scrum enn jeg var i begynnelsen.

6 *Ulike fortolkninger*: Det ble utført intervjuer med alle på prosjektet for å skape høy grad av triulangasjon på resultatene. Dette ble gjort for å være sikker på at informasjonen som kom frem var så riktig som mulig. Det ble også brukt flere forskjellige datagenereringsmetoder for å øke troverdigheten ytterligere, men det ble ikke brukt lyd- eller video-opptak. All informasjon ble samlet i feltnotater, som vil si at det bare er notater å gå tilbake på om det er ting jeg var usikker på hvordan skjedde. I studiet ble notatblokken brukt så lite som mulig i løpet av møtene, for at ikke prosjektmedlemmene skulle føle seg usikre og oppføre seg annerledes enn de ville normalt ha gjort. Derfor var det ofte slik at notatene ikke ble skrevet før etter hendelsene fant sted. Det er da ikke sikkert at alt er 100 % gjengitt i disse notatene, som kan ha truet troverdigheten til resultatene. Det er likevel god grunn til å si at essensen av sitatene ble videreført i studiet, men at bredden i observasjonene kan ha blitt påvirket.

7 *Mistanke* (bias): Siden min tilgang til prosjektet var gjennom en PO ved IT-avdelingen ved NTNU som ikke var en del av prosjektet kan dette ha farget prosjektgruppas syn på å bli observert. Det er ikke sikkert at hele gruppa var komfortabel med å bli observert, og var dermed mer motvillig til å dele informasjon med meg. Siden dette bare er andre gang jeg utfører en slik type studie har jeg ikke lært alle metodene som skal til for å utføre observasjon på en diskret måte. Dette kan ha gjort at teamet har oppført seg annerledes med meg tilstede fordi jeg ikke var usynlig som en flue på veggen. Det er også slik at jeg var mye yngre enn de i prosjektgruppa. Dette kan ha påvirket resultatene i den grad at intervjuobjektene kan ha oppført seg annerledes til meg enn de ville til en jevnaldrende eller eldre observatør. I tillegg har man den faktoren at det kan ta en stund før folk vender seg til en utenforstående observatør. Denne tiden varierer fra person til person, derfor ble intervjuene med prosjektmedlemmene utført så sent i studiet som mulig for å minske påvirkningskraften denne faktoren hadde.

3.2 Bakgrunn

14C var et prosjekt som ble utført av konsulenter fra IT-avdelingen ved Norges teknisk-naturvitenskaplige universitet (NTNU), på vegne av kunden *NTNU Vitenskapsmuseet*. NTNU Vitenskapsmuseet har det eneste laboratoriet i Norge som kan utføre aldersbestemmelse av prøver av organisk materiale ved hjelp av ^{14}C-metoden. Da studiet startet var NTNU Vitenskapsmuseet i gang med en renovering av hele laboratoriet for å oppgradere utstyret slik at man kunne behandle prøver mer effektivt og med større presisjon i dateringen av disse prøvene. Dette skulle realiseres ved å selv ta i bruk en ny teknikk for datering: et akseleratorbasert massespektrometer (AMS). Tidligere ble prøver sendt til et laboratorie i Uppsala i Sverige for å få brukt denne teknikken. Ved å selv kunne utføre denne delen av dateringen ville NTNU Vitenskapsmuseet spare mye tid ved å slippe posttiden for å sende prøver til og fra Uppsala. Oppgraderingen ble også gjort med et ønske om å modernisere seg og og kunne være en attraktiv konkurrent til internasjonale laboratorier som utfører ^{14}C-datering. Datering hos enkelte av disse internasjonale laboratoriene var dyrere enn hos NTNU Vitenskapsmuseet, men behandlingstiden var også kortere. I forbindelse med renoveringen hos NTNU Vitenskapsmuseet gikk noen av deres kunder over til utenlandske laboratorier, men det var et håp om at flere av kundene planla å vente til renoveringen var ferdig og bestille datering fra NTNU Vitenskapsmuseet som før[18].

Sammen med renoveringen så man på muligheten for å få til et webgrensesnitt for bestilling av datering. Etter at denne saken var oppe til lederforum ble det bevilget midler til å å iverksette arbeidet med webgrensesnittet som et ledd i renoveringsprosessen [18]. 14C-prosjektet ved IT-avdelingen på NTNU gikk ut på å realisere denne webløsningen. Grunnen til at IT-avdelingen fikk denne oppgaven var at IT-avdelingen hadde vært med på å utvikle et annet system for NTNU Vitenskapsmuseet hvor de ansatte kunne legge inn prøveinformasjon i en database. Dermed hadde IT-avdelingen allerede noe domenekunnskap om hvordan ^{14}C-prøver ble behandlet hos NTNU Vitenskapsmuseet og slapp på denne måten å sette seg inn i et helt nytt domene når de skulle utvikle den nye webløsningen.

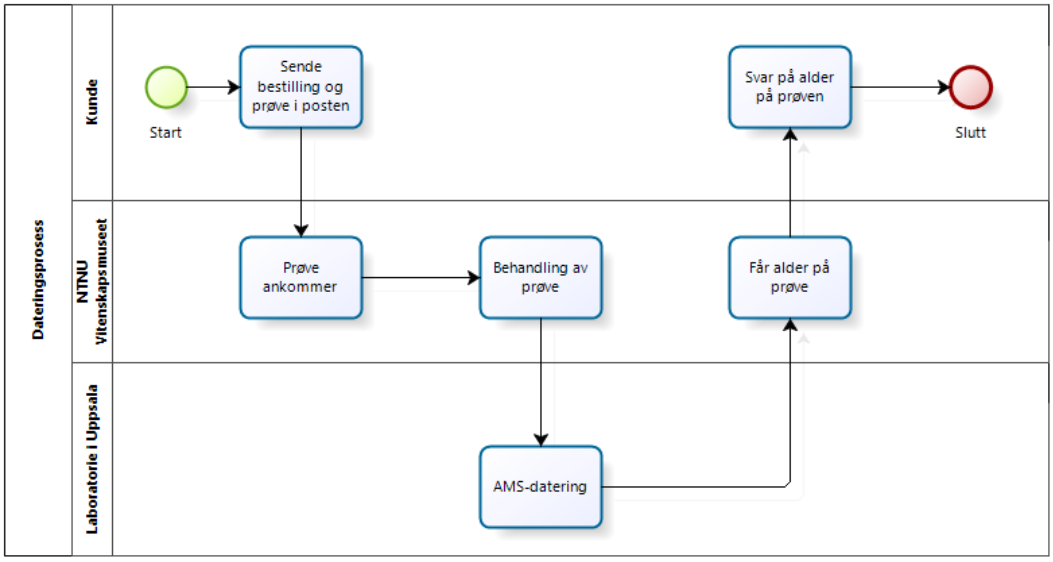
Før laboratoriet ble midlertidig stengt grunnet renoveringen var det hektisk drift, hvor NTNU Vitenskapsmuseet fikk inn masse prøver fra både arkeologer, andre museer og fylkeskommuner over hele Norge som ønsket prøvene sine datert ved hjelp av ^{14}C-datering. I løpet av ett år fikk NTNU Vitenskapsmuseet inn 1 500 prøver til datering hvor hver prøve hadde minimum 1 måned behandlingstid. I tillegg var det ventetid i forbindelse med forsendelse

av prøven til laboratoriet i Uppsala. Dermed ble det fort flaskehalser i de forskjellige stegene på laboratoriet. Med slike flaskehalser bestemte NTNU Vitenskapsmuseet seg for å utvide kapasiteten og effektivisere arbeidet sitt. En ny webløsning ville gjøre at NTNU Vitenskapsmuseet hadde kontroll over alle prøver som til enhver tid var på vei inn for datering. Dermed kunne man forberede laboratoriet for innkommende prøver ved for eksempel å planlegge forbehandling på beinprøver til neste dag, om mange beinprøver var meldt på tur inn til laboratoriet. På den nye webløsningen ville det bli slik at brukerne av systemet ville sende inn bestilling av ^{14}C-datering på en prøve digitalt, før de sender inn den fysiske prøven ved hjelp av posten. På den måten ville NTNU Vitenskapsmuseet kunne vite hvilke prøver som til enhver tid var på tur inn til laboratoriet og kunne dermed forberede seg bedre til arbeidet som ventet i tiden fremover.

Med introduseringen av AMS-teknikken på laboratoriet ville NTNU Vitenskapsmuseet også kunne klare å utføre ^{14}C-datering på prøver med enda mindre masse enn tidligere [18]. For kundene av NTNU Vitenskapsmuseet ville dette bety at de kunne få datert funn av mindre størrelse enn tidligere mulig. Den nye webløsningen ville også gi brukerne mulighet til å søke i frigitte dateringsresultater. Derfor trodde NTNU Vitenskapsmuseet at de ville få en ny gruppe brukere med den nye løsningen, i tillegg til de opprinnelige kundene: "vanlige" personer som ville bruke søkefunksjonen til for eksempel å se på funn som er gjort i nærheten av sitt nærområde [18].

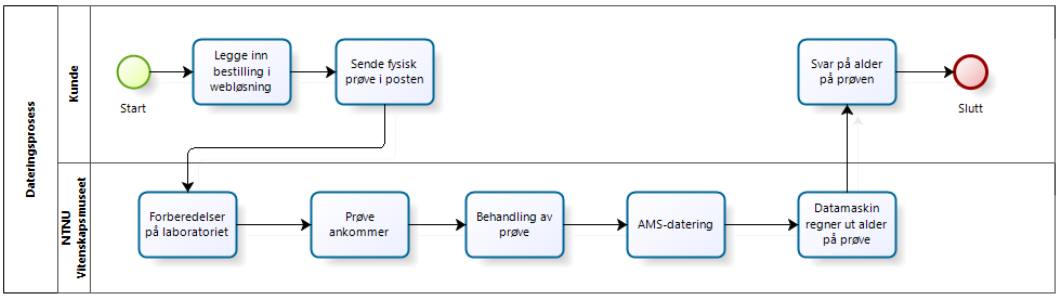
Tidligere laboratorieprosess

Før renoveringen av laboratoriet og utviklingen av den nye webløsningen ble dateringen hos NTNU Vitenskapsmuseet utført som vist i figur 3.3. For å bestille ^{14}C-datering av prøver var brukerne (som i hovedsak var arkeologer, fylkeskommuner og andre museer) nødt til å sende inn papirer sammen med prøven. Disse papirene inneholdt informasjon om prøven, som for eksempel funnsted, type materiale (tre, bein osv.), og hvordan prøven ble samlet inn (som utgraving eller løsfunn). Når prøven og papirene kom inn til NTNU Vitenskapsmuseet ble hver prøve tildelt et identifikatornummer (TRa-nr.) på laboratoriet som den beholdt gjennom hele dateringsprosessen. Med den nye webløsningen skal brukerne selv merke prøven med en egen referanse, mens prøven vil fortsette å få tildelt en egen identifikator når den ankommer laboratoriet. Dette fordi de laboratorieansatte må se den fysiske prøven før de kan godkjenne den for datering (med hensyn på for eksempel mengde og kvalitet).



Figur 3.3: Tidligere dateringsprosess

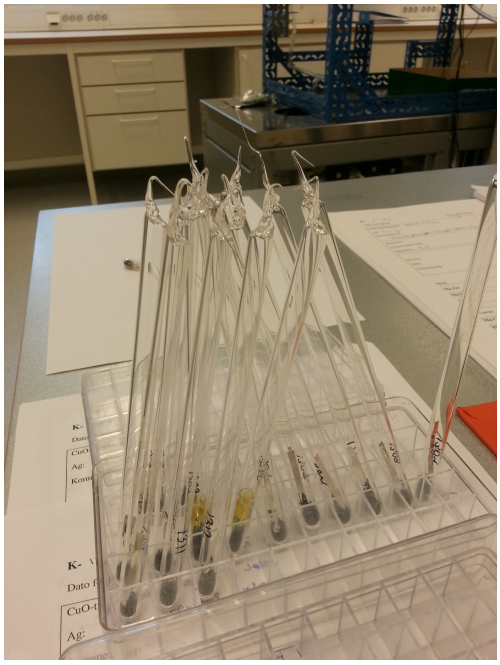
Til sammenligning, vil renoveringen og den nye webløsningen føre til at dateringsprosessen vil utføres som vist i figur 3.4.



Figur 3.4: Ny dateringsprosess

Første steg i ^{14}C-dateringen av en prøve var forbehandling av prøven. Denne prosessen skal fortsette som før etter renoveringen, bare i større skala. Målet med forbehandlingen var å fjerne forurensninger i prøven og selve prosessen var ulik basert på hvilket materiale prøven besto av: en prøve av bein hadde annen forbehandlingsprosedyre enn en prøve av tre. De fleste materialer gjennomgikk en syre-base-syre behandling som ble gjort i en lukket avtrekksboks (se figur 3.5), før prøvene ble lagt i reagensrør og merket med sin identifikator (se figur 3.6). Gjennom dateringsprosessen ble prøven fulgt av et laboratorieskjema hvor man fylte inn informasjon relatert til prøven etterhvert som prøven gikk gjennom de ulike stegene. Laboratorieskjemaene ble brukt for å sikre at prøvene fikk rett behandling i AMS-maskinen. Se figur 3.7 for eksempel på laboratorieskjema.

Figur 3.5: Datering: forbehandling



Figur 3.6: Forseglede reagensrør med prøvemateriale, kobberoksid og sølvfolie. Identifikatoren sitter nederst på reagensrørene.

Med det nye systemet vil bruken av laboratorieskjemaer bli noe endret for de laboratorieansatte. For å oppfylle ønsket om å gi brukeren kontinuerlig tilbakemelding når prøven går gjennom de ulike prosessene blir det nødvendig å legge inn informasjon i det andre datasystemet etterhvert som prøven gikk gjennom de ulike stegene i dateringsprosessen. NTNU Vitenskapsmuseet tenker på sikt at prøveinformasjon skal legges direkte inn i databasen, slik at bruken av fysiske laboratorieskjemaer etterhvert fases helt ut.

K-

Dato forsegling: Dato grafittering:

CuO-type/mengde:

Ag:

Kommentarer:

Materiale:

Forbehandlet dato: Overført dato (For OX/bkg gass):

Mg prøve:

Knuser:

H20:

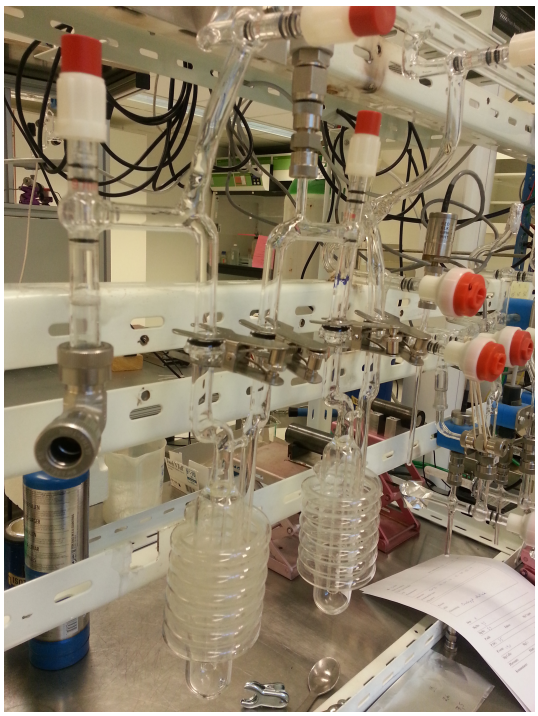
Gassrensing:

Ovn:		
Mg Zn:		
Mg Fe:		
P mål:	**Faktor:**	**Mg C i gass:**
P 13C:		
P->ovn:		**mg C i ovn:**
Mg C+Fe:	**Mg C :**	
Pforventet:	**Pslutt:**	**%utbytte(vekt):**

Kommentarer:

Figur 3.7: Laboratorieskjema

Etter forbehandlingen gikk prøvene gjennom forbrenning, slik at karbongass ble frigjort, og deretter steg med rensing (figur 3.8). Gassen ble omgjort til grafitt (figur 3.9), og presset inn i targets (små ampuller), som vist i figur 3.10. Utstyret som er vist i figur 3.8 og 3.9 ville i løpet av renoveringen bli ombygget og utvidet, for å kunne behandle enda flere prøver i parallell.



Figur 3.8: Datering: rensing

Figur 3.9: Datering: grafittering

Figur 3.10: Datering: Ampuller

Ampullene ble deretter sendt til et laboratorie i Uppsala i Sverige for å gjøre dateringen i et akseleratorbasert massespektrometer (AMS). Etter dette var gjort fikk NTNU Vitenskapsmuseet tilbake resultat på datofesting av prøven samt eventuelle rester av selve prøven. Både resultatene og selve prøven ble så sendt videre til kunden som hadde bestilt dateringen. Eksempel på resultatene av prøven er vist i figur 3.12. Som man ser fra denne figuren var

informasjonen som kunden fikk fra prøven veldig enkel med lite beskrivelser. I den nye webløsningen var det planlagt at det kunne legges ved kommentarer til resultatene, slik at NTNU Vitenskapsmuseet skulle kunne komme med ekstra kommentarer til resultatene om de føler at dette er nødvendig. Dette kunne for eksempel være for å forklare hvorfor en spesifikk prøve hadde veldig stor usikkerhet i sin tidsfesting. I figur 3.12 hadde for eksempel prøve T-17923 en alder mellom 1895 og 1955, hvor det kunne være viktig for kunden å vite mer nøyaktig når prøven stammet fra siden det var stor forskjell på verden i 1895 og i 1955. Det som forårsaker usikkerhet ved bruk av AMS-teknikken er at ^{14}C-innholdet i atmosfæren har forandret seg over tusenvis av år, grunnet endringer i solaktivitet, vulkanutbrudd og andre klimaendringer. Med den store utviklingen av industrien i verden de siste 300 årene er det veldig stor usikkerhet i prøver fra denne tiden, slik at dateringsprøver for AMS-teknikken bør være eldre enn 300 år. For prøver eldre enn 300 år beregnes sannsynlig datering basert på historiske data av ^{14}C-innhold i atmosfæren, med en sikkerhet på 99 %. Dermed vil de ulike prøvene kunne få forskjellig størrelse på tidsrommet de stammer fra, se figur 3.7.

I begynnelsen av studiet var NTNU Vitenskapsmuseet i gang med renoveringen og hadde i den forbindelse allerede fått installert sitt eget AMS og startet med datering av egne prøver. Resten av dette delkapittelet inneholder derfor informasjon om hvordan NTNU Vitenskapsmuseet brukte sitt AMS i forbindelse med datering. Etter at grafitten var presset inn i ampuller ble disse ampullene montert i NTNU Vitenskapsmuseets AMS (figur 3.11. Her ble karbonet ionisert, akselerert i elektriske felt og bøyet av i magnetiske felt. Dermed ble karbonisotopene splittet i tre stråler; for ^{12}C og ^{13}C ble strømstyrken målt, mens antall ^{14}C-atomer ble tellet i en detektor. Ut fra dette kunnen man beregne forholdet mellom isotopene, og dermed alderen for materialet. Opptil 50 ampuller ble satt inn i AMS samtidig, plassert ut i fra informasjonen på laboratorieskjemaene fra de tidligere prosessene. Ved å fortelle datasystemet koblet til akseleratoren hvilken prøve som lå i hvilken av de 50 plassene ble dateringen automatisk koblet til riktig ampulle og prøve. Med ^{14}C-dateringen av prøven ferdigstilt, ble resultatene og eventuelle rester av prøven sendt tilbake til brukeren. Resultatene (se figur 3.12) som ble sendt tilbake til kunden inkluderte ^{14}C-alder med usikkerhet, δ^{13}C-verdi og kalibrert alder.

Figur 3.11: Datering: Akselerator

Lab.ref.	Oppdragsgivers ref.	Materiale	^{14}C alder før nåtid	Kalibrert alder	δ^{13}C (‰)
T-17923	T23209-10, sjakt 1 Munkeby kloster Levanger, Nord-Trøndelag	Tre	40 ± 50	AD1895-1955	-26.1*
T-17924	T23209-16, sjakt 2 Munkeby kloster Levanger, Nord-Trøndelag	Trekull	805 ± 75	AD1170-1285	-26.1*
TUa-5531	T23209-14, sjakt 4 Munkeby kloster Levanger, Nord-Trøndelag	Trekull Gran	1075 ± 35	AD965-1010	-26.1*
TUa-5532	T23209-17, sjakt 3 Munkeby kloster Levanger, Nord-Trøndelag	Trekull Gran	995 ± 35	AD1015-1035	-26.1*

Figur 3.12: Eksempel på resultattabell som sendes til kunden

For at akseleratoren skulle regne ut riktig datering på prøvene hadde de ansatte ved laboratoriet kjørt mange prøver med kjent alder, slik at man hadde kontrollprøver for dateringen. I tillegg ble det kontrollert at det ikke var forurensninger i prepareringslinjer og akselerator ved at man kjørte bakgrunnsprøver og moderne standarder sammen med kontrollprøvene. Når kontrollprøvene ble ferdig testet var laboratoriet til NTNU Vitenskapsmuseet klar for nye, ukjente prøver sendt inn fra brukerne. Forventet åpning av laboratoriet for prøver fra brukerne var sommeren 2014.

3.3 Resultater

Dette delkapittelet viser resultatene fra studiet av prosjektgruppen, hvor alle hendelsene er kategorisert etter hvor i sprintløpet de oppsto. To hele sprinter ble observert, og hendelsene er her trukket sammen og presentert som en sprint. For eksempel har to møter med sprint-planlegging blitt observert, hvor da beskrivelsene av disse har blitt samlet under et eget avsnitt. Fokuset på utviklingen og forskjellene mellom disse to møtene blir videre utforsket i kapittel 4, før en samlet konklusjon kommer i kapittel 5. Navnet på prosjektdeltakerne er erstattet med deres rolle i Scrum: [teammedlem], T_{PO} [1], PO, eller SM. Dette ble gjort for å sikre anonymitet til prosjektdeltakerne i tillegg til å vise hvem som gjorde hva. Rollen til deltakerne er mer interessant enn det spesifikke individet.

Roller

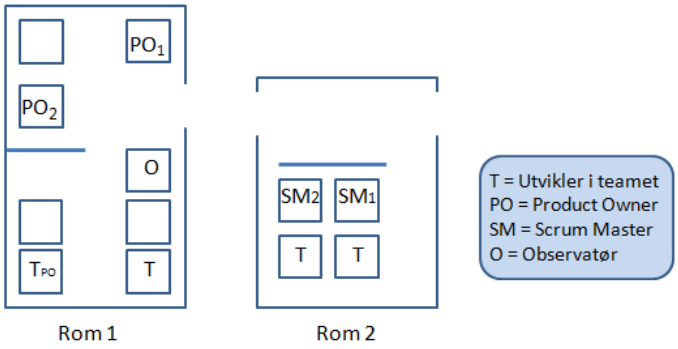
Dette prosjektet ble tildelt et team på 6 teammedlemmer og en PO, alle fra IT-avdelingen ved NTNU. Rollen som SM var tenkt å gå på rundgang, men to av teammedlemmene tok på seg å bytte på denne rollen siden de andre medlemmene ikke hadde lyst til å påta seg rollen som SM. Flere i teamet fortalte at "man er ansatt som utvikler og vil ikke ha arbeidsoppgaver som går på noe helt annet." De to som byttet på rollen som SM hadde rollen i to sprinter før man byttet ansvar. I tillegg var den aktive SM en del av teamet i den form at han var med på utviklingsarbeidet på lik linje med de andre i teamet. Dette ble ikke sett på som noe problem siden det var "[...] egentlig veldig lite ekstra jobb å være SM hos oss. Det meste ordner teamet selv" - [SM_1].

Medlemmene på dette prosjektet hadde også andre prosjekter som de arbeidet på i parallell med 14C-prosjektet, samt driftsoppgaver som måtte utføres. Dette gjorde at dagene kunne være veldig forskjellige. Ved IT-avdelingen ble prosjekter tildelt til hele team, slik at alle 6 teammedlemmene jobbet på de samme prosjektene. Dette var imidlertid ikke tilfellet med rollen som PO. PO ble tildelt prosjekter, ikke team, noe som førte til at dette teamet i utgangspunktet hadde to forskjellige PO å forholde seg til. Utover i studiet ble det også klart at det ene teammedlemmet (T_{PO}) fungerte som PO i 14C-prosjektet sammen med PO_1. Dette medlemmet var med på møter med kunden og laget oppgaver som ble lagt inn i Jira[2] ut fra tilbakemeldingene

[1] Teammedlem som i tillegg fungerte som PO

[2] Koordineringsverktøy fra Atlassian brukt til både product og sprint backlog

på disse møtene. Grunnen til at T_{PO} fungerte som PO i 14C-prosjektet var at dette medlemmet hadde hatt et stort ansvar i forbindelse med utviklingen av den tidligere løsningen for de labansatte ved Vitenskapsmuseet. Flere i teamet har pekte at dette påvirket arbeidet, blant annet ved at T_{PO} følte et større ansvar for produktet enn resten av teamet. Det viste seg også at PO_1, som egentlig var den eneste PO på 14C-prosjektet, følte det var vanskelig å fordele ansvaret mellom de to som fungerte som PO. T_{PO} var selv klar på at dette var "[...] ingen gunstig permanent løsning. Det ble høyere temperatur med denne rollen". Alle involverte i utviklingsdelen av prosjektet var fordelt på to rom, vist i figur 3.13. De kvadratene uten bokstav signaliserer arbeidsplasser til andre ansatte ved avdelingen som ikke var en del av teamet, men som satt på samme rom.



Figur 3.13: Den interne organiseringen av prosjektgruppa.

Prosess

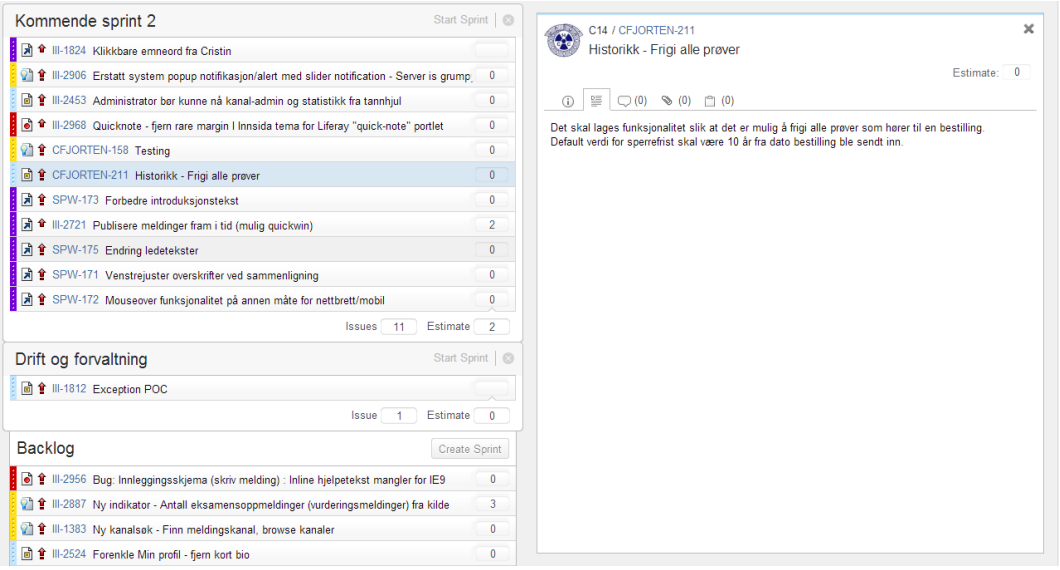
Teamet som jobbet med 14C-prosjektet tok i bruk Scrum med alle de overordnede aktivitetene vist i figur 2.4. Selv om de overordnede aktivitetene var med, gjorde IT-avdelingen noen endringer på bruken av disse aktivitetene i forhold til "teoretisk" Scrum. I dokumentet *Pistasj-Scrum* (se vedlegg A) hadde avdelingen skrevet ned sine egne regler over hvordan Scrum skulle brukes av teamet. I tillegg hadde teamet i dette prosjektet tatt med en ny aktivitet med fokus på kunnskapsdeling (mer om dette senere i kapittelet).

Visjon

Utviklerne ved IT-avdelingen hadde ikke i noe innsikt i målene NTNU Vitenskapsmuseet hadde med den nye webløsningen. Utviklerne forholdt seg utelukkende til sprint backlog og en pdf-fil de hadde fått fra kunden med informasjon om hvordan de ville at løsningen skulle se ut. Det kom godt frem fra flere intervjuer at utviklerne ikke visste hva prosjektet gikk ut på utover det som sto i pdf-filen, for eksempel kom SM_1 med en kommentar om at "[...] hele teamet burde kanskje ha vært hos kunden en tur og sett det som foregikk på laboratoriet".

Product backlog

For å holde styr på Product backlog ble det i dette prosjektet tatt i bruk verktøyet Jira, som var et digitalt samarbeidsverktøy laget av Atlassian. Bruken av dette verktøyet var utbredt blant flere team ved IT-avdelingen, blant annet fordi det tillot både teamet og PO å se det nåværende bilde av backlog'en. Figur 3.14 viser et skjermklipp tatt fra Product backlog i Jira, tilhørende de prosjektene som teamet arbeidet på. Som man ser i figuren var oppgaver som hadde blitt gjort klar og skulle inn i neste sprint liggende under "Kommende sprint 2", mens resten av Product backlog lå under "Backlog". Alle disse oppgavene ble laget av PO etter møte med kunden, og fungerte som en oversikt over alt kunden ønsket å ha med i det ferdige produktet.



Figur 3.14: Skjermutklipp av product backlog fra oppgavestyringsverktøyet Jira [3]

Under møtene med planlegging av neste sprint ble nye oppgaver fra "Backlog" overført til "Kommende sprint" slik at de skulle bli utført i løpet av den neste sprinten. Oppgavene i Backlog var ikke endelig prioriterte siden teamet i samarbeid med PO overførte flere av oppgavene som var langt ned i lista. Det var PO som hadde en egen oversikt over hvilke oppgaver som var klare til å tas med i sprinten, og formidlet dette videre til SM og resten av teamet.

Sprint planning

På planleggingsmøtene for sprintene var hele teamet, samt PO for alle prosjektene i neste sprint, tilstede. I løpet av studiet var dette alltid de samme personene: hele teamet samt PO_1 og PO_2. Grunnen til at to PO var tilstede var at disse var ansvarlig for hvert sitt prosjekt som teamet jobbet på. I tillegg var PO med på møter hvor prosjektstart og prosjektslutt ble diskutert sammen med ledelsen ved IT-avdelingen. I disse møtene var frister og fordeling av arbeidskraft med på å bestemme hvilke prosjekter som skulle prioriteres de neste månedene. Det var derfor POene som fortalte til teamet hvilke prosjekter som skulle være i fokus fremover. Under Sprint planning begynte PO med å fortelle om hvilke prosjekter som hadde prioritet de neste månedene, slik at teamet fikk en oversikt over hva de kom til å jobbe med. PO tok et sterkt ansvar med å lede planleggingen av sprinten, blant annet ved å sette i gang møtet, fortelle om hva som skulle foregå i prosjektene og hvor fokuset skulle ligge fremover. Etter at prosjektene var presentert kom det en del spørsmål fra resten av prosjektgruppa som førte til mer detaljerte oppgaver i forhold til prosjektene. Før møtet hadde PO laget noen større oppgaver til hvert prosjekt (stories), som deretter ble brutt ned i mindre oppgaver sammen med teamet under diskusjoner.

Møtet gikk etterhvert over i en estimeringsfase hvor oppgavene for neste sprint ble estimert ved hjelp av story points og planning poker[3]. Før estimatene ble hver oppgave presentert på nytt av noen som hadde god kontroll på oppgaven (som oftest en av POene), slik at alle hadde en ide om hva oppgaven gikk ut på. Det var også ofte slik at PO selv kom med et estimat på oppgaven uten at teamet hadde noe de skulle sagt. Dette gjaldt særlig under første sprint planning. Jeg ble også fortalt at man tidligere ikke hadde drevet med estimering i 14C-prosjektet, fordi man ifølge PO "[...] ville bruke mer tid på å gjøre ting skikkelig enn å passe på at man ikke brukte over estimert tid på oppgaven". Dette vises godt i skjermutklippet fra oppgavene i sprinten vist i figur 3.15, hvor alle oppgavene har estimatet 0. Under første sprint planning ble det gjort et overordnet estimat på hele

[3]www.planningpoker.com

14C-prosjektet, slik at man kunne se hvor mye tid man kunne bruke på de andre prosjektene. Det ble her introdusert et nytt prosjekt (parkeringsapp) og noen bugfiksingsoppgaver. Her ble oppgavene i parkeringsapp-prosjektet og bugfiksingsoppgavene estimert på vanlig måte, selv om 14C-oppgavene ikke ble det. I andre sprint planning ble alle oppgaver estimert, uavhengig av prosjekt.

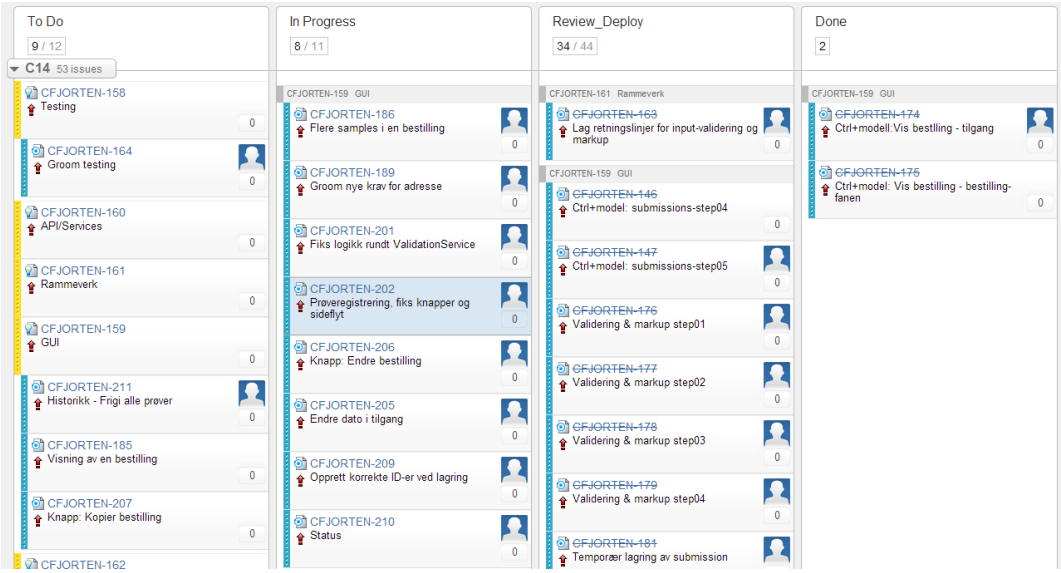
I løpet av begge Sprint planning-møtene var SM anonym og fungerte mer som en sekretær. PO tok til seg ansvaret med å lede møtet fremover, og kom flere ganger med forslag til teamet som "Dere kan kaste kort om dere er uenige" og "Totale story points for neste sprint nærmer seg 40 nå. Da begynner det å bli nok å gjøre".

Sprint backlog

I tillegg til å bruke Jira for product backlog ble Jira brukt til å holde styr på Sprint backlog. Alle oppgavene i Sprint backlog ble lagt inn i et "agile board", som vist i figur 3.15, hvor alle oppgavene startet i første kolonne: "To Do". Det var da opp til teammedlemmene selv å velge oppgaver de ville gjøre. I teorien ble oppgaven da flyttet over til "In Progress", slik at andre ikke ville ta denne oppgaven. Det viste seg gjennom studiet at man flere ganger glemte å bruke Jira aktivt på denne måten, slik at dobbeltarbeid forekom.

Når det gjaldt selve sprint backlog var oppgavene i denne i prioritert rekkefølge, slik at de som var høyest opp hadde størst prioritet. Dett var veldig ofte bugfiksningsoppgaver i forbindelse med andre, tidligere prosjekter som teamet måtte løse i parallell med nåværende prosjekter. Disse oppgavene ble sett på som kjedelige og uinteressante og teammedlemmene ville heller gjøre andre oppgaver. Dette løste SM ved å la hele teamet jobbe med bugfiksing samtidig: "Da foreslår jeg at vi etter lunch i dag setter av tid til å løse disse bugsene". Dette ble gjort som et mottiltak til at man i tidlige sprinter hadde igjen flere bugfiksingsoppgaver mot slutten av sprinten, selv om disse egentlig hadde høyere prioritet enn andre utførte oppgaver.

Selve oppgavene i sprint backlog ble utformet og laget av den PO som var ansvarlig for prosjektet. På oppgavene var det en kort tekst som beskrev oppgaven (se figur 3.16), men flere i teamet pekte på at "[...] disse beskrivelsene var mer utfyllende i tidligere prosjekter med annen PO, nå har det vært vanskeligere å vite akkurat hva en oppgave har gått ut på". Tidligere ble det brukt skjermutklipp som ble lenket sammen med oppgavebeskrivelsen,



Figur 3.15: Skjermutklipp av sprint backlog fra oppgavestyringsverktøyet Jira [3]

slik at man enklere kunne se hva oppgaven omhandlet. Teammedlemmene valgte selv hvilke oppgaver de skulle jobbe på. Det ble dermed slik at hvert medlem fikk sitt spesialfelt som de jobbet på i prosjektene: Samme person valgte ofte oppgaver som lignet på hverandre.

Sprint

Teamet i dette studiet brukte sprinter på 2 uker. Innenfor disse sprintene jobbet teamet parallelt med flere forskjellige prosjekter, noe som gjorde at dagene ble veldig varierende. Noen dager jobbet alle på samme prosjekt, mens andre dager kunne et teammedlem være eneste person som jobbet på for eksempel 14C-prosjektet. Selv om teamet jobbet på forskjellige prosjekter var det mye samarbeid i form av parprogrammering og hjelp til ting man lurte på: om de ikke arbeidet på samme rom gikk de over i det andre rommet for å finne den kompetansen de ønsket. Det var imidlertid slik at et par teammedlemmer påpekte at "det er jo ikke langt å gå, men det merkes at man er delt."

Alle dager i sprinten ble startet med standup-møte klokken 0845. Dette møtet ble holdt fast på et tredje rom hvor alle i teamet møttes for å oppdatere hverandre på sin status i arbeidet. Hvert møte ble startet av SM og alle i teamet fortalte kort hva de hadde arbeidet med siden sist og hva

Figur 3.16: Skjermutklipp av oppgavebeskrivelse fra oppgavestyringsverktøyet Jira [3]

de planla å arbeide med etter møtet. Flere ganger tok teammedlemmene opp planlagt fravær om de skulle være borte. Det var også slik at man under dette møtet kom med kommentarer til de andre gruppemedlemmene om man hadde spørsmål til det som var gjort. Om denne diskusjonen viste seg å bli dyp ble det utsatt til etter møtet. Alle dager i studieperioden ble standup ferdig på under 15 minutter, og som oftest under 7 minutter. Disse møtene var en ren verbal oppdatering på status, det ble ikke presentert og vist frem burndown chart for oppgavene i sprinten. En dag kontaktet en PO meg og ville forklare hvorfor burndown charts ikke ble brukt. Det var fordi man "[...] ikke har brukt estimater i 14C-prosjektet, og burndown var ikke interessant for oss. Vi ville heller ha fokus på kvalitet i det som gjøres, i stedet for å studere burndown".

Sprint evaluering

Det ble holdt to sprint reviews i perioden studiet pågikk. Her ble kunden invitert til møtet den første gangen, og teamet presenterte det de hadde gjort. Under dette møtet var det T_{PO} som presenterte løsningene, mens resten av teamet kom med innspill. Den andre sprinten ble preget av mindre arbeid på 14C-prosjektet, noe som førte til at det ikke var noe kunde med på review av det lille som hadde blitt gjort siden sist. Det var heller ingen presentasjon om de overordnede målene for sprinten ble oppnådd eller ikke. I stedet pre-

senterte teamet status på de oppgavene som ennå lå under "To Do" og "In progress"-fanene i sprint backlog, for å oppdatere PO om hvordan prosjektet lå ann. Her viste det seg at flere av oppgavene var ferdige, mens andre ble flyttet over i neste sprint. Det endte med en kjapp visning av hvordan webløsningen så ut på dette tidspunktet.

For å evaluere sprinten ble det også holdt Sprint retrospective etter hver sprint. Hovedelementet her var at SM satte teamet i gang med å skrive lapper med ting som hadde vært bra og dårlig i løpet av sprinten, antallet lapper var selvvalgt. Etter lappeskrivingen fikk hvert teammedlem presentere hva de hadde skrevet. PO var også med på dette møtet, men utesto fra lappeskrivingen. Ved første sprint retrospective var det veldig mange positive tilbakemeldinger til de andre i teamet, hvor enkeltindivider ble pekt ut. Andre gjengangere var "[...] positivt å jobbe med ny teknologi" og at "[...] oppgavene var artige". På den negative siden ble bruken, eller rettere sagt den manglende bruken, av Jira tatt opp. Teamet ønsket at medlemmene skulle bli flinkere til å vise i Jira at en oppgave var tatt og igangsatt. Deretter kom PO inn i bildet, med spørsmål som for eksempel "Har dere noen forklaring på hvorfor dere ikke flytter på saker i Jira?". Dette ble særlig klart under andre sprint retrospective, hvor PO fulgte opp med å komme med forslag til forbedringer som teamet kunne gjøre i neste sprint. Ved slutten av møtene var det ikke noe offisielt valg av ting som man aktivt skulle jobbe med for å forbedre seg i neste sprint. Det virket som om teamet brukte denne lappeskrivingen til å få oversikt over hva de andre syntes om arbeidet, og for å få en slags avslutning på sprinten.

Kunderollen

En kunderepresentant var valgt ut for å representere kunden, NTNU Vitenskapsmuseet, i forbindelse med utviklingen av webløsningen. Denne kunderepresentanten ble valgt fordi hun var sterkt involvert i den andre dataløsningen som IT-avdelingen hadde utviklet for NTNU Vitenskapsmuseet. Denne representanten var også en av de fra kunden med tid og interesse for å kunne påta seg denne rollen. Kunderepresentanten var ansvarlig for å kommunisere NTNU Vitenskapsmuseets krav og ønsker til prosjektgruppa og webløsningen.

Generelt sett var begge parter, både kunderepresentanten og prosjektgruppa, fornøyd med kommunikasjonen og samarbeidet mellom partene, men kunderepresentanten begynte å miste troa på at kartintegrasjon rakk å bli med til lanseringen av webløsningen. Tilgjengeligheten var høy og geografisk nærhet gjorde det enkelt å avtale møter. Begge partene syntes at det var

enkelt å forstå den andre partens ønsker og tanker, men kunderepresentanten ønsket at hun hadde et mer teknisk grunnlag slik at det hadde vært enklere å forstå mulighetene med webløsningen. Kunderepresentanten var imidlertid klar på at de jevne møtene mellom partene var veldig nyttig og at det var interessant å se utviklingen på produktet [18]. Når det gjaldt kunderepresentanten sitt samarbeid med resten av de ansatte ved NTNU Vitenskapsmuseet var kunderepresentanten misfornøyd med at det tok så lang tid før resten av kundegruppa kom på banen og sa hva de mente om det foreløpige produktet. Dette førte til mange nye endringer for prosjektgruppa i uka før teamets presentasjon (sprint review) til kunderepresentanten.

Kunden hadde noen klare ideer om hvordan webløsningen skulle bli. Prosjektgruppa fikk overlevert en pdf-fil fra kunden, med forslag til hvordan brukergrensesnittet til løsningen skulle se ut. Dette forslaget hadde kunden utarbeidet etter å ha kjørt testversjoner på papir på fire av sine brukere. NTNU Vitenskapsmuseet planla å gjøre mer omfattende testing nå som URL til webløsningen hadde blitt utlevert fra prosjektgruppa. Det var altså kunden selv, ikke utviklerne, som utførte testene i forbindelse med løsningen.

Kunnskapsflyt

I 14C-prosjektet var teamet fordelt på to rom, som kunne være et mulig hinder for kunnskapsdeling. Selv om teammedlemmene vandret mellom rommene for å få tak i den kunnskapen de lette etter er det tydelig at det hadde vært mye lettere om hele teamet jobbet i umiddelbar nærhet til hverandre. Det var også veldig ofte at teammedlemmer gikk til PO_2 med spørsmål om prosjektene, for eksempel "[PO_2] kan du komme til min arbeidsplass en tur, så kan vi ta en kikk på GUI og bestemme hvordan det skal være?" For å fremme kunnskapsdeling hadde teamet i dette prosjektet noen møter, like etter standup var ferdig, hvor hensikten var å dele kunnskap man hadde opparbeidet seg. Disse møtene ble bare brukt i starten av første sprinten i løpet av studiet og jeg ble fortalt av flere teammedlemmer at slike møter "var tenkt brukt i starten av prosjekter hvor vi jobber med ny teknologi. Siden vi setter oss inn i forskjellige ting er det en veldig fin måte å sørge for at alle får noe kunnskap om de forskjellige teknologiene." 14C-prosjektet var det første prosjektet hvor denne formen for kunnskapsdeling ble utprøvd.

Med tanke på kunnskapen i forbindelse med oppgavene til webløsningen var det ofte slik at de forskjellige teammedlemmene hadde sine faste deler å arbeide med. Om en person for eksempel tok oppgaven om informasjon-

ssending mellom skjemaer i webløsningen så ble alle lignende oppgaver utført av samme person. Flere i teamet pekte på at "[...] man ble spesialisert, men forsøkte å velge andre typer oppgaver når dette passet". Utfra observasjonene kan virket det som at andre typer oppgaver ble valgt med jevne mellomrom, siden det flere ganger forekom problemer ved at et teammedlem begynte på en oppgave og fant senere ut at oppgaven allerede var påbegynt av et annet teammedlem som ikke hadde indikert dette i sprint backlog. Dette skapte ekstra og unødvendig arbeid, som enkelt kunne ha vært unngått.

Kunnskapsflyt mellom kunde og utvikler spilte i dette studiet, som i de fleste prosjekter, en viktig rolle for det endelige produktet. Kunden hadde sine prosesser og intern kunnskap om hva som fungerte på laboratoriet hos dem, og formidlet dette videre til prosjektgruppa. Denne kunnskapen ble forsøkt formidlet gjennom møter med både PO_1 og T_{PO}. Det kom tydelig frem i løpet av studiet at disse to var de eneste som hadde denne kunnskapen, mens resten av prosjektgruppa forholdt seg til sprint backlog. De fleste i teamet hadde overordnet kunnskap om 14C-prosjektet, som at løsningen skulle "[...] hjelpe kunden med å bestille ^{14}C-datering" - [T]. De hadde ingen informasjon om hvorfor kunden ønsket denne løsningen eller spesifikt hva kunden ønsket å få ut av å ta i bruk en slik løsning. Det var et par i teamet som etterlyste mer informasjon om dette: "[...] kanskje skulle også resten av teamet ha vært på lab'en og sett hvordan de opererer" - [T].

Implementasjon

Ideen om å ta i bruk Scrum i dette prosjektteamet kom fra nyansatte arbeidere som ble satt inn i teamet. Disse nyansatte hadde med ideen fra den gamle jobben sin, hvor metoden hadde fungert bra, og fikk overtalt IT-avdelingen til å ta i bruk Scrum. Dette skjedde i underkant av 10 år siden, når Scrum for alvor begynte å bli utbredt: "Det var Scrum som var i vinden og andre muligheter ble ikke undersøkt" - [T]. Etter at det ble besluttet å ta i bruk Scrum fikk prosjektgruppa opplæring i metoden av en ekstern Scrumlærer. I begynnelsen ble Scrum brukt som beskrevet i teorien, men ble etterhvert utviklet og endret av prosjektgruppa. For eksempel som å ta i bruk møter for kunnskapsdeling eller involveringen av PO. Noen i teamet var imot denne endringen av metoden. De syntes at man "[...] bør bruke Scrum som beskrevet når man har bestemt å ta i bruk metoden" - [T]. Etterhvert som bruken av Scrum endret seg for prosjektgruppa ble et dokument opprettet som beskrev hvordan prosjektgruppa brukte Scrum (se vedlegg A). Det har vært PO_2 som har skrevet og oppdatert denne beskrivelsen, men i løpet av studiet var det flere bevis på at dette dokumentet ikke ble fulgt. Det sto

for eksempel at teamet var "ansvarlig for oppdatering av sine oppgaver i Jira, og for å initiere Review steget", men Review-steget ble aldri initiert i løpet av studiet. Et teammedlem eksemplifiserte dette godt: "Slik jeg bruker Jira er å velge oppgave, gjøre den og flytte til Review-kolonnen. Ferdig." - [T]. Dette førte videre til at veldig mange oppgaver lå under denne kolonnen ved slutten av sprintene, og som SM bare flyttet over til Done-kolonnen uten noe videre informasjon eller sjekk av oppgavene.

Kapittel 4

Analyse

4.1 Hvor og hvordan avvek metoden fra teorien?

Prosjektet jeg studerte brukte ikke Scrum helt som teorien sa det skulle brukes. Det som er interessant å se er imidlertid ikke om metoden ble fulgt, men hvorfor den ikke ble fulgt. Tabell 4.1 viser i hvilken grad de ulike delene av Scrum ble gjennomført i tråd med det som står i teorien. Venstre kolonne inneholder de ulike delene i Scrum, høyre kolonne inneholder graden. En grad som er "Lav" betyr at prosjektet i liten grad fulgte den teoretiske beskrivelsen. "Middels" betyr at prosjektet i noen grad fulgte teorien, men at det også var deler som ikke ble fulgt. "Høy" betyr at prosjektet i stor grad fulgte det som skrives i teorien. Denne evalueringen er gjort på bakgrunn av informasjonen som hittil er presentert i rapporten. Som det kommer frem fra tabellen var det mange deler av Scrum som ikke ble brukt som anvist i teorien: store endringer i bruken av Scrum var innført og gjennomført av prosjektgruppa. Det er også verdt å merke seg at ingen av delen vurderes til å følge teorien i "lav" grad. Det er altså slik at alle delene til en viss grad følger den teoretiske beskrivelsen, med noen avvik på ulike prosesser. Graderingene i tabellen er basert på min opplevelse og vurdering av bruken av Scrum, og det er mulig at andre personer ville ha gradert noe annerledes. Det er uansett udiskutabelt at mye av Scrum i denne prosjektgruppa brukes, i ulik grad, forskjellig fra den teoretiske beskrivelsen.

Teoridel	Grad
Roller	Medium
Prosess	Høy
Visjon	Medium
Product backlog	Høy
Sprint planning	Medium
Sprint backlog	Medium
Sprint	Høy
Sprint evaluering	Medium
Kunderolle	Høy
Kunnskapsflyt	Høy
Implementasjon	Medium

Tabell 4.1: I hvilken grad de ulike delene av Scrum gjennomføres som anvist i teorien

Resten av dette delkapittelet presenterer mine begrunnelser for graderingen i tabellen.

Roller

Mye av rolleansvaret i dette prosjektet ble utført og tildelt på en ny måte i forhold til hvordan teorien beskrev de ulike rollene. Rollen som Scrummaster ble delt mellom to personer som byttet på å ha denne rollen, i tillegg til at mye av ansvaret til SM var overtatt av en PO. Det var slik at selv om man hadde rollen som SM var man en del av utviklingsteamet og var aktivt med på utviklingsarbeidet. Teamet hadde flere ulike PO å forholde seg, som i ulik grad utførte SM sine plikter. Dette gjaldt i stor grad PO_2 som i tillegg til å lede planleggingsmøter og ha en stor påvirkning i Sprint retrospective, også hadde stor innvirkning på bruken av Scrum i prosjektgruppa. PO_2 var med på å utarbeide hvordan Scrum skulle brukes ved NTNU gjennom dokumentet "Scrum ved NTNU" vist i vedlegg A. I løpet av studiet var det også PO_2 som tok kontakt med meg og forsøkte å forklare hvorfor prosjektgruppa tok i bruk Scrum som de gjorde. Dette kom godt frem i hennes syn på bruken av Jira og burndown charts: "[...] vi har ikke fokus på burndown. [...] Standup kan da brukes til ting som gjøres, uten å bruke tid på burndown. Det er ikke interessant." PO_2 var generelt veldig interessert i hvordan teamet brukte Scrum, og hadde sterke meninger om hvordan Scrum burde bli, og ble, brukt innad i prosjektteamet. Grunnen til at denne teoridelen fikk graden "Medium" var at både PO og SM hadde ansvar og oppgaver ulikt det som ble beskrevet i teorien. PO involverte seg i mye større grad enn teorien beskrev, mens SM

hadde mindre ansvar enn beskrevet i teorien.

Prosess

I studiet viste det seg at prosjektgruppa hadde med alle de overordnede prosessene som tilhører Scrum og tok i bruk sprinter som varte i 2 uker. I tillegg hadde prosjektgruppa utvidet denne biten til å ha med en ekstra del i prosessen: møtene med informasjonsdeling. Siden disse møtene bare ble holdt i starten av prosjektet, i første sprint, var ikke dette nok til å senke graden og denne delen ble vurdert til "Høy".

Visjon

Prosjektet hadde liten grad av inkrementell utlevering. Etter første sprint ble arbeidet så langt presentert til kunden, men ingen inkrementell leveranse av produktet ble utført. Men, helt på slutten av studiet fikk kunden utlevert en URL slik at de kunne utføre mer omfattende testing av løsningen. Dette var den eneste utleveringen, og ingen ny utlevering var planlagt før hele løsningen skulle leveres i midten av 2014. Det at 14C-prosjektet var relativt lite gjorde at inkrementell levering ikke sto i sentrum hos prosjektgruppa. Med et lite prosjekt virket det veldig begrenset hvor mange leveranser man kunne forvente, med stadig ny funksjonalitet. Til tross for liten grad av inkrementell leveranse ble denne delen vurdert til "Medium" fordi prosjektgruppa hadde innhentet en kravliste fra kunden før prosjektet begynte.

Product backlog

Product backlog ble aktivt brukt og oppdatert av de ulike PO'ene i prosjektene ved at nye oppgaver ble lagt til og eksisterende oppgaver ble oppdatert og mer detaljert. I product backlog lå ikke oppgavene i prioritert rekkefølge. Det var under sprint planning at den ansvarlige PO ga beskjed om hvilke av oppgavene som skulle inn i neste sprint. Det var ikke noe automatikk i at det var de som lå øverst i product backlog som skulle med neste gang. Siden prosjektgruppa jobbet på flere prosjekter til samme tid var det en product backlog for hvert prosjekt. Disse ble samlet og lagt inn i Jira, slik at resten av teamet kunne gå inn og se på hvilke oppgaver som lå i product baklog om de ønsket dette. Product backlog fulgte teorien i "Høy" grad.

Sprint planning

PO ledet disse møtene og hadde allerede før møtet bestemt hvor stor del av neste Sprint som skulle gå med til hvert prosjekt. Det var ingen klar todelt oppdeling i under planleggingsmøtene. På PO sitt initiativ gled møtene over i noe estimering, før man gikk tilbake og gjorde endringer i beskrivelsen for oppgavene. I noen av prosjektene, som 14C-prosjektet, var det heller ikke vanlig å gjøre noe form for estimering. I disse møtene var det ikke teamet selv som bestemte hvordan de ville løse oppgavene sine i sprinten. Det kom noen innspill fra teammedlemmene i løpet av møtene med tanker og spørsmål til det PO presenterte, men den innvirkningen de hadde var å estimere oppgavene sine. Selv om PO også her var involvert ved å komme med sitt eget estimat. De endelige estimatene avgjorde om flere oppgaver ble lagt til i sprinten, eller om noen oppgaver ble tatt ut og utsatt til neste sprint. Det at sprint planning ble vurdert til "Medium" var i stor grad på grunn av involveringen PO hadde i disse møtene. De to møtene som ble studert i løpet av studiet var ganske ulike, men PO hadde alltid kontrollen.

Sprint backlog

Sprint backlog ble vurdert til "Medium" fordi bruken av dette artefaktet ikke var god. Selv om artefaktet hadde oppgaver i prioritert rekkefølge ble ikke de høyest prioriterte oppgavene utført først. Graden var også påvirket av at utviklerne med jevne mellomrom jobbet på samme oppgave uten å vite det, fordi de ikke markerte at oppgaven var tatt ved å flytte den i neste kolonne i Jira. Noe av dette dobbeltarbeidet oppsto fordi oppgavene var for dårlig beskrevet. Flere i teamet pekte på at oppgavene var dårlig beskrevet i 14C-prosjektet og at det hadde vært bedre i andre prosjekter, med en annen PO. Selv om det er lett å legge ansvaret på PO spilte også mindlessness en rolle her. Det at utviklerne valgte oppgaver innen sitt spesialfelt og hadde mer fokus på å løse oppgavene enn å bruke Jira på en ansvarlig måte kan tyde på at oppgaveløsingen var rituelt utført. Det lave fokuset på bruken av Jira kan tyde på at teamet egentlig bare hadde lyst til å løse oppgaver, men brydde seg ikke noe særlig om organiseringen rundt oppgavene.

Sprint

De to sprintene som ble overvært i studiet var relativt ordinære og i "Høy" grad etter teorien grunnet jevn, kort lengde og daglige standupmøter. Det at PO_1 var med på disse standupmøtene var mot teorien, men siden hun bare observerte og ikke blandet seg inn i prosessen var ikke dette nok til at

"Sprint" fikk en lavere grad.

Sprint evaluering

Som nevnt tidligere var de to sprint evalueringsmøtene i studiet veldig ulike. Dette i seg selv fikk meg til å stille spørsmål om prosjektgruppa hadde noen tanke om hva de ønsket å få ut av disse møtene. Ved siste sprint evaluering ble det bestemt der og da hvordan møtet skulle være. PO stilte spørsmål til gruppa om de ønsket å gjennomføre retrospective-delen av møtet. Om jeg ikke hadde vært der tror jeg at den delen hadde blitt droppet. I sprint retrospective utførte prosjektgruppa de oppgavene som ble beskrevet i teorien, uten å bruke informasjonen de skaffet seg videre. På den måten var også denne delen utført som et rituale, med manglende dedikasjon fra prosjektgruppa. Dette var meget uheldig siden det er i sprint retrospective at muligheten til å være mindful er størst, og hvor det faktisk blir lagt til rette for å være mindful. "Sprint evaluering" fikk til slutt graden "Medium" og ikke "Lav" fordi møtene ble holdt og utført til en viss grad etter teorien. Selv om den viktigste delen manglet: forberede seg i fremtidige sprinter.

Kunderolle

Dedikasjonen fra kundens side var udiskutabel og både prosjektgruppe og kunderepresentant var fornøyd med kommunikasjonen og samarbeidet mellom partene. Kunderepresentanten var tilgjengelig for jevnlige møter og syntes selv den fikk mye tilbake for alle møtene, både i Sprint review og andre møter med PO_1 og T_{PO}. Det kom også frem at kunderepresentanten skulle ønske den hadde noe bedre teknisk grunnlag for enklere å forstå muligheter og valg i forbindelse med den tekniske løsningen.

Kunnskapsflyt

Prosjektgruppa forsøkte å legge godt til rette for kunnskapsflyt blant annet ved å arrangere egne møter for kunnskapsdeling, og ved å ta kommunikasjonen med kunden gjennom PO-rollene (for å minske det tekniske gapet mellom partene). Kunnskapen innad i utviklingsteamet ble noe spesialisert, slik at hvert teammedlem hadde dyp kunnskap på hvert sitt felt, men det at teammedlemmene også påtok seg oppgaver utenfor sitt område viste at overførbarheten i oppgaver allikevel var tilstede.

Implementasjon

Selve Scrum-metoden ble tatt i bruk fordi nyansatte med gode erfaringer

med bruk av metoden fikk solgt inn ideen. Andre mulige metoder ble ikke undersøkt for å finne ut hva som passet best: Scrum ble tatt i bruk som eneste alternativ. I begynnelsen ble Scrum brukt som beskrevet i teorien, men etterhvert ble det gjort endringer på bruken av metoden for å passe prosjektgruppa bedre. Det at prosjektgruppa tok i bruk Scrum fordi "det var det som var i vinden" gjør at denne delen vurderes til å følge teorien i "Medium" grad, siden det er en vital del når man skal ta i bruk en ny metode at den passer til settingen. I dette tilfellet var man heldig at Scrum i stor grad kan tilpasses de fleste situasjoner og klarte i løpet av årene å gjennomføre denne tilpasningen til en viss grad. Man kan si at Scrum ble tatt i bruk relativt mindless, men at prosjektgruppa senere utviklet og modnet metoden mindfully.

4.2 Hvorfor oppsto avvikene?

Hovedgrunnen til at bruken av Scrum i dette prosjektet avvek fra teorien var rollen PO spilte. Prosjektgruppa forsøkte å mindully tilpasse Scrum etter rollen og ansvaret som PO hadde ved IT-avdelingen. Denne rollen fikk vel mye ansvar og kontroll slik at prosjektgruppa til slutt utførte enkelte deler av Scrum rituelt, uten noe særlig mening med det de gjorde. I løpet av studiet forsøkte jeg å finne ut hvorfor PO involverte seg så mye som de gjorde men fikk ikke noe skikkelig svar. Spørsmål angående dette temaet ble stilt til både POene selv og teamet, men svarene jeg fikk var i all hovedsak "Det har bare blitt sånn". Selv om jeg ikke kom frem til noe eksplisitt svar på hvorfor PO involverte seg i så stor grad, blir det i dette avsnittet presentert noen ideer om hva som kan ha vært grunnen. Organiseringen av IT-avdelingen ved NTNU gjorde at alle PO spilte en viktig rolle i hvordan avdelingen jobbet. PO var med på møter med ledelsen hvor man planla hvilke prosjekter som måtte gjøres når i løpet av det neste året. Når de ulike prosjektene skulle være i fokus ble bestemt ut fra bestillingdatoer og leveringsfrister. Siden PO var med på å legge dette løpet følte POene et ekstra ansvar for prosjektene, og kan være mye av grunnen til at de involverte seg i mye større grad enn beskrevet i teorien om Scrum. POene hadde et sterkt ønske om at prosjektene skulle bli suksessfulle, både med tanke på kostnadsrammer, frister og innhold. Derfor involverte PO seg i alle deler av prosjektene, fra å være med å planlegge prosjektet i detalj til å ha syn på hvordan Scrum skulle brukes innad i teamet. PO_2 var den som opprettet og laget dokumentet som beskrev hvordan Scrum skulle brukes ved IT-avdelingen, og var den PO som var mest interessert i å involvere seg i teamets arbeid.

Som en følge av involveringen til PO ble rollen som SM mindre enn opprinnelig beskrevet i teorien, for eksempel ved at det ofte var PO som dro møter videre og sørget for at prosjektgruppa hadde fokus på riktig del, og at PO var den personen som offisielt avsluttet og startet nye sprinter. Dette vistes godt i evalueringen av sprintene, hvor det var PO som skapte diskusjoner rundt hva man kunne forbedre i neste sprint ved å komme med spesifikke forslag. Dette viser også at PO involverte seg i vel stor grad. Siden ingen PO var med på selve utviklingen av webløsningen burde det ha vært utviklingsteamet selv som kom med forslag til hva de kunne gjøre bedre i neste sprint. Det var teamet som satt med størst innsikt i egen arbeidsdag mens POene sto på utsiden og kom med forslag basert på hvordan de opplevde situasjonene. Når dette er sagt kunne teamet ha vært flinkere til å evaluere seg selv og forplikte seg mer til mulige forbedringer. I løpet av studiet hadde denne delen av sprint evalueringen gått over til å bli mer som et rituale, hvor man

fortalte hva man syntes hadde gått bra med sprinten, og hva som ikke hadde gått så bra. Og det var det. Det var veldig lite fokus på å faktisk innføre og gjennomføre forbedringer i senere sprinter.

Slik SM var i dette prosjektet var rollen uten særlig ansvar og spesifikke oppgaver knyttet til rollen. Den reelle forskjellen på SM og utviklerne i dette studiet var at SM satte i gang standupmøtene. Siden PO allikevel var med på disse møtene kunne også dette ansvaret ha gått til PO, slik at SM kunne ha vært utvikler på fulltid. PO hadde i dette studiet allerede tatt over mange av oppgavene til SM, for eksempel med å lede sprint planning og sprint evaluering, slik at en komplett sammenslåing av PO og SM hadde vært naturlig. På denne måten kunne PO ha fokusert på PO og SM oppgaver og latt teamet utføre sine oppgaver uten innblanding, som for eksempel å komme med egne forslag til forbedringer i sprint retrospective. Når det er sagt var ikke hovedproblemet i sprint retrospective at PO kom med forslag til forbedringer, men at ingen av forslagene ble tatt med videre og brukt i neste sprint. Spring retrospective var i dette studiet drevet som et rituale, hvor man gjorde mye av det som var beskrevet i teorien (analysere hva som hadde gått bra, og hva som ikke hadde gått så bra) men uten å bruke denne informasjonen til for eksempel å forbedre seg i neste sprint. Dermed ble denne prosessen ekstra arbeid som ikke ga noe tilbake til teamet. Det at deler av Scrum blir drevet som et rituale kom også frem i forstudiet som ble utført høsten 2013. Der var det standupmøtene som ble drevet rituelt ved at teamet hadde fokus på å bruke så lite tid som mulig, uten å bry seg nevneverdig om innholdet i møtene.

Når det gjelder delen om "Visjon", hvor mangelen var den jevne, inkrementelle utleveringen av produktet, var dette påvirket av måten IT-avdelingen organiserte seg selv og størrelsen på prosjektene. I løpet av studiet var prosjektgruppa alltid involvert i minst 2 prosjekter i tillegg til at de hadde driftsoppgaver som måtte gjøres. Flere små prosjekter gjorde at inkrementell utlevering ikke var like aktuelt, siden prosjektene ikke var store nok til å få mer enn en utlevering med ny funksjonalitet. Med flere mindre prosjekter varierte arbeidsmengden på prosjektene fra sprint til sprint. I første sprint var det 14C-prosjektet som hadde fokus og man holdt et Sprint review med kunderepresentanten ved slutten av sprinten. I den andre sprinten var det andre prosjekter som hadde prioritet over 14C-prosjektet. Dette gjorde at arbeidet som ble utført i de ulike prosjektene var av veldig ulik mengde fra sprint til sprint og den inkrementelle utleveringen ble vanskelig. Denne typen organisering gjorde også at teamet fikk en manglende eierfølelse av produktene. Det var veldig få innad i teamet som hadde kunnskap om hvorfor NTNU Vitenskapsmuseet ønsket å få utviklet en webløsning for bestilling av ^{14}C-datering.

Oppgavene i sprint backlog ble laget og beskrevet av PO, selv om teorien sier at det er teamet selv som bryter opp de overordnede oppgavene gitt av PO slik at teamet selv bestemmer hvordan de skal jobbe i en sprint. Flere i teamet pekte på at mange av oppgavene var dårlig formulerte slik at det var "[...] vanskelig å vite hva som skulle gjøres i en oppgave" - [T]. Noen teammedlemmer mente dette gjorde at mer arbeid ble gjort på en oppgave enn det oppgaven inneholdt, slik at de også i praksis hadde utført en annen oppgave uten at de hadde kontroll over dette. Den andre oppgaven lå da fremdeles under "To Do" i Jira, slik at det så ledig ut for andre teammedlemmer. I løpet av studiet viste dette seg flere ganger ved at teammedlemmene gjorde dobbeltarbeid, grunnet dårlig kommunikasjon. Dette kunne vært unngått med grundigere beskrivelser av oppgavene i sprint backlog men da burde disse blitt skrevet av noen med større innsikt i utviklingsprosessen, for eksempel et teammedlem eller SM.

Implementasjonen av Scrum hos IT-avdelingen ved NTNU foregikk uten å utforske andre muligheter. Ideen om Scrum ble introdusert av nyansatte som hadde brukt metoden hos tidligere arbeidsgivere og ønsket å ta i bruk metoden også når de kom til IT-avdelingen. Gode erfaringer med metoden på den tidligere arbeidsplassen gjorde at man trodde man ville få like gode erfaringer ved å ta i bruk metoden ved IT-avdelingen. Scrum var for tiden den metoden som "[..]var i vinden" - [T] og litteraturen konkluderte med at Scrum var det nye og store [31, 40]. Flere bedrifter tok da i bruk metoden som et symbol til omverdenen for å vise at de var til å stole på [17], uten mye tanke over om metoden faktisk passet. Dette skjedde også hos IT-avdelingen, men her klarte de etterhvert å tilpasse metoden slik at den fungerte bedre til denne settingen. Man kan si at de delene av metoden som ble tilpasset prosjektgruppa og dens setting, kom som følge av mindful-tenking. Andre deler av metoden gled mer og mer over til å bli ritualer, som følge av ubevissthet og mindlessness i prosjektgruppa.

4.3 Kunderollen

Kunderepresentanten fra NTNU Vitenskapsmuseet var valgt ut til å representere kundegruppen i forbindelse med utviklingen av den nye webløsningen. Basert på samtaler med denne representanten hadde det fungert fint å representere en stor kundegruppe, bortsett fra noen problemer med å få tilbakemeldinger på det som var gjort på løsningen. Kunderepresentanten syntes det var uheldig at hun sent i utviklingsprosessen kom med nye ønsker fra resten av kundegruppen, slik at utviklingsteamet måtte gjøre om mye av arbeidet sitt. Hos kunden var det slik at kunderepresentanten oppfylte med kravene til involvering og dedikasjon som etterlyses i teorien, mens resten av kundegruppen ikke var klar over hvor viktig kundens involvering i utviklingsarbeidet var. I prosjektet ble disse nye kravene løst relativt fort og skapte få problemer, mye takket være et tett samarbeid med kunderepresentanten gjennom hele utviklingsprosessen.

Kunderepresentanten ble involvert i den grad at hun forholdt seg til GUI og bestemte detaljer i løsningen, med tanke på både utseende og sideflyt, og hadde lite involvering i bakgrunnsprosessene i webløsningen. Kunderepresentanten uttalte selv at hun skulle ønske hun hadde større teknisk innsikt, for å lettere kunne se mulige løsninger. Med større teknisk innsikt ville kunderepresentanten kanskje også ha følt at hun kunne påvirke den endelige løsningen i større grad. Et større teknisk grunnlag for kunden ville resultert i et bedre produkt: en mer involvert kunde med god innsikt ville kunne ta diskusjoner vedrørende tekniske løsninger på en god måte sammen med prosjektgruppa, og kunderepresentanten kunne ha vært enda mer presis i beskrivelsene og kravene til produktet. I 14C-prosjektet var kravspesifikasjonen satt sammen av krav til innhold og utseende på GUI. Prosjektgruppa involverte dermed kunderepresentanten på disse delene i utviklingen mens de tok andre valg selv, for eksempel angående sideflyt og sending av bakgrunnsinformasjon. I tråd med teorien ble kunderepresentanten involvert der hun gjorde minst skade [22], noe som ikke gjenspeiler smidige metoders promotering av viktigheten med kundeinvolvering. Her ble kunden involvert på detaljnivå i samme grad som brukerne ble i de tradisjonelle utviklingsmetodene. For å sikre at produktet falt i smak hos de fremtidige brukerne tok kunden selv på seg oppgaven med å teste løsningen med brukerne, både ved bruk av enkle prototyper og webløsningen utviklet av IT-avdelingen ved NTNU.

Det var også interessant at teamet selv ikke hadde mye informasjon om hensikten med produktet og hvorfor NTNU Vitenskapsmuseet ønsket seg denne løsningen. Utviklerne forholdt seg til sprint backlog uten å stille

spørsmål om disse dypereliggende grunnene, unntatt når de ble introdusert til problemstillingen: "Kanskje resten av teamet også skulle ha vært nede på laboratoriet og sett hvordan de jobber der" - [T]. Med større innsikt i hvorfor NTNU Vitenskapsmuseet ønsket den nye løsningen ville utviklerne hatt et bedre utgangspunkt for å lage en god løsning som tilfredsstilte kravene til kunden, og hadde ikke vært så avhengig av alle møtene med kunderepresentanten som ble holdt i løpet av studiet.

4.4 Kunnskapsflyt

Møtene mellom kunderepresentanten og PO fra prosjektgruppa ble holdt jevnlig og ofte, selv om man mot slutten av prosjektet hadde færre møter grunnet lavere prioritering på 14C-prosjektet. Det kan tyde på at det var et gap mellom kunderepresentanten og prosjektgruppa med tanke på teknisk kunnskap. Kunderepresentanten pekte på at hun gjerne skulle hatt større teknisk kunnskap i møtene med prosjektgruppa slik at hun lettere kunne se løsninger og muligheter. Når det er sagt virket det som at kommunikasjonen mellom kunderepresentanten og PO var god: begge var fornøyde med dedikasjonen og kommunikasjonen mellom de to partene. Under sprint review kom ikke kunderepresentanten med så mange innspill og spørsmål som forventet, noe som kom av forskjellen i teknisk ekspertise mellom henne og prosjektgruppa. I dette studiet var det kun gjennom sprint review at hele prosjektgruppa hadde kontakt med kunderepresentanten, ellers var det slik at PO hadde jevn kontakt med kunden og formidlet ønsker og krav videre til utviklerne. Det er klart at en to-ledds kommunikasjon mellom utvikler og kunde hadde vært enklere enn en tre-ledds kommunikasjon mellom kunde, PO, og utvikler, men da hadde man også vært avhengig av å minske gapet i teknisk kunnskap mellom kunden og utviklerne slik at de kunne kommunisere effektivt. Det ekstra leddet åpner for en ekstra mulighet for misforståelser og feiltolkninger.

Et forstyrrende moment i kommunikasjonen mellom kunderepresentanten og utviklerne var, i dette studiet, det faktum at utviklerne var spesialiserte og arbeidet på hver sin del i utviklingsarbeidet. Kunderepresentanten var da tvunget til å forholde seg til mange ulike personer ved diskusjoner angående løsningene i stedet for å forholde seg til en PO (i dette studiet: forholde seg til to PO). Spesialisering av utviklerne går imot smidig utvikling hvor man kan risikere at en oppgave bare kan utføres av en bestemt person. Dette kan fort føre til flaskehalser som hindrer utviklingen i å være smidig. Et viktig begrep innenfor smidig utvikling var "embrace change" [5, 23], noe som motarbeides ved å ta i bruk spesialisering av utviklerne. Det at alle utviklere kan utføre alle oppgaver fungerer godt for mindre prosjekter, men når man kommer inn på større prosjekter vil dette skalere veldig dårlig. Ved å bruke spesialisering av utviklere er man avhengig av en god kunnskapsflyt mellom de ulike utviklerne, slik at man får utnyttet ekspertisen best mulig. I dette studiet ble dette gjort ved egne møter hvor utviklerne møttes og delte ideer og ny kunnskap de hadde tilegnet seg. I tillegg ble koordineringsverktøyet Jira brukt for enklere å holde styr på hva de ulike utviklerne arbeidet med og legge til rette for kunnskapsflyt innad i teamet. Som nevnt tidligere hadde

dette varierende suksess fordi utviklerne ikke alltid oppdaterte oppgavene i Jira når de for eksempel begynte på en ny oppgave.

I løpet av studiet kom det også frem at kommunikasjonen mellom PO og resten av prosjektgruppa var meget hyppig. PO var sterkt involvert i møtene som ble holdt, i alt fra standup til sprint evaluering. PO ønsket å få et godt produkt, men gikk vel langt utenfor sitt ansvarsområde for å sørge for dette når de begynte å blande seg inn i hvordan teamet arbeidet i sprinten. Om det skal fortsettes med bruk av PO på denne måten innad i prosjektgruppa kunne PO med fordel ha vært en del av utviklingsteamet. I dette tilfellet ville da T_{PO} vært en god kandidat til en slik rolle. Denne personen hadde innsikt i hvordan teamet fungerte i løpet av en sprint, i motsetning til PO_1 eller PO_2 som sto på utsiden og hadde sterke meninger på hvordan teamet skulle arbeide. En annen mulighet er å slå sammen rollen som PO og SM, og la denne rollen stå utenfor utviklingsdelen. På den måten vil den nye rollen være en del av teamet på grunn av SM-ansvaret, mens den opprettholder kundekontakten gjennom PO-ansvaret.

PO kan ikke fortsette å bruke sin rolle slik det ble gjort i dette studiet. Kanskje er det, som Berglund [8] presenterte, på tide å fjerne rollen som PO og la hele teamet dele denne rollen. Ved å la teamet selv påta seg denne rollen kunne hele prosjektgruppa fått et nærmere forhold til kunden og produktet som ble utviklet. Kommunikasjonen mellom partene ville blitt bedre og en større vilje innad prosjektgruppa for å lage et godt produkt ville oppstå. Ved å flytte PO-rollen på denne måten ville man også løst problemet med at utenforstående personer hadde innspill på hvordan teamet selv jobbet i sprintene. En stor fare identifiseres ved denne løsningen: mange tekniske begreper brukt av utviklerne kan være nye og ukjente for kunderepresentanten. PO sin rolle var å fungere som et mellomledd mellom kunde og utvikler, dermed må man endre mentaliteten ved en slik løsning. Utviklerne må være klar over at kunden har mindre teknisk ekspertise enn de selv og må derfor tilpasse sin dialog mot kunden slik at løsninger og muligheter blir forstått av begge parter. Det åpner også for at kunderepresentantene bør ha noe høyere teknisk ekspertise enn tilfellet i dette studiet, men det er heller usannsynlig at en bedrift vil investere mye tid og penger i en slik opplæring. Løsningen kan da være at kunderepresentanten automatisk oppnår høyere teknisk innsikt ved å tilbringe tid med utviklerne. Et enda tettere samarbeid med prosjektgruppa kan gi fordeler til begge parter: høyere teknisk innsikt hos kunderepresentant som fører til bedre tilbakemeldinger og mer givende dialog med prosjektgruppa. Det vil da bli enklere for prosjektgruppa å forstå akkurat hva kunden ønsker ut av prosjektet, og enklere for kunden å forstå

valgene som blir tatt i forbindelse med løsningen.

Det at hele teamet deler ansvaret til PO har mange fordeler, men er en urealistisk mulighet siden det skaper mye ekstra ansvar for alle i teamet. Grunnen til at dette ekstra ansvaret gjør løsningen urealistisk er at flere av utviklerne i prosjektet ikke hadde lyst på andre oppgaver, de jobbe som utviklere. Den beste løsningen er dermed å slå sammen PO og SM, og la en person dele dette ansvaret mens resten av teamet kan fokusere på utviklingen av nye produkter. I mindre prosjekter, som 14C-prosjektet, går det bra å dele så mye ansvar på en person, men denne løsningen skalerer ikke godt til større prosjekter. I store prosjekter med mange prosjektdeltakere vil det i praksis være umulig å dele så mye ansvar uten å være en høy risiko for prosjektet, slik at man bør holde seg til de faste rollene beskrevet i teorien.

4.5 Scrum som rituale

I dette prosjektet ble Scrum først tatt i bruk relativt mindless, ved at man tok i bruk metoden fordi "alle" andre gjorde det, før man senere utviklet og tilpasset metoden til å passe prosjektgruppa. Mye av denne tilpasningen ble gjort på bakgrunn av organiseringen og hvordan IT-avdelingen ble drevet: man hadde gode grunner og baktanker ved å avvike fra teorien. Men i løpet av studiet viste det seg at prosjektgruppa også brukte noen sider av Scrum rituelt hvor prosessene og metodene ble utført med en viss automatikk uten å tenke over som egentlig ble gjort. Dette gjaldt for eksempel Sprint retrospective i hovedstudiet og bruken av standupmøter i forstudiet. Dette kan tyde på at man manglet en grunn for å utføre disse aktivitetene: man fikk ikke lenger noe tilbake av å utføre disse, som for eksempel diskusjonen om forbedringer som ikke endte i noen endringer i evalueringen av foregående sprint. Siden det var i retrospective-delen av sprint evalueringen at man hadde mulighet til å mindfully gjøre endringer på sin bruk av metoden kan det få store konsekvenser om denne delen fortsetter å miste sin mening.

Det at Scrum ble tatt i bruk basert på hvilke metoder andre prosjekter tok i bruk på den tida, uten å utforske flere muligheter, gjorde at prosjektgruppa kanskje gikk glipp av muligheter som kunne ha passet bedre til gruppa. Både Martha Feldman & James March [17] og Michael Power [35] peker på at det innad i organisasjoner ofte kan være slik at riktig type informasjon blir presentert slik at man har grunnlag for at valgene som ble tatt var de riktige. Problemet her var bare at mye av denne informasjonen ble innhentet etter valgene ble gjort, og man kun presenterte den informasjonen som underbygde valget: informasjon som gikk mot valget ble bevisst tilbakeholdt. Slik type informasjon ble presentert for å "bevise" for utenstående personer at valget som ble tatt var det riktige. I dette prosjektet kom det frem gjennom deltakerne at Scrum ble tatt i bruk fordi nyansatte hadde forsøkt metoden i sin forige jobb og opplevde gode resultater der. På den tiden var Scrum et "buzzword"[1] som mange bedrifter ønsket å forbinde seg med, for å vise verden at de var oppdaterte på metoder og var tidlig ute med å ta i bruk det nyeste innen fagfeltet. I løpet av studiet var jeg ute av stand til å finne ut i hvor stor grad dette ble gjort i prosjektgruppa, men jeg fikk også noen indikatorer. Man hadde bestemt seg for å ta i bruk Scrum, uavhengig av hvor godt det passet inn i hverdagen til prosjektgruppa: "Scrum egner seg dårlig når vi har driftsoppgaver. Disse er så varierende [i omfang] og vanskelige å estimere" - [T]. Dette kan tyde på at man ikke analyserte hvordan Scrum ville passe inn.

[1] Buzzword: ord som er ofte går igjen både i media og innad i samfunnet

Prosjektgruppa har lenge hatt ansvar for driftsoppgaver. Kanskje hadde det vært andre metoder som hadde fungert bedre, for eksempel Kanban som ikke har med estimeringsdelen på samme måte som Scrum har. Med så mange avvik fra teorien i bruken av Scrum var dette også et tegn på at Scrum i utgangspunktet ikke var det beste alternativet for prosjektgruppa. Da Scrum først ble tatt i bruk gikk man tett på teorien og fulgte den, men man måtte etterhvert gjøre tilpasninger slik at Scrum passet bedre for prosjektgruppa. Det at man forsøkte å bruke Scrum uten avvik fra teori, for senere å gjøre endringer tyder på at det ikke ble hentet inn nok informasjon som kunne hjelpe IT-avdelingen med å ta et informativt valg angående ny utviklingsmetode. På denne måten ble Scrum tatt i bruk mindless, men man ble mer mindful etterhvert og skaffet informasjon som gjorde at man kunne tilpasse metoden for å passe bedre inn i prosjektgruppas setting.

Scrum ble altså tatt i bruk for å få gode resultater, ikke nødvendigvis for at det var denne metoden som passet best inn i prosjektsettingen. Senere klarte denne prosjektgruppen å tilpasse mye i bruken av Scrum til sine behov (mindfully), i motsetning til i pilotstudiet hvor Scrum i stor grad ble brukt som et rituale og etter hvert avviklet til fordel for en annen utviklingsmetode. I 14C-prosjektet var det enkelte deler av Scrum som ble utført rituelt, som bruken av Sprint retrospective. Her ble det gjort en rundgang hvor teamet fortalte hva som hadde vært bra og dårlig med foregående sprint, men tok ingen aktive grep for å bedre seg i de neste sprintene. Det at bruken av Scrum avvek fra teorien var ikke noe problem, så lenge denne tilpasningen skjedde mindfully med en tanke bak valgene slik at det hadde en hensikt. Et eksempel på et positivt avvik som skjedde mindfully var bruken av kunnskapsdelingsmøter. Disse møtene ble opprettet og holdt fordi teamet mente at man ved å bruke litt ekstra tid på et slikt møte ville kunne oppnå enda bedre resultater i forbindelse med utviklingsarbeidet. Dette til tross for at Scrum og andre agile metoder ønsker at fokuset skal ligge på utviklingsdelen. Dette viste seg å fungere så bra for teamet at slike møter ville fortsette i fremtidige prosjekter.

Dette leder inn på en ide om at metoder som blir tatt i bruk mindless har lettere for å foregå rituelt enn metoder tatt i bruk mindfully. Uten å ha en analyse av mulighetene er det enkelt å gjøre et valg bare fordi andre har gjort det samme valget, noe som fort kan lede til at metoden brukes rituelt uten at man tenker over hvordan og hvorfor man bruker metoden. Prosjektgruppa i dette studiet har klart å gå fra å ta i bruk metoden mindless, til å bli relativt mindful med tanke på den nåværende bruken av metoden. Nå som Scrum begynner å bli så utbredt at metoden brukes i veldig mange forksjellige vari-

anter er det enda viktigere enn før å være mindful om, og eventuelt når, man skal ta i bruk Scrum. Siden metoden blir brukt med store forskjeller fra prosjekt til prosjekt vil det være vanskelig å finne en versjon som passer godt til et nytt prosjekt. Det vil være nødvendig å utforske sine muligheter, om Scrum faktisk kan passe inn i organiseringen og prosjektene man har, samt hvordan man skal ta i bruk Scrum. En vurdering må gjøres om man skal følge Scrum helt etter teorien eller følge teorien i liten grad. Det viktigste er at disse vurderingene blir gjort mindfully slik at man gjør det som er best for prosjektgruppa. Kanskje må man da gjøre vurderingen isolert slik at man ikke får påvirkninger og ideer fra hva andre har gjort, men tenker mest på hvordan metoden kan passe til sin egen situasjon. Siden det kan være vanskelig å se hvordan alle deler av metoden bør bli brukt i et prosjekt, vil det kunne være nødvendig å gjøre endringer etterhvert som man ser ting som ikke fungerer.

Kapittel 5

Konklusjon

Metoder med teoretiske beskrivelser har eksistert i lang tid, i mange ulike disipliner, og det har ofte vært slik at praksis ikke følger teorien. At dette også gjelder innen smidige metoder kommer ikke som noen overraskelse. Basert på kapittel 4 presenteres tre hovedpoenger ved bruken av Scrum i dette studiet.

5.1 Scrum som rituale

Det som var spesielt i dette studiet var at det var mange aktiviteter og deler av Scrum som ikke fulgte teorien. Selv om alle de obligatoriske aktiviteten var tatt med, avvek innholdet i mange av aktivitetene fra teorien. Disse avvikene kom som et resultat av både mindful og mindless tenking fra prosjektgruppa. Ved mindful tenking avvek gruppa fra teorien med vilje i et forsøk på å tilpasse metoden til sin situasjon, mens mindless tenking gjorde at gruppa ufrivillig gled bort fra teorien. Her gikk prosjektgruppa bort fra teorien uten å kanskje være klar over det selv, som om prosessbitene var så naturlig å gjøre at de gradvis gled over til å bli et rituale hvor man også gled bort fra den teoretiske beskrivelsen av Scrum. Et godt eksempel var sprint evaluering, hvor man i stor grad gjorde hovedaktiviteten beskrevet i teorien (finne ut hva som gikk bra i forrige sprint, og hva som ikke gikk så bra) mens man ikke brukte denne informasjonen videre til å forbedre seg i fremtidige sprinter. Dette var en aktivitet som ble utført uten å tenke noe mer på hvorfor den ble gjort og hva prosjektgruppa ønsket å få ut av den. Slik denne aktiviteten ble brukt i prosjektet kunne den ha blitt fjernet helt forid den i praksis ikke tjente noen hensikt. Dette var meget uheldig for prosjektgruppa, siden det er i denne aktiviteten at Scrum faktisk oppfordrer brukerne av metoden til å være bevisste og tilpasse metoden til sin situasjon.

Ved å ta i bruk en metode, eller deler av en metode, mindless er det lettere at metoden går til å bli et rituale. Da er det et problem at de fleste bedrifter mener og tror at de selv er mindful, selv om de skulle vise seg å være mindless. I løpet forstudiet og hovedstudiet viste det seg at begge prosjektene mente selv at de var mindful og klar på hvorfor de utførte aktivitetene som de gjorde, mens det i realiteten var flere aktiviteter som ble utført mindless. For å hindre at aktiviteter går til å bli rituelle handlinger burde prosjektgruppa med jevne mellomrom analysere sin bruk av Scrum som metode og se på hva de ulike aktivitene har gitt tilbake til gruppa. I hovedstudiet gikk prosjektgruppa fra å være mindless ved introduseringen av metoden til å bli mer mindful ettersom tiden gikk, selv om den også ved slutten av studiet hadde innslag av mindlessness. Derfor er det ingen grunn til at prosjektgruppa ikke kan gå andre vegen, fra å være mindful til å bli mindless. Det kan til og med være lettere. Det vil være viktig å analysere sin bruk av metoden gjennom hele tidsløpet, ikke bare ved implementeringen av metoden. Derfor er det viktig at sprint retrospective utføres etter hver sprint slik at prosjektgruppa kontinuerlig kan analysere sin bruk av Scrum. Sprint retrospective er en av de viktigste delene i Scrum som må brukes mindfully, og ikke bli en rituell aktivitet.

5.2 Kunderollen

Det viste seg i løpet av studiet at selv om kunderepresentanten forsto viktigheten av god og jevn kontakt med utviklerne, var dette vanskeligere å forstå for resten av kundegruppen. Denne gruppen kom veldig sent med sine tanker og meninger om produktet, til kunderepresentantens skuffelse. Når det gjaldt kommunikasjonen mellom kunderepresentanten og prosjektgruppa var begge parter fornøyde, selv om det var stor forskjell i den tekniske ekspertisen mellom partene. Dette samsvarte med teorien som sa at de tidligere kundene hadde et større teknisk grunnlag siden disse gjerne var andre utviklere [22, 38]. Denne forskjellen i teknisk ekspertise bidro også til at kunderepresentanten ble inkludert i valg som hadde med GUI og andre detaljer å gjøre, i motsetning til det store bildet med tanke på for eksempel bakgrunnsprosesser i webløsningen. En bekreftelse på at kunden ble inkludert hvor *"[...] de negative konsekvensene av deres krav er så små som mulig"* [22, s. 14]. I bruken av Scrum i dette studiet var ikke kundeinvolveringen på det store nivået som ble presentert i teorien. Teorien forteller om viktigheten ved å ha en god og jevn kommunikasjon med kunden, noe som

ikke ble fulgt da kunden kun ble konsultert vedrørende GUI. Ekstra kommunikasjon har lite hensikt når kunden allikevel bare ble konsultert vedrørende grunnleggende valg, akkurat slik som brukerne ble involvert i de tradisjonelle metodene.

Dette, kombinert med teamets manglende innsikt i bakgrunnen for at NTNU Vitenskapsmuseet ønsket en ny webløsning, viser noen av utfordringene i forbindelse med kunnskapsflyt og kommunikasjon mellom partene. Rollen som PO var ment å forenkle denne kommunikasjonen og legge til rette for god kunnskapsflyt, men dette ekstra leddet mellom kunde og utvikler kan også virke mot sin mening ved å åpne for misforståelser og feiltolkninger i et ekstra ledd. I mindre prosjekter, som 14C-prosjektet, mener jeg at rollen som PO kunne ha gått til en SM som var en del av teamet og hadde enda bedre innsikt i produktet. Dermed kunne kunderepresentanten ha hatt en kontinuerlig dialog direkte med en utvikler i teamet (SM), uten å måtte gå gjennom et ekstra ledd i kommunikasjonen. Det vil da være veldig viktig at SM lærer seg kundekommunikasjon på samme nivå som PO har i dag: vite at kunden har mindre teknisk ekspertise og ta dette med i betraktningen under kommunikasjon med kunden. Jeg mener at dette vil være vanskelig å gjennomføre i større prosjekter, hvor man er avhengig av ulike roller for å fordele det massive arbeidet på mange personer. Der vil en sammenslåing av SM og PO vil skape for mye arbeid, trykk og ansvar på en rolle. I større prosjekter bør man derfor fortsette med bruken av alle rollene som presenteres i Scrum. Da må prosjektgruppa være klar på at rollene må holde seg til oppgavene sine og ikke gjøre som i dette hovedstudiet og gå langt utenfor sitt ansvarsområde.

5.3 Forslag til endringer på Scrum

Det er ikke rart at Scrum er så utbredt når veldig mange bedrifter har sin egen variant av metoden. Med så mange forskjellige utgaver av Scrum er det også interessant å se at alle kaller det for "Scrum", slik at det er noen kjente trekk ved det og noe man kan skryte av utad til omverdenen [17]. Slik det er nå fungerer Scrum mer som et fleksibelt skall (hvor innholdet bestemmes selv) enn en utviklingsmetode. Kanskje er det da også på tide å presentere Scrum som det det er: et skall med noen faste aktiviteter hvor detaljene i disse aktivitetene bestemmes av bedriftene selv. Fra Scrum ble introdusert har det gjennom årene kommet mindre, detaljerte endringer på metoden, for eksempel ved å spesifisere eksplisitt at møter kan ende før den planlagte lengden om hensikten med møtet har blitt oppfylt [43]. Etter mange år med

små endringer og oppdateringer er det på tide å ta et steg i en annen retning, siden Scrum allikevel ikke blir fulgt til hver minste detalj i praksis. Det vil være viktigere å presentere hva som er hensikten med aktiviteten i stedet for å beskrive detaljert hva man skal gjøre. Dette kan også hjelpe bedrifter å bli mer mindful ved valg og bruk av Scrum som utviklingsmetode ved å tvinge de til å tenke gjennom hva man ønsker med å ta i bruk Scrum.

Basert på informasjonen presentert tidligere i dette kapittelet har jeg kommet frem til noen konkrete forslag til endringer på Scrum som metode. Hovedideen er at Scrum nå finnes i veldig mange forskjellige former, at det kan være vanskelig for bedrifter å finne ut hvordan de bør bruke Scrum for å få mest mulig ut av metoden. Som sagt tidligere i avhandlingen bør bedrifter være mindful og tenke gjennom valgene de tar for å få mest mulig ut av metoden, og de må selv finne ut hvordan de vil følge de ulike delene av Scrum. For å gjøre dette litt enklere for bedrifter som er nye til Scrum presenterer jeg noen forslag til hvordan Scrum bør brukes basert på hvilken situasjon bedriften befinner seg i, siden det ikke er slik at opprinnelig, teoretisk Scrum passer til alle situasjoner. Figur 5.1 viser forslag til hvordan Scrum bør brukes i ulike bedrifter.

Type prosjekt	Tilpasning i Scrum
Lite prosjekt, mindless	Bred kunnskap (ikke spesialisering), mindre fokus på inkrementell levering av produktet, sammenslått PO og SM rolle, en fast oppskrift som skal følges i stor grad (ikke ulikt i dagens Scrum)
Lite prosjekt, mindfull	Bred kunnskap (ikke spesialisering), mindre fokus på inkrementell levering av produktet, sammenslått PO og SM rolle, Scrum som et skall hvor innholdet bestemmes selv og tilpasses prosjektet
Stort prosjekt, mindless	Spesialisering, viktig med inkrementell levering, holde på de tradisjonelle rollene, en fast oppskrift som skal følges i stor grad (ikke ulikt i dagens Scrum)
Stort prosjekt, mindfull	Spesialisering, viktig med inkrementell levering, holde på de tradisjonelle rollene, Scrum som et skall hvor innholdet bestemmes selv og tilpasses prosjektet.

Figur 5.1: Forslag til situasjonsbasert bruk av Scrum

Som man kan se fra figuren skiller jeg mellom størrelsen på prosjektet, stort eller lite, og om prosjektgruppa er mindful eller ikke i sitt valg av og arbeid med metoden. Det å finne ut om et prosjekt er stort eller lite er noenlunde greit å skille på. For eksempel kan et prosjekt sees på som lite når er satt til å vare mindre enn 4 måneder og stort om det er over 4 måneder. Det er vanskeligere å avgjøre om man er mindful eller mindless. De fleste

berifter vil tro at de selv er mindful, også de som ikke er det. Det som er fint med denne modellen er at de bedriftene som anser seg selv som mindful og følger disse rådene fra tabellen blir tvunget til å være mindful. Ved å ta i bruk Scrum som et skall blir man tvunget til å bestemme innholdet i metoden selv, slik at man presses til å tenke gjennom og analysere valgene sine. Forslagene for tilpasning av Scrum går på om prosjektet bør være spesialisert eller ikke, om inkrementell utlevering skal realiseres, bruken av PO og SM, og bruken av resten av metoden. Dette siste punktet er det mest vage i tabellen. Her går valget på å bruke Scrum som en fast oppskrift, slik Scrum blir presentert i teorien i dag, eller å bruke Scrum som et skall. Ved å bruke Scrum som et skall vil man typisk bli presentert hovedaktivitetene i Scrum (sprint planning, sprint evaluering og lignende) uten noen fast oppskrift på hvordan disse skal gjennomføres. Ved å presentere hva som er hensikten med disse aktivitetene og understreke viktigheten av de, kan prosjektgruppa selv bestemme hvordan de vil utføre aktivitetene for å oppfylle hensikten med de. På denne måten blir prosjektgruppa tvunget til å være mindful, uansett om man faktisk var mindful eller bare mente at man var mindful. Om prosjektgruppa mente at den ikke var mindful ville den gått på en av radene merket "mindless" og brukt Scrum som en fast oppskrift. Ved å legge større vekt på sprint retrospective enn det er i teorien i dag, vil også bedrifter som er mindless kunne bli mer mindful gjennom god bruk av sprint retrospective.

5.4 Begrensninger i oppgaven

Forsker var i dette studiet alene om å utføre forskningen, slik at ressursene satt til oppgaven har vært noe begrenset. I tillegg var forsker avhengig av tilgang til prosjektgruppa for å utført studiet, slik at omfanget ble naturlig begrenset av prosjektgruppas egne tidsfrister og gjøremål. Forstudiet ble utført som en forberedelse til hovedstudiet ved å la forsker opparbeide gode metoder for både innhenting og prosessering av data. Dette ble gjort for at begrensningene skulle få så liten påvirkning som mulig på den endelige rapporten.

5.5 Videre forskning

Selv om jeg i figur 5.1 har kommet med et forslag til hvordan Scrum kan legges opp ut fra gitte situasjoner, vil det være nødvendig med videre studier

for å kartlegge i større grad hvilke situasjoner som bør være med i en slik oversikt og hvordan Scrum bør brukes i de ulike situasjonene. Informasjonen presentert i dette studiet er basert på et studie av et prosjekt som var lite og relativt mindful. Det bør også studeres prosjekter som passer inn i de andre beskrivelsene, for eksempel et stort prosjekt som ikke anser seg selv som mindful. Nye studier kan avdekke ny informasjon som påvirker modellen presentert i figur 5.1, for eksempel ved å avdekke andre aspekter av Scrum som kan variere basert på situasjon eller ved at man må skille mellom enda flere ulike typer prosjekter. Det må også forskes videre på hvordan man kan presentere Scrum som et skall. Hvor mye informasjon skal med, hvilken type informasjon skal inkluderes, skal man ha med noen forslag slik at bedriftene kan få en ide over hvordan ting kan løses?

Referanseliste

[1] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, and Juhani Warsta. Agile software development methods: Review and analysis, 2002.

[2] Pekka Abrahamsson, Juhani Warsta, Mikko T Siponen, and Jussi Ronkainen. New directions on agile methods: a comparative analysis. In *Software Engineering, 2003. Proceedings. 25th International Conference on*, pages 244–254. Ieee, 2003.

[3] Atlassian. Skjermbilde fra oppgavestyringsverktøyet jira, 6. februar 2014.

[4] F Terry Baker. System quality through structured programming. In *Proceedings of the December 5-7, 1972, fall joint computer conference, part I*, pages 339–343. ACM, 1972.

[5] Kent Beck and Cynthia Andres. *Extreme programming explained: embrace change*. Addison-Wesley Professional, 2004.

[6] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. The agile manifesto. *The agile alliance*, 200, 2001.

[7] T. E. Bell and T. A. Thayer. Software requirements: Are they really a problem? In *Proceedings of the 2nd international conference on Software engineering*, ICSE '76, pages 61–68. IEEE Computer Society Press, 1976.

[8] Tim Berglund. First, kill all the product owners, september 2013. URL `http://jz13.java.no/presentation.html?id=159f8623`.

[9] Barry Boehm and Wilfred J Hansen. Spiral development: Experience, principles, and refinements. Technical report, DTIC Document, 2000.

[10] Barry W. Boehm. A spiral model of software development and enhancement. *Computer*, pages 61–72, 1988.

[11] Alistair Cockburn and Jim Highsmith. Agile software development, the people factor. *Computer*, pages 131–133, 2001.

[12] Mike Cohn and Doris Ford. Introducing an agile process to an organization. *Computer*, pages 74–78, 2003.

[13] O. J. Dahl, E. W. Dijkstra, and C. A. R. Hoare, editors. *Structured programming*. Academic Press Ltd., 1972.

[14] Pete Deemer, Gabrielle Benefield, Craig Larman, and Bas Vodde. The scrum primer, 2012. URL `http://www.scrumprimer.org/scrumprimer20_small.pdf`.

[15] Torgeir Dingsyr, Tore Dyb, and Nils Brede Moe. *Agile Software Development: Current Research and Future Directions*. Springer Publishing Company, Incorporated, 2010.

[16] Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and software technology*, pages 833–859, 2008.

[17] Martha S Feldman and James G March. Information in organizations as signal and symbol. *Administrative science quarterly*, 26(2), 1981.

[18] Kristin Foosnæs, 13. februar 2014. Personlig intervju.

[19] Athula Ginige and San Murugesan. Web engineering: An introduction. *Multimedia, IEEE*, pages 14–18, 2001.

[20] Diego Lo Giudice. Forrester: Agile development, februar 2012. URL `http://www.computerweekly.com/opinion/Agile-Development-A-Perfect-Fit-For-Todays-Business-Challenges`.

[21] Fred Grossman, Joe Bergin, David Leip, Susan Merritt, and Olly Gotel. One xp experience: introducing agile (xp) software development into a culture that is willing but not ready. In *Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research*, pages 242–254. IBM Press, 2004.

[22] Morten Hatling and Knut H Sørensen. Social constructions of user participation. *The spectre of participation. Technology and work in a welfare state*, pages 171–188, 1998.

[23] Jim Highsmith and Alistair Cockburn. Agile software development: The business of innovation. *Computer*, 34(9):120–127, 2001.

[24] Rashina Hoda, James Noble, and Stuart Marshall. The impact of inadequate customer collaboration on self-organizing agile teams. *Information and Software Technology*, pages 521–534, 2011.

[25] Ben Hughes. Making scrum work for you, november 2013. URL `http://www.computerworld.com/s/article/9244195/Making_Scrum_work_for_you`.

[26] Andrea A Janes and Giancarlo Succi. The dark side of agile software development. In *Proceedings of the ACM international symposium on New ideas, new paradigms, and reflections on programming and software*, pages 215–228. ACM, 2012.

[27] Heinz K Klein and Michael D Myers. A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS quarterly*, pages 67–93, 1999.

[28] Mikko Korkala, Minna Pikkarainen, and Kieran Conboy. Distributed agile development: A case study of customer communication challenges. In *Agile Processes in Software Engineering and Extreme Programming*, pages 161–167. Springer, 2009.

[29] Luigi Lavazza, Sandro Morasca, Davide Taibi, and Davide Tosi. Applying scrum in an oss development process: an empirical evaluation. In *Agile Processes in Software Engineering and Extreme Programming*, pages 147–159. Springer, 2010.

[30] Jingyue Li, Nils B Moe, and Tore Dybå. Transition from a plan-driven process to scrum: a longitudinal case study on software quality. In *Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement*. ACM, 2010.

[31] Mikael Lindvall, Dirk Muthig, Aldo Dagnino, Christina Wallin, Michael Stupperich, David Kiefer, John May, and Tuomo Kahkonen. Agile software development in large organizations. *Computer*, pages 26–34, 2004.

[32] Nils Brede Moe, Torgeir Dingsøyr, and Tore Dybå. A teamwork model for understanding an agile team: A case study of a scrum project. *Information and Software Technology*, pages 480–491, 2010.

[33] Briony J Oates. *Researching information systems and computing*. Sage, 2005.

[34] Colm O'hEocha, Kieran Conboy, and Xiaofeng Wang. So you think you're agile? In *Agile Processes in Software Engineering and Extreme Programming*, pages 315–324. Springer, 2010.

[35] Michael Power. The audit society: Rituals of verification. *OUP Catalogue*, 1999.

[36] Linda Rising and Norman S Janoff. The scrum software development process for small teams. *Software, IEEE*, pages 26–32, 2000.

[37] Daniel Robey and Sundeep Sahay. Transforming work through information technology: A comparative case study of geographic information systems in county government. *Information systems research*, pages 93–110, 1996.

[38] Winston W Royce. Managing the development of large software systems. In *Proceedings of IEEE WESCON*, 1970.

[39] Kevin Rutherford, Paul Shannon, Craig Judson, and Neil Kidd. From chaos to kanban, via scrum. In *Agile Processes in Software Engineering and Extreme Programming*, pages 344–352. Springer, 2010.

[40] Ken Schwaber. *Agile project management with Scrum*. O'Reilly Media, Inc., 2004.

[41] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*. Prentice Hall Upper Saddle River, 2002.

[42] Ken Schwaber and Jeff Sutherland. The scrum guide, 2011.

[43] Ken Schwaber and Jeff Sutherland. The scrum guide, 2013.

[44] Ken Schwaber and Jeff Sutherland. The definitive guide to scrum: The rules of the game, juli 2013. URL `http://www.scrum.org/Scrum-Guide`.

[45] Scrum Alliance. The state of scrum: benchmarks and guidelines, 2013.

[46] Mali Senapathi. Adoption of software engineering process innovations: The case of agile software development methodologies. In *Agile Processes in Software Engineering and Extreme Programming*, pages 226–231. Springer, 2010.

[47] Kai Stapel, Eric Knauss, Kurt Schneider, and Matthias Becker. Towards understanding communication structure in pair programming. In *Agile Processes in Software Engineering and Extreme Programming*, pages 117–131. Springer, 2010.

[48] Jeff Sutherland. Agile can scale: Inventing and reinventing scrum in five companies. *Cutter IT Journal*, pages 5–11, 2001.

[49] E Burton Swanson and Neil C Ramiller. The organizing vision in information systems innovation. *Organization science*, 8(5):458–474, 1997.

[50] E Burton Swanson and Neil C Ramiller. Innovating mindfully with information technology. *MIS quarterly*, pages 553–583, 2004.

[51] Hirotaka Takeuchi and Ikujiro Nonaka. The new new product development game. *Harvard business review*, pages 137–146, 1986.

[52] Patrick Thibodeau. John deere plows into agile, januar 2012. URL `http://www.computerworld.com/s/article/9223664/John_Deere_plows_into_agile`.

[53] Patrick Thibodeau. Feds turn to agile development as budget cuts loom, januar 2013. URL `http://www.computerworld.com/s/article/9235966/Feds_turn_to_agile_development_as_budget_cuts_loom`.

[54] VersionOne. 7th annual state of agile development survey, 2012. URL `http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf`.

[55] Ina Wagner. A web of fuzzy problems: confronting the ethical issues. *Communications of the ACM*, pages 94–101, 1993.

[56] Geoff Walsham. Knowledge management: The benefits and limitations of computer systems. *European Management Journal*, pages 599–608, 2001.

Vedlegg

Vedlegg A

Scrum ved NTNU

Dette vedlegget er innholdsmessig en blind kopi av dokumentet laget for IT-avdelingen ved NTNU angående deres bruk av Scrum.

Pistasj-Scrum

NTNU driver ikke med Vanilla-Scrum (unmodified Scrum). Web-app teamets egen variant, Pistasj-Scrum, tar utgangspunkt i det som har fungert bra for oss (use what's good, throw out the rest). Teamet kombinerer Scrum med par-programmering og brukerinvolvering, og har opplevd mange gode bi-effekter, inklusive:

- et velfungerende team som fungerer i dag
- kompetanseoverføring mellom teamets medlemmer skjer på en naturlig måte og gjør organisasjonen mindre sårbar
- medvirkning - utviklernes innspill til kravene, og samarbeid med interessentene under hele prosessen gir gode resultater.

Team

- 5-9 medlemmer
- Selvorganiserende - Teamet bestemmer selv hvem som gjør hva og når for å løse oppgaver i sprinten.
- Er ansvarlig for å møte opp på daglig Standup
- Er ansvarlig for gjennomføring av oppgavene i sprinten
- Er ansvarlig for å bryte ned oppgaver i plenum (Sprint Planlegging)

- Er ansvarlig for estimering, med bruk av Story Points, i plenum
- Er ansvarlig for oppdatering av sine oppgaver i Jira, og for å initiere Review steget.
- Er ansvarlig for testing og kvaliteten

Sprint

- Vanligvis 2 uker. Teamets avgjørelse.
- Om sommeren bruker vi gå over til en "jog", som opererer etter Kanban prinsipper.

Standup

- ukedager kl. 8:45 - 9:00
- rundgang med hva du gjorde i går; hva du skal gjøre i dag; eventuelle hindringer; innsikt som kan være nyttig for andre

Om det er behov for diskusjon, det tas etter rundgangen.

Scrum master

Har ansvar for sprint-prosessen, men er en del av teamet. (Se seksjon om Vanilla Scrum vs. Pistasj)
For Web-app teamet betyr dette:

- Sørger for at standup, planlegging, retrospective og review finner sted. (Møteinnkalling - 15 minutt; avbryter diskusjoner som tar fokus bort fra planlegging)
- Sørger for at noen har lagt oppgavene i Jira før Sprinten begynner (30-60 minutter hver Sprint).
- Sørger for at teamet jobber med det de har forpliktet seg til (jage havørn, si ifra når team-medlemmene ikke har fokus på sprint-oppgavene, kjøre review)
- Som oftest krever dette lite tid. Alt som er nødvendig er en bemerkning av og til. Varsle fra om hinder (til [Navn], til Produkteier).

ScrumMaster ansvar på rundgang

Dette har vi testet - både rundgang og ansvarsdeling. Det fungerer hvis de utvalgte er motivert, men det er ingen langtidsløsning. Det er aktuell i perioder, for team som jobber bra sammen.

Product Owner (produkteier)

Har ansvar for produktet.

- kommunikasjon med interessentene (systemeier, øvrige prosjektmedlemmer, brukere)
- bringe frem og kommunisere krav i rett tid
- prioritering av oppgavene i backloggen
- talsmann for brukerne (Pistasj-scrum komponent)

På NTNU kreves det at Product Owner har domene-kunnskap (må kjenne til brukerbehovene og kravspesifikasjoner). Fungerer som mellomledd mellom teamet og interessentene. Ved NTNU er PO-rollen ivaretatt av enten (a) prosjektlederen som er komfortabel med rollen, egnet til å ta tekniske avgjørelser og tilgjengelig nok for teamet, eller (b) et prosjektmedlem som kan stå for den tekniske prosjektledelsen.

Task-board

Anbefalt: Digitalt task-board i Jira.

Nye team må avgjøre om de skal bruke en fysisk task-board eller et digitalt task-board. En fysisk tavle fungerer veldig bra når teamet er samlet, og er metoden med minst overhead.

Men det er også mange fordeler til en digitalt task-board:

- oppgavene kan lett berikes med kommentar, lenker, test-cases, skjermbilder og dokumenter
- at det fungerer som teknisk dokumentasjon, også for testing og bugfiksing

Bug-fiksing og "unplanned" oppgaver

Utviklingsteamet har også ansvar for bugfixing og feilhåndtering. Ansvar for ServiceDesk overvåking går på rundgang. Teammedlemmer som er ServiceDesk ansvarlig har ansvar for:

- å overvåke ServiceDesk saker
- Håndtere brukerforespørsel som kommer inn i løpet av uken
- Undersøke bugs og dokumenter funn i en nye bug-tickets
- Estimere omfang av bug-fiksen i Story Points
- Avgjøre om en bug er kritisk (håndteres umiddelbart).
  - Dersom det er en kritisk bug, legg en "label" (tag) ServiceDesk oppgaver med "unplanned"
  - Legg kritiske oppgaver inn i sprinten. (Hvis det ikke er kritisk, gjør PO oppmerksom på at det må være med i neste Sprint)

Grooming

Egne *Grooming møter* brukes

- etter behov, i forkant av Sprint planlegging
- og alltid når nye prosjekter starter
- varer 1 til 3 timer

Grooming skal resultere i en løsningsstrategi og grove oppgaver, samt grove estimater som Product Owner trenger for å ta avgjørelser..

Vanilla Scrum vs. Pistasj-Scrum

Hovedforskjellene

	Vanilla	**Pistasj**
Sprint planlegging	4 timer. To-delt sprintplanlegging; del 2 uten Product Owner	Mellom 1 og 4 timer. Product Owner er med for å svare på spørsmål.
Scrum Master	Passer på burndown og fremgang under hele sprinten og coacher teammedlemmer for å oppnå optimale resultater. Er sammen med teamet hver dag. Fjerning av hinder kan ta mye tid, og dette krever autoritet og mandat. ScrumMaster er megler når teamet er uenig og når det oppstår konflikter, noe som krever objektivitet og tillitt. **Er ikke en del av teamet.** Hovedgrunn: ScrumMaster rollenhar ansvar for coaching av teamets medlemmer inkludert konfliktløsning, og i disse situasjonene kan objektivitet og avstand være viktig. Pga. stress ovenfra om å være en slags leder og pga. den menneskelige natur vil det i mange organisasjoner være umulig for ScrumMaster å unngå en lederrolle. Product Owners som er ivrig etter raske leveranser og lave kostnader, eller uerfarne Product Owners som trenger coaching, gjør at ScrumMaster også må være en mellommann, og skjerme teamet.	Er ansvarlig for å si ifra (til teamet, til PO eller til sjefen) når ting begynner å skli ut i prosessen, og når hinder oppstår. Bruker 1-2 timer tilsammen på backlog, og møteinnkalling. Sørger for at backlog er klar, men trenger ikke være den som legger inn sakene. **Er en del av teamet,** og bruker 90% av tida si på utvikling. Teamets medlemmer har ikke behov for coaching eller konflikt-håndtering (som håndteres i linjen). ScrumMaster trenger dermed, ikke like mye avstand eller objektivitet som i Vanilla-Scrum. Iht. teamets dynamikk, ScrumMaster klarer fint å være likemann, og hever seg ikke over teamet. Har ikke hatt behov for megling mellom PO og teamet. Fordi PO er tilgjengelig og med på mange møter har man ikke hatt behovet for å være mellommann i like stor grad.

Coaching	Coaching og konflikthåndtering er en vesentlig del av ScrumMaster rollen.	Coaching er ikke en del av jobben. Teamet tåler konflikt og medlemmene har håndtert det selv.
Burndown	ScrumMastere og teamet har veldig fokus på å oppnå optimale burndowns, og optimale prosesser, også fordi ledelse og kundene er opptatt av ROI og har ubøyelige leveransedatoer.	Burndown er interessant, men ikke alfa og omega. Oppgavene heller enn burndown har fokus. Optimalisering av prosessen er hele teamets ansvar.
Tracking	Timer brukt føres på hver oppgave.	Timer brukt er ikke ført på oppgaver i Jira.
Product Owner	Må ha en solid forståelse av brukere og bruksscenario. Pleier å være enten en superbruker av systemet eller noen fra markedsføringsavdeling eller prosjektledelse.	Ved NTNU er PO-rollen ivaretatt av enten (a) prosjektlederen som er komfortabel med rollen og er egnet til å ta tekniske avgjørelser, eller (b) et prosjektmedlem som kan stå for den tekniske-prosjektledelsen.
Retrospective	Product Owner er ikke med.	Product Owner er med, men ikke uten unntak. Dette er teamets møte. Resultat: Har hittil ikke hatt noen konflikt mellom PO og ScrumMaster (noe som er uvanlig). Fordelene med at PO er med: Det gir PO innsikt i hva som fungerer bra og det som teamet ønsker å endre. PO får en anledning til å si "bra jobbet". Eliminerer behov for egne møter mellom SM og PO.

Scrum of Product Owners		Uformell etter behov. Ved tvil om prioritering vil dette avgjøres i møte med produkteiere ([PO1] og [PO2]), [seksjonsjef] og [overingeniør]. Nødvendig kun når et team har 2 eller flere backlog samtidig, og når nye prosjekter ønsker å bruke teamets kapasitet.

Tabell A.1: Sammenligning Vanilla- og Pistasj-Scrum, av NTNU

Scrum egner seg ikke hvis...

- Scrum er ikke en "good fit" hvis det er mange avbrytelser og krisesituasjoner.
- Hvis bugfiksing og andre oppgaver står for mer enn 20% av teamets kapasitet, og varierer ofte.
- Hvis oppgavene er repeterende / rett frem, slik at time-boxing og estimering ikke har noen mening og bare blir til hinder for det som skal gjøres. (Bruk Kanban)
- Hvis teamet er større enn 9 medlemmer.
- Hvis det ikke er behov eller rom for medvirkning

Om Scrum

> Scrum is a simple framework, the basic definition can be understood in less than a day. Scrum does not try to give all the answers or be one size fits all.
> There is a balance in Scrum between prescriptive and freedom, strongly demanding certain, few practices and leaving the business to self-organize the rest.
> - Alan Dayley

Scrum har ingen tekniske forskrifter

> Any good implementation of Scrum needs to borrow technical practices from some other method like XP. The suite of technical practices that should be added probably include: TDD, Continous integration, Acceptance testing, Pair programming, Refactoring.
> - Chris Brookins

Kanban brukes om sommeren

1. Arbeidsflyten: arbeidsflyten visualiseres på en taskboard.
2. Pågående arbeid: Man skal begrense hvor mange oppgaver som ligger i "In progress", per person og totalt
3. Kanban estimerer *ikke* tiden for oppgavene. Dersom du vil måle/track noe, mål antall saker som er gjennomført, og ledetid på gjennomførte arbeid.

Vedlegg B

Intervjuspørsmål

Dette vedlegget inneholder de veiledende intervjuspørsmålene for de semi-strukturerte intevjuene som ble utført i løpet av case studiet. Med veiledende menes at spørsmålene ofte ble spurt på en annen måte, men noe i nærheten av de veiledende spørsmålene. I tillegg ble nye spørsmål stilt om intervjuene kom inn på nye interessante temaer. Før intervjuene startet ble hvert intervjuobjekt opplyst om anonymisering av svarene, samt mulighet til å unnlate å svare på spørsmål og avslutte intervjuet om det blir ubehagelig. Alle intervjuobjektene svarte på alle spørsmål de ble stilt. Noen av de veiledende spørsmålene var felles for alle type roller, mens andre spørsmål var tilpasset de enkelte rollene. Her ble det brukt 4 forskjellige roller: utvikler, Scrummaster, produkteier, og utvikler med PO-ansvar.

Felles spørsmål

- Kan du fortelle litt om 14C-prosjektet?
- Dere har taskboard i Jira, kan du si noe om hvordan dette blir brukt?
- Hva synes du om måten Scrum blir brukt på i dette prosjektet, er det ting du ville ha forandret på om det var opp til deg?
- Til slutt, har du noen sterke meninger angående 14C-prosjektet som vi ikke har kommet inn på her?

Spørsmål for utviklerne

I tillegg til de felles spørsmålene ble utviklerne i teamet stilt følgende spørsmål:

- Dere bruker Scrum på alle prosjekter dere jobber med, kan du si noe om hvordan man begynte å bruke Scrum?
- Kan du fortelle litt om hvordan og hvorfor dere gikk bort fra ren "Vanilla Scrum"?
- Har du lyst til å fortelle meg litt mer om morgenmøtene dere har holdt etter standup?

Spørsmål for Scrummaster

I tillegg til de felles spørsmålene og spørsmålene for utviklere ble de to Scrummasterene stil følgende spørsmål:

- Hvordan synes du det har gått å dele denne rollen med en annen i teamet?
- Har det oppstått situasjoner hvor det har vært vanskelig å holde seg objektiv i rollen som Scrummaster?

Spørsmål for produkteier

I tillegg til de felles spørsmålene ble produkteieren stilt følgende spørsmål:

- Vil du fortelle meg litt om hvordan kommunikasjonen med kunden er? Hvor enkelt er det å innhente krav og ønsker, og forstå behovene?
- Hvordan synes du det har gått med to personer som fungerer som en slags produkteier på dette prosjektet?

Spørsmål for utvikler med PO-ansvar

Utvikler med PO-ansvar ble stilt de felles spørsmålene samt spørsmålene for produkteier og utvikler.