Jean-Pierre V. Strand
Torbjørn Steine Sæle

DevOps i Norden

Masteroppgave i Digital Samhandling
Veileder: Knut Arne Strand
Mai 2022

Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Jean-Pierre V. Strand
Torbjørn Steine Sæle

DevOps i Norden

Masteroppgave i Digital Samhandling
Veileder: Knut Arne Strand
Mai 2022

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Forord

Denne masteroppgaven er et avsluttende prosjekt på det toårige masterprogrammet «Digital samhandling» ved NTNU i Trondheim. Det har vært et meget lærerikt prosjekt, og vi har nå en mye dypere kunnskap om dette svært interessante og relevante temaet som vil være nyttig å ta med seg ut i arbeidslivet.

Vi hadde som mål å undersøke hvordan personer som jobber med DevOps opplever metoden, og hva de mener definerer konseptet DevOps.

Vi ønsker å gi en takk til alle de andre medstudentene på trinnet vårt som skrev på sin masteroppgave på samme tidspunkt som oss, og som delte sine erfaringer som de har hatt under prosjektet med oss. Det har vært to fine år vi har delt sammen.
Vi vil også takke bedriften som har vært i fokus og som har vært oppdragsgiver i denne oppgaven. De har behandlet oss bra, og gav oss anledning til å bruke lokalet deres for å skrive oppgaven, samt til å spise lunsj sammen med dem. De ansatte har vært meget hyggelige og imøtekommende mot oss, og for dette er vi meget takknemlige. Til slutt vil vi takke Knut Arne Strand som har hatt rollen som veileder i dette forskningsprosjektet. Tilbakemeldingene og rådene han gav underveis har vært uvurderlige, og dette forskningsprosjektet hadde ikke vært mulig uten at han hadde etablerte kontakt med oppdragsgiveren og sørget for at denne problemstillingen ble en mulig masteroppgave.

Sammendrag

DevOps er en videreutvikling av agile metoder, der utviklere og driftere skal jobbe tettere for å tette gapet som har eksistert mellom disse faggruppene med tradisjonelle programvareutviklingsmetoder. Om man prøver å finne en tydelig definisjon av DevOps i litteratur, så kan man finne uklare og tvetydige definisjoner som kan være forskjellige avhengig av hvilken kilde man bruker.

I denne masteravhandlingen vil det bli presentert en studie som har blitt gjennomført med fokus på hvordan de som bruker DevOps vil definere begrepet, og hvilke konsekvenser en implementering av DevOps vil medføre i en organisasjon. Konteksten til denne studien har vært en utviklingsavdeling som tilhører en internasjonal programvarebedrift som begynte å implementere DevOps for rundt fem år siden. Avdelingen er lokalisert i Trondheim, og for å få tillatelse av bedriften til å bruke dem som en case måtte vi skrive under på en NDA.

Metoden som er brukt er case-studie med bare en case. Flere semi-strukturerte intervjuer ble utført med totalt åtte informanter, og blant informantene kan man finne rollene prosjektleder, utvikler, konsulent og systemarkitekt. Alle intervjuene ble tatt opp. For å analysere dataene ble de transskribert til tekst, for så å bli delt opp individuelt i koder som igjen ble brutt ned i mindre koder. Programmet NVIVO ble brukt for å gjøre denne prosessen lettere.

Resultatet av case-studien vår viser at ansatte i en bedrift som bruker DevOps ofte støter på problemer når de skal prøve å definere begrepet. Om man skal definere DevOps, så kan det være nyttig å snakke om DevOps-kultur og teknisk DevOps. Når bedrifter implementerer DevOps, så endrer kulturen seg til å passe beskrivelsen med økt samarbeid, ansvar og involvering fra alle parter i utviklingssyklusen. Man opplever flere fordeler som blant annet bedre flyt i leveringen av produkter, automatisering som fører til sparing av tid og mulighet til å rette feil på kort tid. Det var også noen utfordringer/ulemper assosiert med DevOps, som blant annet vanskeligheter med å bryte gamle rollemønster, prioritering av tid, mangel på relevant kompetanse og utfordringer med å fornye legacy-systemer. Det virker som at de som bruker DevOps liker det, og misnøye eller motstand oppstår når den nye måten å gjøre det på gjør gammel kompetanse overflødig. For å implementere DevOps på en god måte burde man være klar over at en god DevOps-kultur som tilrettelegger for økt samarbeid vil være viktig for at implementeringen blir en suksess. Det vil også være viktig at implementering er flytende, og at man implementerer det

som faktisk vil være til nytte for bedriften. I tillegg må man ha både de ansatte og ledelsen med på implementeringen.
I konklusjonen konkluderer vi at definisjonen av DevOps har blitt klarere, og at konsekvensene av DevOps har blitt klarere med noen retningslinjer for implementering.

Abstract

DevOps in northern Europe

DevOps is a continued development of agile methods, where employees in development and operations will work closer than with traditional software development methods. If you try to find a clear definition of DevOps in literature, you may find unclear and ambiguous definitions which may differ depending on the source you use.

In this master thesis, a study will be presented which has been conducted with a focus on the users' definition of the term DevOps, and what consequences an implementation of DevOps will entail in an organization. The context of this study was a department of an international software company that began implementing DevOps five years ago. The department is located in Trondheim, and in order to get permission from the company to use them in this study, we had to sign an NDA.

The method used in the study is a case study with one case. Several semi-structured interviews were conducted with a total of eight informants, and among the informants one can find the roles of project manager, consultant, developer, and system architect. All the interviews were recorded. To analyze the data, they were transcribed to text, and then divided individually into codes which in turn were broken down into smaller codes. A program called NVIVO was used to make this process easier.

The results of our case study shows that employees of a company which uses DevOps often encounter difficulties when trying to define the term. If you are defining DevOps, it can be helpful to separate it into DevOps culture and technical DevOps. When companies implement DevOps, the culture changes to fit the description with increased collaboration, responsibility and involvement from all parties in the development cycle. There are several advantages with DevOps, such as better flow in the delivery of products, automation which leads to saving time and the ability to correct errors in a short time span. There were also some challenges and/or disadvantages associated with DevOps, such as difficulties in breaking old role patterns, prioritization of work time, lack of relevant expertise and challenges in renewing legacy systems. It seems like those who use DevOps tends to like it, and any discontent or resistance occurs mostly when the new way of doing things, or the new product makes old skills and knowledge

redundant. In order to implement DevOps in a successful manner, one should be aware that a good DevOps culture which facilitates increased collaboration will be important for the implementation to be a success. It will also be important that the implementation happen in a fluid way, and what's implemented is beneficial to the company. In addition, both the employees and management must be included in the implementation.

In conclusion, we conclude that the DevOps definition has become clearer, and that the consequences of DevOps has also become clearer with some implementation guidelines.

Innhold

Liste over figurer

Liste over tabeller

Tabell over begrep og forkortelser med forklaringer

Begrep	**Forklaring**
Drifter	En arbeidstittel for noen som blant annet har ansvar å holde et system operativt etter at det har blitt utgitt. Operations på engelsk.
Legacy	Brukes for å beskrive gammel programvare som ofte er en essensiell del av et system, eller hele systemer. Programvaren er utdatert på ett eller flere områder som sikkerhet eller andre standarder. Den blir ikke like mye oppdatert som nyere programvare.
Release	Utgivelser/publisering av programvare, når programvaren/systemet blir tilgjengelig for de som skal bruke det (kunden).
Kontinuerlige utgivelser	En praksis der en utgir produkter/kode i et høyt tempo.
NDA	Står for **N**on-**D**isclosure **A**greement. En kontrakt der partene blir enige om tilgang til noe som ikke skal bli delt videre, for eksempel forretningshemmeligheter.
NSD	**N**orsk **S**enter for forsknings**D**ata. Et nasjonalt senter og arkiv for forskningsdata. Tilrettelegger for deling og gjenbruk av data om mennesker og samfunn (NSD, n.d.).
Norden	Når vi bruker Norden i dette dokumentet mener vi Norge, Sverige, Danmark og Finland.
Arkitekt	Brukes for stillingen systemarkitekt eller «Software/system architect». Ligner på systemutvikler, men har mer ansvar enn

Begrep	Forklaring
	systemutviklere innen blant annet design og planlegging.
Standup-møter	Standup-møter er samlinger som holdes regelmessig, vanligvis daglig. Teammedlemmer deler statusrapporter om arbeidet sitt og hva neste på agendaen er.
Agile/Agil	Agil er en metode for prosjektledelse, benyttet spesielt innen programvareutvikling. Agil går ut på oppgavedeling i korte arbeidsfaser og hyppige revurderinger og tilpasning av planer.
Deploy/Distribuere	Deploy betyr å distribuere en ny utgivelse til en eller flere maskiner, oppdatere den gjeldende versjonen. I webutvikling betyr dette at man oppdaterer versjonen som ligger på produksjonsserverne.
Build/Bygg	En build eller programvarebygg er et sett med kjørbar kode som er klar til bruk av kunder.
Skyløsning	Skybasert løsning refererer til applikasjoner, lagring, on-demand tjenester, datanettverk eller andre ressurser som er aksessert med en internettforbindelse gjennom en annen leverandørs delte cloud computing-rammeverk.
Rider og Visual Studio	Dette er programmeringssystemer som benyttes av utviklere.
NVivo	NVivo er et programvareprogram som brukes for kvalitativ forsking og forskning med blandede metoder. Det brukes spesifikt til å analysere ustrukturert tekst, video, bildedata,

Begrep	**Forklaring**
	intervjuer, fokusgrupper, undersøkelser, sosiale medier og tidsskriftartikler.
Azure DevOps	Var kalt Team Foundation Server(TFS) før. Det er en skyløsning som tilrettelegger for samarbeid mellom teammedlemmer i programvareutvikling.

Tabell 1: En tabell som forklarer begreper og forkortelse

1 Introduksjon

Programvarebedrifter står i dag overfor et økende press for å redusere utviklingskostnader og samtidig opprettholde kvalitet på leveransene. Behovet for å levere effektiv programvare krever nye måter å administrere systemutvikling på. Fortløpende endring av forbrukerkrav til praktisk tjenesteleveranser har tvunget bedrifter til å ta i bruk nettbaserte tjenester. Dette har gitt bedrifter muligheten til å allokere flere ressurser til utvikling av vellykket programvare for sine kunder. Til tross for store påløpte kostnader viste det seg at de fleste prosjektene ikke ble vellykkede. For å levere produkter som bedre møter et skiftende kundebehov, ble en fleksibel metodikk foreslått. Denne metodikken er kjent som agil arbeidsmetodikk eller agile methodology på engelsk. I denne oppgaven benytter vi oss av det norske ordet agil. Denne arbeidsmetodikken tilrettelegger for endring av planer underveis, samarbeid med kunden(e), kontinuerlige leveranser og mer (Fowler & Highsmith, 2001). Agil arbeidsmetodikk kan forklares som et sett med prinsipper som et team bruker til å utføre et prosjekt. Prosjektet deles inn i faser som krever at man har kontinuerlig kommunikasjon med interessenter, utfører kontinuerlig utvikling og kontinuerlig iterasjoner. Agil programvareutvikling er basert på reaksjonshastigheten til endringer og er ment å gi bedrifter konkurransefortrinn (Fowler & Highsmith, 2001).

Den overordnende ideen innen agil arbeidsmetodikk er at arbeidsgruppen (teamet) skal bli mer effektive til å reagere på endringer. Agil arbeidsmetodikk skal bidra å redusere kostander forbudent med å flytte på informasjon mellom personer, tiden det tar å ta valg og observere endringene. Agil setter søkelys på egenskapene «soft skills» som fokuserer på å inkludere kompetanse, ferdigheter og kommunikasjon. Kompetanseutvikling er derfor en sentral del av agil arbeidsmetodikk. Desto dyktigere og mer talentfull en person er, jo mer verdi vil kunne leveres på kortere tid. Med dette sagt vil hastighet, hyppige endringer og strevet med å holde takt med disse kunne skape flaskehalser i markedet. Det som agil ikke fordyper seg i er forholdet mellom personer som jobber som utviklere og driftere. Her kan forsinkelser og forvirring ofte oppstå. Det har i senere tid blitt klart at det var behov for en arbeidsmetodikk eller tilnærming som fordyper seg tettere inn på integrasjon mellom Utviklere (Dev) og Drift (Ops). Utviklere har som oppgave

å utvikle noe nytt eller videreutvikle noe eksisterende, mens driftere har som oppgave å sørge for at produktet eller tjenesten som tilbys er stabilt og kjører som det skal.
I en bok med tittelen *Sense & Respond* beskrives det hvordan alle bedrifter i dag må kunne utvikle programvare til en hvis grad. Det blir derfor et større behov for å finne en beste praksis for programvareutvikling og drift (Gothelf & Seiden, 2017). Selv om agil på mangemåter ble ansett som revolusjonerende for programvareutvikling, klarte det ikke å løse konflikter mellom utviklere og driftere. Det fortsatte å utvikle seg siloer rundt tekniske ferdighetssett og spesialiteter, og utviklere leverte fortsatt kode til driftsfolk for å distribuere og drifte. I 2008 møttes to programvareingeniører, Andrew Clay Shafer og Patrick Debois og diskuterte frustrasjonene over den konstante konflikten mellom utviklere og driftsfolk. Sammen lanserte de det første offisielle DevOpsDays-arrangementet i Belgia for å skape en bedre og mer smidig måte å tilnærme seg programvareutvikling på (Freeman, 2019).

Denne tilnærmingen fikk et fast feste, og DevOps har de siste årene blitt et bredt diskutert fenomen innen programvareutvikling. DevOps tar for seg det som beskrives som tomrommet mellom utviklings og driftspersonell. DevOps fører til et organisatorisk skifte der istedenfor å ha distribuerte separate grupper som utfører funksjoner separat, kjøres det nå kryssfunksjonelle team som jobber med kontinuerlig operativ funksjonsleveranser. Denne tilnærmingen bidrar til å levere verdi raskere og kontinuerlig, redusere problemer på grunn av feil kommunikasjon mellom teammedlemmer, i tillegg til å akselerere problemløsning.

Mange har positive forventninger til DevOps og bedrifter blir stadig mer interesserte i fenomenet og hvordan man kan utnytte de mulige fordelene. Likevel er begrepet DevOps omgitt av tvetydighet. Selve målet med DevOps er tydelig, vi ønsker å bygge en bro mellom utviklings- og driftspersonalet. Definisjonen derimot er ikke så tydelig. Det finnes fortsatt mange tolkninger av hva DevOps betyr (Smeds, Nybom, & Porres, 2015). Tolkning eller interpretivisme er et filosofisk paradigme som handler om at mennesker tolker virkeligheten forskjellig, og forskning bør utføres i det naturlige konteksten der det som skal undersøkes faktisk oppstår (Oates, 2006). Etter et søk i litteraturen så kan en finne mange casestudier om DevOps internasjonalt, men det ble raskt tydelig at det er vanskelig å finne slike studier gjort hjemme i Norge eller i Norden generelt.

En som ble gjort i Norge hadde fokus på testing i forbindelse med DevOps og ikke nødvendigvis DevOps som en filosofi/tilnærming (Cruzes, Melsnes, & Marczak, 2019). Det ble tydelig for oss at det i stor grad er rom for ytterligere forskning på DevOps i Norden. Det er ikke gitt at mennesker som jobber med DevOps i Norge vil ha en lignende erfaring fra deres arbeid med denne metoden, og derfor er det et potensiale for at den kollektive oppfatningen og forståelsen av DevOps kan forbedres. Med dette som basis mente vi at det var verdt å undersøke hvordan norske ansatte opplever å jobbe i en DevOps bedrift, og om det var eventuelle forskjeller og likheter med resultater fra utlandet. For å teste ut om dette muligens kunne være noe vi kunne basere en masteroppgave på, gjennomførte vi en pilotstudie. Dette fungerte som en utprøving i liten skala av de metodene vi hadde tiltenkt til selve masteroppgaven.

Pilotstudien foregikk i samarbeid med en norsk programvareutviklingsbedrift som har kontorer lokalisert i Trondheim, Oslo, Sverige og Finnland. Denne bedriften har nylig gått over til arbeidsmetoden DevOps. På kontoret i Trondheim fikk vi tilgang til ansatte som har opplevd overgangen til DevOps, og dermed erfaringer i hvordan det er å jobbe med og uten DevOps. Metoden vi valgte å bruke i pilotstudien for datainnsamling var litteratursøk og intervjuer. Vi var meget fornøyde med hvordan resultatet fra pilotstudien vår ble og så tydelig potensialet for å forske videre på DevOps. Vi valgte derfor å skrive masteroppgaven om lignende tema i samarbeid med denne bedriften. Den store forskjellen mellom denne masteroppgaven og pilotstudien vår er mengden teori og data vi har samlet inn og en tilpasset problemstilling/forskningsspørsmål. I pilotstudien fikk vi bare intervjuet to personer, og det var ikke mange sider med teori som ble samlet inn grunnet begrensninger på antall sider som var tillatt på rapporten. Forskningsspørsmålet i denne oppgaven er todelt:

1. Hva blir DevOps definert som av de ansatte?
2. Hva er konsekvensene av å innføre DevOps?

For å belyse disse forskningsspørsmålene har vi gjennomført litteraturstudier og intervjuer med 8 informanter. Samlet sett håper vi at de innsamlede dataene vil gi oss et godt grunnlag for å svare på forskningsspørsmålene og gi en bedre forståelse av hvordan vi i Norden definerer DevOps. Frem til nå har det vært meget begrenset med forskning på DevOps i Norden. Denne oppgaven

vil derfor være et viktig bidrag til det eksisterende forskningsfeltet. Oppgaven består av 6 kapitler.

1.1 Problemstilling/forskningsspørsmål

Om man prøver å finne informasjon om hva DevOps er for noe, så kan dette være utfordrende fordi man kan finne mange forskjellige definisjoner. Derfor hadde det vært nyttig å lage en overordnet definisjon av DevOps, om det er mulig. I tillegg, så er det vanskelig å finne case-studier fra Norden om DevOps og effekten det har på de ansatte. Hva synes de som bruker DevOps om det? Arbeidsmiljøer og perspektiver på arbeidsmetoder kan variere avhengig av hvor man er lokalisert i verden. Betyr dette at det er mulig at det er en nordisk definisjon av DevOps? For å kunne svare på denne problemstillingen har vi brukt noen forskningsspørsmål.

Forskningsspørsmål:
F1: Hvordan defineres begrepet DevOps av forskjellige ansatte i en programvarebedrift?

For å svare på dette forskningsspørsmålet har vi intervjuet flere ansatte i en bedrift om hvordan de vil definere DevOps. I tillegg vil vi inkludere litteratur om dette temaet fra hele verden for å lage en felles definisjon. Case-studier fra Norden blir det lagt ekstra vekt på for å lage en nordisk definisjon av DevOps.

F2: Hva er konsekvensene av å implementere DevOps i en programvarebedrift?

Dette forskningsspørsmålet vil vi svare på ved å spørre intervjuobjektene våre om hvilke endringer de har merket seg etter implementeringen av DevOps. Vi vil også inkludere konsekvenser beskrevet i litteraturen fra hele verden, for å ha noe å sammenligne de lokale resultatene med. Det er også to underforskningsspørsmål som skal sette fokus på noen områder innen dette temaet.

F2.1: Hva syntes de ansatte om DevOps i en bedrift som bruker det?

For å finne svaret på dette underforskningsspørsmålet vil vi spørre deltakerne i case-studien vår om hva de synes om DevOps etter å ha brukt det, og bruke litteratur for å sette resultatene fra studien opp mot hverandre.

F2.2: Hva er nødvendig for å implementere DevOps på en god måte i en programvarebedrift?

Dette forskningsspørsmålet vil vi svare på ved å inkludere spørsmål i intervjuguiden vår om hva som er kritisk for å implementere DevOps i en organisasjon, samt bruke litteratur for å finne råd.

2 Metode

En metode er en fremgangsmåte, et middel til å løse problemer og komme frem til ny kunnskap. Et hvilket som helst middel som tjener formålet, hører med i arsenalet (Dalland, 2020).
I dette kapittelet vil vi beskrive hvordan vi har gått fram for å løse problemstillingen og finne svar på forskningsspørsmålene vi har satt. Dette innebærer en beskrivelse av begrunnelsen for forskningsmetode, samt hvilket filosofisk paradigme som vi har basert forskningen på, forskningsdesign, datainnsamling, analyse av data, metodekvalitet og etikk relatert til forskning. Dette kapittelet skal gi en innsikt i hvordan og hvorfor vi har gjennomført datainnsamlingen, og gi leseren en mulighet til å vurdere om masterprosjektet har blitt gjennomført på en god måte.

2.1 Valg av forskningsstrategi og filosofisk paradigme

Det er fullt mulig å bruke flere strategier i et forskningsprosjekt for å oppnå strategitriangulering, men vi valgte å bare bruke en strategi fordi vi følte vi ikke hadde tid til å gjennomføre flere (Oates, 2006). Strategien vi valgte var case-studie, som også var strategien vi brukte i pilotstudien vår. Ved å fokusere på en instans av en hendelse, i dette tilfellet implementering av DevOps i en bedrift, så får en mer innsikt i de komplekse prosessene og det som ellers skjer i den aktuelle hendelsen (Oates, 2006). Hendelsen blir studert der det faktisk skjer, som i vårt tilfelle var i kontorlokalet til avdelingen vi skrev om (Oates, 2006). DevOps virket for oss som noe som påvirker forholdene mellom de ansatte på en arbeidsplass og den generelle kulturen. Vi følte derfor at et dypdykk med case-studie ville være en god strategi for vårt formål. Valget av case-studie som strategi var også påvirket av litteratursøk etter case-studier i Norge, og etter hvert Norden. Siden det var fem case-studier vi fant på Google Scholar med noenlunde samme problemstilling som også tydelig hevdet at de var basert på bedrifter i Norden samt var skrevet på engelsk, så følte vi at en case-studie i Norge med vår problemstilling var noe relativt unikt.

Etter å ha gjennomført pilotstudien og hatt lange diskusjoner med flere av de ansatte på avdelingen i forbindelse med pilotstudien, ble vi i studentgruppen enige om at en kvalitativ framgangsmåte var den beste fremgangsmåten for vår problemstilling der vi samlet inn kvalitative data. Kvalitative data vil da si data som ikke er numeriske, og vi foretrakk dette over kvantitativ data fordi vi følte at vi lettere kunne se enkeltpersoners meninger og få innsikt i hvordan de forstod ulike tema med dem (Oates, 2006).

Litteratursøket i forbindelse med dette forskningsprosjektet begynte tidlig, og siden vi valgte case-studie som strategi var det naturlig å søke etter andre case-studier. I pilotstudien fant vi flere kilder som ble vurdert som meget interessante for dette temaet, og teorien vi fant i pilotstudien ble fundamentet for teorien vi brukte i dette forskningsprosjektet. Det første vi lette etter var case-studier utførte i Norden. Dette gjorde vi ved å søke etter case-studier som ble utførte i de individuelle landene i Norden ved hjelp av Google Scholar. Når vi hadde funnet de få case-studiene som ble utført i Norden, så utvidet vi søket til å inkludere case-studier i hele verden. På dette tidspunktet begynte vi å dele litteratursøket i to: den ene delen bestod av å finne ut hvordan de ansatte opplevde DevOps med fordeler og ulemper, og den andre bestod i å prøve å finne en definisjon på DevOps. Her inkluderte vi også artikler som ikke var case-studier, blant annet litteraturstudier.

2.1.1 Filosofisk paradigme

Vi brukte interpretivismen som det filosofiske paradigmet vi baserte forskningen på, så det passet bra å bruke case-studie med kvalitative metoder som strategi. Paradigmet passer bra fordi det er bedre å bruke når en skal studere tema som kultur og oppførsel, og disse metodene baserer seg på å tolke dataene som samles inn og erkjenner at virkeligheten er subjektiv slik som interpretivismen (Oates, 2006). Det betyr også at når vi utfører en case-studie i Norge/Norden når det er mangelfullt med slike tilgjengelig, så kan vi forbedre den overordnede forståelsen av DevOps ved å legge til en tolkning som ikke eksisterte fra før av. Case-studier er ofte basert på interpretivismen, men trenger ikke nødvendigvis å være det (Oates, 2006). De kan også være baserte på et av de andre paradigmene vi vurderte å bruke og som det er populært å basere forskningen på: positivisme. Positivisme kort oppsummert: om man bruker dette paradigmet skal

man være så objektiv som mulig i forskningen, og en vet at verden består av naturlover og/eller mønster som en prøver å finne (Oates, 2006). Positivismen passer bedre til å forske på naturfenomener enn den gjør til å studere sosiale fenomener som for eksempel kultur og forholdet mellom personer (Oates, 2006). Siden sosiale fenomener viste seg å være viktig for mange som har skrevet om DevOps, så følte vi at dette paradigmet ikke passet for dette forskningsprosjektet.

2.2 Forskningsdesign

Når en skal utføre en case-studie, så har en ifølge Oates (2006) generelt tre typer å velge mellom: en utforskende studie, en beskrivende studie eller en forklarende studie. Case-studien vår er en beskrivende studie, der vi lagde en detaljert beskrivelse av hendelsen vi skulle studere som kontekst med hendelsesforløp og hvordan de involverte forstod og opplevde det (Oates, 2006). Om vi hadde hatt en utforskende studie så hadde vi brukt studien til prøve å forstå et forskningsspørsmål der det mangler litteratur og prøve å legge til rette for en oppfølgende studie (Oates, 2006). Vi kan kalle case-studien vi utførte i forbindelse med pilotstudien vår for en slik utforskende studie der vi brukte den for å tilrettelegge for denne case-studien ved å skaffe mer informasjon om både litteratur og om DevOps i praksis. En forklarende studie vil være enda mer nøyaktig i sine beskrivelser enn en beskrivende studie, der en prøver å finne hvilke faktorer som påvirket hvilke ting eller sammenligner det som skjedde med eksempler fra litteraturen (Oates, 2006). Vi følte at en beskrivende studie var det som best beskrev det vi ville oppnå, der vi brukte en blanding av studie av det som hadde skjedd i fortiden og det som skjedde i tidsrommet vi var til stede for å studere det vi var ute etter. Med case-studier så må en også ha en hendelse eller flere som en skal studere. Noen eksempler på hva som får noen til å velge en spesiell hendelse er: den er typisk eller atypisk for slike hendelser, forskeren kan ha sett en sjanse for å studere noe unikt eller det er praktisk å bruke den hendelsen (Oates, 2006). Vår hendelse kan beskrives som en typisk hendelse, en instans av implementasjon av DevOps. Det var praktisk for oss å bruke den i en case-studie fordi veilederen vår hadde kontakter som gjorde det mulig å studere casen. I vår case-studie inkluderte vi bare en av avdelingene til bedriften som var plasserte i Norge fordi vi ville gjøre en dypere undersøkelse på hvordan enkeltpersoner opplever og forstår DevOps, noe som kanskje ville forsvunnet om vi inkluderte flere personer. Det hadde også blitt vanskeligere å

gå dypere i temaet om flere var inkludert i undersøkelsen. Ulempen med dette er at resultatene fra denne case-studien kanskje ikke er representativt for forståelsen av DevOps i Norge eller Norden, men bare reflekterer denne ene avdelingens tanker og opplevelser. For å kompensere fant vi andre case-studier å sammenligne resultatene fra vår case-studie med, samt diskutere hvilke likheter og ulikheter vi fant.

For å samle inn data til denne case-studien planla vi å bruke intervjuer og observasjoner av flere personer hos avdelingen som metode. Ved å bruke flere metoder oppnår man metodetriangulering (Oates, 2006). Vi fikk tillatelse til å skrive oppgaven i kontorlokalene deres, samt bruke dem til å utføre datainnsamlingen. For å få lov til dette måtte vi skrive under på en NDA slik at vi ikke gav ut forretningshemmeligheter i resultatene eller andre steder. Dette gav oss en nærhet til de som skulle delta i case-studien, som det hadde blitt vanskelig å skaffe på andre måter. For å velge hvem som skulle inviteres til å delta i case-studien fikk vi en liste over ansatte ved avdelingen, og hvilken rolle/stilling de hadde. Ved hjelp av denne listen sendte vi ut informasjonsskrivet med samtykkeerklæringen til de som vi vurderte som interessante. Etter en diskusjon med kontaktpersonen vår ved bedriften, der han kom med noen anbefalinger om hvem som ville være mest interessante, gjorde vi en vurdering om hvem vi ville prøve å inkludere som deltakere i denne case-studien. De vi vurderte som interessante å inkludere var først og fremst de som hadde erfaring med å jobbe i avdelingen før implementeringen av DevOps begynte, i tillegg til arbeidserfaring med DevOps nå. Vi inkluderte også deltakere med forskjellige roller fordi vi ville ha med ulike perspektiver på DevOps, og ikke nødvendigvis bare de som har dyp forståelse av hva DevOps er. De som vi var interesserte i å ha som deltakere ble sendt informasjonsskrivet og samtykkeerklæringen som ligger i vedlegg 2, før vi ventet på at de skulle sende samtykkeerklæringen tilbake med svar. Observasjonene vi planla å utføre var mer spontane enn intervjuene, der vi ble spurt om vi ville være med på ulike møter som deltakerne i case-studien skulle være med på litt før de startet. Intervjuene ble som oftest avtalt flere dager i forveien siden de ikke er en naturlig del av hverdagen til de som skulle være med, og de ansatte måtte ta tid ut av arbeidsdagen sin for å delta.

2.2.1 Intervjuer

I dette kapittelet kommer en nærmere beskrivelse av intervjuene som ble utført i sammenheng med dette forskningsprosjektet. Det inkluderer en beskrivelse av forarbeidet til intervjuene i tillegg til beskrivelse av selve intervjuene. Ifølge Oates (2006) så passer intervjuer bra når forskere vil blant annet få tak i:

- Detaljert informasjon.
- Erfaringer og følelser som ikke kan direkte observeres.
- Svar på åpne eller komplekse spørsmål som forskjellige personer kanskje har ulike svar på.

Dette passer bra for våre behov for undersøkelse av forskningsspørsmålene, som handler om de ansattes personlige meninger og tanker om DevOps.

2.2.1.1 Planlegging

På vårt studieprogram har vi ikke fått mye spesiell trening på hvordan en utfører et intervju, hvordan man oppfører seg mens man intervjuer noen og hvordan man lager et bra intervju. Det har vært et tema som har blitt kort nevnt i noen få forelesninger, men ikke noe mer enn det. Mer kunnskap og kompetanse på dette området kunne gitt oss en høyere sjanse for at intervjuene ble nyttige og at både intervjuobjektet og den som ledet intervjuet var komfortable under selve intervjuet (Oates, 2006).

I pilotstudien og i litteratursøket til denne masteroppgaven fant vi flere case-studier med lignende problemstillinger og forskningsspørsmål. Disse ble brukt som inspirasjon når vi skulle lage vår egen intervjuguide, spesielt en som inkluderte sin egen intervjuguide (Lwakatare, et al., 2019). Intervjuene beregnet vi på forhånd til å ta en time og 30 minutter, og transskriberingen ville etter erfaring også ta lang tid. Dette fikk vi erfare når vi utførte et par intervjuer i pilotstudien vår, der vi ikke kom igjennom hele intervjuguiden når vi hadde satt av en time til ett intervju. Derfor økte vi tiden vi satt av til hvert intervju opp til en time og 30 minutter. Etter gjennomføringen av det første intervjuet kuttet vi ned på intervjuguiden når vi så at noen av spørsmålene ble litt repeterende, noe som vi også håpte skulle minke tiden vi brukte på hvert

intervju. Tiden vi brukte på intervjuene minket etter dette, og intervjuene tok nå mellom 50 og 20 minutter. Transskriberingen tok også lang tid i pilotstudien, og var ikke så fullstendig som vi hadde ønsket fordi vi bare skrev ned notater underveis, og vi ikke tok lyd- eller videoopptak. Siden vi ikke var så fornøyde med resultatet av transskriberingen i pilotstudien, så ønsket vi å bruke en diktafon under intervjuene i dette forskningsprosjektet. Det er typisk at intervju tar lang tid å både utføre og transskribere (Oates, 2006). Intervjuguiden som vi brukte i våre intervju, ligger i vedlegg 1.

Vi prøvde å holde spørsmålene så åpne som mulig selv om noen av spørsmålene i intervjuguiden er mer åpne enn andre, noe som er anbefalt (Oates, 2006). Fra pilotstudien vår fikk vi erfaring med å bruke en intervjuguide, og intervjuguiden som ble brukt til dette forskningsprosjektet er for det meste lik den vi brukte i pilotstudien. Endringene som ble gjort er for eksempel at vi fjernet noen spørsmål som ikke ble sett på som relevante, og noen ble lagt til som virket relevante for den nye problemstillingen/forskningsspørsmålene. Intervjuguiden er delt opp etter de ulike forskningsspørsmålene i tillegg til noen introduksjonstema. Dette for å gjøre det mer oversiktlig for både oss og intervjuobjektene som ble tilsendt intervjuguiden før intervjuet via epost, slik at de kunne forberede seg. I pilotstudien fikk vi tilbakemeldinger fra forelesere ved NTNU om at vi kanskje ikke burde nevne DevOps i alle spørsmålene vi stilte, spesielt de tidlige spørsmålene i intervjuet, noe som vi også endret på. Vi startet intervjuguiden med noen lette og generelle spørsmål, før vi gikk over til de mer kompliserte spørsmålene senere i intervjuguiden, noe som er anbefalt (Oates, 2006).

Det er lurt å samle informasjon om intervjuobjektet og konteksten rundt det som intervjuet handler om på forhånd av intervjuet, som rollen til intervjuobjektet og nyheter om bedriften som intervjuobjektet tilhører (Oates, 2006). Før intervjuene startet hadde vi skaffet oss en del informasjon om situasjonen som vi skulle bruke i case-studien, samt om avdelingen og selskapet generelt. Pilotstudien var en viktig kilde i denne sammenhengen. Vi hadde også hatt en del uformelle samtaler med flere av de som jobbet der i flere sammenhenger. Alt dette gav oss en del innsikt i hvordan avdelingen var organisert, hva slags roller som de ansatte hadde og hva de tenkte om DevOps.

2.2.1.2 Type intervju og gjennomførelse

Vi bestemte oss for å gjennomføre intervjuene med bare ett intervjuobjekt om gangen, selv om det er mulig å ha flere med i ett intervju. Gruppeintervjuer med tre til seks intervjuobjekter der de diskuterer temaer sammen gir forskeren muligheten til å få innsikt i konsensusen om temaer og intervjuobjektene kan komme med flere svar til spørsmål fordi svarene blir diskutert fram og tilbake blant intervjuobjektene (Oates, 2006). Selv om slike intervjuer har noen fordeler, så har de ifølge Oates (2006) også noen ulemper:

- Noen personer er mer dominerende i sosiale kontekster enn andre, noe som kan gjøre at de overdøver andre i diskusjonen.
- Ikke alle personer er komfortable med å snakke om alle mulige tema når andre kan høre på.
- Intervjuobjektene kan føle at de ikke kan komme med meninger utenom de som er aksepterte av gruppen.

Siden vi var mer interesserte i å finne ut de ansattes meninger om DevOps, så følte vi det var best å bare intervjue én person om gangen slik at de følte seg trygge på å dele sine egne meninger om DevOps, som kanskje ikke deles av alle de ansatte. Om vi hadde utført gruppeintervju, så hadde vi kanskje bare fått vite definisjonen av DevOps fra de som har meget god innsikt og ikke de som ikke følte de hadde fullt så god oversikt. Det hadde vært meget uheldig om vi mistet slike meninger og tanker fra enkeltpersoner på grunn av måten vi utførte intervjuer på.

Vi utførte intervjuene ansikt til ansikt i møterom i kontorlokalet til avdelingen med de intervjuobjektene som jobbet der fysisk. Noen av intervjuobjektene som ble invitert til å delta jobbet ikke i kontorlokalene fysisk, men fra et annet lokale. Vi ønsket å inkludere dem i dette forskningsprosjektet fordi de var drifterne, men ingen av disse ansatte var interesserte i å delta. Det hadde vært mulig å gjennomføre alle intervjuene digitalt, noe som hadde muligens hadde gjort det litt lettere å sette opp intervjuene, men siden vi tidlig i prosessen fikk tilbud om å bruke kontorlokalene til avdelingen til slike formål, så følte vi at dette var et godt tilbud som vi burde

ta i bruk. Ett av intervjuene ble gjennomført på Teams fordi intervjuobjektet som vanligvis jobbet fra kontoret som de andre, denne dagen var på reise. Dette gikk bra.

Etter at intervjuobjektene hadde godkjent å bli med, men før selve intervjuet sendte vi intervjuguiden til intervjuobjektet via epost for å gi dem tid til å tenke over hva de ville svare på de forskjellige spørsmålene. Dette kan være nyttig i slike sammenhenger (Oates, 2006). Under intervjuene brukte vi intervjuguiden som en liste over tema og spørsmål som vi ønsket å snakke om, og underveis kom vi med tilleggsspørsmål om det dukket opp noe som virket relevant for forskningsspørsmålene våre. Denne typen intervjuer kalles semi-strukturerte intervjuer (Oates, 2006). Vi valgte å bruke denne typen intervju fordi det er noen spesielle tema og spørsmål vi ønsket svar på, men vi ville også fange opp interessante data som vi ikke hadde kommet på i forkant. Det var også fordi vi ikke ville gå glipp av interessante ting som vi ikke kom på i forkant ved å bruke strukturerte intervjuer, men vi ville heller ikke bare nevne et tema og la intervjuobjektet snakke fritt om det uten å bryte inn som i ustrukturerte intervjuer, fordi vi hadde visse ting vi visste vi ville ha svar på (Oates, 2006). Det kan hende vi hadde spart tid om vi hadde brukt strukturerte intervjuer og om vi hadde brukt ustrukturerte intervjuer så hadde kanskje enda flere meninger dukket opp. Etter en diskusjon kom vi fram til at en mellomting mellom strukturerte og ustrukturerte intervjuer virket som den typen intervju som ville gi oss best mulig resultat.

For å lagre det som ble sagt i intervjuene brukte vi en diktafon, noe som vi spurte om intervjuobjektet var komfortabelt med før intervjuet startet. Ingen av intervjuobjektene hadde noe imot at vi tok opptak av intervjuet. Intervjuet som ble gjennomført digitalt ble tatt opp ved hjelp av opptaksfunksjon i Teams, som i tillegg til lyd også tar opptak av kameraene i møtet. Også her spurte vi om intervjuobjektet var komfortabel med å bli tatt opp på forhånd. Vi informerte også alle intervjuobjektene om at i de fysiske og digitale intervjuene så ville alle opptak bli slettet så fort som mulig, det vil si så snart transskriberingen var ferdig og intervjuobjektet hadde godkjent resultatet av transskriberingen. Når vi hadde gått gjennom alle temaene og spørsmålene vi ville snakke om i intervjuene, spurte vi alltid om intervjuobjektene hadde noe annet å si om det vi hadde snakket om i intervjuet så langt. Dette gjorde vi for å forsikre oss om at intervjuobjektet hadde fått sagt alt de hadde å si om det vi hadde gått gjennom.

Totalt var det 8 ansatte som deltok i case-studien vår. Disse hadde forskjellige roller i bedriften. Rollene til informantene våre var:

Informant 1: Prosjektleder

Informant 2: Konsulent

Informant 3: Utvikler

Informant 4: Konsulent

Informant 5: Arkitekt

Informant 6: Arkitekt

Informant 7: Arkitekt

Informant 8: Utvikler

2.2.2 Observasjoner

Observasjoner kan brukes til å undersøke hva folk faktisk gjør i stedet for hva de sier de gjør (Oates, 2006). Dette er grunnen til at vi følte at dette var et bra tilskudd til intervjuene vi skulle gjennomføre slik at vi fikk høre hva folk mente og hva de faktisk gjorde. Vi hadde planlagt å utføre flere observasjoner, men vi fikk ikke anledning til å utføre disse på en måte som gav oss data som var verdt å bruke. Det var ikke mange ansatte ved avdelingen som ønsket å bli observert i ulike situasjoner. Møtene vi planla å observere involverte en rekke forskjellige ansatte, og etter hvert ble det åpenbart at vi ikke kunne ta notater fra noen av statusmøtene eller lignende fordi det alltid var noen som ikke ønsket å delta i dette forskningsprosjektet på denne måten. Vi vurderte også å utføre noen observasjoner av hvordan utviklerne samarbeidet og delte kunnskap i arbeidshverdagen utenom disse møtene. Igjen, så ble det vanskelig å få nok samtykker fra ansatte til å få meningsfulle data. Vi så en sjanse til å utføre noen få observasjoner der utviklerne samarbeidet i par der vi hadde samtykke fra begge to, men disse var ikke mange. Derfor valgte vi å kutte ut observasjon som metode for datainnsamling i forskningsprosjektet, og heller bruke tiden på noe annet. På grunn av dette mistet vi metodetrianguleringen som vi hadde håpet å oppnå, men det ble nødvendig å stoppe med datainnsamlingen og fortsette med det som gjenstod å gjøre i forskningsprosjektet siden vi hadde en innleveringsfrist som vi måtte nå.

2.3 Dataanalyse

Nå som vi har beskrevet hvordan vi samlet inn data i forrige kapittel, så vil vi i dette kapittelet beskrive hvordan vi gikk fram for å analysere det vi samlet inn. Kapittel 18 om analyse av kvalitative data i Oates (2006) har vært stor hjelp for oss når vi har planlagt hvordan vi skal gå fram i analysen av dataene vi skulle samle inn. Erfaringene vi fikk med å samle inn og analysere data i pilotstudien var også meget verdifull i denne sammenhengen. I dette forskningsprosjektet var vi mye mer forberedt på denne delen av prosjektet enn vi var i pilotstudien. I dette kapittelet beskriver vi først hva som kjennetegner analyse av kvalitative data og hvordan vi prepararte dataene våre til analyse, deretter hva som var spesielt rundt analysen av dataene fra intervjuene.

2.3.1 Analyse av kvalitative data

I motsetning til analyser av kvantitative data der en kan bruke statistikk og matematikk til å analysere dataen, så er det ikke like lett med kvalitative data der forskeren må klare å finne temaer og mønstre ut fra ren tekst (Oates, 2006). Det er fullt mulig å bruke slike metoder på kvalitative data, men dette er mer uvanlig og ikke noe som vi brukte noe særlig i dette forskningsprosjektet (Oates, 2006). Siden en samler inn så mye tekst i stedet for bare tall, så kan det fort bli veldig mye data som en må analysere, noe som vi også opplevde med side på side med tekst etter transskriberingen var ferdig (Oates, 2006).

2.3.2 Preparasjon av data

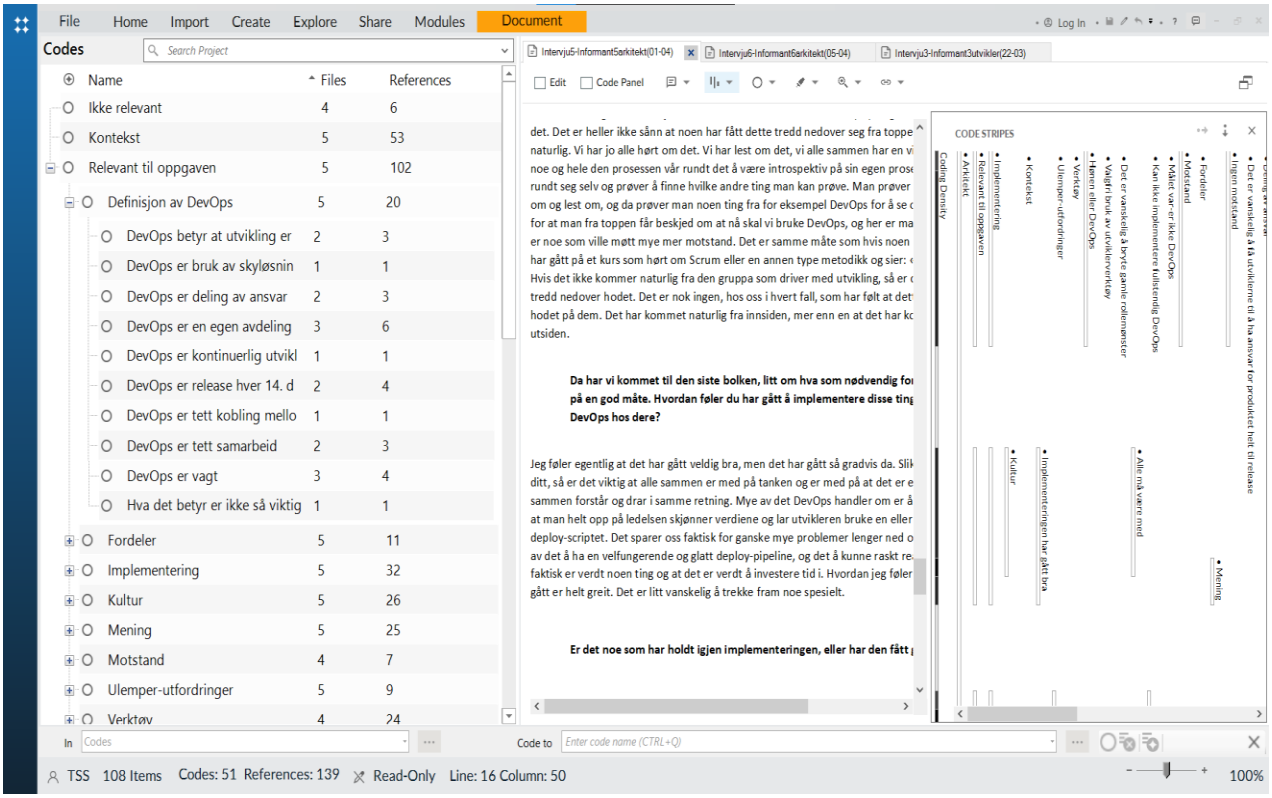
For å kunne analysere dataene våre bestemte vi oss for at alt skulle konverteres over til ren tekstform. Det er mye lettere å kategorisere og søke gjennom dataene når de står på denne formen i stedet for å måtte tolke korte notater eller lytte gjennom lange lydopptak (Oates, 2006). Alt av opptak og transskriberinger ble lagret på Microsoft Teams-løsningen som NTNU har gitt studenter som oss tilgang til. Vi hadde ekstra kopier av disse filene slik at de originale uendrede filene ikke skulle gå tapt ved et uhell.

Lydopptakene ble transskribert til tekst så fort som mulig etter hvert intervju var ferdig. Etter at vi følte at vi hadde transkribert ferdig et lydopptak, så sendte vi resultatet til intervjuobjektet på epost for godkjenning om de ønsket dette. Vi ønsket å forsikre oss om at vi ikke feiltolket intervjuobjektet, og at intervjuobjektet var komfortabelt med alt som stod der. Vi tilbydde å fjerne alt som intervjuobjektet ikke ville ha med. Dette er god praksis (Oates, 2006). Etter transskriberingen var godkjent av intervjuobjektet, så begynte vi å analysere dem.

2.3.3 Analyse av data

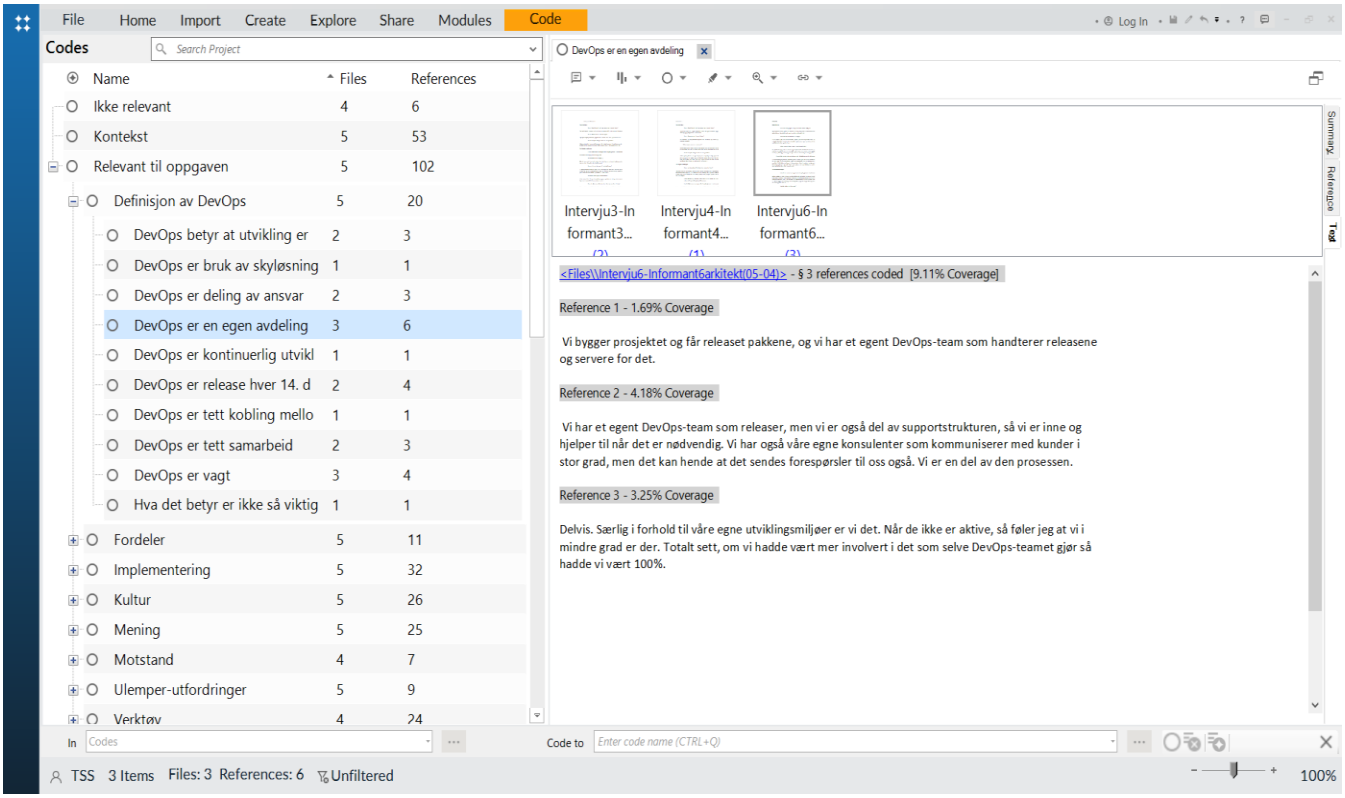
Etter at opptaket ble konvertert til ren tekst, så ble den først delt opp i tre temaer: irrelevant for oppgaven, relevant for forskningsspørsmålene og beskrivelse av konteksten. Med denne første oppdelingen fullført gikk vi gjennom det som hadde relevans for forskningsspørsmålene og delte det inn i kategorier ut ifra hva delene av intervjuet handlet om. Når en skal dele inn data på denne måten, så har en to metoder for å velge hvordan man skal kategorisere de ulike delene: deduktiv eller induktiv tilnærming (Oates, 2006). En induktiv tilnærming innebærer at en bare lager kategorier som kan utledes fra teksten en skal analysere, og bare derfra (Oates, 2006). Deduktiv tilnærming er det motsatte, der en bare bruker kategorier en har funnet fra litteratur en har lest på forhånd (Oates, 2006). Når vi kategoriserte våre data så brukte vi en blanding av begge tilnærmingene. På forhånd hadde vi lest litteratur rundt DevOps slik at vi var klar over noen kategorier som mest sannsynlig ville komme opp, og i pilotstudien opplevde vi at det oppstod noen kategorier som vi ikke forventet når vi bare hadde lest litteraturen. Derfor brukte vi kategorier fra både litteraturen/teorien vi hadde lest og fra selve tekstdokumentet i dette forskningsprosjektet. For å lage noen grovere kategorier/temaer bruke vi den deduktive tilnærmingen for å dele dataene inn etter det vi søkte etter. Disse temaene var fordeler, ulemper/utfordringer, mening, motstand, verktøy, implementering, definisjon av DevOps og kultur. Videre gikk vi gjennom kategoriene vi hadde laget, og sjekket om det var mulig å dele dem inn i mindre kategorier eller om det var mulig å kombinere noen av dem. Her brukte vi en induktiv tilnærming, og vi delte dataene inn etter det som faktisk stod i teksten for å lage mindre koder som viste hva som kom fram i mindre deler av teksten. Det ligger en liste med alle kodene som ble brukt i dette prosjektet under vedlegg 4.

Vi bestemte oss for å bruke NVIVO for å få bedre oversikt over kodene vi brukte, og vi hadde hørt fra medstudenter at dette programmet gjorde at det ble enklere og raskere å både kode og gå gjennom koden i ettertid. Dette var mye bedre enn å bruke kommentarfunksjonen i Word som vi gjorde i pilotstudien. NTNU gir sine studenter muligheten til å laste ned og bruke NVIVO gratis. Ett problem som oppstod var at det er ingen måte å skrive sammen på den samme filen samtidig i versjonen av NVIVO som NTNU gav oss tilgang til, og vi er to studenter som utfører dette forskningsprosjektet sammen. For å løse dette delte vi intervjuene mellom oss der vi kodet disse hver for oss. Etter den første inndelingen diskuterte vi hvilke koder som vi burde dele det relevante opp i, før vi delte våre tildelte intervju opp i disse. Dette skjedde i flere runder. Når vi var fornøyde med kodene dataene var delt inn i, så delte vi prosjektfilene med hverandre og slo de sammen slik at begge hadde tilgang til alle de fullstendig kodede dataene. Til slutt satt vi igjen med en kategorisert oversikt over intervjuene, der de forskjellige intervjuene er delt opp i mindre koder, slik at det er lettere å se hva de inneholder. Dette er en vanlig måte å analysere data på, som har blitt beskrevet av Oates (2006). Figur 1 viser et skjermbilde fra vårt prosjekt i NVIVO. Til venstre kan man se kodene som vi har lagt inn i prosjektet, og at noen koder er underlagt andre. I midten ligger transskriberingene av intervjuene, og til høyre ser man hvilke deler av teksten som har fått tildelt visse koder. Dette oppsettet er ett eksempel på hvordan programmet så ut når vi brukte det.



Figur 1: Skjermbilde I fra NVIVO

Man kan for eksempel gå inn i selve kodene og bare se det som er merket med den koden. Dette var svært nyttig når vi skulle skrive om spesifikke tema i resultat- og diskusjonsdelen. På Figur 2 kan man se et eksempel på dette. Til venstre ligger igjen alle kodene, og nederst til høyre ligger alt som er merket med den valgte koden i et valgt transskriberingsdokument som man kan se over teksten.



Figur 2: Skjermbilde II fra NVIVO

Et problem vi støtte på i bruken av NVIVO var at en av oss brukte Mac-versjonen og den andre brukte Windows-versjonen av programmet. En kan ikke slå sammen prosjektfiler fra disse forskjellige versjonene. For å gjøre det må en konvertere filene til samme versjon ved å laste ned en egen konverterer. En må også ha samme versjonsnummer i programmet en bruker som det versjonsnummeret programmet som prosjektfilen ble opprettet i for å åpne den, noe som også var forskjellig mellom versjonene vi brukte. Etter en del prøving og feiling, så ble konverteringsprogramvaren lastet ned og versjonene ble oppdatert slik at vi kunne slå sammen prosjektfilene.

2.4 Metodekvalitet

Etter beskrivelsen av hvordan gjennomføringen av forskningsprosjektet har foregått, så kommer det nå et kapittel som handler om kvaliteten av metodene vi har brukt. Kapittelet skal overbevise

leseren om at de forskjellige stegene i prosessen har blitt utført på en måte som gjør at resultatet oppleves som troverdig. Først vil vi gå gjennom hva som kjennetegner kvalitet ved bruk av interpretivisme og forskjellen fra positivisme. Videre kommer litt drøfting rundt det med å tolke data korrekt når en bruker kvalitative metoder.

2.4.1 Kvalitet og interpretivisme

Siden vi har tatt utgangspunkt i interpretivismen og brukt en kvalitativ metode, så blir vurderingen av kvaliteten litt annerledes enn den ville blitt om vi brukte positivismen (Oates, 2006). Vi inkluderer ikke et gjennomsnitt på hva alle har sagt som en kan gjøre med kvantitative data, men vi tolker dataene gjennom analysen vi har gjort. I positivismen er det visse kriterier som sjekkes når en vurderer kvaliteten på et forskningsprosjekt: validitet, objektivitet, intern/ekstern validitet og pålitelighet (Oates, 2006). Det eksisterer en lignende liste over kriterier for forskning som er basert på interpretivismen: troverdighet, overførbarhet, pålitelighet, bekreftelighet og grad av dokumentasjon av prosess (Oates, 2006). Som en kan se så har kriteriene til interpretivismen mye mer fokus på at leseren skal bli overbevist om at prosessen ble utført på en god måte, og hvor mye en kan stole på resultatet. Problemet er at disse kriteriene kan kritiseres for å bare eksistere for å kunne se på interpretivismen på samme måte som positivismen (Oates, 2006). Dette gjør at kriteriene egentlig ikke passer til å vurdere kvaliteten til forskningsprosjekt basert på interpretivismen (Oates, 2006). Det en kan gjøre er å bruke essensen av disse kriteriene, som er: en må utføre forskningen i et forsøk på å faktisk forstå de som deltar i prosjektet i tillegg til at en har dokumentert prosessen på en slik måte at den som leser dokumentene blir overbevist at forskeren(e) ikke har funnet på det som står der (Oates, 2006). Som validering har vi derfor grundig dokumentert hele prosessen vi gikk gjennom i dette prosjektet steg for steg slik at alle som leser dette dokumentet kan følge hele prosessen fra start til slutt. Alle intervjuene ble utført med den samme intervjuguiden som vi har inkludert i vedlegg 1. Eventuelle problemer vi støtte på har også blitt dokumentert for å vise vår troverdighet. Valgene vi har tatt i metoden har også blitt diskutert, med potensielle positive og negative følger. Selv om vi har dokumentert prosessen og gitt grunner for hvorfor vi har tatt de valgene vi tok, så er det mange som vil kritisere et slikt forskningsprosjekt fordi kvalitative metoder med

interpretivismen ikke klarer å tilfredsstille kriteriene for kvalitet satt for positivisme på samme måte som positivismen og da ikke er god forskning i deres øyne (Oates, 2006).

2.4.2 Rett tolkning av data

Når vi hadde så stort fokus på tolkning av data på grunn av at vi samlet inn kvalitative data, så stod vi i fare for at våre egne syn ville få mye mer vekt enn intervjuobjektene. Om dette skjedde ville vi ha presentert våre meninger som resultatet av undersøkelsen i stedet for de som faktisk deltok i forskningsprosjektet. Derfor var det viktig at vi gav intervjuobjektene muligheten til å lese gjennom transskriberingene våre slik at vi kunne med sikkerhet begynne å analysere noe som intervjuobjektet hadde godkjent som sin mening. Ved å bruke metodetriangulering kunne vi bruke intervjuer og observasjoner til å validere hverandre og for å kunne se om det var forskjell mellom det deltakerne sa i intervjuene og hva de faktisk gjorde når vi observerte dem (Oates, 2006). Vi valgte å bruke flere metoder slik at vi hadde en sjanse til å øke kvaliteten på forskningsprosjektet (Oates, 2006). Når vi bestemte oss for å ikke inkludere observasjoner som en metode mistet vi metodetriangulering. Dette senket vår evne til å validere det som ble sagt i intervjuene, og for å prøve å veie opp for dette stilte vi flere spørsmål om konteksten og arbeidsforholdene under de resterende intervjuene. Siden vi ikke fikk noen fra den tekniske avdelingen til å delta, så mister vi også et viktig perspektiv.

2.5 Etikk

Nå er vi kommet til slutten av metodekapittelet, og det inneholder en gjennomgang av hvordan vi forsikret oss om at dette forskningsprosjektet ble gjennomført på en etisk måte. Først og fremst var vi ærlige med alle som vi snakket med eller hadde interaksjoner med, i tillegg til at vi respekterte løfter vi gjorde og ønsker de hadde i forbindelse med deres deltakelse. Dette en sentral del av å være en etisk forsker (Oates, 2006). Vi måtte også søke tillatelse fra NSD før vi begynte å samle inn data, der vi oppgav detaljer om hvordan vi hadde tenkt å gjennomføre prosjektet/datainnsamlingen og hvordan vi skulle behandle dataene vi samlet inn. De får tilsendt mange slike søknader, og de sjekker om forskningsprosjekter blir utført skikkelig med tanke på datainnsamling og lagring av data. Søknaden vår ble godkjent av NSD, og godkjennelsen ligger

under vedlegg 3. Det er viktige å ha rettighetene til deltakerne i tankene når man gjennomfører slike forskningsprosjekter, og de kan ifølge Oates (2006) deles inn i:

- Rett til å ikke delta
- Rett til å trekke tilbake deltakelse
- Rett til å gi informert samtykke
- Rett til anonymitet
- Rett til konfidensialitet

Vi vil gå gjennom hvordan vi dekket hver enkelt av disse rettighetene i løpet av dette kapittelet.

2.5.1 Deltakelse og tilbaketrekking av deltakelse

Deltakelse i forskningsprosjektet for de ansatte på avdelingen var helt frivillig. Vi fikk en liste over ansatte fra ledelsen på avdelingen, men det var opp til oss hvem vi ville inkludere i forskningsprosjektet og de ble ikke beordret til å delta av ledelsen. Alle de vi ville inkludere fikk en epost med informasjonsskrivet og samtykkeerklæringen. Vi snakket også en del ansikt til ansikt med flere ansatte ved avdelingen ved flere anledninger der vi snakket om hva forskningsprosjektet innebar, og de fleste var generelt nysgjerrige og positive til å delta i forskningsprosjektet. Vi informerte også i informasjonsskrivet vårt at deltakerne kunne trekke seg fra undersøkelsen når som helst ved å ta kontakt med oss, eller ikke svare på spørsmålene våre. Dette var viktig å inkludere, selv om det kunne hindre prosjektet (Oates, 2006).

2.5.2 Informert samtykke

Deltakere i et forskningsprosjekt skal ha muligheten til å gi et informert samtykke, som betyr at de skal ikke gi samtykke til å delta i prosjektet før de har fått en full forståelse for hva en slik deltakelse vil innebære og hva prosjektet handler om (Oates, 2006). Dette inkluderer hensikten til datainnsamlingen samt hvorfor og hvilke fordeler forskeren håper å få utav den, hvem som utfører og er ansvarlig for den, hvem som er involvert og hvordan dataen vil blir brukt (Oates,

2006). Alt dette var inkludert i vårt informasjonsskriv som ble sendt ut til alle vi ville ha som deltakere i case-studien som også ligger under vedlegg 2, og før vi ba om samtykke til de vi ville ha med som deltakere. Noen forskningsprosjekter som for eksempel inkluderer eksperimenter kan av og til slippe å måtte gi testpersonene sine informerte samtykker fordi dette ville endre resultatet (Oates, 2006). Dette var ikke relevant for oss, siden vi brukte case-studie med intervjuer som metode. Derfor var det et krav at våre deltakere måtte få gjøre informert samtykke.

2.5.3 Anonymisering

Anonymisering var viktig for oss siden datainnsamlingen bestod av intervjuer, og deltakerne har rett på beskyttelse av sin identitet og lokasjon (Oates, 2006). Ingen enkeltpersoner på avdelingen som deltok i case-studien er nevnte med navn i dette dokumentet. Dette gjorde vi for å unngå at enkeltpersoners syn og utsagn kommer klart frem. Siden denne case-studien ble utført ved en relativt liten avdeling i et internasjonalt selskap, så var det en sjanse for at utsagn fra deltakerne kunne være identifiserende nok til at enkeltpersoner ble gjenkjent (Oates, 2006). Dette var noe vi nevnte i søknaden vår til NSD. Etter beste evne har vi prøvd å unngå dette når vi har brukt utsagn fra deltakerne ved å bare bruke utsagn som er minst mulig identifiserende, men dette var ikke alltid like lett siden det kan være vanskelig for oss som utenforstående å vite hvilke utsagn som kan identifisere enkeltpersoner. Om vi ikke anonymiserte dataene våre godt nok, og om for eksempel noen som kom med utsagn som ikke ledelsen i bedriften likte, så kunne det vært en mulighet for at de ble straffet. I noen tilfeller uttrykte intervjuobjektene at noe av det som stod i transskriberingen muligens kunne brukes for å identifisere dem, men at de ikke syntes dette var et problem. I tillegg informerte vi intervjuobjektene våre på forhånd om hvilken grad av anonymisering dataen vi innhentet ville ha slik at de følte seg trygg til å dele meningene sine med oss i stedet for å holde tilbake hva de mente, som en kan se i informasjonsskrivet vårt som vi sendte ut til de vi ville inkludere i forskningsprosjektet vårt. Disse ble utarbeidet fra malen som NSD har lagt ut i forbindelse med søknader til forskningsprosjekt på deres hjemmeside. Vi har også valgt å anonymisere navnet på bedriften i forskningsprosjektet.

2.5.4 Konfidensialitet

I forbindelse med dette forskningsprosjektet skrev vi under på en NDA med bedriften, slik at de var sikre på at vi ikke gav ut noen forretningshemmeligheter som vi fikk vite om mens vi var der. Dette ble diskutert i oppstarten av forskningsprosjektet, og alle parter ble enige om hva som ikke kunne inkluderes i resultatene. Siden resultatene ikke inneholder noen sensitive data om bedriften, så ble vi og representanten for bedriften enige om at dette dokumentet kunne offentliggjøres med en gang. Innsamlet data fra intervjuene har blitt lagret på løsninger vi har blitt gitt av NTNU, som OneDrive og Teams. Intervjuene vi utførte ble tatt opp på en diktafon vi fikk låne av vår kontaktperson. Disse ble kopiert til en av de tidligere nevnte lagringsløsningene, før kopien på diktafonen ble slettet. Dette ble gjort fordi slik data skal lagres på en sikker måte (Oates, 2006). Vi skrev fra kontorlokalene til avdelingen med våre egne bærbare datamaskiner, og passet på å lukke igjen maskinene når vi for eksempel gikk til lunsj eller til toalettet, slik at ingen som gikk forbi kunne se noe de ikke skulle. Inngangsdøren til kontorlokalet var også låst, og bare ansatte kunne komme inn. Vi fikk nøkler som gav oss tilgang til kontoret når vi skrev på prosjektet. Utstyret vi brukte var dermed godt sikret mens vi jobbet med forskningsprosjektet. Vi informerte deltakerne på forhånd om at bare det som de godkjente kom til å være med i forskningsprosjektet, som en kan se i informasjonsskrivet. I etterkant av intervjuene sendte vi transskriberingen av intervjuet til intervjuobjektet om de ønsket dette, der de hadde mulighet til å si om noe måtte endres for å passe på at det intervjuobjektet var komfortable med at vi tok med det som stod i transskriberingen. Flere endringer ble lagt inn i transkriberingene i etterkant etter intervjuobjektenes ønsker, men ikke alle intervjuobjektene ønsket å se gjennom transskriberingen i etterkant.

3 Teori

DevOps har blitt et bredt diskutert fenomen innen programvarebransjen de siste årene. DevOps tar for seg det som ofte beskrives som tomrommet mellom utvikling og driftspersonell. Mange har positive forventninger til DevOps og bedrifter blir stadig mer interesserte i fenomenet og hvordan man kan utnytte de potensielle fordelene. Likevel er begrepet DevOps omgitt av tvetydighet. Selve målet med DevOps er tydelig, vi ønsker å bygge en bro mellom utviklerne og driftspersonalet. Definisjonen derimot er ikke så tydelig, det finnes fortsatt mange tolkninger av hva DevOps faktisk betyr (Smeds, Nybom, & Porres, 2015). Å ta i bruk DevOps er kanskje ikke så rett frem som man tror da det kan kreve at en organisasjon gjør prosess, personell og teknologiske endringer. I hvert initiativ for forbedring av programvareprosesser, vil veien til en vellykket DevOps adopsjon være unik for hver organisasjon (Smeds, Nybom, & Porres, 2015).

For å få en bedre forståelse hva DevOps betyr, har vi samlet inn litteratur fra ulike kilder og sammenlignet dem. Litteraturen som er plukket ut viser tydelig at det å komme frem til en klar definisjon av begrepet DevOps er utfordrende. Definisjonene i litteraturen varierer, noen er meget vage og andre er begrenset til en bestemt kontekst.

I en artikkel forklarer forfatteren Christofer Ekbert at DevOps integrerer det som tidligere har blitt ansett som to adskilte verdener, utvikling og drift. Dette gjøres ved bruk av automatisert utvikling, utrulling og monitorering av infrastruktur. DevOps fører til et organisatorisk skifte der istedenfor å ha distribuerte separate grupper som utfører funksjoner separat, kjøres det nå kryssfunksjonelle teams som jobber med kontinuerlig operativ funksjonsleveranser. Denne tilnærmingen bidrar til å levere verdi raskere og kontinuerlig, redusere problemer på grunn av feil kommunikasjon mellom teammedlemmer, samt akselerere problemløsning. DevOps betyr et kulturskifte mot samarbeid mellom utvikling, kvalitetssikring og drift (Ekbert, Gallardo, Hernantes, & Serrano, 2016).

Ekbert forklarer videre at verktøy er nødvendig for å automatisere og implementere DevOps. Kvalitetsleveranser med kort syklustid trenger en høy grad av automatisering. Så, å velge riktige

verktøy for miljøet eller prosjektet er viktig når man flytter over til DevOps. (Ekbert, Gallardo, Hernantes, & Serrano, 2016).

I en annen artikkel forklarer Emily Freeman at det er vanskelig å gi en eksakt DevOps beskrivelse, fordi det ikke eksisterer. Her blir DevOps referert til som en filosofi som guider programvareutvikling, en filosofi som prioriterer personer ovenfor prosesser og prosesser over verktøy. DevOps bygger en tillitskultur, samhandling og legger til rette for kontinuerlig forbedring. Denne artikkelen går også inn på kultur. Det beskrives at som kultur ser DevOps på utviklingsprosessen på en helhetlig måte og tar hensyn til alle involverte: Utviklere, testere, driftspersonell, sikkerhetsfolk og infrastrukturingeniører. DevOps setter ikke noen av disse gruppene over de andre, og rangerer heller ikke viktigheten av arbeidet deres. I stedet behandler et DevOps selskap hele teamet av medarbeidere som kritiske for å sikre at kunden får en best mulig opplevelse (Freeman, 2019).

I artikkelen "Adopting DevOps Practices in Quality Assurance fremmes det også at det ikke finnes en standard definisjon for DevOps. James Roche tar for seg to blogginnlegg som diskuterer hva DevOps vil si. På den ene siden diskuteres det at DevOps kan være en jobbeskrivelse. Kort oppsummert blir det i den ene bloggen forklart at utviklere jobber for det meste med koding, driftspersonell jobber mest med drift av systemer, og at DevOps er en blanding av disse to områdene. Den andre bloggen er ikke helt imot denne oppfatningen, men argumenterer for at DevOps ikke er en jobb (ved at du ansetter en «DevOp» i seg selv), men heller at ånden til DevOps taler til et voksende behov i det moderne programvareutviklingslandskapet (Roche, 2013).

I en blogg skrevet av en John Willis, presenteres det fire aspekter som han mener er de definerende karakteristikkene på DevOps. Han har snakket og intervjuet mange av tankelederne i det han kaller for DevOps bevegelsen. Det han kom frem til er at DevOps i stor grad handler om CAMS. Dette står for culture, automation, measurement og sharing (Willis, 2016).

- **Culture** - Mennesker og prosesser først. Hvis du ikke har en samhandlingskultur, vil alle automatiseringsforsøk være resultatløse (Willis, 2016).

- **Automation** - Dette er en av stedene man kan starte når man har fått en forståelse for kulturen. På dette punktet kan verktøyene begynne å sette sammen en automatiseringsinfrastruktur for DevOps. Verktøy for administrasjonsutgivelse, klargjøring, konfigurasjon, administrasjon, systemintegrasjon, overvåking og kontroll samt orkestrering blir viktige brikker når det bygges en DevOps infrastruktur (Willis, 2016).
- **Measurement** - Hvis du ikke kan måle, kan du ikke forbedre deg heller. En vellykket DevOps-implementering vil måle alt den kan så ofte den kan. Her er det snakk om ytelsesmålinger, prosessmålinger og til og med personmålinger (Willis, 2016).
- **Sharing** - Deling er tilbakekoblingen i CAMS-syklusen. Å skape en kultur der folk deler ideer og problemer er avgjørende. Det er viktig at utviklerne og driftere ser på problemene som fienden og ikke hverandre (Willis, 2016).

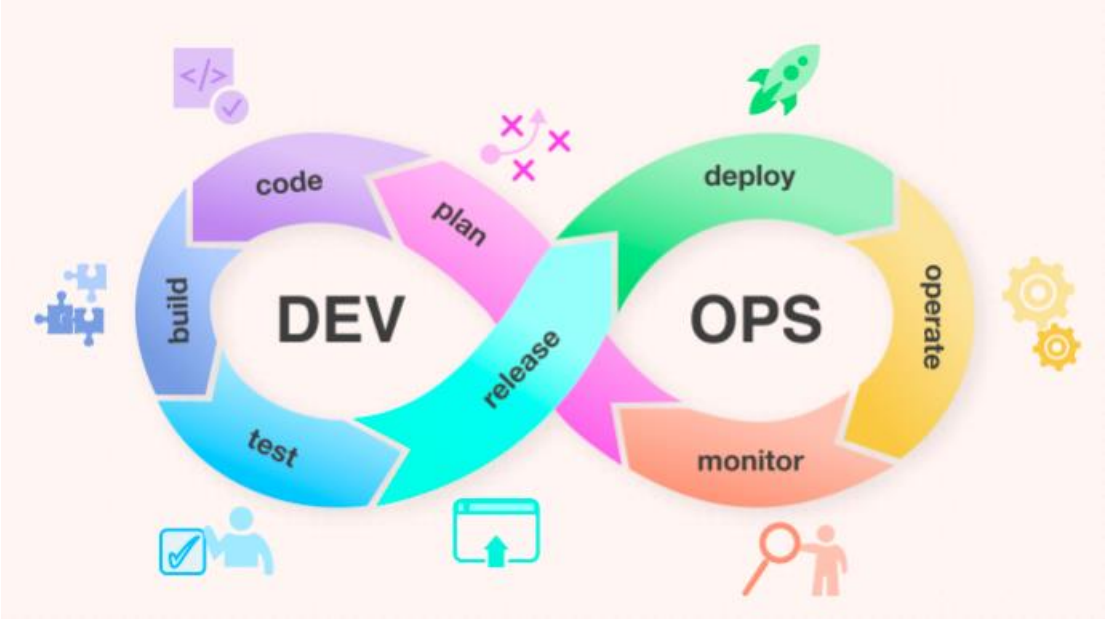
DeGrandis antyder at det å ta i bruk DevOps er en organisatorisk revolusjon, "Revolusjonen under utvikling". Det forklares at DevOps er et skifte fra fokus på separate avdelinger som jobber uavhengig til et organisasjonsomfattende samarbeid. Dette er en tilnærming som refereres til som "systemtenkning". Det handler om å adressere alt arbeidet som en helhet, versus å se på det som biter. Det handler om arbeid som flyter på tvers av funksjoner versus i lukkede siloer. Som en del av en "systemtenkende" tilnærming handler DevOps om respekt, samarbeid og tillit blant individene som gjør arbeidet kontra ledelses drevet kontroll. (DeGrandis, 2011)

I boken "Building a DevOps Culture" beskriver Walls DevOps som en "kulturell bevegelse kombinert med en rekke programvare relaterte praksiser som muliggjør rask utvikling". Senere i boken beskriver Walls DevOps kulturen med fire sentrale kulturelle kjennetegn: Åpen kommunikasjon, samordning av insentiver og ansvar, respekt og tillit.

Videre beskrives det i boken hvordan man skal nå en så kalt DevOps kultur. Walls mener at begrepet ble introdusert for å definere en ønsket kultur som organisasjoner kan sikte mot (Walls, 2013). Han antar at kulturen i en organisasjon kan endres mot en målkultur ved å følge et sett med trinn. Walls tar til slutt opp at begrepet DevOps er nytt, men at DevOps har eksistert lenge

før begrepet ble kjent. Han mener at siden det er kun begrepet som er nytt, så kan det muligens være grunnen til at det finnes ulike tolkninger og vage definisjoner (Walls, 2013).

DevOps livssyklus beskrives som en serie automatiserte utviklingsprosesser eller arbeidsflyter innenfor en iterativ utviklingslivssyklus. DevOps følger en kontinuerlig tilnærming og livssyklusen symboliseres derfor i form av en uendelig sløyfe. Denne sløyfen presenterer den samarbeidende og iterative tilnærmingen gjennom hele anvendelsen av livssyklusen, bestående av verktøy og teknologistabler for hvert trinn (Dhaduk, 2022). Se figur nedenfor.



Figur 3 DevOps livssyklus (Dhaduk, 2022).

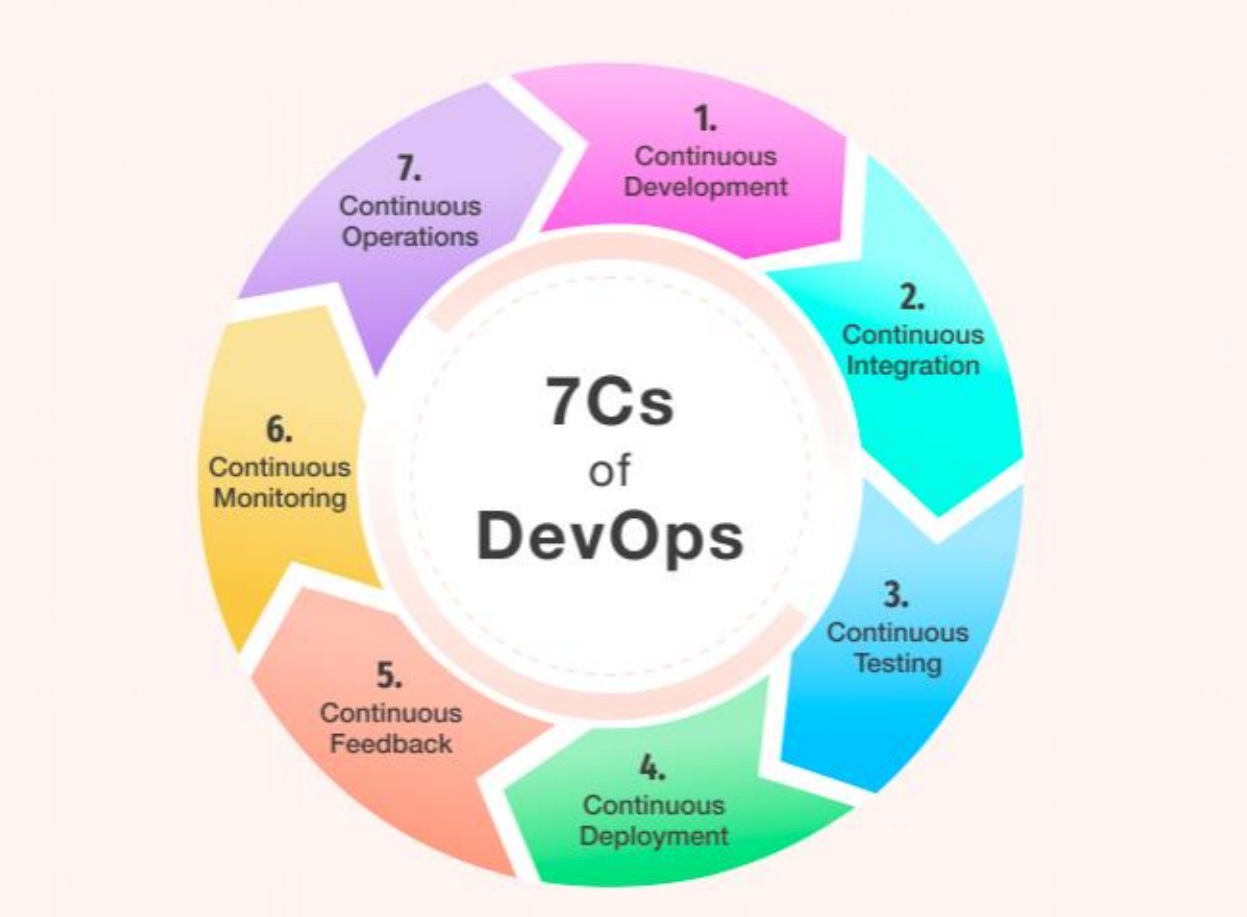
Den venstre delen tar for seg programvareutvikling og testing. Den motsatte siden av sløyfen representerer utgivelses- og driftssyklusen utplasserings- (Dhaduk, 2022). Kort forklart fungerer stegene DevOps livssyklusen slik:

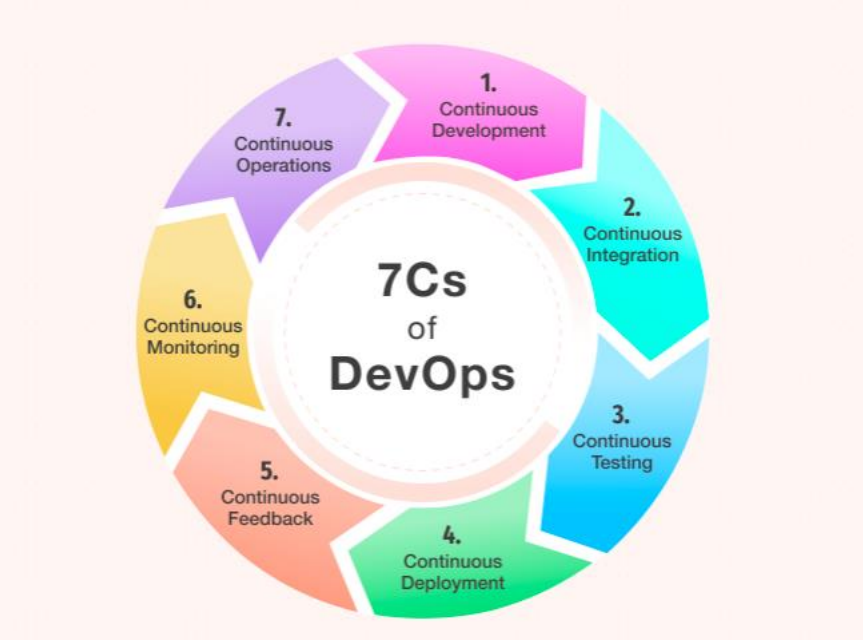
1. **Plan**: På dette stadiet identifiserer teamene forretningskravet og samler tilbakemeldinger fra sluttbrukere. De lager et prosjektveikart for å maksimere forretningsverdien og levere det ønskede produktet i løpet av denne fasen (Dhaduk, 2022).

2. **Kode:** På dette stadiet foregår kode utviklingen. Utviklingsteamene bruker verktøy og plugins som Git for å strømlinjeforme utviklingsprosessen, noe som hjelper dem å unngå sikkerhetsfeil og dårlig kodepraksis (Dhaduk, 2022).
3. **Bygg:** Når utviklerne er ferdige med oppgaven, legger de koden til det delte kodelageret ved å bruke byggeverktøy som for eksempel Maven og Gradle (Dhaduk, 2022).
4. **Test:** Når koden er bygget ferdig, distribueres den til testmiljøet først for å utføre flere typer tester som brukeraksepttest, sikkerhetstest, integrasjonstest og ytelsestest. Dette gjøres ved å bruke verktøy som Selenium og JUnit som også hjelper med å sikre programvarekvalitet (Dhaduk, 2022).
5. **Utgivelse:** I denne fasen er programvaren klar til å distribueres i produksjonsmiljøet. Når programvaren har bestått alle testene, planlegger driftsteamet utgivelsene eller distribuerer flere utgivelser til produksjon, avhengig av organisasjonens behov (Dhaduk, 2022).
6. **Distribusjon:** I denne fasen hjelper Infrastructure-as-Code med å bygge produksjonsmiljøet og distribuerer deretter programvaren ved hjelp av forskjellige verktøy (Dhaduk, 2022).
7. **Betjene:** På denne fasen i sløyfen er utgivelsen live for bruk av kunder. På dette stadiet tar driftsteamet seg av serverkonfigurering og klargjøring ved hjelp av ulike verktøy (Dhaduk, 2022).
8. **Overvåking:** Her overvåkes DevOps-pipelinen basert på data som er samlet inn fra kundeadferd, applikasjons ytelse, osv. Overvåking av hele miljøet hjelper teamene med å finne flaskehalsene som påvirker utviklings- og driftsteamenes produktivitet (Dhaduk, 2022).

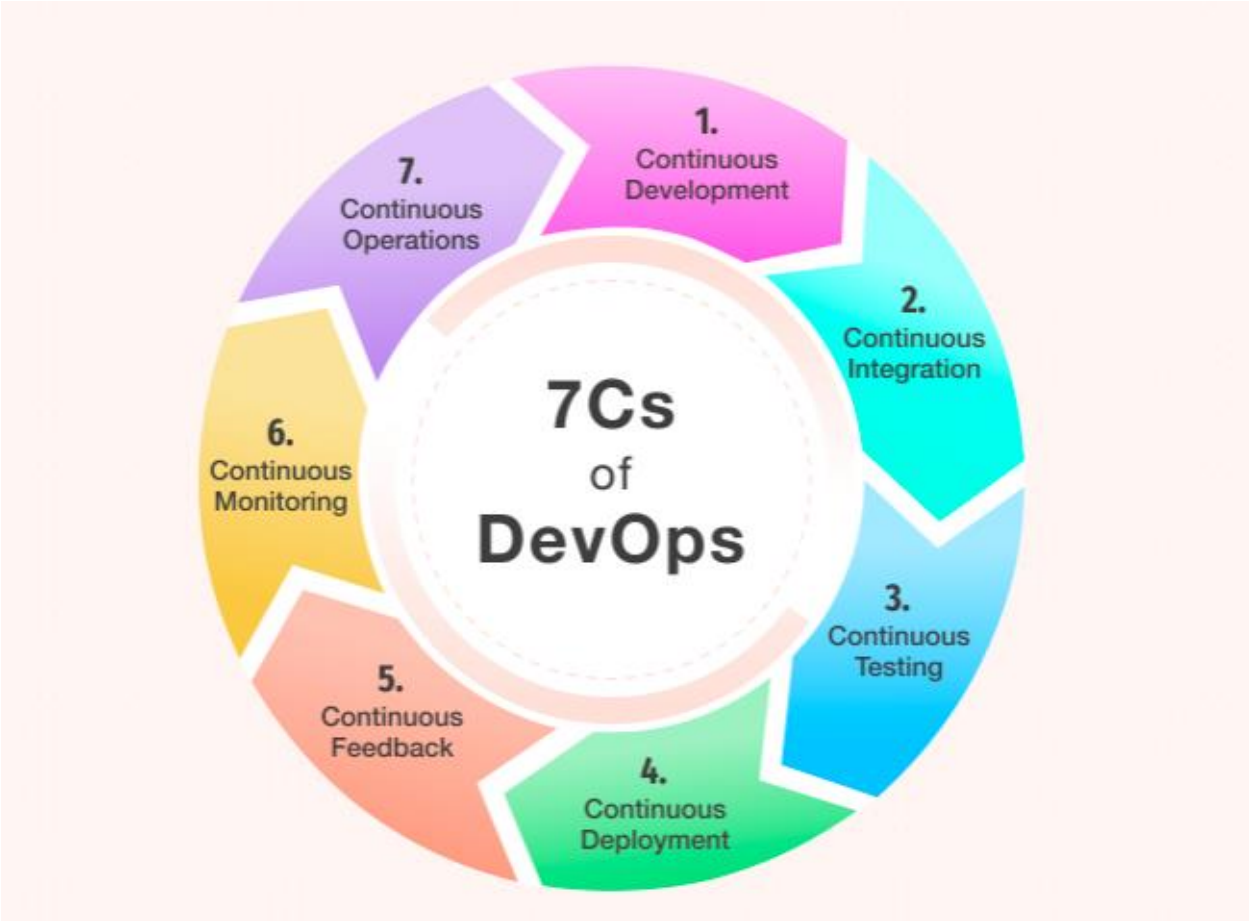
Slik det ble nevnt tidligere er alt kontinuerlig i DevOps, helt fra planlegging og frem til overvåkning. Videre kan livssyklusen brytes ned i syv ulike faser der kontinuitet ligger i kjernen. Enhver fase i livssyklusen kan gjentas flere ganger gjennom prosjektene, helt til man får det ønskede resultatet (Dhaduk, 2022). Figuren under viser de 7 fasene av

DevOps.:


1.
Continuous
Development
2.
Continuous
Integration
3.
Continuous
Testing
4.
Continuous
Deployment
5.
Continuous
Feedback
6.
Continuous
Monitoring
7.
Continuous
Operations
7Cs
of
DevOps



Figur 4 DevOps 7 faser (Dhaduk, 2022).



1. Kontinuerlig utvikling:

Denne fasen spiller en sentral rolle i å avgrense visjonen for hele programvareutviklingssyklusen. Den fokuserer først og fremst på prosjektplanlegging og koding. I denne fasen samles prosjekt kravene og diskuteres med interessenter. I tillegg opprettholdes produkt backloggen basert på tilbakemeldinger fra kunder som er brutt ned i mindre utgivelser og milepæler for kontinuerlig programvareutvikling. Når teamet er enige om forretningsbehovene, begynner utviklingsteamet å kode for å møte de ønskede kravene. Det er en kontinuerlig prosess der utviklere er pålagt å kode når det skjer endringer i både prosjektkrav eller i tilfelle ytelsesproblemer (Dhaduk, 2022).

2. Kontinuerlig integrasjon:

Kontinuerlig integrasjon er den mest avgjørende fasen i hele DevOps livssyklusen. I denne fasen blir oppdatert kode eller tilleggsfunksjonaliteter og funksjoner utviklet og integrert i den eksisterende koden. I denne fasen blir det i tillegg blir det oppdaget og identifisert bugs i koden på hvert steg gjennom enhetstesting, kildekoden blir modifisert deretter. Dette trinnet gjør integrasjon til en kontinuerlig tilnærming der kode testes ved hver "commit". Videre blir de nødvendige testene også planlagt i denne fasen (Dhaduk, 2022).

3. Kontinuerlig testing:

Noen team gjennomfører den kontinuerlige testfasen før integrasjonen skjer, mens andre gjør det etter integrasjonen. Kvalitetsanalytikere tester kontinuerlig programvaren for feil og problemer i løpet av dette stadiet ved å bruke «docker-container» som er en åpen kildekode containeriseringsplattform. Ved en bug eller en error sendes koden tilbake til integrasjonsfasen for modifikasjoner. Automatisert testing reduserer også tiden og innsatsen for å levere kvalitets resultater. Team grupper bruker verktøy som Selenium på dette stadiet. Videre forbedrer kontinuerlig testing test evalueringsrapporten og minimerer provisjons og vedlikeholdskostnadene til testmiljøene (Dhaduk, 2022).

4. Kontinuerlig utgivelse

Dette er den mest avgjørende og aktive fasen i DevOps livssyklusen. Dette er hvor sluttkoden blir levert til produksjonsservere. Den kontinuerlige leveringen innebærer konfigurasjon ledelse for å gjøre levering av kode på serverene så nøyaktige og smidig som mulig. Utviklingsteam leverer koden til servere og planlegger oppdateringer for servere, noe som holder konfigurasjonene konsistente gjennom hele produksjonsprosessen. Containeriseringsverktøy hjelper også i distribusjonsprosessen ved å gi konsistens på tvers av utvikling, test, produksjons og utgivelsesmiljøer. Denne praksisen gjør det mulig for kontinuerlig utgivelse av nye funksjoner i produksjonen (Dhaduk, 2022).

5. Kontinuerlig tilbakemelding

Kontinuerlig tilbakemelding ble til for å analysere og forbedre applikasjonskoden. Gjennom denne fasen blir kundeatferd evaluert regelmessig for hver levering. Dette er for å forbedre fremtidige leveranser og utplasseringer. Bedrifter kan enten velge en strukturell eller ustrukturert tilnærming for å samle inn tilbakemeldinger. I den strukturelle tilnærmingen samles tilbakemeldinger gjennom spørreundersøkelser og spørreskjemaer. I en ustrukturert tilnærming mottas tilbakemeldingene gjennom sosiale medieplattformer. Totalt sett er denne fasen avgjørende for å gjøre kontinuerlig levering mulig og introdusere en bedre versjon av applikasjonen (Dhaduk, 2022).

6. Kontinuerlig monitorering

I løpet av denne fasen blir applikasjonens funksjonalitet og egenskaper monitorert kontinuerlig for å oppdage systemfeil, slik som lite lagringsplass, ikke-tilgjengelig server, osv. Denne prosessen hjelper IT-teamet raskt med å identifisere problemer knyttet til appytelse og årsaken bak det. Hvis IT-teamet finner et kritisk problem, går applikasjonen gjennom hele DevOps-syklusen på nytt for å finne løsningen. Sikkerhetsproblemene kan imidlertid oppdages og løses automatisk i denne fasen (Dhaduk, 2022).

7. Kontinuerlig drift

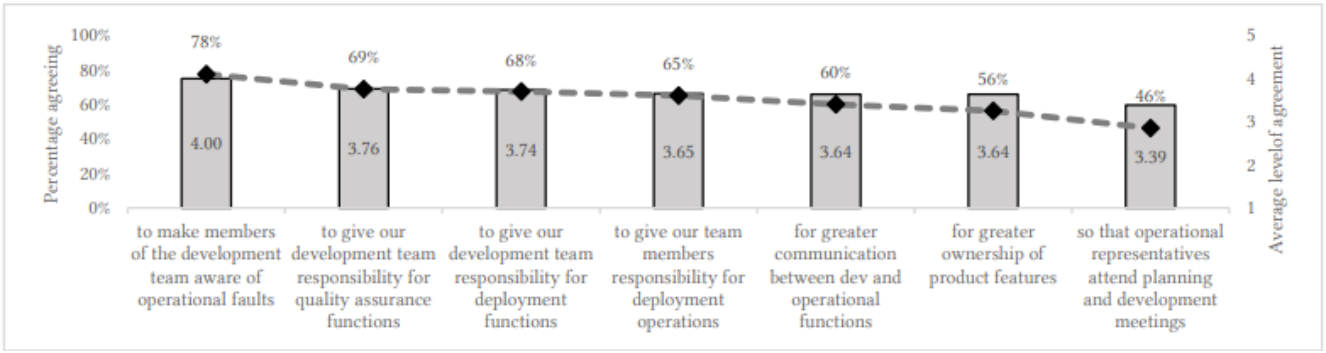
Den siste fasen i DevOps-livssyklusen er avgjørende for å redusere den planlagte nedetiden, for eksempel planlagt vedlikehold. Vanligvis er utviklere pålagt å ta serveren offline for å gjøre

oppdateringer, noe som øker nedetiden og kan til og med forårsake betydelig tap for selskapet. Etter hvert vil kontinuerlig drift automatisere prosessen med å starte applikasjonen og dens oppdateringer. Den bruker container administrasjonssystemer som Kubernetes og Docker for å eliminere nedetid. Disse containeradministrasjonsverktøyene hjelper til med å forenkle prosessen med å bygge, teste og distribuere applikasjonen i flere miljøer. Hovedmålet med denne fasen er å øke applikasjonens oppetid for å sikre uavbrutte tjenester. Gjennom kontinuerlig drift sparer utviklere tid som kan brukes til å akselerere applikasjonens "time-to-market" (Dhaduk, 2022).

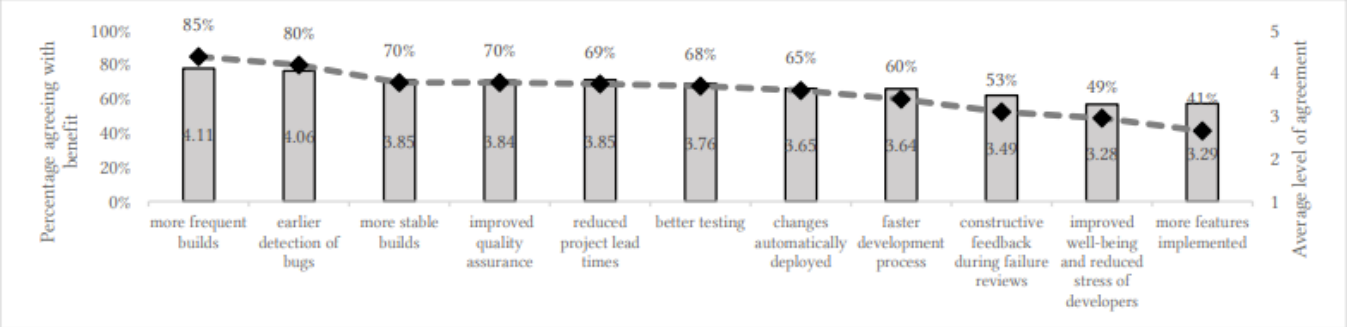
3.1 Erfaringer med DevOps i case-studier og andre artikler fra verden

Noe som er viktig for en bedrift, er at de ansatte er motiverte til å utføre oppgavene de er tildelt. Når en bruker DevOps, så kan en ifølge en case-studie se at teamene som jobber på denne måten er gladere og mer engasjert i arbeidet (Senapathi, Buchan, & Osman, 2018). Blant noen som bruker DevOps i Sør-Afrika, så mener de at det oppstår en kultur der det er mer delt ansvar blant de ansatte når en bruker DevOps (Rowse & Cohen, 2021). I en case-studie fant forfatterne ut at med DevOps så får de ansatte bedre oversikt over hele konteksten slik at de får se det faktiske ringvirkningene på det de gjør, de føler seg mer verdsatte enn tidligere, miljøet blir mer positivt siden det blir mindre fingerpeking mellom ansatte og det økte samarbeidet blir godt likt av de som jobber der (Senapathi, Buchan, & Osman, 2018). Noe av det som kan føre til økt samarbeid er at de ansatte blir mer engasjert i hele prosessen på grunn av kontinuerlige utgivelser som gjør at de ansatte føler seg ansvarlige for hva som blir sendt videre (Siqueira, Camarinha, Wen, Meirelles, & Kon, 2018). Noe av det som går inn under det nye delte ansvaret, er for eksempel at utviklere får ansvar for kvalitetssikring og distribusjon, som man kan se i Figur 5 (Rowse & Cohen, 2021). Figuren viser noen av resultatene fra en spørreundersøkelse i Sør-Afrika blant noen som jobber med DevOps, der de ble spurt om hva DevOps-kulturen gjør med bedriften. Det eksisterer forskjellige meninger om hvorvidt implementasjon av DevOps fører til mindre stress, der det blir hevdet av noen at det er en merkbar forskjell og noen mener at det ikke er det (Rowse & Cohen, 2021). Utviklere liker generelt å utforske nye teknologier og måter å arbeide på, og siden DevOps trenger både nye verktøy og metoder for å fungere, betyr det at de har en forutsetning for å like denne måten å jobbe på (Jones, Noppen, & Lettice, 2016).

Organisasjonskulturen kan sies å være viktig for at DevOps skal fungere. Det er funnet antydninger til at en kultur som passer med DevOps-prinsippene, metodene og verktøyene er mer sannsynlig å se de forventede fordelene fra DevOps, samtidig som en unngår flere av utfordringene (Rowse & Cohen, 2021). Det blir rett og slett lettere å implementere resten av det som går inn under DevOps, om man klarer å få på plass en organisasjonskultur som passer med DevOps (Muñoz, 2021).



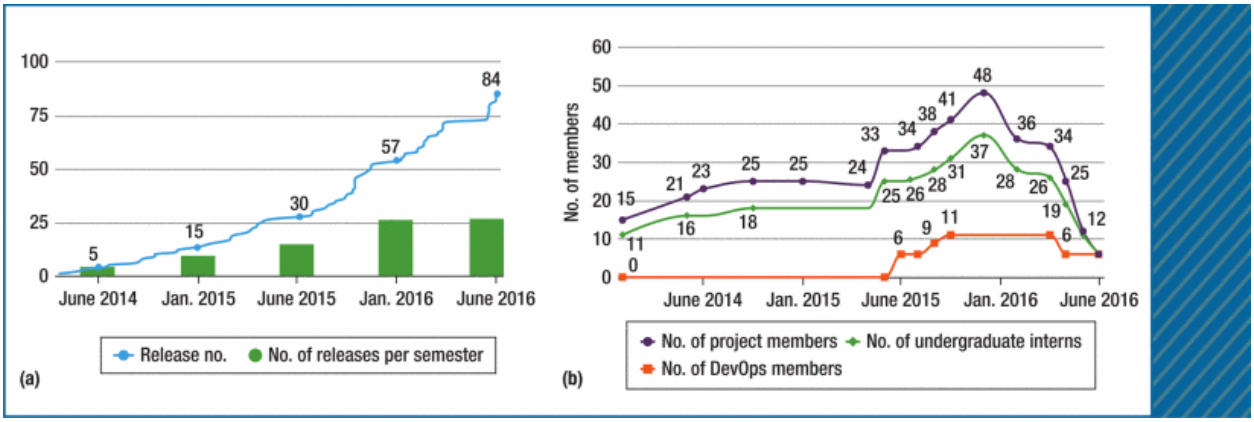
Figur 5: Konsekvenser av DevOps-kultur (Rowse & Cohen, 2021).



Figur 6: Fordeler med DevOps (Rowse & Cohen, 2021).

En annen fordel med DevOps som blir beskrevet er at det er oftere release med DevOps, noe som gjør at det blir lettere å få ut flere funksjoner og risikoen med slike små utgivelser blir beskrevet som mindre, noe man kan se et eksempel i på Figur 7 (Senapathi, Buchan, & Osman, 2018). Figuren viser utviklingen av antall utgivelser i prosjektet til den offentlige brasilianske programvareportalen (a), og utviklingsteamets distribusjon av medlemmer (b). De siste 12 månedene brukte prosjektet DevOps. Med kontinuerlige utgivelser blir teamet bedre rustet til å reagere på endringer i krav og prioriteringer fra ledelsen og/eller kunden, som også gjør at prosjektet de jobber på har en større sjanse for å bli utført (Siqueira, Camarinha, Wen, Meirelles,

& Kon, 2018). Det er viktig å nevne at ikke alle som bruker DevOps nødvendigvis er enige i at DevOps fører til at produktet får flere funksjoner enn uten DevOps (Rowse & Cohen, 2021). Kontinuerlige utgivelser blir sett på som en god måte å opprette tillit når det eksisterer mistillit mellom DevOps-teamet og ledelsen/kunden, fordi det gir mer oversikt over hva som faktisk blir jobbet med, og skaper en dialog mellom partene om hva som kreves (Siqueira, Camarinha, Wen, Meirelles, & Kon, 2018). Tilbakemeldingene til produktet som blir gitt etter DevOps-implementasjon blir sett på som bedre enn med tidligere metoder (Rowse & Cohen, 2021). En kan også se i flere instanser at både behandlingstid og uforutsett arbeid minker etter innføring av DevOps-arbeidsmetoder (Muñoz, 2021), (Rowse & Cohen, 2021). Noe som også mange som bruker DevOps er enige om er at i DevOps finner man feil i programvaren både tidligere og oftere, i tillegg til at produktene blir mer stabile, noe man kan se av Figur 6 (Rowse & Cohen, 2021). Figuren viser noen resultat av en spørreundersøkelse i Sør-Afrika blant noen som jobber med DevOps om hvilke fordeler en får når en bruker DevOps.



Figur 7: Utviklingen av antall utgivelser og distribusjon av medlemmer i et DevOps-prosjekts. (Siqueira, Camarinha, Wen, Meirelles, & Kon, 2018).

Siden både utviklere og driftere jobber på en annen måte med DevOps sammenlignet med tidligere, så kan det oppstå noen problemer når det er snakk om kompetanse. Ett av disse problemene er at det er få personer som har kompetansen som er nødvendig, det vil si kunnskap og kompetanse innen felt som infrastruktur, overvåkning, koding for å nevne noen (Senapathi, Buchan, & Osman, 2018). Tidligere trengte ansatte bare kompetanse som enten utvikler eller drifter, men nå som hele teamet skal ha kunnskap om hele prosessen et produkt går gjennom

fører dette til at det blir vanskelig å finne ansatte som faktisk har den nødvendige kunnskapen og kompetansen (Riungu-Kalliosaari L., 2016). Å skaffe ansatte med DevOps-kompetanse er i en artikkel beskrevet som den største utfordringen forbundet med DevOps ifølge de som ble spurt, men å implementere selve teknologien som trengs samt å bruke automatiseringsverktøy blir sett på som de minste utfordringene (Rowse & Cohen, 2021). Noe som også gjør det vanskelig å bruke DevOps-praksiser er når arkitektene lager monolittiske applikasjoner som er meget vanskelig å inkludere i kontinuerlige praksiser, og en må også ha alle avhengighetene til applikasjonen med i beregningen (Shahin, Babar, & Zhu, 2016). Kunnskapen som er nødvendig å vite er også i stadig endring, med nye verktøy og metoder som blir innført, så er det til tider vanskelig å få skikkelig innsikt i alt som er nødvendig for å gjennomføre DevOps-prosessene (Senapathi, Buchan, & Osman, 2018). Det er vanskelig å lage applikasjoner som skal kunne fungere i forskjellige miljøer og tilpasses flere forskjellige verktøy, og dette blir en hindring når man skal prøve å implementere kontinuerlige utgivelser (Shahin, Babar, & Zhu, 2016).

Det som alle endringer i bedrifter og andre organisasjoner kan støte på er motstand fra de ansatte. Dette er noe som også skjer når DevOps blir implementert. Noen ansatte vil bare jobbe med det de har fått som jobb, for eksempel en utvikler skal skrive kode og en drifter skal ikke skrive kode, og motstand oppstår om en bare setter ulike ansatte med forskjellig kompetanse sammen i en gruppe og forventer at de skal jobbe bra sammen uten videre (Senapathi, Buchan, & Osman, 2018), (Jones, Noppen, & Lettice, 2016). Med så mange endringer opplever ansatte også at det blir usikkerhet rundt hvem som har ansvar for ulike oppgaver, og resultatet er at arbeid blir ikke gjort fordi alle tror at noen andre har gjort det (Senapathi, Buchan, & Osman, 2018). Feiljusteringer mellom avdelinger og infleksibilitet i kommunikasjonsprosessene er også et problem som en ser oppstår når DevOps blir implementert, i tillegg til å være en hindring for at DevOps blir innført skikkelig i bedriften (Díaz J., 2019). En kan også finne eksempler på at det er de ansatte som prøver å få DevOps implementert, der ledelsen ikke etterkommer forespørsler om nødvendige ressurser som trengs for å kunne gjennomføre DevOps-aktiviteter på en tilfredsstillende måte og heller ikke fullt forstår nytten eller vitsen med DevOps (Jones, Noppen, & Lettice, 2016). Det kan være lettere å bevise nytten av DevOps og kontinuerlige utgivelser gjennom resultater i stedet for diskusjoner, der en har en oversiktlig utviklingsprosess som viser at en faktisk kan utføre det en lover til ledelsen (Siqueira, Camarinha, Wen, Meirelles, & Kon,

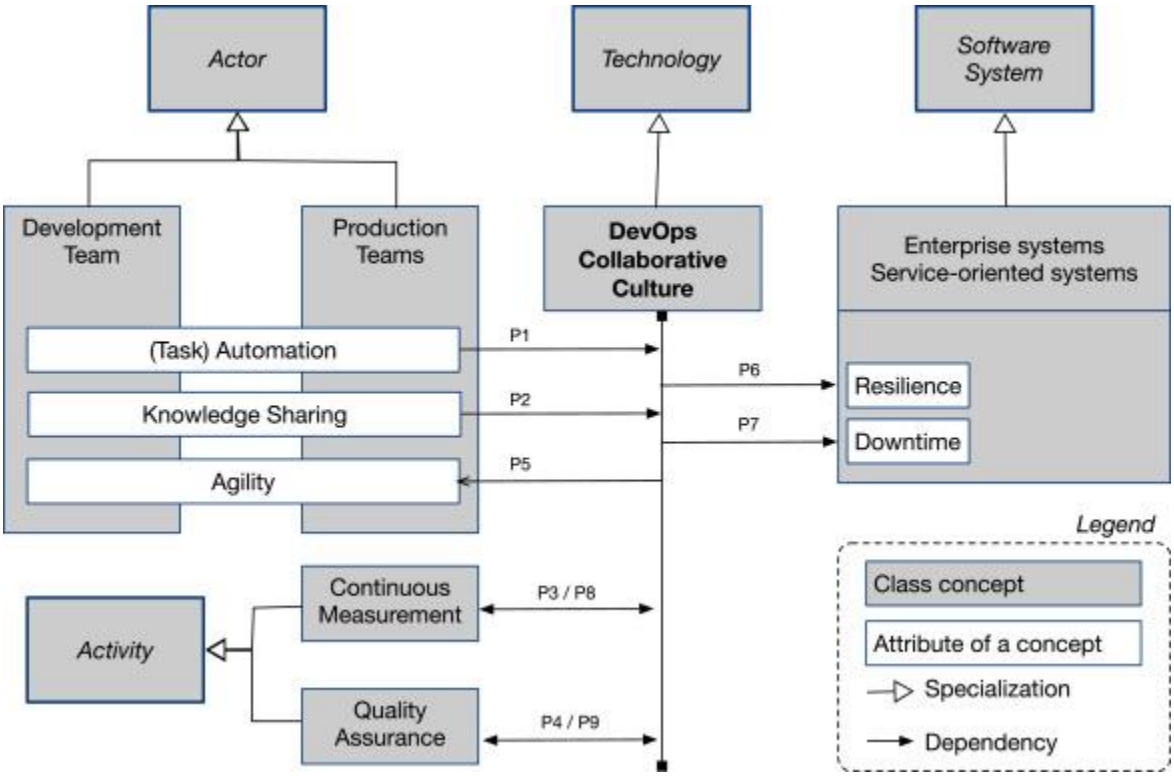
2018). Når team blir avhengige av hverandre kan en støte på problemer når alle prøver å release samtidig, for ett teams versjon av koden kan føre til at ett annet teams nye kode ikke lenger fungerer som den skal (Shahin, Babar, & Zhu, 2016).

En sentral del av DevOps er at utviklere og driftere er satt sammen i samlokaliserte teams, men dette er ikke alltid mulig i praksis for internasjonale selskaper. Dette vil påvirke kommunikasjonen mellom de som ikke deler samme lokaler, der blant annet mengden av kommunikasjon mellom disse ansatte vil minke (Diel, Marczak, & Cruzes, 2016). En studie sier at man får mye bedre kommunikasjon mellom teammedlemmene når man sitter sammen, der en kommuniserer mye mer naturlig og misforståelser blir lettere oppklart (Senapathi, Buchan, & Osman, 2018). Stor geografisk avstand mellom teammedlemmer gjør at det oppstår uvisshet om hva de andre gjør, og kulturelle forskjeller kan også oppstå der samarbeidet blir mer utfordrende (Diel, Marczak, & Cruzes, 2016). Om en har store geografiske avstander mellom teammedlemmer, så må en også håndtere at teammedlemmer arbeider i forskjellige tidssoner. Avhengig av hvor stor tidsforskjellen er, så gjør dette at teammedlemmene kanskje ikke har muligheten til å arbeide samtidig og dette gjør synkron kommunikasjon nesten umulig (Diel, Marczak, & Cruzes, 2016). En kan se at alt dette også fører til misforståelser mellom teammedlemmene når en bruker indirekte kommunikasjon (Diel, Marczak, & Cruzes, 2016). På Figur 8 kan en se sammenhengen mellom forskjellige problemer og hvilke kommunikasjonsutfordringer som får dem til å oppstå i en internasjonal bedrift som bruker DevOps.

	A Geographical distance	B Socio-Cultural distance	C Temporal distance	D Frequency	E Direction	F Modality	G Content
Teams are not available at the same time	X		X		X		
It is not possible to talk to the dev on call immediately, QA sessions always take at least 24hrs	X		X	X		X	X
No previous notice of releases		X		X	X	X	X
Few/None training on the changes in the application	X	X	X	X		X	X
A team does not know the routine of another	X	X	X			X	X
Lack of prioritization of bugs open by ops	X	X		X	X	X	X
There is no formal communication channel between the teams for feedback	X	X	X	X	X	X	X

Figur 8: Sammenheng mellom problemer og kommunikasjonsutfordringer (Diel, Marczak, & Cruzes, 2016).

I Figur 9 kan man se en modell for implementering av DevOps som er laget etter en undersøkelse av 15 instanser av suksessfulle implementeringer av DevOps, og viser forholdet mellom ulike faktorer (Luz, Pinto, & Bonifácio, 2019). I artikkelen figuren er hentet fra ble en samarbeidskultur trukket fram som det sentrale for implementering av DevOps (Luz, Pinto, & Bonifácio, 2019). Det kan sies å være fem kategorier som muliggjør DevOps: automasjon, kontinuerlig målinger, gjennomsiktighet, kvalitetssikring og deling (Luz, Pinto, & Bonifácio, 2019). Smidighet og motstandsdyktighet er trekt fram som utfall av å implementere DevOps (Luz, Pinto, & Bonifácio, 2019).



Figur 9: Modell for implementering av DevOps (Luz, Pinto, & Bonifácio, 2019)

I Figur 10 kan man se en oppsummering av de ulike kategoriene i en anbefalt DevOps-implementering (Luz, Pinto, & Bonifácio, 2019).

Figur 10: Forhold mellom kategorier (Luz, Pinto, & Bonifácio, 2019)

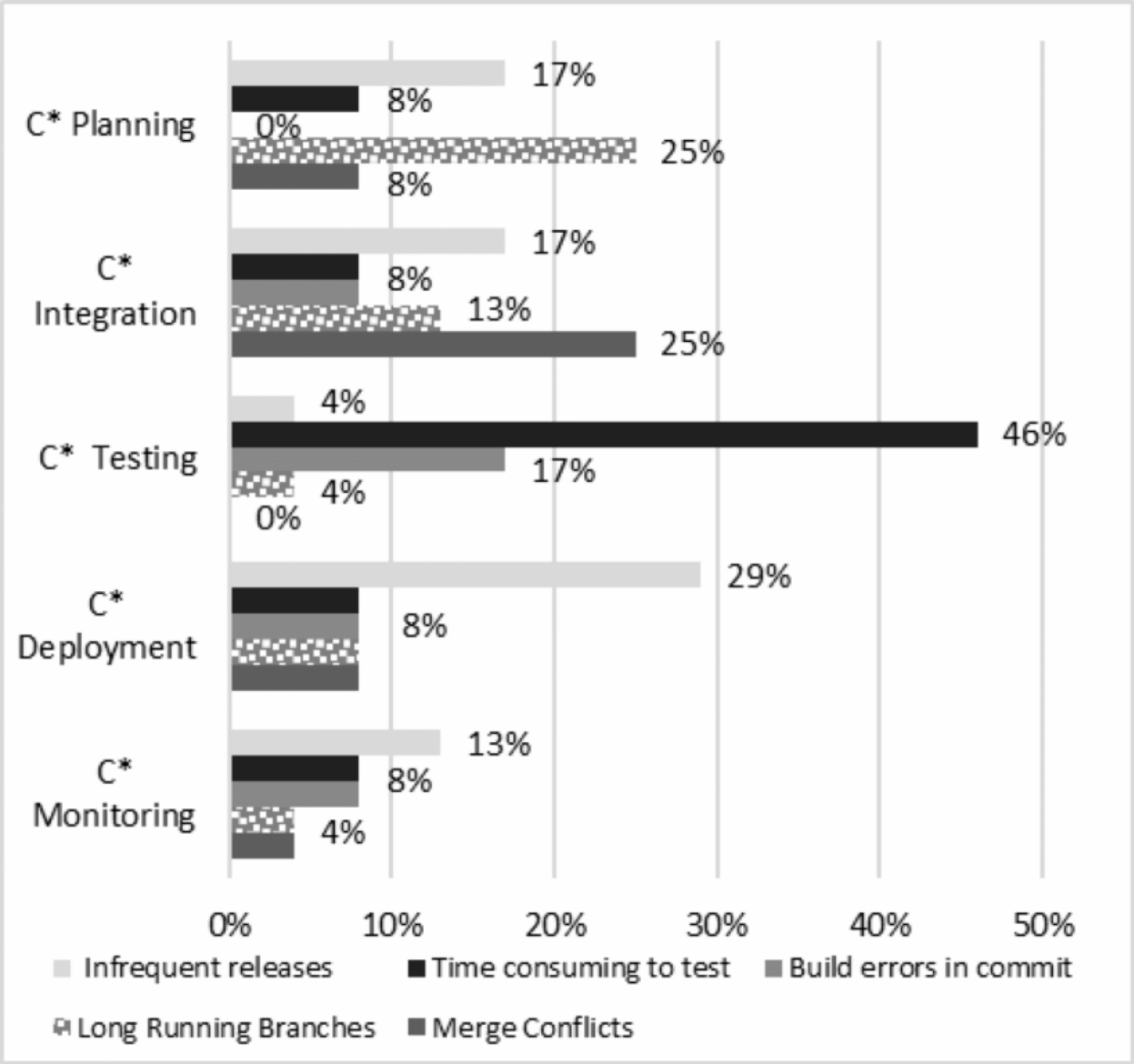
3.2 Case-studier fra Norden

Det er ikke mange av case-studiene som er utførte i Norden som spesifikt undersøker hva DevOps blir definert som av de ansatte. I en av artiklene som inkluderer flere bedrifter, betyr DevOps for deltakerne eierskap, og at utviklerne har ansvar for alle prosessene i utviklingssyklusen fra design til drift (Lwakatare, et al., 2019). Hovedmålet med DevOps er ifølge de som ble spurt i artikkelen, at man skal få ut endringer i programvaren fortest mulig slik at man kan få tilbakemeldinger så raskt som mulig (Lwakatare, et al., 2019). Driftsavdelingen og utviklerne kommuniserer når det trengs som oftest på en uformell måte, og driftsavdelingen hjelper utviklerne med automasjon, kunnskapsdeling og lignende som utviklerne ikke tradisjonelt gjør (Lwakatare, et al., 2019).

Fra andre case-studier i Norden, kan vi se at de som jobber med DevOps opplever flere fordeler med arbeidsmetoden. En av disse er at de opplever mindre stress etter implementasjonen av DevOps, i forhold til tidligere arbeidsforhold (Lwakatare, et al., 2019), (Riungu-Kalliosaari L., 2016). Det blir også hevdet at helsen til de ansatte generelt ble bedre på grunn av DevOps, noe som skjedde fordi det var mindre angst rundt større release som var vanlig før DevOps ble implementert (Riungu-Kalliosaari L., 2016). Det er ikke mange case-studier å hente data fra, men det er flere bedrifter som er involverte i case-studiene. Det er derfor en synlig tendens der arbeidsmetoden ikke bare påvirker selve arbeidet, men også de ansatte på en positiv måte.

Det er også noen utfordringer som går igjen over flere av case-studiene. Et problem bedrifter som implementerer DevOps støter på er eldre systemer/programmer, som ofte kalles «legacy»-systemer eller «legacy»-programvare. Problemene oppstår fordi slike produkter ofte er lokalisert

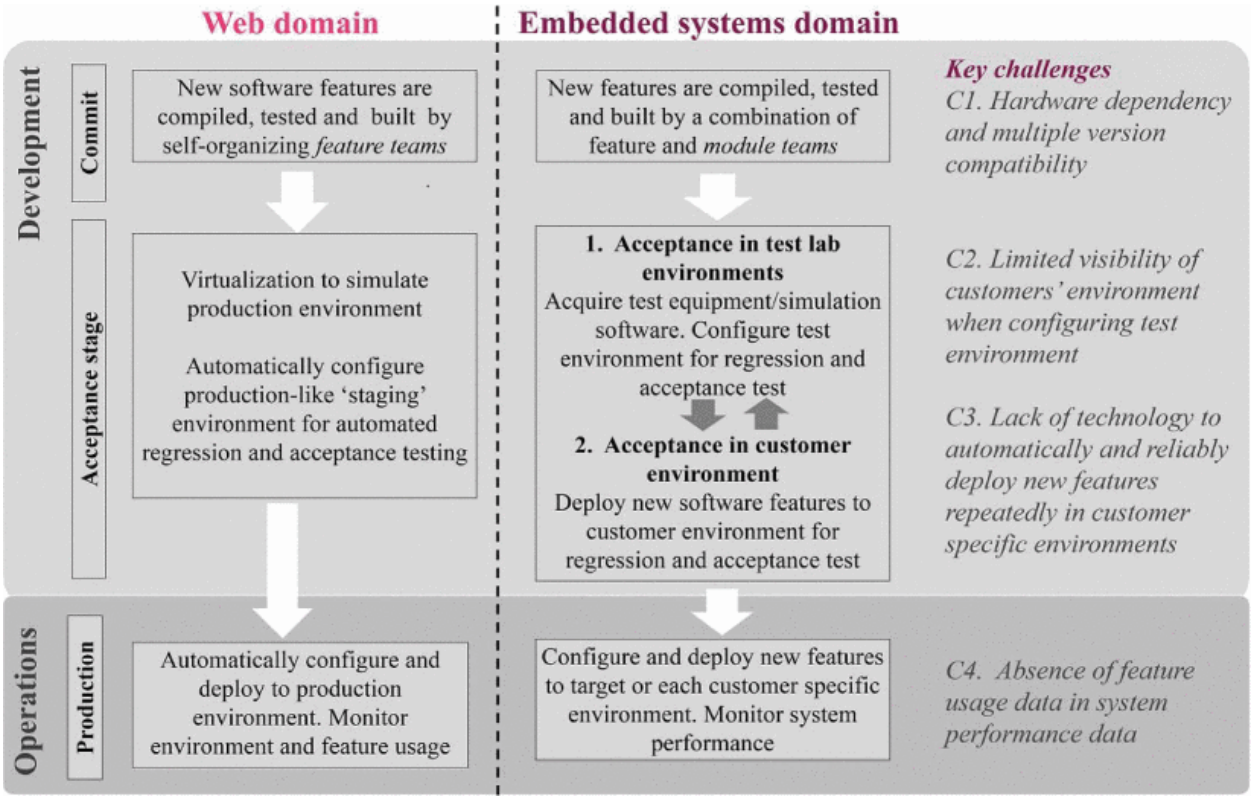
hos kundene og det passer dårlig å oppdatere dem kontinuerlig, mens nye systemer kan ofte lettere legges inn på skyløsninger med alle fordeler det bringer med seg (Lwakatare, et al., 2016), (Riungu-Kalliosaari L., 2016), (Kuusinen, et al., 2018). Forskjellige kunder kan ha forskjellige versjoner av den samme programvaren, noe som gjør det ekstra vanskelig å levere oppdateringer raskt nettopp fordi det er mange forskjellige versjoner å holde kompatible (Lwakatare, et al., 2016). Når en skal automatisere arbeidsprosessene i tråd med DevOps-metoden, så kan en også støte på vanskeligheter med legacy-kode. Det som er problemet er at slik kode gjerne ikke er designet for å brukes på denne måten, og en kan bli nødt til å gjøre ekstra manuelt arbeid på grunn av feil som oppstår (Lwakatare, et al., 2019). Resultatet av dette er i ett tilfelle at de ansatte føler at den eneste måten å få innført DevOps på er å fullstendig erstatte slike systemer med nye, siden systemene blir sett på som en hindring (Kuusinen, et al., 2018). Som en kan se fra Figur 11, så er det ikke stor tro fra de som deltok på en spørreundersøkelse at man kan fikse problemer ved å bare innføre kontinuerlige praksiser når man jobber med innebygde systemer og legacy-kode. Det er ingen av alternativene som har fått over 50%. Figuren viser hva noen som jobber med DevOps svarte når de ble spurt om kontinuerlige metoder kunne fikse noen av problemene deres (C* står for continuous(kontinuerlig)).



Figur 11: Kan kontinuerlige metoder løse problemer? (Kuusinen, et al., 2018).

Flere av case-studiene fra Norden nevner DevOps-kultur og den eksisterende organisasjonskulturen. Et eksempel var at med DevOps-kulturen ble det å finne feil i koden sett på som positivt, noe som ikke var tilfelle før de begynte med DevOps (Cruzes D.S., 2019). Det ble også fortalt at kulturen bestod av blant annet tillit, inklusivitet, samarbeid og åpenhet for læring (Lwakatare, et al., 2019). I samme kilde kunne en se positive endringer i hele bedriften på grunn av denne kulturen (Lwakatare, et al., 2019). Problemer kunne oppstå i implementeringen av DevOps når den originale organisasjonskulturen kom i konflikt med DevOps-kulturen

(Riungu-Kalliosaari L., 2016). En kan ikke bare sette utviklere og driftere sammen i et team og forvente at alle skal jobbe på en ny måte uten problemer (Lwakatare, et al., 2016). Kundene er heller ikke alltid klare for et tett samarbeid som er vanlig i DevOps (Cruzes D.S., 2019). Faktisk så er det foreslått at endring i organisasjonskulturen kunne være en av de største utfordringene med å implementere DevOps (Riungu-Kalliosaari L., 2016). For noen utviklere og driftere kan det være meget krevende å plutselig måtte fundamentalt endre måten de arbeider på og ansvaret de har, spesielt om de har jobbet på en spesiell måte over lengre tid (Riungu-Kalliosaari L., 2016). Noen aktiviteter kan være mindre interessante for utviklere, som for eksempel testing, og når man skal prøve å inkludere og gi utviklerne innsikt i disse aktivitetene så kan man støte på problemer (Cruzes D.S., 2019). Noe som ser ut til å forbedre organisasjonskulturen i denne forbindelsen var at de som jobbet sammen jobbet på samme stedet, altså at teamene var samlokalisert (Lwakatare, et al., 2019). Det er foreslått at i store bedrifter der en sliter med å innføre DevOps, så er det viktig at alle i organisasjonen tar til seg hva en DevOps-kultur innebærer, og følger den personlig (Kuusinen, et al., 2018).



Figur 12: Egenskaper og utfordringer knyttet til DevOps (Lwakatare, et al., 2016).

Problemer kan også komme fra andre sentrale deler av DevOps. Automasjon blir sett på som en sentral del av DevOps gjennom hele livet til et produkt, fra start til slutt (Cruzes D.S., 2019). Når en automatiserer noe som testing, så støter en på flere utfordringer: manuelle tester gir testerne mer innsikt i programvaren og de som bruker den, testene kan bli utdaterte, testene kan inneholde feil siden de er kodet og det kan være komplisert å lage automatiserte tester som dekker alle behovene (Riungu-Kalliosaari L., 2016), (Lwakatare, et al., 2016), (Cruzes D.S., 2019). Som en kan se i Figur 12, så er det en del utfordringer som oppstår når en prøver å gå gjennom DevOps-prosessen med innebygde systemer i motsetning til web-domener. Figuren viser egenskapene til DevOps i web-domene og i domene for innebygde systemer. Det er ikke bare problemer som oppstår når bedrifter prøver å automatisere forskjellige arbeidsprosesser. Når en først har fått til automatisering av forskjellige prosesser, så føler de som utfører prosessene at de sparer mye tid (Lwakatare, et al., 2019). Man kan også støte på problemer med tid, der man får problemer med å effektivisere eksisterende prosesser fordi man har prosjekter med tidsfrister man må innfri, og drifterne har ikke nok tid til å hjelpe utviklerne (Lwakatare, et al., 2019).

Når bedrifter har implementert DevOps, så har de som jobber der merket seg at kvaliteten på programvaren har økt, blant annet fordi de nå får raskere tilbakemeldinger fra kundene og kan lettere rulle tilbake feil siden release/utgivelsene er mindre enn tidligere (Lwakatare, et al., 2019), (Riungu-Kalliosaari L., 2016).

4 Resultater

I dette kapittelet vil vi presentere resultatene vi fant i case-studien vi gjennomførte i forbindelse med dette forskningsprosjektet. Kapittelet er delt inn i underkapittel som skal svare på forskningsspørsmålene vi presenterte i kapittel 1.1 om problemstillingen:

F1: Hvordan defineres begrepet DevOps av forskjellige ansatte i en programvarebedrift?
F2: Hva er konsekvensene av å innføre DevOps i en programvarebedrift?
F2.1: Hva syntes de ansatte om DevOps i en bedrift som bruker det?
F2.2: Hva er nødvendig for å implementere DevOps på en god måte i en programvarebedrift?

4.1 Definisjon av DevOps (F1)

En sentral del av spørsmålene vi stilte under intervjuene vi utførte handlet om definisjonen av DevOps, med fokus på hva intervjuobjektet definerte DevOps som. Dette var en del av intervjuguiden for at studien skulle kunne svare på forskningsspørsmål F1, og i dette kapittelet kommer resultatene som synliggjør hva de som deltok i studien definerer DevOps som. Svarene som ble gitt av intervjuobjektene var forskjellige, men det var mulig å finne noen mønster. Noe av det som flere av deltakerne i case-studien vår uttrykte når de ble spurt om hva de definerte DevOps som, så syntes de at begrepet DevOps var vagt:

> *«Det er et navn som dukker opp av og til, men nøyaktig hva det betyr er litt fjernt».*
>
> Informant 2 (konsulent)

Dette gjaldt spesielt for konsulentene vi spurte som samarbeider med utviklerne, men som selv ikke er utviklere. Informant 4 måtte ha en beskrivelse fra oss om hva vi faktisk ville snakke om siden konsulenten var usikker på hva som faktisk var temaet. De sa:

> *«Uttrykket er litt «cloudy» for meg. For meg er DevOps den avdelingen der de på teknisk jobber. De jobber fra et annet kontor lenger sør. Dere må nok si*

hvilken retning dere vil gå med intervjuet slik at jeg vet hvilken vinkel dere vil ha på den. Det er mange uttrykk som går igjen, og det er litt uklart hva som ligger bak hvert uttrykk»

Informant 4 (konsulent)

Etter en kort forklaring fra oss om hva DevOps er for noe, kunne Informant 4 gå nærmere inn på temaet. Her er forklaringen:

«Det er litt vanskelig å definere, det er litt hva du selv tenker hva DevOps er. Det er på en måte at drift og utvikling skal jobbe sammen … Det starter gjerne med smidig/agilt som scrum, også er det en flytende overgang til DevOps»

Jean-Pierre V. Strand (student)

Det var ikke bare konsulentene som hadde vanskeligheter med å kunne gi en nøyaktig beskrivelse av DevOps:

«DevOps for meg er et litt vagt konsept. Jeg føler kanskje at det ikke er mindre vagt enn før, men jeg liker å være involvert i prosessen: få tilbakemeldinger, høre om feil, påvirke måten vi deployer på og bygge ting på ...».

Informant 6 (arkitekt)

«Det er et ganske vagt spørsmål, det vi har kommet frem til er at det ikke finnes noe spesifikk definisjon for DevOps. Så det er litt personlig, hva du selv tenker at det er.»

Informant 1 (prosjektleder)

Til og med noen som har jobbet i bedriften i flere år slik som prosjektlederen har gjort, kan føle at begrepet DevOps er vanskelig å definere. Selv om noen av deltakerne i case-studien følte at begrepet DevOps var et vagt konsept, så kunne de fremdeles beskrive noe som de følte var DevOps. Disse definisjonene varierte fra tett samarbeid med hverandre til automatisering av prosessene rundt programvaren på grunn av skyløsninger og hva de ansatte burde være i stand til å gjøre:

«Å jobbe med noen som kontinuerlig utvikler noe er vel det som det betyr for meg. Å kontinuerlig komme med tilbakemeldinger og åpen dialog til enhver tid. Å snakke om hva som funker, hva som ikke funker og hva som mangler.»

Informant 2 (konsulent)

> *«For meg betyr det at jeg vet at systemet blir oppdatert, om det fungerer på en kunde så fungerer det på den andre også. Hvis det ikke fungerer, så har man glemt en fil eller noe sånt. Det har jo skjedd før. Det fungerer her, men ikke der. Hvorfor? Da må det ha skjedd en feil. Før var det inhouse, produktet var installert hos kundene og noen hos oss. Da visste vi aldri hvilken tilstand systemet var i.»*

Informant 4 (konsulent)

> *«I mine tanker så er det jo det smidige og tverrfaglige, kanskje litt feil ord men … man har et team hvor alle parter i teamet klarer å få utviklingsløpet til å gå ifra selve utviklingen til den er hos kunde. En selvgående gruppe hvor egentlig de fleste kan gjøre en del ting. Utover det, i mitt hode så burde det tekniske miljøet ha muligheten til å gå inn å gjøre en enkel fiks i kode dersom de får beskjed om at her har vi en fillefeil og jeg er på fjellet og får ikke gjort det, kan ikke du gå inn? Så burde det være mulig at de andre i teamet kan gå inn å gjøre det som kanskje tekniske ikke gjør til vanlig. Eller kanskje jeg burde hatt muligheten til å pushe noe ut i et miljø. Vi skal klare å gjøre alt selv uten noe støtte utenfra. Så er det smidig, i mitt hode så er det å være smidig viktig.»*

Informant 1 (prosjektleder)

Noen andre som også hadde et mer klart syn på hva DevOps kunne defineres som, var blant annet en av utviklerne. I motsetning til noen av de andre informantene, så hadde de et litt mer teknisk syn på dette. På spørsmålet om hva definisjonen av DevOps var, så mente de at DevOps handlet om infrastruktur og automatisering, spesielt knyttet til release:

> *«Tett kobling mellom release-infrastruktur og koding egentlig. Bygge features og mer fra spec til monitorering når det er deployet.»*

Informant 3 (utvikler)

> *«Det er litt som du sa tidligere, føler liksom at Scrum er veldig greit fordi Scrum har et sett med klare regler og roller. Jeg føler ikke at DevOps er der, jeg føler at DevOps er løsere og mer en tankegang. Når jeg tenker på DevOps*

så er det jo enda mer automatisering av oppgaver og kanskje enda litt tettere på det her deploy som kan nesten bare være en knapp som gjør det enda lettere for meg som prosjektleder å få dette ut i produksjon og få dette automatisert i bakgrunnen, og det er selvsagt bygget opp av PowerShell og diverse ting av gutta på teknisk»

Informant 1 (prosjektleder)

Andre som var mer sikre på definisjonen hadde en oppfatning av DevOps der det var mer fokus på samarbeid, involvering og delt ansvar fra starten av utviklingen til utgivelse:

«For meg, så handler DevOps mest om den ideen om, at du har en ansvarslinje hele veien gjennom. Ja, det er det som jeg føler er kanskje den den viktigste tanken rundt DevOps. Det er at alle i leveranselinjen er ansvarlig for at sluttproduktet kommer i hendene til brukeren og faktisk fungerer. Det er dette som jeg tenker på som DevOps. Det finnes ingen avlevering, det finnes ingen gates, det finnes ingen plasser der: «nå er det i hendene på driften, nå er det i hendene på support, nå er det i hendene på noen andre enn meg». Alle har ansvar. Det endelige målet er ikke å levere til drift. Det endelige målet er å sørge for at brukerne har en applikasjon som funker. Det er en sånn måte å tenke på som hører til DevOps»

Informant 5 (arkitekt)

«Det betyr egentlig at alle er involvert helt fra ledelsen og helt ned til dem som sitter og installerer. Det er hvert fall den definisjonen jeg har landet på. Så det er på en måte, istedenfor å være den mer tekniske greia der de som deployer er en del av utviklingsteamet så tenker jeg at det er en hel filosofi der du får alle involvert.»

Informant 7 (arkitekt)

«For min del, så betyr det at utvikling sørger for å være med på hele prosessen fra start til mål. Vi planlegger i stor grad det vi skal implementere, kanskje ikke hva som skal implementeres, men hvordan det skal implementeres,

prosessen med workitems og alt sånt. Det går hele runden med utvikling og helt inn til at vi har bygd.»

Informant 6 (arkitekt)

Så en kan se at det er en del forskjellige meninger her, der noen mener at det betyr noe teknisk der det viktigste er infrastrukturen i utviklingen og hvordan produktet og prosessen er automatisert. Andre mener at essensen av DevOps er samarbeidet og ansvarsdelingen gjennom hele utviklingssyklusen som trengs mellom ansatte. En av arkitektene var klar over denne oppdelingen av definisjoner:

«Jeg tror kanskje det er 2 ulike linjer. Det ene er at alt skal være automatisert og ingen involvert og kanskje at du når continuous deploy der man kan deploye mange ganger per dag. Men det andre er det tankesettet at alle er med og tar et ansvar for totaliteten.»

Informant 7 (arkitekt)

Ifølge en av konsulentene, så har egentlig ikke slike definisjoner så mye å si for de på kontoret som ikke jobber direkte med å utvikle produktet. Det viktigste for dem blir da framstilt som konsekvensene av DevOps, og ikke så mye hva det faktisk er for noe:

«… for BPO-ansatte er ikke de begrepene så viktige. Vi jobber som vi jobber, og prosessen som skjer på den andre siden ser vi ikke så mye til. Vi vet at det oppdateres hver andre uke»

Informant 4 (konsulent)

En annen ting som kom fram, var at bedriften har en egen avdeling som blir kallet for DevOps-avdelingen eller DevOps-teamet slik som informant 4 beskrev tidligere. Dette virker som en beskrivelse av en driftsavdeling, som vanligvis har ansvaret for programvaren etter at den er ferdig utviklet.

Dette er i tillegg til de som jobber på utvikling, og samarbeidet med dem ble beskrevet av andre informanter slik:

«Vi har et egent DevOps-team som releaser, men vi er også del av supportstrukturen, så vi er inne og hjelper til når det er nødvendig. Vi har også

våre egne konsulenter som kommuniserer med kunder i stor grad, men det kan hende at det sendes forespørsler til oss også. Vi er en del av den prosessen.»

Informant 6 (arkitekt)

«Kanskje de [teknisk avdeling] ikke gjør så veldig mye av det vi gjør, men vi hjelper dem i mye større grad enn før. Blant annet bygge inn automatisering i produktet eller pakketere ting sånn at det er enklere for dem å installere, så det er veldig god kommunikasjon med de teknikerne.»

Informant 7 (arkitekt)

«… jeg jobber jo på en måte i den Dev biten av DevOps og så har du et par nøkkelpersoner som er litt mer involvert i den Ops biten og så har du de som jobber i Ops.»

Informant 8 (arkitekt)

En annen ting vi spurte intervjuobjektene om var hvorvidt måten de jobbet på nå fulgte definisjonen av DevOps som de hadde gitt. Når vi fikk informasjon om casen og bedriften, så ble vi informert om at de brukte DevOps i arbeidshverdagen. Om det var forskjell mellom hva de mente var DevOps og det som var de faktiske forholdene var et interessant tema for oss. Definisjonene til intervjuobjektene og forholdene i bedriften var ofte forskjellige, men de følte at de fulgte definisjonen i hvert fall delvis:

«Særlig i forhold til våre egne utviklingsmiljøer er vi det. Når de ikke er aktive, så føler jeg at vi i mindre grad er der. Totalt sett, om vi hadde vært mer involvert i det som selve DevOps-teamet gjør så hadde vi vært 100%.»

Informant 6 (arkitekt)

«Der er alltid forbedringer. Sånn som når du nevner målinger f.eks så kunne man kanskje hatt eller funnet noe fornuftig å måle på og vi kunne alltids hatt en større automatiseringsgrad for eksempel, så det finnes forbedringspotensial, men det er jo litt det som kanskje er filosofien, man blir aldri ferdig. Det at man begynner å forbedre er en del av prosessen i seg selv også»

Informant 7 (arkitekt)

«Vi kommer veldig fort tilbake til dette her at ting bare skal flyte, være smidig og være enkelt, at flere kan gjøre de forskjellige tingene som kanskje ikke er samme rolle. Men jeg mangler fortsatt de klare reglene hvor jeg kan gå ned i sjekkbokslisten og si at vi har oppfylt 12 av 15.»

Informant 1 (prosjektleder)

Intervjuobjektene hadde også noen tanker om hvor nødvendig det er å implementere alt som inngår DevOps, og om de ønsker dette i bedriften. I kapittelet om forskningsspørsmål F2, så vil vi gå nærmere inn på hva som kom fram om dette.

En av informantene mente også at DevOps kommer til å bli definert som standard praksis i fremtiden etter at bedrifter har prøvd konseptet i praksis. Versjonen av DevOps som sirkulerer nå er ifølge dem en teoretisk ide som kommer til å endres for å tilpasses den virkelige verdenen:

«Litt som med Scrum, så er det en sånn ting som går seg litt til. De første variantene er en sånn ekstremvariant, et teoretisk ideelt scenario, hvor man har kontroll på alle variabler og har en tekstbokidé om hvordan man skal kunne gjøre det, som har blitt til hvilke ting kan man og hvilke ting burde man droppe eller tilpasse fordi at den ideen faktisk treffer den virkelige verden. Det er nesten sånn at DevOps på et eller annet tidspunkt ikke kommer til å være DevOps. De tingene en har lyst til å beholde kommer til å være så åpenbare at det ikke kommer til å være et slikt navn på dem lengre. Det kommer bare til å bli måten man gjør det på. Man blir fort litt låst på slike ting som prøver å beskrive flyter og slikt, som Scrum, Lean og Kanban. Alle disse tingene kan flyte litt over i hverandre til det punktet at man bare sier at man driver agilt, og alle kan si at de driver agilt på en eller annen måte. DevOps kommer nok til å bli en sånn ting også. Ingen kommer til å gjøre det slik som de første tankene var og slik som det er beskrevet i skolebøker, men de aller fleste kommer til å bruke store deler for det er bare slik man gjør det og det er den eneste måten man gjør det på. Det vil være veldig dumt å ikke gjøre det på den måten. Hvis man stopper og tenker seg litt om, så kommer man fram til den samme konklusjonen uansett.»

Informant 5 (arkitekt)

4.2 Konsekvensene av å innføre DevOps(F2)

I dette kapittelet kommer resultatene som svarer på forskningsspørsmål F2, samt underforskningsspørsmålene F2.1 og F2.2. Dette vil være konsekvensene av implementeringen av DevOps, med ekstra fokus på meningene til de ansatte og hva som gjør en slik implementering suksessfull i underkapitlene. Før disse underkapitlene, så kommer noen av resultatene om bedriftskulturen, fordeler og ulemper med DevOps som kom fram i studien vår.

4.2.1 Kultur

Et av temaene som intervjuguiden vår fokuserte på var kulturen i bedriften, og eventuelle endringer som en følge av implementeringen av DevOps. Alle intervjuobjektene hadde noe å si om temaet, men det var forskjellige aspekter som ble trukket frem. Pandemien ble trukket frem som en faktor av flere:

> *«Ja, det har vært en pandemi som har gjort at mye har endret seg i forbindelse med hvordan man jobber. Spesielt med hjemmekontor. ... Jeg hadde bortekontor, men egentlig ikke hjemmekontor. Jeg tror jeg hadde en dag på fire til fem år med hjemmekontor før pandemien. Jeg hadde hjemmekontor i går for eksempel.»*
>
> Informant 6 (arkitekt)

De har for tiden få problemer med at noen ikke sitter lokalt på kontoret, og de har tidligere jobbet sammen med den tekniske avdelingen:

> *«Det har jo vært situasjonen vår veldig lenge, også før vi begynte med DevOps. Jeg tror egentlig ikke at det er en kjempestor utfordring. Det er nok at man er tilgjengelig for hverandre når man trenger å være tilgjengelig. Det er gode nok kommunikasjonsveier her slik at det er sjeldent utfordringer med at man ikke er fysisk i samme rom.»*
>
> Informant 5 (arkitekt)

> *«Før ville jeg sagt at det påvirket arbeidet, før corona kom. Det var vanskelig når alle ikke var på kontoret.»*

Informant 4 (konsulent)

Selv om det ikke nødvendigvis er et problem at noen ikke arbeider lokalt, så mener flere at det å sitte sammen i det samme lokalet er det beste for samarbeidet:

> *«Så ønsket vårt er at nå som pandemien er over så får vi flere tilbake, det er lettere enn å sitte på teams da man bare kan snu seg dersom det er noe. Så det er et ønske, men det har fungert godt [jobbe i MS-teams], noen kommer sikkert til å fortsette med det og det avtales med leder. … jeg har inntrykket av at de fleste trives best med å være på kontoret sammen med folk»*

Informant 1(prosjektleder)

> *«Det aller beste hadde vært om alle satt sammen her …»*

Informant 2 (konsulent)

> *«Det hadde vært bedre om alle var på samme plass ja, men det har jo gått fram med nettmøter og alt mulig sånt. Var med i et prosjekt i 2004/2005 før man hadde Slack og Skype og da fortsatt Messenger var det som var moderne. Da var det helt umulig å jobbe på tvers, men nå sitter man i daglige møter med andre ressurser på andre plasser. Så det har blitt mer en hverdag. … det er ingenting som slår å sitte i samme rom.»*

Informant 7 (arkitekt)

> *«Med tanke på kommunikasjon og samarbeid så hadde det sikkert vært enklere om alle har vært under samme tak. Personlig synes jeg det er veldig godt å sitte på hjemmekontor for når jeg sitter her så blir jeg ofte forstyrret av folk som lurer på ting når jeg da er på en måte opptatt på teams så blir jeg ikke kontaktet like mye sånn at det å ha noen sånn der i fraværstimer så har vi 3 møter i uka: mandag, onsdag og fredag som var et 3 timers møte. Der sitter vi ikke nødvendigvis og prater i det hele tatt, men vi er tilgjengelige for hverandre hvis at vi har noen innspill eller noen spørsmål og slike ting, også er vi rød på Teams så da får vi jobbet hele den tiden der uten at vi blir forstyrret. Det er veldig godt slik, og jeg synes er mye enklere da også på hjemmekontor enn å være her på kontoret.»*

Informant 8 (utvikler)

Hierarkiet i bedriften blir beskrevet som flatt, og kulturen gjør at det ikke er stor avstand mellom ansatte i forskjellige roller:

> *«Som sagt er det veldig flat struktur her i Trondheim, så det er ikke noe vanskelig å tørre å snakke med ledelsen, arkitekter eller avdelingsleder. Han avdelingslederen sitter jo her og hvert fall som det er nå så sitter jo avdelingslederen med en høyere rolle enn for Trondheim også. … da sitter vi enda tettere på en av de høyere lederne i selskapet. Han er jo veldig fremoverlent, han er jo ikke redd for å prøve ting.» ¨*

Informant 1 (prosjektleder)

De som jobber der kommer godt overens med hverandre og kjenner hverandre godt, men dette er ikke nødvendigvis en nylig endring:

> *«Jeg har en fordel med at jeg kjenner mange personlig, så jeg vet hvem jeg skal kontakte om jeg har et problem. … Nei, jeg tror ikke den[kulturen] har endret seg. Det har jeg ikke merket. I denne bedriften er det nesten som en stor familie, det er ikke bare arbeidsgiver og ansatt. En er veldig nær til hverandre, er veldig kjent med hverandre og snakker direkte med folk. Det er ikke stivt og formelt. Det er veldig komfortabelt å jobbe sånn.»*

Informant 4 (konsulent)

> *«Flat struktur har det vært hele tiden. Men det har blitt en helt annen måte å jobbe iterativt på blant annet. Det skjedde hvertfall når vi begynte med den nye web versjonen i slutten av 2015. Da gikk vi all inn på den nye måten å tenke på, med å ha med kunden på dialog og sånne typer ting.»*

Informant 7 (arkitekt)

Siden flere av de ansatte er så godt kjent med hverandre, så hender det ofte at de ikke bruker support-systemet som er satt opp når de støter på et problem, men snakker i stedet direkte med noen som har kompetansen til å fikse problemet:

> *«Derfor oppretter jeg ikke noen supportsak eller lignende med en gang, men i stedet går jeg bare direkte til den personen jeg vet kan hjelpe meg. … Dette er*

kanskje å omgå systemet litt, men det hjelper meg ofte å få resultat kjapt. Om det går veldig tregt på den ene kunden, men ikke hos den andre så sender jeg en melding til en på teknisk: «Er det noe som sitter på tråden til den kunden eller? Det går tregt hos bare den ene kunden.». Da kan han sjekke det. Om jeg oppretter en sak, så kan det av og til ta lang tid. Jeg har ganske direkte kontakt med dem. Det er kanskje ikke så veldig populært, men det funker. Jeg får beskjed om jeg må opprette en supportsak og da gjør jeg det. De på utvikling har sine oppgaver, men vi har andre oppgaveområder. Om det er noen her som har problemer i BPO, så kan det hende de kommer til meg først. De tror jeg har teknisk kompetanse, men jeg har egentlig ikke det. Jeg har bare en annen forståelse av programmet. Når noe ikke virker, så må jo jeg ha hjelp. Jeg prøver å hjelpe, og om jeg ikke kan gjøre noe så forteller jeg dem hvem de bør kontakte for å fikse det, eller så tar jeg kontakt med noen på vegne av dem. Dette skjer av og til, men det er litt avhengig av hva det er.»

Informant 4 (konsulent)

Intervjuobjektene kunne fortelle at bedriften har ansatt mange nye folk de siste årene, og disse ansatte hadde en annen forventing til hvordan arbeidet skulle utføres og hvordan arbeidsmiljøet skulle være:

«… det har vært et generasjonsskifte de siste årene. Det som kommer inn, er nye unge folk som kommer i fra studiet der det har vært snakk om masse smidig så egentlig er den tankegangen her i hodet deres når de kommer hit. Så de vil nok forvente at det blir sånn man jobber. Sånn sett er det stort sett de eldre karene som føler på dette som en endring.»

Informant 1 (prosjektleder)

«I forhold til utviklingen, mitt hjørne av bedriften, så har det skjedd veldig mye. Vi har blitt veldig mange flere folk, vi har blitt en helt annen type folk. Før vi begynte med det nye produktet var det primært folk der som har jobbet sammen siden 80-tallet som har jobbet på den samme applikasjonen.»

Informant 5 (arkitekt)

«… da jeg ble ansatt så var jeg den yngste på kontoret, da var jeg rett ut fra høyskolen og det var, snittalderen var sikkert 70 sant. Jeg tar i litt, men nå i det siste så har vi jo ansatt veldig mange nye som er mye yngre enn meg og da kjenner jo jeg det at etter at de har kommet inn, så har på en måte kulturen eller hvert fall miljøet blitt litt annerledes. Litt mer sånn tulling og litt lettere egentlig synes jeg, det var litt sånn tungrodd tidligere, så det har blitt lettere egentlig.»

Informant 8 (utvikler)

Samtidig som bedriften har fått mange nye ansatte, så beskriver flere av intervjuobjektene at kulturen har endret seg over årene. Det er vanskelig å si om endringene i kulturen som er observert de siste årene eksisterer fordi at noen i bedriften har bestemt seg for å endre kulturen, eller om det er fordi at alle disse nye ansatte har andre holdninger:

«Vi har mye unge folk. Vi er et helt annet team. Det er veldig vanskelig å si om det er firmakulturen som har endret seg, eller om det er folkene i firmaet har blitt byttet ut, og det har blitt mange flere. Det har blitt en helt annen aldersprofil på folk. Ting har endret seg på måter som ikke er relatert til DevOps som kanskje er en større endring enn ellers. Hadde de vært akkurat de samme folkene og bare prosessene som hadde endret seg, så hadde det vært enklere å se hva som er hva.»

Informant 5 (arkitekt)

«Det har jo blitt det[endringer], og det er jo fordi at vi har ansatt nye. Det er ikke nødvendigvis det at vi har gått over til andre måter å jobbe på så siden det er på en måte veldig mange sånne ting som har skjedd så er det vanskelig å si hva som er årsaken til at ting føles bedre nå enn det var før da. Ja vi var jo ikke så mange tidligere så da var det på en måte de som hadde ansvaret som var litt sånn «high and mighty» mens vi som kommer fra skolebenken var litt usikre. Men, det jeg legger merke til er at folk kommer veldig fort inn i det og får veldig mye hjelp fra de som er rundt seg.»

Informant 8 (utvikler)

Dette kan sies å være en endring som ikke har skjedd de siste årene fordi en av informantene som bare har jobbet noen få år her, ikke svarte noe spesielt når vi spurte om endringer i kulturen:

> *«Ikke noen spesielle tanker om det, ikke som jeg vet om siden jeg ikke har jobbet her så lenge.»*
>
> Informant 3 (utvikler)

4.2.2 Fordeler

Et annet tema intervjuguiden inneholdt, var fordeler og ulemper/utfordringer med DevOps. En av fordelene som ble nevnt er at man får mer innsikt i hele prosessen på tvers av avdelingene. Informant 8 som jobber som en utvikler forklarer at han kan be driftsavdelingen om å kjøre en spørring for å kunne se hvilke data som ligger i ulike kundebaser. Dette er ikke noe man kunne gjøre før, for da var alt «on premises» og avdelingene jobbet mer adskilt. DevOps legger til rette for et tettere samarbeid mellom avdelingene, noe som gjør det enklere å få innsikt i prosessene knyttet til utvikling og drift:

> *«Det er veldig kjekt da å faktisk kunne få vite når jeg sitter og ser på koden hva er det egentlig kunden har av data i basen sin? For å se hvilken veier koden går da, så sånn sett så er det veldig kjekt. … Du blir jo litt mer obs på akkurat det her Ops bildet. Sånn for eksempel for database operatør oppgraderinger så har vi bygd verktøy for synkronisering av script for både oss selv og de gutta på teknisk, slik at man kan holde miljøene sine oppdatert på en mye bedre måte enn hvordan det var før.»*

Informant 8 (utvikler)

> *«At folk har mer ansvar for sin egen kode, og flere får mer innsyn i det en skriver.»*

Informant 3 (utvikler)

> *«Totalt sett, så er det vinning å være deltakende i hele prosessen for min og andres del.»*

Informant 6 (arkitekt)

En annen fordel gitt av informantene er at man får bedre flyt i levering av produkt. Dette kommer av at man tar mer kontroll og ansvar for sitt eget arbeid fra start og frem til levering. Prosesser blir også i større grad automatisert:

> *«Større kontroll over hele greia er en ting hvertfall. Det å få endret ting på og det å få ansvar ikke minst. Det å bli kvitt det kast over gjerdet greiene. Jeg vil si at man har blitt mere effektiv og, fordi det er også litt av den automatiseringstingen eller script tingen eller hvordan man skal si det med den DevOps bølgen»*
>
> Informant 7 (arkitekt)

> *«Det har mer vært at det er en logisk konklusjon på at vi har gjort mye forskjellige ting for å prøve å gjøre flyten i systemene og måten utviklingsteamet leverer produkt på bedre.»*
>
> Informant 5 (arkitekt)

> *«Jeg liker at det blir gjort automatisk på en måte. Det er jo ikke helt automatisk, men for oss er det automatisk. Vi trenger ikke tenke på det, vi får informasjon og det blir oppgradering. Jeg setter det i en kalender, og da må man logge av. «Du kan ikke jobbe da, for da må det oppgraderes.» Det er flott å vite at vi ikke trenger å bry oss om det.»*
>
> Informant 4 (konsulent)

Bedre oversikt over produktene og at man ikke trenger å vente lenge på små endringer blir også nevnt som en fordel av informantene:

> *«Nå så er det litt mer sånn at du koder ting og så har du kanskje i hvert fall jeg da, tilgang til et par kunde miljø. Da kan jeg gå inn der og se OK på ekte reell data hvordan har det funket, fikset denne buggen eller hvordan var den featuren her egentlig … Vi har begynt å samarbeide mer med kunder og kan dele litt skjerm over teams og de kan stille spørsmål og vi kan stille*

oppfølgingsspørsmål og svare på ting og sånt, veldig mye bedre enn at man bare koder en ting, ferdig tar tak i neste ting.»

Informant 8 (utvikler)

«Dette var fordi noen hadde installert løsningen i eget hus, noen hadde kjøpt det som en sånn Citrix remoting løsning, remote desktop-løsning fra oss. Det å lage og deploye en fiks på det, var ikke noe som man bare kunne gjøre. Nå kan det være fra en bug blir oppdaget, til at vi kan ha en ha en fiks ute i produksjon på en time.»

Informant 5 (arkitekt)

«Så har det med det nye systemet blitt enklere å bryte ned ting og komme med små endringer slik at en ikke trenger å vente på en stor release. Det er også greit å kunne fikse feil fortløpende i stedet for to ganger i året, med mindre det er store ting det er snakk om.»

Informant 4 (konsulent)

Det at det tar kortere tid å utføre en endring har videre ført til fordelen at de ansatte føler mindre på stress rundt utgivelse:

«Og det gjør jo også at det er mindre stress momenter og det er lavere puls på alle, fordi hvis det går galt og det går alltid galt, så er ikke det en meget stor kilde til stress for alle. Det har gjort at det er lavere skuldre på hele linjen.»

Informant 5 (arkitekt)

Fritt valg av verktøy blant utviklerne blir trukket frem som en fordel. Utviklerne har muligheten til å ta i bruk nye verktøy eller holde seg til de verktøyene de er vant til å bruke.

«Vi er jo mye mer fleksibel og åpen på verktøy og «tooling»-en nå enn vi var før. Ikke bare fordi at vi har endret på prosesser, men også fordi at det er mye mer «tooling» tilgjengelig….Vi har frihet til å kunne endre litt på hvilke verktøy vi bruker, og vi trenger ikke engang å bruke de samme verktøyene. Det er ikke alle utviklere som bruker de samme verktøyene lenger. Alle står helt

fritt til å bruke det de vil ha. Noen sitter i Rider, noen sitter i Visual Studio og noen bruker command line «tooling». Noen bruker Visual «tooling», noen bruker Visual Code.»

Informant 5 (arkitekt)

En annen fordel som ble diskutert var muligheten til å forme egen arbeidshverdag. Slik informantene forklarer, er håpet at DevOps strukturen skal gi de ansatte mer ansvar og frihet til å bestemme selv hvordan de skal påvirke utviklingen:

«Vi håper på mange måter at DevOps skal være med å løse en del ting, vi har hos andre bedrifter enkelte nøkkelpersoner som blir flaskehalser, hovedarkitekten er nok så ettertraktet med tanke på kunnskap og oversikt. Jo flere det blir her som skal ha en bit av han jo mindre tid blir det egentlig. Så håpet er jo at DevOps strukturen også skal gjøre hver celle eller team mer selvgående, at de tar litt mer ansvar selv....man får litt mer muligheten til å være med å bestemme på mange måter, man får ikke bare kastet oppgaver etter seg.»

Informant 1 (prosjektleder)

«Man kan være med å forme sin egen arbeidshverdag. Dette er verktøyet vårt. Å være med på å påvirke utviklingen av verktøyet slik at det blir slik som vi vil.»

Informant 2 (konsulent)

4.2.3 Ulemper/utfordringer med DevOps

En utfordring med å gå over til DevOps, er at det ikke alltid er mulig å få alt over til den nye måten, og en ender opp med at en er nødt til å tilby to versjoner av programvaren sin til kundene:

«vi begynte å gjøre litt oftere releaser enn sånn 2-3 ganger i året, men det var bare en liten kort tidsperiode til vi gikk rett over til 14 dagers. Men der også har det jo vært i en overgangsfase nå, så har og kundene vært delt. Det vil si at

noen har bare kjørt den gamle Windows versjonen og noen har kjørt en miks av det nye Web og vanlig Windows og da har vi oppgradert dem i 2 grupper. En for dem som kjører web og har hatt hver 14 dag og så lå de kundene som bare kjører Windows litt bak sånn at vi tok dem opp en gang eller 2 i kvartalet, men nå så ligger vi mer at de ligger 1 til 2 releaser bak om, så de oppgraderer annenhver gang for eksempel.»

Informant 7(arkitekt)

En annen utfordring knyttet til overgangen til en DevOps hverdag er at det er vanskelig å bryte ut av det gamle rollemønsteret. Utviklere har vært vant til å kode noe ferdig og så sende det videre før de begynner på noe nytt. Slik er det ikke nå lenger:

«Utviklere liker tankegangen: «gjør en ting, bli ferdig med det, ferdig!». Du skal lage en ting, så har du lagd en ting og så er du ferdig med det. Da får du det ut av hodet ditt, for du skal være klar for nye ting. Det at du har ikke lagd en ting før den faktisk er i bruk av noen, og selv etter det så er det din ting de bruker. Hvis de sitter og trykker på den, og så funker det kanskje ikke. Eller så gjør den ikke det den skal, eller så gjør den ikke bra nok eller lignende. Og hvordan skal du vite om den gjør det bra? Den måten å tenke på der man liksom har ansvaret hele veien oppover, den sitter langt inne.»

Informant 5 (arkitekt)

En utfordring med DevOps er at det tar mye arbeidstid og visse ansatte får mye å jobbe med. Bedrifter som jobber med DevOps jobber ofte smidig med rundt 14 dagers utgivelsessykluser. Det er mye som skal gjøres mellom de dagene og da må det gjøres prioriteringer for å komme i mål:

«Vi har en grei størrelse på utviklingsteamet til å være et norsk softwarefirma i den bransjen vi er i, men vi har ikke tusenvis av utviklere. Vi har rett og slett ikke tid til å levere de tingene vi skal levere og samtidig gjøre dette, så vi må prioritere. Da må man finne ut hvordan man skal prioritere tiden til de som sitter på utvikling? Hvordan skal man prioritere tiden til alle som sitter i den

stacken? Hva har vi faktisk tid til å gjøre eller ikke? Og alle har alltid lyst til å gjøre mer enn de har tid til. Det er bare sånn det.»

Informant 5 (arkitekt)

Det informant 5 beskriver stemmer godt med det som ble sagt av informant 1. Med den 14 dagers syklusen blir det utfordrende å rekke alle gjøremål og man må derfor prioritere det viktigste frem til neste utgivelse:

«Hvis vi skal ha det testregimet vi hadde før der vi hadde kode freeze 3 uker før vi slapp noe så hadde ikke det gått an, da hadde vi en tester som satt her og hun satt stort sett å testet jevnlig hele tiden men når vi da kom til slutten av versjonen så ble det litt sånn test splid så ble det sendt ut testing til alle utviklere ... Vi brukte veldig mye tid på det. Det greier vi ikke nå når vi har så kort tid mellom hver versjon. Så du sjekker det viktigste, du sjekker at en lønnsspredning blir kjørt at det ikke ser spinnvilt ut, du får lagt til personer, du får snakket med Altinn. Det er egentlig en liten liste.»

Informant 1 (prosjektleder)

Legacy systemer gjør det utfordrende å implementere DevOps i en bedrift. Legacy systemer er som oftest ikke kompatible med en DevOps tilnærming og krever at oppdateringen eller endringer gjøres manuelt on prem. Overgangen til et nytt sky basert system kan derfor bli meget tidskrevende da bedrifter ofte må fortsette å drifte legacy systemet frem til kunder og data er migrert over til det nye. Dette trekker arbeidskraft og ressurser bort fra det nye systemet:

«Ja, rett og slett for å holde kundene i sync fordi vi vet at alle kundene som kjører på den versjonen skal over på den nye og det er samme databaseskjema og samme bakgrunns logikk og alt mulig rart. Så for å hindre at det løfter blir så stort og de kan ta i bruk en del av det nye på den gamle plattformen.»

Informant 7 (arkitekt)

«Det er jo legacy kode vil jeg tro, det er jo et eldgammelt system som ligger til grunn for alt og vi har jo pensjonister som jobber for oss for å refaktorere og

sånn så som er kjent med koden fra før av. ...vi har jo ting som må skrives om og fornyes og gjøres tilgjengelig for den nye løsningen og da er det jo mye arbeid som går med på det.»

Informant 8 (Utvikler)

Pålegg fra myndigheter kan skape utfordringer for det nye skybaserte systemet. Ettersom det nye systemet utvikles og driftes med en DevOps tankegang, vil pålegg fra myndighetene også påvirke implementasjonen av DevOps:

«Det er kanskje det å få tilgang til de underliggende tjenestene. Det tar tid å bytte virtualiseringsplattform og sånn type ting, også det at vi ikke har lov til å bruke public cloud, har holdt det tilbake....det er på grunn av GDPR hensyn, for der finnes det masse tjenester som er ferdig innbokset for den type bruk, men når du skal levere fra ditt eget datasenter i steden så er det litt mere eller krever litt mere å finne produktene og plattformene å bruke.»

Informant 7(arkitekt)

«Den største utfordringen er når det kommer nye pålegg fra myndighetene, og vi må gjøre ting på en annen måte. Da har de kanskje implementert det og ikke gitt beskjed til leverandørene. Eller så har de en ny implementasjon og ingen spesifikasjoner på hvordan det skal se ut. Da skal utvikling plutselig fikse og tilrettelegge uten å teste det. Hvordan skal du programmere noe uten at du vet hvordan det blir tatt imot? Og da forventer vi i BPO at det bare fungerer. Vi ser ikke hva som er involvert bak utviklingen.»

Informant 4 (konsulent)

4.2.4 Verktøy

Bruk av verktøy er en stor del av DevOps filosofien. Det ble derfor naturlig å inkludere noen spørsmål om verktøy i intervjuguiden. Et av hovedtemaene vi tok for oss var om informantene hadde merket noen endringer i bruk av verktøyene de bruker nå sammenlignet med før det nye

systemet var på plass. Svarene vi fikk fra informantene varierte fra ingen opplevede endringer til store endringer i verktøy:

> *«Nei, egentlig ikke. Vi har vært en Microsoft-partner siden vi var den gamle bedriften, vet ikke hvordan det er i den nye bedriften som har kjøpt oss opp. Så vi har brukt Microsoft produkter hele tiden, helt siden TFS nett. Da har vi jo brukt det som Source og tracking punkt og alt det her. TFS ble mer og mer en skyløsning tror det fortsatt het TFS i skyen de første årene men for noen år siden endret de navnet til Azure DevOps. Da hadde vi allerede brukt web løsningen en periode så det er egentlig bare noe som ble med. Så kanskje største endringen var når det gikk fra å være en klient til å være en web løsning. Sånn sett har vi ikke skiftet noe fordi det skal være DevOps, vi har jo skiftet litt fordi vi har gått over til en webløsning.»*

Informant 1 (prosjektleder)

> *«Egentlig ikke. Vi har jo blitt kjøpt opp av et europeisk selskap, og det er noen semantiske endringer. I stedet for å bruke Conference, så bruker vi noen Microsoft-program. Jeg bruker ikke det så veldig mye.»*

Informant 3(utvikler)

> *«Det er verktøy som vi bruker til å registrere feil som egentlig forandrer seg over tid. Ellers nei, egentlig ikke. Det er mer utvikling som har merket endringer»*

Informant 4 (konsulent)

Overgangen til den nye skyløsningen regnes som en endring i verktøy av informantene. Nå utvikles, leveres og driftes tjenestene via skyløsningen. Slik var det ikke før:

> *«Før var det inhouse, produktet var installert hos kundene og noen hos oss. Da visste vi aldri hvilken tilstand systemet var i.»*

Informant 4 (konsulent)

> *«Absolutt, meget store endringer. Det gamle systemet kan ikke sammenlignes med det nye bortsett fra at mye av den samme funksjonaliteten ligger der, men den funker på en annen måte og det er mye ekstra funksjonalitet. Det har jeg lagt mye merke til.»*
>
> Informant 2 (konsulent)

Overgangen til skyløsningen har gitt bedriften muligheten til å levere programvare som en tjeneste og ikke bare som et produkt:

> *«Ja, så det som vi leverer er en ren tjeneste. Det er ingen som får binærfiler fra oss blant dem som bruker produktet vårt. De kjøper produktet som en tjeneste, de abonnerer på det produktet som en tjeneste fra oss. Vi leverer ikke produkt, vi leverer tjenester.»*
>
> Informant 5 (arkitekt)

Automatisering blir trukket inn som en av de største endringene av verktøy. Det nye systemet er lagt til rette for automatisering av prosesser, men kan kjøres manuelt dersom det er ønskelig:

> *«Hele UI er totalforandret, det er ingenting som ser ut som det gjorde før. Alle snarveier er forandret. Helt nye prosesser, spesielt med automatisering av en god del prosesser som var veldig manuelle før. Det er den største endringen: automatisering av ting. En kan bruke det nye systemet helt manuelt også, men det er lagt opp til at alt skal være automatisk. En merker tidsforskjellen mellom manuell utførelse og automatisert utførelse av prosessene. En kan sette prosesser til å begynne på et tidspunkt man gjør noe annet, og så kan man kontrollere om det har gått bra eller feilmelding har kommet i ettertid. Kanskje får du en arbeidsliste over det som må gjøres. Vi kontrollerer resultatet før vi produserer filer, og en stor forandring er at i stedet for at vi får en lang liste over rapporter som vi har kontrollert manuelt, så i det nye systemet så blir disse kontrollert i bakgrunnen. Om det blir funnet en feil, så kommer det en oppgave i arbeidslisten.»*
>
> Informant 2 (konsulent)

> *«Jeg liker at det blir gjort automatisk på en måte. Det er jo ikke helt automatisk, men for oss er det automatisk. Vi trenger ikke tenke på det, vi får informasjon og det blir oppgradering. Jeg setter det i en kalender, og da må man logge av. «Du kan ikke jobbe da, for da må det oppgraderes.» Det er flott å vite at vi ikke trenger å bry oss om det.»*
>
> Informant 4 (konsulent)

Selv om det nye systemet er lagt opp til automatisering er det ikke alle prosesser som er automatisert enda, slik som utgivelse:

> *«Det hadde sikkert vært greit om det hadde vært mer automatisert release, det virker som en veldig manuell prosess. For hver release, så sitter vi alltid og kikker gjennom casene som skal gjennom og venter på ting. Så må en person lage en branch, og bygger det på en byggeserver og deployer det. Så må en gå gjennom manuell riggetesting med Devops-folkene. Det går jo fort en del timer der vil jeg tro.»*
>
> Informant 3 (utvikler)

En annen endring som har funnet sted er friheten til å velge verktøy selv. Informant 5 forklarer at før bedriften jobbet med DevOps var de mye mer låst i alt av verktøy. Mye av dette skyltes at prosessene var tyngre, men også tiden de levde i. Det var mindre utvalg av verktøy innen .NET-infrastrukturen de jobber i, man kunne ikke bare bytte ut utviklerverktøy.

> *«Vi er jo mye mer fleksibel og åpen på verktøy og «tooling»-en nå enn vi var før. Ikke bare fordi at vi har endret på prosesser, men også fordi at det er mye mer «tooling» tilgjengelig.»*
>
> Informant 5 (arkitekt)

Nå er det en mye mer pragmatisk tilordning til «tooling» fra for eksempel Microsoft. De var meget proteksjonistiske før og krevde at man kjørte Microsoft hele veien fra programvare til maskinvare. Omtrent når bedriften startet med DevOps gikk Microsoft gjennom sin egen lille

revolusjon i forhold til open source. Fokuset beveget seg bort fra å selge lisenser til for eksempel Visual Studio og heller til å tjene penger på å selge Azure-tid:

> *«Det er positivt for oss fordi at vi er nå veldig frie til å kunne endre litt på «tooling». Vi har fri til å kunne endre litt på hvilke verktøy vi bruker, og vi trenger ikke engang å bruke de samme verktøyene. Det er ikke alle utviklere som bruker de samme verktøyene lenger. Alle står helt fritt til å bruke det de vil ha. Noen sitter i Rider, noen sitter i Visual Studio og noen bruker command line «tooling». Noen bruker Visual «tooling», noen bruker Visual Code. Man kan ha et helt annet pragmatisk forhold til det, men veldig lite av det har egentlig med prosess å gjøre, veldig mye av det har med hva som i det hele tatt er tilgjengelig.»*
>
> Informant 5(arkitekt)

4.2.5 Hva de ansatte syntes om DevOps (F2.1)

I dette underkapittelet vil alle resultatene som viser hva intervjuobjektene tenkte om DevOps. Dette vil være resultatene som hører til forskningsspørsmål F2. Resultatene er delt inn i positive meninger og motstand/negativ oppfatning til DevOps.

4.2.5.1 Positive til DevOps

Fra dataene vi samlet inn ser det ut som alle var positive til DevOps på en eller annen måte. Enten liker de hvordan DevOps gir dem større frihet:

> *«Jeg synes det er veldig positivt, og endringen har vært at jeg har fått mer forståelse for hva DevOps handler om og hva som kreves. Vi har en gyllen mulighet til å påvirke hvordan verktøyet vårt skal være og få med i programmet hva vi ønsker samt hva vi trenger»*
>
> Informant 2 (konsulent)

Eller så liker de at de får være med på mer av prosessen:

> *«… . Jeg liker å være involvert»*

Informant 6 (arkitekt)

Informant 5 (arkitekt) har hatt kjennskap til DevOps en stund, og mener fremdeles at det er en god ide:

> *«Jeg synes det høres ut som en god idé fremdeles. Jeg synes det hørtes ut som en god ide når jeg hørte om det første gangen. Litt som med Scrum, så er det en sånn ting som går seg litt til»*

Informant 5 (arkitekt)

Deltakerne i case-studien var positive til DevOps, og gjerne til spesielle fordeler som kommer som følge av DevOps til og med før de begynte å arbeide på en slik måte. Fordelene blir nevnt i et tidligere kapittel. Det var også noen som var mer skeptisk til DevOps da de først hørte om det, men har endret mening etter å ha prøvd det i praksis:

> *«Ja, opprinnelig når jeg hørte om å release hver andre uke fra en testleders perspektiv, da var det skremmende. Jeg vet jo hvor mye tid det tar å utvikle ting.»*

Informant 4 (konsulent)

Dette var på grunn av at ting som kortere sykluser kan virke vanskelig å gjennomføre når en er vant til å gjøre det på gamlemåten.

4.2.5.2 Motstand

Det kan være vanskelig å tilpasse seg nye måter å jobbe på, og DevOps slik som vi har beskrevet det tidligere i dette dokumentet kan være meget forskjellig fra tradisjonelle metoder. Derfor kan det oppstå motstand når en endrer slike ting i bedriften, og det som blir tatt opp i neste kapittel blir viktig å ha i tankene. En overgang til en kortere utviklingssyklus er et eksempel på en av de mer radikale forskjellene, og ble nevnt når vi spurte om motstand:

> *«Jeg tror det var noe skepsis til det [ny utviklingssyklus], for tidligere så var det jo 3 release i året. Da hadde du jo samlet opp da gjerne 4 eller 5 måneder med databaseendringer som skulle ha inn i den versjonen. Hvis en ting gikk dårlig der, så gikk det jo dårlig, og man skyndet seg ut med en hotfix, og så*

videre og så videre. Det var på en måte en risiko for kunden å skulle for eksempel ta inn en oppgradering rett før de skulle kjøre lønn, så da måtte man som regel vente med nå har vi kjørt lønn, da kan vi gjøre oppgraderingen hvis ting går dårlig og så får vi en ny versjon kanskje i løpet av 2, 3 eller 4 uker som fikser problemet sånn at det er klart neste gang du skal kjøre lønn. Så jeg tror det var noen som var litt skeptisk til at OK hver 14. dag nå som kommer databaseoppdatering, kanskje. Det tror jeg det var noen som reagerte på i begynnelsen.»

Informant 8 (utvikler)

Det var heller ikke alle som var like interesserte i å ta i bruk alt som har blitt implementert av arbeidsmetoder, spesielt de som ikke jobber med det nye systemet:

«Det går litt tilbake til det at vi var på vei en stund før vi begynte å bruke ordet DevOps. Da blir det en del av disse Scrum-tingene vi har ført inn som standup, det hadde vi ikke før. Noen synes at standup er en fin måte å holde seg oppdatert på og noen føler seg litt mer overvåket og synes ikke dette er morsomt i det hele tatt, spesielt de gamlekarene. De har på mange måter meldt seg ut, de er sjeldent med på standup, de er sjeldent med på sånne ting. Men så sitter de egentlig bare og jobber på den gamle Windows klienten. Vi er lett tilgjengelig dersom de lurer på noe, men som oftest er ikke det vi driver med viktig for dem. Dette går litt utenfor DevOps da man egentlig skal inkludere alle, Vi har også prøvd å få med de tekniske inn på standup-møter men det er de ikke med på. Vi har ikke presset så hardt for å få med dem heller fordi de egentlig har lite konkrete oppgaver. I løpet av dagen blir ulike ting kastet mot dem av typen "gjør det her", de følger som oftest en mailliste.»

Informant 1 (prosjektleder)

Det er ikke bare utviklingssyklusen som ble trukket frem som en kilde til motstand. Det nye systemet som bedriften har tatt i bruk var også en kilde til motstand:

«Ut ifra min oppfatning har det vært noen som ikke har vært villige til å bruke det nye systemet og vært negative rundt det. Det har blitt en trykkende sak, og nå har de begynt å stenge funksjonalitet i det gamle systemet slik at man

tvinges til å bruke det nye systemet. Da har det blitt litt irritasjon her og der. Det en del personer som dette handler om. Jeg som bruker det nye systemet, klarer ikke å se på saken på samme måten som dem. Jeg ser på systemet som positivt, og ønsker at andre hadde det samme synet på denne saken. Sånn er det dessverre ikke.»

Informant 2 (konsulent)

«… kanskje er det noen som opplever at det er vanskelig å tilpasse seg så fort. For de eldre ansatte så var det vanskelig å få tankegangen over fra det gamle systemet over til det nye systemet fordi de kjente alle kodene og alle de tingene som blir sjeldent brukt og slikt.»

Informant 4 (konsulent)

Denne motstanden er rettet mot bruken av det nye systemet, og ikke nødvendigvis måten en jobber på med DevOps. Det ble også trukket frem motstand som var rettet mot samarbeidet og tilbakemeldingene som DevOps oppfordrer til. Dette ble sagt om situasjoner hvor konsulenter gav tilbakemeldinger til utviklerne og ble spurt oppfølgingsspørsmål:

«De som spør, lurer gjerne på hvorfor de blir intervjuet på denne måten. Jeg har snakket med noen av de andre konsulentene der de sier at de blir utspurt og at dette er ubehagelig, at de blir svar skyldig i stedet for at de forstår at utvikling er nødt til å spørre.»

Informant 2 (konsulent)

Informanten mente også at det var blitt en negativ endring i kulturen:

«Egentlig har det blitt en negativ endring i kulturen fordi det ikke har vært noen spesiell kommunikasjon og samarbeid mellom utvikling og konsulentene.»

Informant 2 (konsulent)

I motsetning til dette, så ble det sagt av flere av intervjuobjektene at det var lite eller ingen motstand mot DevOps. På spørsmålet om de visste om motstand mot DevOps i bedriften svarte de:

«Nei, ikke i det hele tatt.»

Informant 6 (arkitekt)

> *«Nei ikke noe spesielt, det har vært veldig toppstyrt det der med DevOps og at det skal være mantraet.»*

Informant 7 (arkitekt)

> *«Nei, det tror jeg ikke fordi det er ingen som har sagt at «Nå skal vi bruke DevOps!».»*

Informant 5 (arkitekt)

En grunn til at det ikke var noen spesiell motstand, var ifølge en informant at det ikke var en endring som kom utenfra:

> *«Hvis man føler at man ikke er med på det, men føler at det er en ting som blir påtvunget fra utsiden og at man kjenner på at det her funker ikke, så tror jeg man vil stritte imot en god del. Det å ha et team som er med på dette er viktig»*

Informant 5 (arkitekt)

> *«. Eller altså det har vært uttalt mange ganger at nå skal vi bruke DevOps, men det er mer at det er noe som vi synes vi burde si. Det har sklidd inn veldig naturlig. Det var ikke «Nå skal vi gjøre DevOps», det var mer «Oi! Vi har innsett at vi egentlig allerede gjør DevOps.» Det er litt flåsete å si. Selvfølgelig har man jo lest om og vet om alle tingene. Det er jo ikke sånn at vi har funnet hele DevOps på egenhånd uten at vi har hørt om det. Det er heller ikke sånn at noen har fått dette tredd nedover seg fra toppen av. Det her har kommet naturlig. Vi har jo alle hørt om det. Vi har lest om det, vi alle sammen har en viss idé om hva det er for noe og hele den prosessen vår rundt det å være introspektiv på sin egen prosess gjør jo at man går rundt seg selv og prøver å finne hvilke andre ting man kan prøve. Man prøver de tingene man har hørt om og lest om, og da prøver man noen ting fra for eksempel DevOps for å se om er en god ide i stedet for at man fra toppen får beskjed om at nå skal vi bruke DevOps, og her er malen implementert. Dette er noe som ville møtt mye mer motstand. Det er samme måte som hvis noen fra toppen av en ledelse har*

gått på et kurs som hørt om Scrum eller en annen type metodikk og sier: «Nå skal vi gjøre det!». Hvis det ikke kommer naturlig fra den gruppa som driver med utvikling, så er det vanskelig å få det tredd nedover hodet. Det er nok ingen, hos oss i hvert fall, som har følt at dette har blitt tredd nedover hodet på dem. Det har kommet naturlig fra innsiden, mer enn en at det har kommet tredd ned fra utsiden.»

Informant 5 (arkitekt)

Informanten mener at så lenge slike ting som Scrum og DevOps blir innført fordi at de ansatte synes de virker som en nyttig måte å arbeide på, så blir det mindre motstand. Mange av intervjuobjektene våre mente at DevOps fungerer bra:

«Fra mitt standpunkt funker det bra»

Informant 2 (konsulent)

«Jeg tror egentlig at alle synes det er ganske greit …»

Informant 3 (utvikler)

«Hvordan jeg føler at implementeringen har gått, er helt greit. Det er litt vanskelig å trekke fram noe spesielt.»

Informant 5 (arkitekt)

4.2.6 Det som er nødvendig for å implementere DevOps på en god måte (F2.2)

Det siste av resultatene våre vil være relevant for det siste forskningsspørsmålet vårt der det som er kritisk for implementering er temaet.

Når vi spurte deltakerne om hva de mener er nødvendig for å implementere DevOps ble det klart at det var vanskelig å komme med en liste med konkrete steg. Flere av deltakerne forklarte at implementeringen av DevOps var en naturlig eller flytene overgang.

Informant 5 forklarer at de tidligere hadde lest og hørt om DevOps men at det ikke var et bevist valg å implementere denne måten å jobbe på. Det forklares videre i intervjuet at hele den filosofien de har rundt det å være introspektiv på sin egen prosess gjør at de ofte prøver å finne hvilke endringer i prosesser de kan gjøre for å få en bedre flyt i systemene. Det kom frem at

DevOps ble mer en naturlig konklusjon på at de hadde gjort mange ulike tiltak for å prøve å gjøre flyten i systemene og måten utviklingsteamet leverer produktene på bedre:

> *«Det har sklidd inn veldig naturlig. Det var ikke «Nå skal vi gjøre DevOps», det var mer «Oi! Vi har innsett at vi egentlig allerede gjør DevOps….Litt av filosofien vi har hatt som arkitekter på det nye systemet har vært at vi er veldig pragmatiske og introspektive rundt egen prosess. Vi er åpne for å prøve alt, vi er villige til å utfordre alt, vi tar ingenting for gitt og vi gjør det som funker. Vi slutter å gjøre det som ikke funker.....I forhold til hvor mange prosent DevOps er vi i dag i forhold til i går, så er det egentlig ikke nå vi har noen mål på. Og det er fordi at det er ikke noe implisitt eller eksplisitt mål for oss å bruke" DevOps. Det som var et mål for oss er å levere produkt på en bra måte. Hvis DevOps gir oss noen ting som gjør dette bedre, så gjør vi det, men vi gjør ikke DevOps for å gjøre DevOps.»*
>
> Informant 5 (arkitekt)

Informant1 mener også at overgangen til DevOps var flytende og at dette var det naturlige neste steget for bedriften:

> *«Vi var på vei dit på mange måter, sånn litt i det flytende. Hovedarkitekten var nok en av de som var dypest inn i det her og pratet med avdelingslederen. Plus att det er litt i tiden, hovedarkitekten holder seg veldig oppdatert. Vi var jo allerede smidig, vi var allerede veldig tett på den tekniske gruppen. Største forskjellen var vel egentlig at det knyttet oss enda nærmere mot teknisk. Så det var bare en naturlig overgang. Vi var på vei dit og det hadde et navn.»*
>
> Informant 1 (prosjektleder)

Når deltakerne ble spurt om nødvendige faktorer for implementering av DevOps ble det også nevnt av flere at det bør legges vekt på god kommunikasjon og dialog mellom utviklere, driftere og ledelsen for å implementere DevOps på en god måte.

Informant 8 ble spurt om han hadde merket noen forskjeller på måten de kommuniserer på nå i den nye webløsningen som følger en mer DevOps orientert filosofi:

> *«Det er jo litt vanskelig å si, det eneste jeg har lagt merke til er jo at jeg har mer kontakt med de som faktisk sitter og vedlikeholder det her, litt mer sånn direkte kontakt med dem heller enn at de bare var en sånn gjeng under plassen som ikke visste noen»*
>
> Informant 8 (utvikler)

Denne informanten beskrev videre at det er viktig å ha god dialog mellom avdelingene for å lykkes med implementasjon og bruk av DevOps:

> *«Jeg tror det å bryte ned siloene mellom utviklere og drift kan være viktig. Hvis du sitter for deg selv som en utvikler og ikke tenker på noen ting annet så jobber du i hvert fall ikke DevOps, du får jo gjort jobben din, men det er noen som sitter på den andre enden og skal implementere det her og ønsker at ting blir utviklet på en sånn måte at det går an å implementere det. Hvis de manuelt må inn og konfigurere ting ved hver release så jobber de sikkert et par uker da for å få det på plass og så kommer en ny release. Så man bør kommunisere litt med dem som faktisk sitter på Ops, for å høre hvilke behov har dere hva er det dere egentlig gjør for noen ting når vi sender en ny versjon til dere?»*
>
> Informant 5 (arkitekt)

Svaret gitt av denne informant 5 stemmer godt med informant 8 angående dialog. Informant 2 forklarer at det nå er mer dialog mellom avdelingene:

> «*Før var det sånn at utvikling lagde en pakke. Så er det opp til operations å finne ut: «Hva i all verden skal man gjøre med den her pakken for å få den i hendene på kundene?» Mens nå er det mye mer dialog om at man skal sørge for at man faktisk har de verktøyene man har. Hvor kan man forbedre, hva kan man lage og scripte, hvordan kan vi endre på den pakken leveres som gjør den*

enklere å drifte til det punktet at det er nesten ingen manuelle steg på nesten noen ting av den deploy-prosessen vår?»

Informant 2 (konsulent)

Informant 2 som jobber som konsulent nevner også at god dialog er viktig mellom avdelingene. Informanten forklarer at han har god dialog med utviklings avdelingen og er fornøyd med at de hører på han sine ønsker. En åpen dialog og mye snakking åpner for den forståelsen om at man jobber på forskjellige måter. Informanten forklarer videre at noen kommer med en forespørsel til utvikling, så blir de utspurt av utvikling. En slik dialog er viktig å ha da utviklingsavdelingen må ha muligheten til å forstå bakgrunnen for forespørselen og om det er noe som kan implementeres:

«God dialog og litt push ifra toppen. … Jeg har jo kjempegod dialog med de på utvikling. Jeg får jo gjennom mine ønsker … De kan ikke bare implementere forespørsler uten videre. De må jo forstå bakgrunnen for det og si ifra om de har en erstatning for det som ønskes.»

Informant 2 (konsulent)

Informant 1 ble også spurt om han har merket noen forskjeller på måten de kommuniserer på nå og hvordan de foregikk før DevOps. Informanten mente at det ikke var noen store forskjeller på måten de kommuniserer på nå annet en statusmøtene som de ulike teamgruppene har:

«Nei, egentlig ikke. Vi har blitt flere, men det har alltid vært en nokså flat struktur, det har vært greit å prate alle veier, ledelsen har sittet tett på. Vi har prøvd ulike møteformer, nå har det blitt så mange at det har blitt vanskelig å følge med, spesielt for avdelingsleder og hovedarkitekt som sitter mye i andre møter. Så har vi hvert av disse små teamene som har sitt eget møte hvor det egentlig rapporteres i statusmøte, hva har skjedd forrige uke, hva skal vi jobbe med neste uke, hva kan vi forvente å se i neste versjon, er det noe man trenger fra våre produktgrupper. Så vi prøver å innføre det fordi det blir vanskelig å følge med etter hvert for hva alle gjør, det blir så mange.»

Informant 1 (prosjektleder)

Overgangen til den nye webløsningen som følger en DevOps fremgang har ført til at bedriften har måttet ansette flere. Slik som informant 1 beskriver har det blitt vanskeligere å følge med på alle de ansatte og har derfor innført statusmøter som et tiltak for å opprettholde dialog med alle teamgruppene.

Informant 4 (konsulent)
Informant 4 tar også opp god kommunikasjon som en viktig faktor. Informanten snakket også om statusmøter og forklarer at:

> *«Det er viktig med god kommunikasjon. Det som mange savner er dokumentasjon på hvordan ting er gjort, hvor man finner ting og hvordan det ser ut, men ting endrer seg så mye at det vil være en fulltidsjobb å oppdatere dokumentasjonen hele tiden. De løser det ved å ha møte annenhver uke hvor de viser de nye funksjonene, statusmøter. De som ikke har jobbet med det nye systemet er mindre interessert i dette, for de er ikke på enda. Utvikling har også forskjellige seminarer en eller to ganger i året, som er veldig nyttig. De varer en halvtime og viser ulike funksjoner.»*
>
> Informant 4(konsulent)

Videre i intervjuene kom det frem at for å implementere DevOps på en god måte er det viktig at man er villig til å endre på at ting og at de ansatte og ledelsen forstår hva det vil si å jobbe på en DevOps orientert måte. I tillegg til dette er det viktig at de tekniske forutsetningene er på plass. Man må ha muligheten til å skape og levere et produkt som er DevOps kompatibelt:

> *«Du må villig til å endre på ting. Du må være villig til å innse at det du synes var en god idé for en måned siden, ikke nødvendigvis er en god ide i dag. Det handler ikke bare om DevOps, dette gjelder generelt også. Hvis man er for firkantet og for lite villige til å gjøre om på de tingene som må gjøres om på, så får man ikke til å gjøre noen form for én ting»*

«Det handler om at alle sammen er med, og at alle sammen ser at dette er en god ide da. At man har den rette måten å tenke på, og at man skjønner hvorfor man gjør det, for det er ikke gitt at alle endringer er bra. Alle sammen må være med, og alle sammen må ha eierskap og på samme måte ha eierskap til produktet man leverer hele veien fra a til å, og så må alle sammen ha eierskap til prosessen man bruker. Hvis man føler at man ikke er med på det, men føler at det er en ting som blir påtvunget fra utsiden og at man kjenner på at det her funker ikke, så tror jeg man vil stritte imot en god del.»

Informant 5 (arkitekt)

Informanten ble videre spurt om man bør ansette folk som har denne holdningen. Informanten forklarer at det er noe man kan gjøre, men at det også er viktig å ha ansatte som er villige til å tilegne seg den kompetansen som trengs:

«Ja, eller gjøre det på en måte som fører til at du får med deg folk. Du må skjønne hvordan det påvirker forskjellige ting som hverdagen og arbeidslivet til alle de ansatte, og da må det presenteres og gjøres på en måte som fungerer. Man kommer også ikke utenom at man må ha den tekniske forutsetningen for å gjøre det. Særlig i vår bedrift, så drev vi lenge med å utvikle web-applikasjoner der de var bygd mer eller mindre som Windows-applikasjoner. Det var gjort ingen innsats i deploy-pipelinen på dem. De som hadde kodet dem hadde ikke peiling på deploy-pipeline. Det er meg inkludert. Jeg var en av dem som kodet på det og jeg hadde ikke peiling på deploy-pipeline på det tidspunktet. Så selv om man bruker web-apps og slikt, så må man ha utviklere som er villige til å gjøre det, og ha eller tilegne seg kompetansen som trengs»

Informant 1 (prosjektleder)

Informant 1 ble spurt om ledelsen motiverte de ansatte til å jobbe på en DevOps-orientert måte. Informanten svarte at det heller var de ansatte som var motiverte og at det var ledelsen som måtte være villige til å jobbe med DevOps, da det krever at man beveger seg bort fra ren fossefallsstruktur der ledelsen styrer og kommer med konkrete oppgaver til de ansatte:

«Jeg føler egentlig at det har gått andre veien. At det kommer fra bunnen og opp så det krever jo en ledelse som er åpen for det og som er åpen til å miste litt kontroll for det var på en måte fordelen med vannfalls metoden. Da kom det en liste og dette er de punktene som skal løses og de skal løses til den datoen»

«Det føles nok litt rart for ledelsen. Det var jo mye mer rapportering typisk, her ser vi at lederne er vant med rapporteringer man har fått gjennom vannfalls metoden tidligere i det her skifte når vi begynte å bli mer smidig men så ser vi at det har blitt mildere og mildere også fordi de på toppen ser at dette fungerer jo faktisk. Så jeg føler at mye av dette har kommet fra bunn fra de unge som har lest og kanskje hatt om dette på skolen og ønsker at vi jobber sånn.»

Informant 1(Prosjektleder)

Informant 1 var på lik linje med informant 5 enig i at man må være villig til å endre på ting, spesielt på ledelses nivå. Informanten bygger videre på det med at man må bevege seg bort ifra en ren fossefalls tilnærming skal man lykkes med implementeringen av DevOps:

«Når vi gikk ifra Scrum og bare sa at det er det øverste på backloggen som til enhver tid skal taes så mister du enda mer oversikt. Det sitter jo arkitekter sammen med hovedarkitekten på toppen og fyller på backloggen, så det planlegges på høyere nivå, lages kjøreplaner, vi skal komme med fraværs løsning da og da og vi skal jobbe med byrå derfra og dit. Den planleggingen for konkrete punkt må de gi slipp på. Så du må først ha en leder som godtar det og lederen over det så er det jo enda vanskeligere å få med de som sitter oppå det igjen å godta at i Trondheim sitter det 20 utviklere og jobber med det de mener viktigst for oss til enhver tid.»

Informant 1(Prosjektleder)

God kultur blir også nevnt som en viktig faktor. Informanten beskriver at det er viktig med en kultur der de ansatte tar ansvar for sitt arbeid og ikke legger det over på andre. Dersom man ikke

tar ansvar over sitt arbeid, vil det være vanskelig å implementere DevOps. Informanten forklarer videre at en god deploy-pipeline også er viktig:

> *«Jeg vil si det er viktig å ha en kultur hvor man kan ta ansvar for sitt egent arbeid. Det er det viktigste. Om man fraskriver seg ansvaret til noen andre, så går det dårligere. Det er ditt ansvar å forstå hva du skal gjøre, gjøre det riktig og få nok tilbakemeldinger til å kunne stå for resultatet og si at «dette er bra». Så deploye det videre etterpå. Om det er ansvarsfraskrivelsekultur i en bedrift, så vil jeg si at det er vanskelig å innføre DevOps. Det er jo det hele DevOps er, å ta ansvar for hele tingen. En annen faktor er en grei deploy-pipeline, eller ha en klar vei fra specc til release. Vite hva du må gå gjennom, ha klare definerte steg: du skal plukke fra backloggen, lage branch med feature, du skal teste og release også videre. Slik at det er enkelt for meg å vite hva jeg skal gjøre, og det er enkelt for meg å vite hva de andre gjør og hvor de er i prosessen. Dermed kan jeg bistå dem, eller de kan bistå meg om nødvendig.»*
>
> Informant 3 (utvikler)

Under intervjuene ble det diskutert hvordan automatisering av prosesser er en del av DevOps. Informant 4 jobber som en konsulent opp mot lønnssystemet i bedriften. Dette systemet skal fases ut og migreres over til en ny versjon. Informanten er i utgangspunktet positiv til automatisering av prosesser, men ser at kompleksiteten til systemet kan gjøre det vanskelig å gjennomføre. Informanten forklarer at de har mange forskjellige kunder, og noen av dem krever forskjellige regelverk. Kundene har kompliserte måter å gjøre ting på, så det er viktig at man streamliner dette før man begynner å automatisere prosessene:

> *«Man kan ikke ta inn gamle prosesser med problemer og automatisere dem uten å løse problemene først...Vi har mange rutinerte lønnsmedarbeidere som har vært her i mange år og de vet hvor de kan finne ting, de kan stole på at det de gjør er riktig. Det nye er mer ukjent, og det er veldig viktig at når de tar det i bruk så fungerer det slik at det blir en positiv opplevelse....Så det er ikke alltid disse eksterne løsningene er klare, og hvordan skal man da automatisere prosessen hvor man laster ned filer fra disse løsningene? Det er greit når det*

fungerer, og det er mer effektivt. Vi er avhengig av alle systemene vi er koblet opp til, og det er mange.»

Informant 4 (konsulent)

Standardisering av verktøy og fornying av tjenester ble trukket frem som et viktig fokus område for vellykket DevOps:

«Det er viktig med standardiserte tjenester og at ledelsen er med på å faktisk innse at man må gjøre noe fornying av verktøy og rutiner. Det er kanskje de to kritiske tingene, fordi det finnes veldig mange valg å gjøre, men det må gjøres noen valg og det må gjøres noen fornyelser. Det gjelder ikke bare produktutvikling og sånn kontinuerlig forbedring og kontinuerlig iterativ måte å jobbe på, men også prosessen å deploye. Alt er en del av samme snurra egentlig.»

Informant 7 (arkitekt)

En av de siste spørsmålene vi hadde i intervjuguiden handlet om hvordan informantene følte at implementeringen av DevOps hadde gått så langt. Informant 7 forklarte at DevOps er noe man aldri blir ferdig med:

«Det er jo en lang vei å gå man kommer aldri i mål. Innen man kommer frem til det som var målet vil målet være endret til noe annet .... I gamle dager så var det bare en on prem installasjon dvs. kundene installerte i eget hus og så gikk man jo i parallelle egentlig over fra at vi hjalp dem til å gjøre en lokal installasjon til at vi også begynte å tilby det via vårt datasenter. Så har det blitt mer og mer hvor vi tilbyr det på våre datasenter og da havnet mer i DevOps hvor også ressurser fra våre teams gjør teknisk installasjon og deploy og onboarding av kunder, og alt sånt.»

Informant 7 (arkitekt)

Informant 3 mener at implementeringen har gått bra, men tror ikke at det vil bli gjort noe mer arbeid for å jobbe mer DevOps enn det de gjør nå. Informanten mener at det ikke er nødvendig å implementere hele DevOps:

> *«Jeg tror ikke det blir gjort noe arbeid for å endre på måten vi gjør det på nå. ... Å følge den retningen selv om en ikke følger den helt nøyaktig slik som vi gjør her, det fungerer jo greit. En trenger ikke å være så veldig streng med det, for eksempel ha to uker sprint, og det må være sånn og sånn. Da blir det plutselig veldig rigid, men jeg har ikke noe erfaring med veldig rigide oppsett heller.»*
>
> Informant 3 (utvikler)

5 Diskusjon

I dette kapittelet vil resultatene fra kapittel 4 drøftes i sammenheng med teorien som er presentert i kapittel 3. Kapittelet skal prøve å svare på forskningsspørsmålene som ble satt i kapittel 1.1 med spesielt fokus på Norden. Kapittelet vil være delt opp i underkapittel som vil handle om forskningsspørsmålene slik som resultatkapittelet.

5.1 Definisjon av DevOps (F1)

Noe av det som kom fram av intervjuene vi utførte i sammenheng med denne studien var at mange av de vi snakket med uttrykte at DevOps kunne være et vagt begrep der det ikke var så klart hva som faktisk inngår i metoden/filosofien. Dette kan også sees i litteraturen vi har funnet, der en finner definisjoner av DevOps og en definitiv definisjon av DevOps (Smeds, Nybom, & Porres, 2015), (Freeman, 2019). Siden vi har funnet artikler der minst ett av forskningsspørsmålene også har handlet om definisjonen av DevOps, så er det trygt å si at det er mange som vil si at begrepet kan oppfattes som vagt (Lwakatare, et al., 2019). I dette kapittelet vil vi svare på forskningsspørsmål F1: Hvordan defineres begrepet DevOps av forskjellige ansatte i en programvarebedrift?

Selv om begrepet er vagt, så er det mulig å finne definisjoner av begrepet DevOps både i eksisterende litteratur og i studien vi har gjennomført. Det som ligger i selve navnet, er at utviklerne og drifterne er involverte. En av tingene som informantene våre fokuserte på når de skulle definere DevOps var en kultur der samarbeid og økt ansvar var viktige trekk, spesielt mellom grupper som tradisjonelt ikke har samarbeidet spesielt mye sammen. Dette blir også nevnt i teorien vi har funnet om DevOps i Norden, der DevOps innebærer at utviklere får ansvar for prosessene som programvaren går gjennom fra design til levering og økt eierskap (Lwakatare, et al., 2019). I studien vår var det utviklerne og drifterne som samarbeidet mer etter implementeringen av DevOps. I tillegg var konsulentene, eller kundene/brukerne, også mer involvert der de kom med mer tilbakemeldinger. Dette kan også sees i definisjoner av agil, der

kunder er mer involvert med tilbakemeldinger (Fowler & Highsmith, 2001). En annen case-studie fra Norden forklarer at det er vanskelig å endre bedriftskulturen til en kultur som passer med DevOps, noe som betyr at kultur var viktig del av DevOps siden de tok bryet med å endre kulturen (Riungu-Kalliosaari L., 2016). En kultur med delt ansvar blir også nevnt i litteraturen fra den verden utenfor Norden (Rowse & Cohen, 2021). Om vi ser på definisjoner utenfor Norden, så kan vi for eksempel se at kultur er en av de fire hoveddelene av DevOps, sammen med automasjon, målinger og deling (Willis, 2016). Man kan også finne definisjoner av DevOps der det blant annet blir beskrevet som en kulturell bevegelse, der kulturen kjennetegnes med åpen kommunikasjon, respekt, tillit og samordning av insentiver og ansvar (Walls, 2013).

En annen del som informantene nevnte når de prøvde å definere DevOps var automasjon. Automatisering av utviklingsprosesser og prosesser knyttet til leveranser av programvaren ble nevnt. Dette kunne vi også finne i litteraturen, der Willis navngir automasjon som en av hoveddelene til DevOps (Willis, 2016). Automasjon vil være et viktig verktøy i å kunne nå hovedmålet til DevOps som vi fant i case-studiene fra Norden: å levere endringer i programvaren så fort som mulig for å få tilbakemeldinger (Lwakatare, et al., 2019). Automatisering av prosesser vil kunne spare tid, som gjør det enklere å nå dette målet og kan sies å være en sentral del av DevOps fra et produkts start og slutt (Cruzes D.S., 2019).

Noe som var forskjellig fra resultatene våre og litteraturen vi har funnet var målinger. Det var ingen av informantene som nevnte målinger som en av tingene som definerer DevOps. I motsetning til dette, så er målinger en av de fire sentrale delene av DevOps ifølge Willis (Willis, 2016). Målinger blir også nevnt som en av kategoriene som muliggjør DevOps (Luz, Pinto, & Bonifácio, 2019). Ingen av informantene kunne nevne noen spesiell bruk av målinger, enten av de ansatte eller hvor langt bedriften hadde kommet i implementeringen av DevOps. Det eneste som ble nevnt var utviklingssyklusen på 14 dager, og noen frister på prosjekter. Med disse kan bedriften se om de ligger i rute, men det var vanlig at ting ikke ble ferdig til deadlinen i bedriften vi inkluderte i studien vår.

Et ord som gikk igjen i både teorien og i resultatene fra studien vår var «kontinuerlig»: kontinuerlige utgivelser, kontinuerlig drift, kontinuerlige leveranser, kontinuerlig operativ

funksjonsleveranser og mer (Siqueira, Camarinha, Wen, Meirelles, & Kon, 2018), (Fowler & Highsmith, 2001), (Ekbert, Gallardo, Hernantes, & Serrano, 2016). Det er et tydelig fokus på at ting skal skje fort, og at man ikke skal vente lenge på at noe skal skje igjen som for eksempel tilbakemeldinger. Med så mange referanser til dette begrepet, så er det trygt å anta at å gjøre ting kontinuerlig er en sentral del av DevOps.

Det kan se ut som det er to forskjellige fokus når det snakkes om definisjoner av DevOps. Som en av informantene nevnte, så kan man identifisere to overordnede deler som DevOps handler om. Den ene delen er det som handler om automatisering og alt det tekniske, som man kan se i eksempelet på livssyklusen assosiert med DevOps vi har inkludert i teorien, der det sentrale i mange av stegene er automatisering og organisering (Dhaduk, 2022). Det andre involverer kultur og tankesett, der man fra Norden kan se at utviklerne og drifterne samarbeider på områder de tradisjonelt ikke har samarbeidet før (Lwakatare, et al., 2019). Det er derfor ikke urimelig å si at DevOps kan deles i to: DevOps-kultur og teknisk DevOps.

Å lage en Nordisk definisjon av DevOps blir vanskelig siden det ikke eksisterer så mye litteratur om en tydelig definisjon, men det er mulig å se noen fellestrekk i resultatene fra studien vår og teorien. Både litteraturen og resultatene er tydelige på at DevOps betyr utvidet ansvar for de ansatte, der utviklerne må arbeide med drift i tankene, og drifterne må inn i arbeidsprosessene tidligere (Lwakatare, et al., 2019). Det er heller ikke bare utvikling og drift som går inn under DevOps. I studien vår fant vi at konsulentene, de som bruker programvaren, også må gjøre ting annerledes med tanke på tilbakemeldinger selv om de ikke nødvendigvis er så opptatt av hva DevOps betyr. I Figur 13, som er hentet utenfor Norden, kan vi se hvor viktig kultur er, og det samme sees i CAMS-modellen, men tekniske faktorer som automatisering og arbeidsprosesser blir også gitt mye oppmerksomhet (Willis, 2016). Det kan argumenteres her at det blir lagt mer vekt på kulturen med blant annet mer ansvar og deling i Norden siden dette var noe mesteparten av informantene i studien vår nevnte, og det ble lagt ekstra vekt på dette i definisjonen fra Norden (Lwakatare, et al., 2019). I det neste kapittelet vil vi gå gjennom mer konkrete konsekvenser av DevOps.

Figur 13: Forhold mellom kategorier (Luz, Pinto, & Bonifácio, 2019)

5.2 Konsekvensene av å innføre DevOps (F2)

Implementering av DevOps i en bedrift vil føre til endringer og konsekvenser for forskjellige deler av organisasjonen som implementerer det. Disse kan rangere fra positive endringer som blir tatt godt imot av alle ansatte til upopulære endringer som fører til stor motstand blant alle involverte partier. I dette kapittelet vil vi svare på forskningsspørsmål F2: Hva er konsekvensene av å implementere DevOps i en programvarebedrift? For å svare på forskningsspørsmålet vil vi først gå gjennom noen av temaene som er sentrale i DevOps og hvilke konsekvenser vi har funnet innen dem. Disse temaene er kultur, fordeler, ulemper/utfordringer og verktøy. Mot slutten av kapittelet vil vi gå gjennom underforskningsspørsmålene for å gi litt mer innsikt i holdninger til DevOps og råd til implementering.

5.2.1 Kultur

En av hoved konsekvensene av å innføre DevOps, er et skifte i organisasjons kulturen. Fra det første kapittel kan vi se at kultur er en viktig del av DevOps, og det vil være naturlig at bedriftskulturen blir påvirket av DevOps. Men det er ikke alltid at slike ting oppfører seg på samme måte når de blir implementert i praksis som de blir beskrevet i litteratur, som er en av grunnene til at vi utførte denne studien.

Ett tydelig eksempel på en endring i kulturen som vi fant i litteraturen var at det å finne feil ble sett på som positivt når det før ble sett på som negativt (Cruzes D.S., 2019). Det ble også hevdet at DevOps-kulturen har ført til positive endringer i hele bedriften, ikke bare i selve avdelingene som utvikler og drifter programvaren (Lwakatare, et al., 2019). I studien vår fant vi også ut at

DevOps-kulturen hadde påvirket ansatte i bedriften som ikke jobbet på utvikling eller på teknisk. Konsulentene vi snakket med hadde en annerledes måte å jobbe på med DevOps. De bemerket at de samarbeidet mer med de ansatte på utvikling ved å blant annet gi tilbakemeldinger på systemet, og brukte merkbart mer tid på dette. Fra litteraturen kunne vi også se at det ble mer kommunikasjon og tilfeller av ansatte gav hjelp til andre utenfor avdelingen de selv jobbet for, og interaksjonene var mer uformelle (Rowse & Cohen, 2021), (Lwakatare, et al., 2019). Uformell kommunikasjon ble også brukt for å beskrive forholdet mellom de ansatte ved bedriften vi utførte studien hos. Det var litt forskjellige meninger om hva som var grunnen til dette. En informant mente at dette ikke var noe spesielt nytt, men at bedriftskulturen var naturlig slik. Fra en annen informant fikk vi vite at miljøet kunne beskrives som litt stivt før de ansatte flere nye, yngre ansatte. Så det er litt forskjellige meninger om hva som forårsaket den uformelle kommunikasjonen mellom de ansatte, men det virker ikke som DevOps var den direkte kilden til dette i denne bedriften. For eksempel så ble koronatiltakene navngitt i studien vår som grunnen til at det føltes mer akseptabelt å jobbe hjemmefra.

Informantene våre kunne fortelle at de hadde opplevd flere av trekkene ved DevOps-kultur, som at ansatte følte høyere grad av ansvar og var mer involverte i flere deler av livssyklusen til programvaren. De sa at bedriftskulturen nå passet med kulturen som vi beskrev tidligere i diskusjonen, og at den ikke nødvendigvis hadde blitt slik de siste årene.

5.2.2 Fordeler

Det fremkommer både i forskningslitteraturen og i resultatene at DevOps-praksis fører til en rekke fordeler. I litteraturen blir det forklart at i en bedrift er det viktig at de ansatte er motiverte til å utføre sine oppgaver på en god måte. I en av case studiene forklarer Senapathi, Buchan, & Osman (2018) at man tydelig kan se at teams som jobber med DevOps er motiverte og mer engasjerte i arbeidet sitt. Dette skyldes i følge Rowse & Cohen (2021) at DevOps krever en endring i kultur som retter seg mot ansvar og involvering. I resultatene blir begge disse referert til som fordeler med DevOps. Ordet ansvar er noe som ofte ble nevnt under intervjuene. En av informantene forklarer at DevOps handler om at du har en ansvarslinje hele veien gjennom. Det vil si at alle som er i leveranselinjen er ansvarlige for at sluttproduktet kommer frem til brukerne og at det fungerer slik det skal. Tradisjonelt hadde man distribuerte grupper som utførte

funksjoner separat, mens DevOps legger til rette for at det skal kunne kjøres kryssfunksjonelle teams som jobber med kontinuerlig leveranser. Det økte ansvaret blant de ansatte gir dem muligheten til å i større grad forme sin egen arbeidshverdag. Flere informanter støttet dette utsagnet og beskrev at de opplever at folk tar mer ansvar for sin kode nå enn det de gjorde før. Håpet er at DevOpsstrukturen skal gi de ansatte mer ansvar og frihet til å bestemme selv hvordan de skal påvirke utviklingen. Informantene forklarer at man får litt mer muligheten til å være med å bestemme på mange måter, man får ikke bare kastet oppgaver etter seg.

Forfatteren (Senapathi, Buchan, & Osman, 2018) forklarer i sin case-studie at DevOps involverer de ansatte mer og at de får en bedre oversikt over alle prosessene slik at de får se ringvirkningene av det de jobber med. Dette stemmer godt med hva som ble sagt av informantene. De trakk frem at de har fått bedre oversikt i prosessene på tvers av avdelingene og ser på dette som en tydelig fordel med DevOps. En informant forteller at utviklere kan nå bedriftsavdelingen om å kjøre en spørring for å se data som ligger i ulike kundebaser. Før var ikke dette mulig da alt var "on premise" og avdelingene jobbet hver for seg. Muliggjøring av et tettere samarbeid mellom avdelingene er også en fordel med DevOps. Tettere samarbeid gjør det vesentlig enklere å få innsikt og en helhetlig oversikt over prosessene knyttet til utvikling- og driftsavdelingene. En informant forteller at det totalt sett er en vinning å være deltagende i hele prosessen, både for egen og andre sin del. I litteraturen sier Siqueira, Camarinha, Wen, Meirelles, & Kon (2018) at noe som kan føre til økt samarbeid er at de ansatte blir mer engasjert i hele prosessen, på grunn av kontinuerlige utgivelser som kan føre til at de ansatte føler seg ansvarlige for hva som blir sendt videre. Dette er i tråd med svar gitt av informantene. De forklarer at kontinuerligeutgivelser fører til bedre flyt i levering av produkt og ansees som en fordel muliggjort av DevOps.

Dette er fordi DevOps handler om å kontinuerlig opprettholde kontinuitet og optimalisere automatisering. I resultatene kommer det frem at bedre flyt i produktleveranser skyldes at man tar mer kontroll og ansvar for sitt eget arbeid, helt fra start og frem til levering. Ved at alle jobber med en slik tilnærming vil man kunne opprettholde kontinuitet og skape flyt i leveransene. Informantene som jobber som utvikler forklarer at de nå gjør mer enn bare koding. Det har blitt mer sånn at utviklere bygger kode, men har i tillegg tilgang til et par kundemiljøer. Der kan de gå

inn og sjekke på ekte reell data hvordan koden fungerer. På denne måten kan utviklere selv sjekke om den nåværende koden fikset en tidligere bug eller om de nye egenskapene fungerer slik de skal. Informantene forklarer at de har begynt å samarbeide mer med kundene der de har samtaler og kan stille spørsmål. Dette mener de er mye bedre enn å kode en ting, for å legge det fra seg og begynne på neste oppgave.

Resultatene viser at DevOps har gitt informantene generelt bedre oversikt over produktene. Dette har videre gitt fordelene ved at man ikke trenger å vente lenge på små endringer av produktet. Informantene forklarer at i det nye systemet så er det enklere å bryte ned ting og komme med små endringer, slik at en ikke trenger å vente på en stor release. Det har blitt mulig å fikse feil fortløpende i stedet for to til tre ganger i året slik det var med det gamle "on premise" systemet. En av informantene forteller at når en bug oppdages kan de ha en fiks ute i produksjon på en time.

Det at det tar kortere tid å utføre en endring har videre ført til fordelen med at informantene føler mindre stress rundt selve utgivelsene. En av informantene forteller at det er laverer puls på alle og at stressmomentet er mindre fordi om noe går galt, noe det ofte gjør, så er det ikke lenger en stor kilde til stress da man har muligheten til å rette opp feilen meget raskt og effektivt. Dette har gitt alle lavere skuldre over hele linjen. I litteraturen viser det seg at Rowse & Cohen (2021) ikke er så sikker på om dette stemmer. De mener at det eksisterer forskjellige meninger om hvorvidt implementasjon av DevOps fører til mindre stress. I deres forskning kom de frem til at det varierer, for noen er det merkbart forskjell og andre mener at det ikke er det. Case-studiene fra norden samsvarer med det informantene føler om stress. Lwakatare, et al. (2019) forklarer at deres resultater viser at de ansatte opplever mindre stress etter implementasjonen av DevOps. Det blir også hevdet at de ansattes helse også ble bedre på grunn av DevOps da det ble mindre angst rundt større utgivelser.

Fritt valg av verktøy blir trukket frem som en fordel med DevOps. Jones, Noppen, & Lettice (2016) forklarer at utviklere liker som oftest å utforske nye teknologier og måter å jobbe på. DevOps trenger både nye verktøy og metoder for å fungere optimalt, noe som gir gode forutsetninger for at utviklerne vil trives å jobbe på denne måten. Utifra resultatene stemmer

dette godt med hva informantene våres mener. De forklarte at utviklerne har muligheten til å ta i bruk nye verktøy eller holde seg til de verktøyene de er vant til å bruke. Informanten forklarer at de har blitt mere fleksibel og åpen på verktøy nå enn vi var før. Dette skyldes ikke bare at de har endret på prosesser, men også at det nå er mye mer verktøy tilgjengelig.

En annen viktig fordel med DevOps er automatisering av prosesser. Slik det ble nevnt litt tidligere i diskusjonen handler DevOps om å kontinuerlig opprettholde kontinuitet og optimalisere automatisering. I forskningslitteraturen forklarer Cruzes D.S. (2019) at automatisering blir sett på som en sentral del av DevOps gjennom hele livet til et produkt, fra start til slutt. Noen av informantene forteller at når de tenker på DevOps så er det automatisering av oppgaver, spesielt tettere på deploy de tenker på. Det blir nevnt at automatisering av deploy skulle nesten bare vært en knapp som gjør det enda lettere for prosjektleder å få det ut på produksjon og hatt det automatisert i bakgrunnen. DevOps og automatisering kan altså gi konsistens på tvers av gjentagende prosesser ved å konfigurere et automatiseringsverktøy og fjerne trusler som menneskelige feil. I tillegg til dette kan de øke hastigheten til teamet fra kodeintegrasjon til distribusjon. Det skaper også et mye mer pålitelig og stabilt system på grunn av det forbedrede samarbeidet mellom utvikler og driftsavdelingen. Automatisering reduserer også feilkommunikasjon da prosesser som før ble kjørt manuelt kjøres nå automatisk med mindre behov for personlig kommunikasjon.

Forskningslitteraturen trekker frem at når en først har fått til automatisering av forskjellige prosesser, så føler de som utfører prosessene at de sparer mye tid (Lwakatare, et al., 2019). Dette stemmer godt med hva informantene mener om automatisering. Informantene mener at en fordel er at man har blitt mere effektive på noen prosesser, men mangler fortsatt automatisering rundt utgivelse. De trekker frem at utgivelsesprosessen er fortsatt meget manuell, for hver utgivelse sitter noen og kikker gjennom casene som skal gjennom og venter på andre deler av koden. Så må en person lage en branch, og bygger det på en byggeserver og deployer det. Automatisering av slike prosesser vil hjelpe bedrifter nå kontinuerlig deployfasen der man kan deploye mange ganger per dag.

5.2.3 Ulemper/utfordringer

Selv om DevOps gir bedrifter store fordeler vil det også være noen utfordringer knyttet implementeringen. En av utfordringene som blir trukket frem av informantene er legacy systemer. De forklarer at legacy systemer kan gjøre implementeringen utfordrende da de ofte ikke støtter eller er kompatible med DevOps. Legacy systemer krever vanligvis at oppdateringen eller endringer på systemet gjøres manuelt og fysisk hos kunden. Dersom legacy systemet ikke er kompatibel med DevOps kan bedrifter blir nødt til å tilby flere versjoner av programvaren sin til kunder. Overgangsfasen til et nytt skybasert system for eksempel, er også meget tidkrevende. Bedrifter må derfor ofte fortsette å drifte legacy systemet frem til kunder og data er migrert over til det nye. Dette trekker arbeidskraft og ressurser bort fra det nye systemet. En av informantene forklarer at legacy systemet de bruker er eldgammelt, men ligger til grunn for alt de gjør. De har pensjonister som jobber for dem for å refaktorere og som er kjent med koden fra før av. Det er en del ting som må skrives om og fornyes og gjøres tilgjengelig for den nye løsningen og det krever mye tid og arbeid. En annen informant forklarer at de oppdaterer og holder liv i legacy systemet rett og slett for å holde kundene synkroniserte da alle kundene som kjører den eldre versjonen av systemet skal over på det nye. For å hindre at det løfter blir så stort får de muligheten til å ta i bruk en del av det nye på den gamle plattformen. Informantene forklarer videre at de i overgangsfasen til det nye systemet begynte å utgi oppdateringer litt oftere enn 2-3 ganger i året før de på en kort tidsperiode gikk over til hver 14 dag som samsvarer i større grad med DevOps. Informantene forklarer at kundene har i overgangsfasen vært delt. De forteller at noen har kun kjørt det eldre Windows systemet og noen har kjørt en kombinasjon av det nye skybaserte Web systemet og gamle Windows systemet, som førte til at kundene ble oppgradert i 2 grupper. En for dem som kjører web systemet med utgivelse hver 14 dag og kundene som bare kjører Windows versjonen med oppdateringer en eller 2 ganger i kvartalet. I forskningslitteraturen fra case studiene gjort i Norden blir også legacy systemer ansett som en utfordring for implementasjonen av DevOps. På lik linje med informantene mener Lwakatare, et al. (2016) at utfordringen oppstår fordi slike systemer ofte er lokaliserte hos kundene og det blir vanskelig å oppdatere dem kontinuerlig i forhold til nye systemer, som kan legges inn på en skyløsning. I den samme casestudien blir det også trukket frem at kunder kan ha ulike versjoner av den samme programvaren, noe som gjør det ekstra vanskelig å levere oppdateringer raskt, nettopp fordi det

er mange forskjellige versjoner å holde kompatible (Lwakatare, et al., 2016). Tatt resultatene og forskningslitteraturen i betraktning ser det ut som det er enighet i Norden om at legacy systemer skaper utfordringer, men vi vil tro at dette gjelder globalt.

En annen utfordring med implementasjonen av DevOps er at det kan være vanskelig å bryte ut av gamle rollemønstre. Programvarebedrifter har tradisjonelt hatt separate siloer der utviklere og driftere jobbet adskilt. En av informantene som jobber som utvikler forklarer at utviklere har vært vant til å kode noe ferdig, sende det videre og så begynne på noe nytt. Utviklere har vært fornøyde med den tankegangen om at man lager en ting og så er man ferdig med det. Man får det ut av hodet slik at man skal være klar for nye ting. Informanten forklarer at tankegangen har blitt annerledes med DevOps. Nå er det slik at man ikke har lagd noe før den faktisk er i bruk av noen og at den fungerer slik det skal. Informanten forklares at måten å tenke på der man har ansvaret hele veien oppover, den sitter langt inne for mange utviklere. I forskningslitteraturen blir dette også diskutert. I litteraturen blir det diskutert hvordan det endrede rollemønsteret mellom utviklerne og driftere skaper utfordringer rundt kompetanse. Senapathi, Buchan, & Osman (2018) forklarer at få personer har kompetansen som er nødvendig. Dette er kunnskap og kompetanse innen felt som infrastruktur, overvåkning og koding. Når man ikke jobbet DevOps trengte ansatte kompetanse bare som utvikler eller drifter. Nå som hele teamet skal ha kunnskap om hele prosessen et produkt går gjennom fra start til slutt blir det vanskelig å finne ansatte som faktisk har den nødvendige kunnskapen og kompetansen (Riungu-Kalliosaari L., 2016). I litteraturen forklares det videre at det å skaffe ansatte med DevOps-kompetanse er den største utfordringen forbundet med DevOps (Rowse & Cohen, 2021).

I litteraturen knyttet til case studiene i Norden blir også rollemønster diskutert som en utfordring. Problemer med endring av rollemønstre kobles til endringen av organisasjonskulturen som bringes av DevOps (Lwakatare, et al., 2019). Organisasjonskultur kan i seg regnes som en utfordring med implementering av DevOps. Lwakatare, et al. (2019) og Riungu-Kalliosaari L. (2016) beskriver i sine studier at en DevOps kultur består av tillit, inklusivitet, samarbeid og åpenhet mot læring. Implementeringen av DevOps kan bli vanskelig når den originale organisasjonskulturen kommer i konflikt med DevOps kulturen. Man kan ikke forvente at man kan sette utviklere og driftere sammen i et team og forvente at de jobbe på en ny måte uten

problemer. Det kan være krevende for utviklere og driftere å fundamentalt endre på deres rollemønster. Riungu-Kalliosaari L. (2016) trekker frem i studiet at dette gjelder spesielt om de har jobbet på en spesiell måte over lengre tid.

Motstand mot endring er når en person eller flere som blir påvirket av endring ikke er positivt innstilt til at det skal gjøres endringer. Dette er en utfordring som også kan skje når DevOps blir implementert. Slik vi har diskutert tidligere vil DevOps skape utfordringer knyttet til endringer i rollemønstre og organisasjonskulturen. Noen ansatte vil skape motstand ved å være negative til endring og andre kan indirekte gi motstand ved at de for eksempel har vanskeligheter med å endre på rollemønster eller at man mangler relevant kompetanse eller forståelse for endringen. Noe av motstanden som ble trukket frem av informantene var skepsisen til den nye utvikling syklusen. Tidligere var det kun 3 utgivelser i året, da hadde man gjerne samlet opp 4 eller 5 måneder med databaseendringer som skulle inn i den nye versjonen. Det å gå over til en 14 dagers syklus der man skal utgi små biter med oppdatert kode skapte skeptiske reaksjoner i begynnelsen. Det nye systemet som bedriften har tatt i bruk har også vært en kilde til motstand. Informantene forteller at det har vært personer som har nektet å bruke det nye systemet og vært meget negative til endringen. Disse personene tvinges nå til å ta i bruk det nye systemet da man har begynt å stenge funksjonalitet i det gamle systemet. Forskningslitteraturen trekker videre frem at motstand kan komme fra at ansatte kun ønsker å jobbe med det som inngår i deres fagfelt, for eksempel at en utvikler skal skrive kode og en drifter skal ikke skrive kode. Motstand kan også komme ved at man setter ulike ansatte med forskjellig kompetanse sammen i en gruppe og forventer at de skal jobbe bra sammen uten videre, (Senapathi, Buchan, & Osman, 2018), (Jones, Noppen, & Lettice, 2016). Dette stemmer godt med hva som ble sagt av informantene, da de forklarte at det ikke var alle som var like interesserte i å ta i bruk alt som har blitt implementert av arbeidsmåter. Dette gjelder spesielt de eldre karene i selskapet som har jobbet med det gamle systemet i mange år. De har for eksempel vist motstand ved å melde seg ut av standup møter. De ønsker egentlig bare å jobbe i fred med den gamle Windows klienten. Informantene forteller at det samme gjelder den tekniske avdelingen. Utviklingsavdelingen har også prøvd å få med den tekniske avdelingen på standup møtene, men det har de ikke vist interesse for. Det diskuteres videre i litteraturen at mange endringer på en gang kan føre til indirekte motstand ved at ansatte blir usikre rundt hvem som har ansvar for ulike oppgaver som

videre fører til at oppgaver ikke blir gjort, da man tror at det er noen andre som har ansvaret. Litteraturen trekker også frem at motstand kan komme fra ledelsen. En av studiene forklarer at det finnes eksempler der ansatte har forsøkt å implementere DevOps, men ledelsen har ikke etterkommet forespørsler om nødvendige ressurser som trengs for å kunne gjennomføre DevOps-aktiviteter (Jones, Noppen, & Lettice, 2016).

En sentral del av DevOps er at utvikle og driftere skal kunne kommunisere og jobbe sammen i samlokaliserte teams. Dersom man ikke får til å kommunisere med samlokalisere teams på en god måte vil det skape store utfordringer for implementasjonen av DevOps. For internasjonale selskaper med kontorer globalt kan dette være krevende. Ifølge forskningslitteraturen vil dette påvirke kommunikasjonen mellom de som ikke deler samme lokaler, blant annet vil mengden av kommunikasjon mellom disse ansatte minke (Diel, Marczak, & Cruzes, 2016). En av studiene i litteraturen sier at man får mye bedre kommunikasjon mellom teammedlemmene når man sitter sammen under samme tak. Der kan man kommunisere mer naturlig og misforståelser blir lettere oppklart (Senapathi, Buchan, & Osman, 2018). Informantene hadde litt ulike meninger om kommunikasjon kan være en utfordring dersom man ikke sitter samlokalisert. Mye har endret seg i forbindelse med hvordan man jobber og kommuniserer med kollegaer hjemmefra. En informant forteller at han ikke syntes kommunikasjon er en stor utfordring når man jobber borte fra kontoret. Han forklarer at det å sitte sammen handler mer om at man er tilgjengelig for hverandre når man trenger det. Informanten forteller videre at det finnes gode nok kommunikasjonsveier nå slik at det er sjeldent utfordringer med kommunikasjon dersom man ikke sitter fysisk sammen. En annen informant forteller at det sikkert hadde vært enklere om alle var under samme tak, men at hjemmekontor fungerer godt. Grunnen til dette er at man ofte blir forstyrret av folk som har spørsmål. Når man jobber hjemmefra på en kommunikasjonskanal som Teams blir man ikke forstyrret like mye. Informanten forklarer videre at når folk jobber hjemmefra kan man kjøre planlagte møter på teams der man er tilgjengelige for hverandre hvis noen vil komme med innspill eller spørsmål. På denne måten mener informanten at man kan opprettholde god kommunikasjon og slippe unna unødvendig kommunikasjon som er forstyrrende. Andre informanter syntes det fungerer greit å jobbe hjemmefra med tanke på kommunikasjon, men synes det er lettere å jobbe sammen fysisk enn å sitte på teams. Informantene forteller at når man sitter sammen kan man bare snu seg dersom det er noe.

En annen utfordring med DevOps er mangel på tid. Med utviklingssyklus på 14 dager får visse ansatte mye å jobbe med. Det er mye som skal gjøres mellom hver utgivelse og da må det gjøres prioriteringer for å komme i mål. En av informantene forteller at de har en god størrelse på utviklingsteamet til å være et norsk programvareselskap, men at de ikke er mange nok. Slik DevOps-utgivelsessyklusen er, har de rett og slett ikke tid til å levere det de skal. Informanten forklarer at man må finne ut hvordan man skal prioritere tiden til de som sitter på utvikling og hva de faktisk har tid til å gjøre eller ikke. Dette er meget vanskelig da de ansatte alltid har lyst til å gjøre mer enn det de har tid til. Før DevOps hadde man 3 ukers kode freeze der man hadde en tester som satt og testet jevnlig hele tiden. Etter hvert ble utviklerne også med på testing. Det ble brukt mye tid på dette og er noe man ikke har tid til nå med en slik 14 dagers syklus. Informanten forklarer at nå handler det om å sjekke det viktigste.

Forskningslitteraturen fra Norden forklarer at automasjon blir sett på som en sentral del av DevOps gjennom hele livet til et produkt, fra start til slutt (Cruzes D.S., 2019). Utforinger med implementasjon av tilnærmingen kan også komme fra sentrale deler av DevOps.
I litteraturen forklares det at dersom man automatiserer prosesser som testing så vil man møte på flere utfordringer. Det hevdes blant annet at manuelle tester vil i motsetning til automatiserte gi mer innsikt i programvaren og de som bruker den. Automatiserte tester kan i tillegg bli utdaterte og inneholde feil da de blir kodet. Å lage automatiserte tester er også kompliserte å lage og det kan derfor være meget krevende å dekke alle behovene. En av informantene forteller at de har fått innvilget å ansette noen til å automatisere testingsprosessen hos bedriften. Utfordringer er slik det blir nevnt i litteraturen at det er meget vanskelig å lage tester som dekker alle behovene. Informanten forteller at de fortsatt ikke har funnet noen med kompetansen til å ta for seg denne jobben.

5.2.4 Verktøy

I forskningslitteraturen forklares det at verktøy er nødvendig for å automatisere og implementere DevOps. Kvalitetsleveranser med kort syklustid trenger en høy grad av automatisering. Så, å velge riktige verktøy for miljøet eller prosjektet er viktig når man flytter over til DevOps (Ekbert, Gallardo, Hernantes, & Serrano, 2016). I CAMS modellen blir også verktøy trukket inn under

automatisering. Det forklares at verktøy kan sette sammen en automatiserings infrastruktur for DevOps med verktøy som kan håndterer administrasjonsutgivelse, klargjøring, konfigurasjon administrasjon, systemintegrasjon, overvåking og kontroll. Dette er viktige brikker når det bygges en DevOps infrastruktur (Willis, 2016). Under forklaringen av DevOps livssyklus sløyfen blir verktøy også presentert som en sentral del av DevOps. Den viser hvordan den samarbeidende og iterative tilnærmingen er bygget opp av verktøy og teknologistabler for hvert trinn i sløyfen (Dhaduk, 2022). Case studiene fra forskningslitteraturen tar også for seg verktøy. Jones, Noppen, & Lettice (2016) forklarer hvordan utviklere liker generelt å utforske nye teknologier og måter å arbeide på og at de derfor vil ha en forutsetning for å like å jobbe med en DevOps tilnærming. Dette er fordi bruk av nye verktøy og metoder er sentralt for at DevOps skal fungere. Case-studien utført av Senapathi, Buchan, & Osman, (2018) diskuterer hvordan verktøy også kan skape utfordringer med implementering av DevOps. Nye verktøy og metoder som blir innført i forbindelse med implementasjonen vil kontinuerlig endre på kunnskapen og kompetansen som er nødvendig for å gjennomføre DevOps prosessene.

Informantene hadde også en del å si om verktøy. Siden bruk av verktøy virker å være en stor del av DevOps filosofien ønsket vi å høre om de har merket noen endringer i bruk av verktøy med det nye skybaserte systemet sammenlignet med det eldre systemet som fases ut. Svarene vi fikk fra informantene varierte fra ingen opplevde endringer til store endringer i verktøy. En av informantene mente at det ikke hadde vært noen endringer i verktøy da de har vært en Microsoft-partner i mange år. De har brukt Microsoft produkter helt siden Team Foundation server (TFS). Da ble det brukt som Source og tracking punkt. TFS ble etter hvert en skyløsning og endret navnet til Azure DevOps. Informanten forklarer videre at endringen må i så fall være at de gikk fra å være en klient til å være en skybasert web-løsning. En annen informant forklarer også at det har vært lite endringer i verktøy, kanskje noen semantiske endringer i forbindelse med at de har blitt kjøpt opp av et europeisk selskap. De fleste informantene trekker frem overgangen til den nye skyløsningen regnes som den største endringen i verktøy. Nå utvikles, leveres og driftes tjenestene via skyløsningen, slik var det ikke før. En informant forklarer at før var alt in house, produktet var installert hos kundene og noen hos oss. På den tiden var det vanskelig å vite hvilken tilstand systemet var i.

En annen informant mener at det har vært meget store endringer i verktøy. Man kan ikke sammenligne det gamle med det nye systemet bortsett fra at mye av den samme funksjonaliteten ligger der, men at det fungerer på en annen måte og det er mye ekstra funksjonalitet. Andre informanter trekker frem at endringen av verktøy i forbindelse med overgangen til skyløsning har gitt bedriften muligheten til å levere programvare som en tjeneste og ikke bare som et produkt. Det forklares av en informant at kundene nå abonnerer på produktet som en tjeneste. De leverer ikke produkt, De leverer tjenester. Slik det ble diskutert i forskningslitteraturen blir også automatisering nevnt av informantene. De forklarer at det nye systemet er lagt til rette for automatisering av prosesser, men kan kjøres manuelt dersom det er ønskelig. En av informantene som jobber med lønnssystemet til bedriften forteller at hele UI er totalforandret og at ingenting ser ut som før. Snarveier er endret, nye prosesser er på plass der flere av prosessene har blitt automatisert. Informanten forteller at den største endringen er verktøy som er knyttet til automatisering.

Når vi diskuterte bruk av verktøy med Informantene, trakk de også frem friheten til å velge verktøy selv. De forklarer at før bedriften jobbet med DevOps var de i større grad låst i alt av verktøy. Mye av dette skyldes at prosessene var tyngre, men også tiden de levde i. Før de gikk over til et skybasert system var det et vesentlig mindre utvalg av verktøy innen infrastrukturen de jobbet i. En av informantene forteller oss at bedriften har i lenger tid vært en Microsoft-partner og at de har nå også fått en mer pragmatisk tilordning til verktøy. Før var de meget proteksjonistiske krevde at man kjørte Microsoft hele veien fra programvare til maskinvare. Nå som flere bedrifter satser på DevOps og skybaserte løsninger har de beveget seg bort fra å selge lisenser til for eksempel Visual Studio og heller til å tjene penger på å selge tidsbruk av skybaserte tjenestene deres, slik som Azure. En av informantene forteller at for dem så har dette vært meget positivt da de har fått friheten til å endre på hvilke verktøy de bruker. Dette stemmer godt med hva Jones, Noppen, & Lettice (2016) forteller om utviklere som liker å utforske nye teknologier og måter å jobbe på. Informantene forteller videre at ikke alle utviklere bruker de samme verktøyene lenger. Alle står helt fritt til å bruke det de vil ha. Noen sitter i Rider, noen sitter i Visual Studio og noen bruker Command Line.

5.2.5 Hva de ansatte syntes om DevOps (F2.1)

En del av studien vår handler om hva noen som bruker DevOps faktisk tenker om det, og i dette kapittelet vil vi svare på underforskningsspørsmål F2.1: Hva syntes de ansatte om DevOps i en bedrift som bruker det?

Informantene som deltok i studien vår, var enstemmige om at de selv likte DevOps. Dette var det eneste som alle informantene var enige om. Noen av dem kunne fortelle at det var noen andre i bedriften som ikke var fornøyde med metoden, men ut ifra det informantene fortalte oss så var flertallet fornøyde med DevOps. Dette er ikke uhørt fra andre bedrifter, og man kan finne eksempler på at de ansatte blir gladere med DevOps, noe som ikke hadde skjedd om de ikke likte denne måten å jobbe på (Senapathi, Buchan, & Osman, 2018). Man kan også se i tidligere underkapitler at det er mulig å finne flere fordeler med DevOps, noe som vil gjøre det enklere for de som bruker det å like det. Vi så dette i studien, der informantene gjerne la til en av fordelene som grunn til at de likte DevOps. Det er ikke veldig overraskende at utviklere liker DevOps, for nye måter å jobbe på i tillegg til nye verktøy er noe som utviklere generelt liker (Jones, Noppen, & Lettice, 2016). Dette er ikke nødvendigvis universelt, og man kan finne ansatte som ikke ønsker å måtte tilpasse seg nye verktøy og arbeidsmetoder, de vil bare gjøre jobben slik de alltid har gjort det (Senapathi, Buchan, & Osman, 2018) (Jones, Noppen, & Lettice, 2016). I studien vår fant vi også tilfeller av dette, der informantene kunne beskrive at noen av den eldre generasjonen ikke var like interesserte i å delta i alle DevOps-relaterte aktiviteter. Det var også motstand mot det nye systemet som ble introdusert, og dette var fordi at mye av kompetansen/kunnskapen ansatte hadde om det gamle systemet kunne ikke overføres til det nye systemet. Det er ikke vanskelig å tenke seg at det kan oppstå negative assosiasjoner til noe som gjør kompetanse man har opparbeidet seg over flere år overflødig. Fra litteraturen kan vi også se at ansatte i programvarebedrifter ikke nødvendigvis tror at DevOps-prinsipper kan være med på å løse alle utfordringer. Kontinuerlig praksiser trekkes frem som et eksempel, som ikke kan løse problemer knyttet til legacy-kode (Kuusinen, et al., 2018).

5.2.6 Det som er nødvendig for å implementere DevOps på en god måte (F2.2)

Som vi har gått gjennom allerede i diskusjonsdelen av dokumentet, så eksisterer det en del fordeler og ulemper. Det er ikke veldig klart hvordan man oppnår fordelene forbundet med DevOps, men samtidig unngå ulempene/utfordringene. I dette kapittelet vil vi svare på forskningsspørsmål F2.2: Hva er nødvendig for å implementere DevOps på en god måte i en programvarebedrift?

Mye av det vi har nevnt av fordeler forbundet med DevOps, i tillegg til den ene delen av definisjonen av DevOps, handler om verktøy og rutiner. Derfor er det ikke overraskende at dette er noe som ble nevnt når vi spurte informantene våre om implementering av DevOps. Verktøy blir nevnt i alle stegene til eksempelet på en DevOps livssyklus som vi fant, og det vil være vanskelig å bruke en slik livssyklus om man ikke har disse verktøyene tilgjengelig (Dhaduk, 2022). Det blir for eksempel vanskelig å kontinuerlig levere kode om man ikke har på plass infrastrukturen som gjør det mulig å hyppig oppdatere programvare.

Kommunikasjon er et tema som blir nevnt i både resultatene våre og i teorien vi har samlet inn. Fra litteraturen kan vi se at teams som ikke sitter samlokalisert vil kommunisere mye mindre, og kommunikasjonen som faktisk blir gjennomført er mindre naturlig i tillegg til at det blir lettere å misforstå hverandre (Senapathi, Buchan, & Osman, 2018), (Diel, Marczak, & Cruzes, 2016). En av kategoriene som er viktig i implementering av DevOps som blir trukket fram i litteraturen er åpenhet og deling, noe som er avhengig av god kommunikasjon for å fungere (Luz, Pinto, & Bonifácio, 2019). I bedriften som vi valgte å bruke i vår case-studie har de noen personer som jobber sammen med kontoret i Trondheim, men de sitter på et kontor på Østlandet. Informantene våre mente at dette ikke var et stort problem, men at det hadde vært bedre om alle satt på det samme kontoret. De sammenlignet kommunikasjonen mellom noen som kun var lokalisert i Trondheim og mellom noen som satt in Trondheim med noen på kontoret på Østlandet, og kommunikasjonen mellom de som var samlokalisert ble beskrevet som mer praktisk, lettere og mindre tidskrevende enn kommunikasjonen med noen på det andre kontoret. Informantene mente at forskjellen ikke var fryktelig stor, men at den var merkbar. Artiklene vi har inkludert har brukt caser der kommunikasjonen skjedde mellom personer som hadde store geografiske avstander mellom seg med alle utfordringene dette brakte med seg, men i vår case er lokasjonene i samme

land (Diel, Marczak, & Cruzes, 2016). De ansatte på de forskjellige lokasjonene er også personlig kjent med hverandre i vårt tilfelle, og møtes noen ganger i året. Så det kan se ut som geografisk avstand er en hindring, men det er et mindre problem om de ansatte kjenner hverandre og avstanden ikke er så stor. Organisasjoner burde derfor unngå store geografiske avstander mellom teammedlemmer som skal jobbe sammen om dette er mulig. I studien vår mente noen informanter at god kommunikasjon kunne erstatte mye dokumentasjon slik at man kunne unngå å bruke så mye tid på å oppdatere dokumenter.

Kultur er et tema som har gått igjen i alle delene av dette dokumentet, og vil derfor også være en naturlig del av dette kapittelet. I en artikkel der implementering av DevOps var temaet, så ble en DevOps-kultur navngitt som hovedkategorien som gjør implementering av DevOps til en suksess (Luz, Pinto, & Bonifácio, 2019). En av informantene i studien vår mente også at dette var viktig for implementeringen. Det som ble nevnt om denne kulturen var ansvar, og hvordan man burde tenke på en måte der man ikke legger ansvaret med å gjøre et produkt ferdig til noen andre.

Ett av rådene informantene gav oss angående implementering av DevOps, var vilje til å både lære seg ting og endre på ting hos de involverte. Siden en sentral del av DevOps er at ting skal skje kontinuerlig, så må de ansatte og ledelsen være nødt til å være villige til å kaste bort ting som er utdatert, som for eksempel gammel kode. Med DevOps kommer nye måter å jobbe på i tillegg til ny teknologi. For å kunne holde seg oppdaterte må de som bruker DevOps derfor være villige til å lære seg nye ting. Dette er ikke noe man kan forvente av de ansatte uten videre, for det å bryte med gamle metoder kan medføre problemer (Lwakatare, et al., 2016). Bedriften i studien vår hadde fått mange nye ansatte som hadde nye syn på ting, så det kan være nyttig å få inn nye unge ansatte som er åpne til nye arbeidsmetoder og verktøy. Ledelsen trenger også å være med på endringene, for ellers så fører dette til vanskeligheter med å innføre DevOps i organisasjonen (Jones, Noppen, & Lettice, 2016).

Flere av informantene i studien vår nevnte at bedriften hadde implementer DevOps på en måte som kan beskrives som flytende. I denne konteksten betyr flytende at DevOps ikke ble implementert over et kort tidsrom, men at det gikk veldig gradvis der de innførte flere ting som passer under DevOps uten at de tenkte på det som å implementere DevOps. De har hatt mest

fokus på å gjøre ting som fungerer. Fra litteraturen vet vi at det er viktig at alle i organisasjonen forstår hva en DevOps-kultur er for noe slik at implementeringen av DevOps skal gå bra (Kuusinen, et al., 2018). En slik flytende overgang til DevOps vil gjøre det lettere for alle involvert å lære seg hva DevOps betyr, noe som vi allerede har diskutert kan være vanskelig å definere. Det gir bedriften også muligheten til å teste ut forskjellige praksiser og verktøy slik at de bruker det som fungerer for dem. Det å velge de riktige verktøyene er viktig når man begynner med DevOps (Ekbert, Gallardo, Hernantes, & Serrano, 2016). Det gir også bedriften en mulighet til å prøve det ut i praksis slik at man kan overbevise de som ikke har tillit til DevOps ved å vise til faktiske resultater og fordelene vi har gått gjennom tidligere (Siqueira, Camarinha, Wen, Meirelles, & Kon, 2018).

I dette kapittelet har vi gått gjennom flere ting som er viktig å tenke på når man skal implementere DevOps i en organisasjon. Bedriften vi utførte studien hos har ikke innført alt som kan klassifiseres som DevOps, og det kom fram at det ikke var sikkert at de kom til å implementere alt. I praksis kan det virke som at det er vanskelig å implementere alt som hører til DevOps, men det er kanskje ikke nødvendig. Bedrifter er forskjellige, og to ulike bedrifter vil ha forskjellige behov. En informant hevdet at man blir aldri helt ferdig med en slik implementering, så det blir en vurderingssak om hva som er mulig å implementere og hva som vil gi en positiv konsekvens for bedriften.

6 Konklusjon

I dette dokumentet har vi presentert en case-studie av implementeringen av DevOps i en avdeling som tilhører en internasjonal bedrift lokalisert i Norge. Ved å svare på forskningsspørsmålene vi har satt, så har vi sett nærmere på en definisjon av DevOps, hvilke konsekvenser implementering av DevOps medfører, hva de som bruker DevOps tenker om det, samt funnet noen retningslinjer for implementering av DevOps.

Det kan være vanskelig å gi en nøyaktig definisjon av DevOps. Dette er fordi man kan finne mange ulike definisjoner i litteraturen der forskjellige områder blir vektlagt annerledes. DevOps kan også inneholde mange forskjellige temaer og fagfelt, så det blir vanskelig for noen som ikke har lest mye fagstoff om DevOps å gi en nøyaktig og omfattende beskrivelse av begrepet. Det vi har kommet fram til, er at DevOps kan deles i to: DevOps-kultur og teknisk DevOps. DevOps-kulturen lager et arbeidsmiljø der samarbeid, involvering og ansvar for store deler av utviklingssyklusen til programvaren. Teknisk DevOps handler om alle rutinene og verktøyene som kreves for å kunne levere programvare så raskt som mulig, som kort utviklingssyklus og skyløsninger med overvåkning og annen funksjonalitet. Med så mange definisjoner av DevOps var det også vanskelig å se om DevOps i Norden var forskjellig fra andre steder, men ut ifra studien vår og litteraturen vi fant, så var det tydelig at kultur ble mer vektlagt i Norden enn resten av verden. Vi kunne også se at ikke alt det som gikk under teknisk var implementert i bedriften vi utførte studien hos, for eksempel brukte bedriften lite eller ingen målinger, noe som ble vektlagt i litteraturen vi fant. Det kan hende at det er fordelaktig å dele DevOps inn i enda mindre deler, som for eksempel DevOps-rutiner og DevOps-verktøy.

Når bedrifter tar valget med å implementere DevOps, så fører dette gjerne til flere konsekvenser for bedriften. Bedriftskulturen i bedrifter som tok steget ble merkbart endret i en retning som lignet DevOps-kulturen, som indikerer at det ikke er umulig å innføre en slik kultur i organisasjoner. Med DevOps implementert kunne man se flere fordeler bli trukket fram av de som brukte det, som mer frihet rundt valg av verktøy og oppsett av arbeidshverdagen, det ble mindre stress rundt utgivelse, de ansatte ble mer involvert i hele livssyklusen til produktene, og

de ble mer ansvarlige. Det var ikke bare positivitet rundt DevOps når det ble implementert, men det er også utfordringer og ulemper som kan oppstå. Mange vil ha problemer med å bryte gamle rollemønster som de er vante med å følge fra tradisjonelle arbeidsmetoder og de kan være uvillige til å måtte ta på seg nye arbeidsoppgaver. Dette kan være på grunn av motstand mot en ny måte å jobbe på, eller så kan det være at de ikke føler de har nok tid til å ta på seg nye arbeidsoppgaver i tillegg til de gamle. DevOps-rutiner og samarbeid kan ta mye tid, og enkelte personer med kompetanse innen visse fagfelt kan få ekstra mye å gjøre som en konsekvens av dette. Disse personene må gjøre mye fordi det er vanskelig å ansette personer som har all kompetansen som kreves for å være med på alle prosessene i livssyklusen til programvaren. Dette kommer i tillegg til at automatisering av forskjellige prosesser kan være meget komplisert om man forsøker å automatisere alle prosesser. Om organisasjonen også bruker legacy-systemer, så medfører dette flere problemer siden man kan bli tvunget til å utvikle/drifte flere versjoner av programvaren. Når samarbeid er så sentralt for DevOps, så blir også kommunikasjon viktig. Dette blir vanskeligere når teammedlemmer sitter på forskjellige lokasjoner. Så var det også endringer i selve verktøyene som ble brukt. En informant i studien vår mente at moderne verktøy gjorde DevOps mulig, og ikke at DevOps gjorde bruk av nye verktøy mulig. Det er også mye mer valgfrihet i verktøy nå for tiden. Skyløsning ble fremhevet som viktig med tanke på verktøy, fordi det gjør ting som overvåkning av programvareløsningen mye lettere å utføre.

Vi undersøkte også hva de ansatte ved avdelingen faktisk tenkte om DevOps, nå som de har erfaring med det i praksis. Informantene var alle positive til DevOps, og kunne ikke fortelle om noen stor motstand blant de andre som jobbet der. Det som noen ansatte viste litt motstand mot var det nye systemet, som gjorde mye av den gamle kompetansen til de som hadde jobbet der en stund overflødig. I litteraturen kunne vi finne eksempler på motstand fra både ansatte og ledelsen, fordi de ikke trodde at det ville gi noen gevinster for bedriften. Det var mulig for oss å finne mange fordeler listet opp fra dem som har prøvd DevOps både i litteraturen og i studien vår, så det er rimelig å anta at det er mange som liker DevOps siden det gir mange fordeler.

Med så mye som går inn under DevOps, så kan det være vanskelig å vite hvordan man skal gå frem med å implementere det i en organisasjon. Dette er grunnen til at vi valgte å undersøke hva som er kritisk for at implementeringen skal bli en suksess. Det som kom fram var at DevOps-

kulturen var kritisk for suksess, der kulturen gjorde det mulig å få fordelene og unngå/dempe ulempene/utfordringene. De ansatte må tenke på en måte som gjør at samarbeid går enklere og at man ikke lemper problemer på andre, men faktisk prøver å fikse dem selv. Med så mange nye arbeidsmetoder og verktøy trenger man ansatte som er villige til å lære seg og bruke noe nytt. Selv om litteraturen vi fant indikerte at de ansatte burde sitte på samme kontor for best mulig kommunikasjon, så var ikke informantene i vår studie like enige. De kommuniserte noenlunde like bra med de som satt på et annet kontor som de på det samme kontoret. Det kan virke som geografisk avstand har noe å si, men jo mindre den er og om de er i samme land så har det mindre innvirkning. Man må også ha de rette verktøyene og rutinene som gjør det mulig å jobbe på en måte som gjør at man kan levere kode raskt. I studien vår la noen av informantene vekt på at de hadde hatt en flytende overgang til DevOps, og dette var en viktig faktor for suksess for dem. De hadde som mål å implementere verktøy og arbeidsmetoder som produserte positive resultater, og ikke nødvendigvis implementere DevOps bare fordi det er noe nytt. Dette gjorde også at de fikk testet ut forskjellige ting i praksis, og ikke innførte noe som ikke fungerte for dem. For oss virker dette som en god måte å gå fram på. Noen informanter i studien vår poengterte også at det ikke nødvendigvis var et mål å implementere alt som kan beskrives som DevOps, men bare det som passet best for dem. Dette virker også som et godt råd siden det er så mye forskjellig som kan klassifiseres som DevOps og forholdene hos ulike bedrifter vil være forskjellige, så bør de implementere det som passer best for dem.

Vår case-studie går ikke veldig dypt inn på kultur eller verktøy/rutiner, noe som kunne vært interessant å vite mer om. Hvilke spesifikke verktøy som er best egnet for DevOps og en nærmere undersøkelse av hvilken rolle verktøy har for DevOps hadde vært et relevant forskningsspørsmål å bruke i en annen studie. En dypere studie på hvordan DevOps-kulturen faktisk blir utført med spesifikke eksempler fra arbeidshverdagen hadde også vært interessant. Vår studie har noen svakheter ved at vi bare har brukt en metode og inkludert én bedrift. Dette gjør det vanskeligere å generalisere funnene våre, og det vi har funnet ut kan kritiseres for å bare gjelde en liten del av bedriftene i Norden. Det eksisterer relativt få case-studier om temaene vi har dekket i Norden, så det hadde definitivt vært gunstig om det ble utført flere case-studier på dette området der ting som meningene til de som faktisk bruker DevOps i arbeidshverdagen blir undersøkt i en større skala.

7 Referanser

Cruzes D.S., M. K. (2019). Testing in a DevOps Era: Perceptions of Testers in Norwegian Organisations. In M. S. (eds), *Computational Science and Its Applications – ICCSA 2019.* Springer, Cham.

Dalland, O. (2020). *Metode og oppgaveskrivning* . Oslo: Gyldendal.

DeGrandis, D. (2011). *DevOps: So You Say You Want a Revolution?* . Cutter Consortium.

Dhaduk, H. (2022, Janaur 13). *DevOps Lifecycle: 7 Phases Explained in Detail with Examples*. Retrieved from Simform: https://www.simform.com/blog/devops-lifecycle/

Díaz J., P. J. (2019). DevOps in Practice – A Preliminary Analysis of Two Multinational Companies. *Product-Focused Software Process Improvement* (p. Lecture Notes in Computer Science). Springer, Cham.

Diel, E., Marczak, S., & Cruzes, D. S. (2016). Communication Challenges and Strategies in Distributed DevOps. *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)* (pp. 24-28). IEEE.

Ekbert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). *ieexplore*. Retrieved from DevOps: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7458761&tag=1

Fowler, M., & Highsmith, J. (2001). The agile manifesto. *Software development 9.8* (pp. 28-35). Agile Software Development Alliance.

Freeman, E. (2019, November 11). *Dummies*. Retrieved from What is DevOps?: https://www.dummies.com/article/technology/programming-web-design/general-programming-web-design/what-is-devops-265584

Gothelf, J., & Seiden, J. (2017). *Sense & Respond: How Successful Organizations Listen to Customers and Create New Products Continuously.* Harvard Business Review Press.

Jones, S., Noppen, J., & Lettice, F. (2016). Management challenges for DevOps adoption within UK SMEs. *In Proceedings of the 2nd International Workshop on Quality-Aware DevOps (QUDOS 2016)* (pp. 7-11). New York: Association for Computing Machinery.

Kuusinen, K., Balakumar, V., Jepsen, S. C., Larsen, S. H., Lemqvist, T. A., Muric, A., . . . Vestergaard, O. (2018). A Large Agile Organization on Its Journey Towards DevOps.

2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 60-63). Prague, Czech Republic: IEEE.

Luz, W. P., Pinto, G., & Bonifácio, R. (2019). Adopting DevOps in the real world: A theory, a model, and a case study. *Journal of Systems and Software,* Volume 157.

Lwakatare, L. E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H. H., Bosch, J., & Oivo, M. (2016). Towards DevOps in the Embedded Systems Domain: Why is It So Hard? *2016 49th Hawaii International Conference on System Sciences (HICSS)* (pp. 5437-5446). Koloa, HI, USA: IEE.

Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., . . . Lassenius, C. (2019). DevOps in practice: A multiple case study of five companies. *Information and Software Technonolgy*, 217-230.

Muñoz, M. R. (2021). A guidance to implement or reinforce a DevOps approach in organizations: A case study. *Software: Evolution and Process*, e2342.

NSD. (n.d.). *om nsd*. Retrieved from nsd: https://www.nsd.no/om-nsd-norsk-senter-for-forskningsdata/

Oates, B. J. (2006). *Researching Information Systems and Computing.* London: Sage Publications Ltd.

Riungu-Kalliosaari L., M. S. (2016). DevOps Adoption Benefits and Challenges in Practice: A Case Study. In J. A. Abrahamsson P., *Product-Focused Software Process Improvement. PROFES 2016. Lecture Notes in Computer Science, vol 10027.* Springer, Cham.

Roche, J. (2013, November). *Adopting DevOps Practices in Quality Assurance*. Retrieved from Queue.acm.org: https://dl.acm.org/doi/pdf/10.1145/2524713.2524721

Rowse, M., & Cohen, J. (2021). A Survey of DevOps in the South African Software Context. *54th Hawaii International Conference on System Sciences* (pp. 6785-6794). Manoa: University of Hawai'i at Manoa.

Senapathi, M., Buchan, J., & Osman, H. (2018). DevOps Capabilities, Practices, and Challenges: Insights from a Case Study. *In Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018 (EASE'18)* (pp. 57-67). New york: Association for Computing Machinery.

Shahin, M., Babar, M. A., & Zhu, L. (2016). The Intersection of Continuous Deployment and Architecting Process: Practitioners' Perspectives. *ESEM '16: Proceedings of the 10th*

ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (pp. 1-10). New York: Association for Computing Machinery.

Siqueira, R., Camarinha, D., Wen, M., Meirelles, P., & Kon, F. (2018). Continuous Delivery: Building Trust in a Large-Scale, Complex Government Organization. *IEEE Software, vol. 35*, 38-43.

Smeds, J., Nybom, K., & Porres, I. (2015). *DevOps: A Defenition and Perceived Adoption Impediments* . Switzerland: Springer International Publishing .

Walls, M. (2013). *Building a DevOps Culture.* Sebastapol, CA: O'Reilly Media, Inc.

Willis, J. (2016, Juli 16). *What Devops Mean to Me*. Retrieved from ProgressChef: https://www.chef.io/blog/what-devops-means-to-me

8 Vedlegg

Oversikt over vedlegg:

Vedlegg 1: Intervjuguide

Vedlegg 2: Samtykkeskjema og samtykkeerklæring

Vedlegg 3: Godkjennelse fra NSD

Vedlegg 1: Intervjuguide

Intervjuguide

Introduksjon:

Hva er din stilling og hvor lenge har du hatt denne stillingen?

Hva er ditt ansvar med denne stillingen?

Hvilket prosjekt/team jobber du med for øyeblikket?

Om implementeringen:

Kan du beskrive noen av stegene dere har gått gjennom i implementeringen?

Hvordan måler dere fremgang?

Hvordan er utviklingssyklusen deres nå?

Måler dere teamets ytelse på noe vis?

Forskningsspørsmål: Hva er DevOps fra de ansattes perspektiv?

Hva betyr begrepet DevOps for deg?

Føler du at dere jobber på en måte som en følger en DevOps-filosofi?

Om ja, når begynte du å føle dette?

Om nei, hva mener du er nødvendig for å jobbe på en måte som følger en DevOps-filosofi?

Forskningsspørsmål: Hva er konsekvensene av å innføre DevOps?

Hva er forskjellen i måten dere arbeider på nå og før dere startet implementeringen av DevOps?

Ikke alle teammedlemmene er samlokalisert, hvordan har dette påvirket samarbeidet/kommunikasjonen i teamene (drift/teknisk og andre)?

Hva var endringen, om noen, i verktøyene dere bruker i jobben?

Hvordan, om noe, har kulturen i bedriften endret seg siden starten av implementeringen?

Forskningsspørsmål: Hva syntes de ansatte om endringene?

Hva synes du om DevOps, og har meningen din endret seg de siste årene?

Hvilke fordeler har dere oppnådd med å implementere DevOps?

Hvilke utfordringer støter dere på med DevOps-metoden?

Har det vært noe motstand mot implementering av DevOps?

Forskningsspørsmål: Hva er nødvendig for å implementere DevOps på en god måte?

Hvordan føler du implementeringen har gått til nå?

Om den har gått bra, er det noe spesielt som har gjort at det har gått bra?

Hva, om noe, har holdt igjen implementeringen?

Har du noen tanker om hva som er kritisk for å implementere DevOps i en bedrift?

Vedlegg 2: Samtykkeskjema og samtykkeerklæring

Vil du delta i forskningsprosjektet

"DevOps i Norden"?

Dette er et spørsmål til deg om å delta i et forskningsprosjekt hvor formålet er å få mer innsikt i hvordan DevOps påvirker de ansatte i Norden. I dette skrivet gir vi deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål
Det er få case-studier på DevOps utført i Norden, og derfor er formålet til dette prosjektet å gi en innsikt som ikke eksisterte fra før av på dette temaet. Omfanget av dette prosjektet er avgrenset til [redacted] avdeling i Trondheim.
Det vi vil finne ut av er hva noen som jobber med DevOps vil definere det som, og hvordan de opplever bruken av DevOps sammenlignet med tidligere metoder.
Dette er et masterprosjekt som utføres av studenter ved NTNU.

Hvem er ansvarlig for forskningsprosjektet?
Institutt for datateknologi og informatikk ved fakultet for informasjonsteknologi og elektroteknikk, NTNU er ansvarlig for prosjektet.

Hvorfor får du spørsmål om å delta?

Vi har hatt kontakt med [redacted] og noen av dere som jobber der allerede. Etter en diskusjon om DevOps-implementasjonen de siste arene vil vi snakke med de som har vært med på prosessen og har fått godkjenning fra han til å gjøre dette.

Hva innebærer det for deg å delta?

Det vi vil er å ha ett intervju med deg som tar opp mot en time og 30 minutter. Du vil for eksempel bli spurt om hva du tenker om arbeidsmetoden du jobber med nå, og hvordan det var før implementeringen av DevOps. Vi vil også observere hvordan du bruker DevOps i hverdagen, for eksempel kan det hende vi vil observere noen møter som holdes.

Det er frivillig å delta
Det er frivillig å delta i prosjektet. Hvis du velger å delta, kan du når som helst trekke samtykket tilbake uten å oppgi noen grunn. Alle dine personopplysninger vil da bli slettet. Det vil ikke ha noen negative konsekvenser for deg hvis du ikke vil delta eller senere velger å trekke deg.

Ditt personvern – hvordan vi oppbevarer og bruker dine opplysninger
Vi vil bare bruke opplysningene om deg til formålene vi har fortalt om i dette skrivet. Vi behandler opplysningene konfidensielt og i samsvar med personvernregelverket. Vi har også skrevet under en NDA med bedriften.

Vi som utfører prosjektet vil ha tilgang til dataene. Veileder, Knut Arne Strand, vil også ha tilgang på disse dataene. Vi vil ikke bruke navnene deres eller direkte nevne dere i sluttresultatet. Alle data vil bli lagret på NTNU sine tjenester.

Hva skjer med opplysningene dine når vi avslutter forskningsprosjektet?
Opplysningene anonymiseres når prosjektet avsluttes/oppgaven er godkjent, noe som etter planen er 20.mai 2022. Etter at prosjektet er ferdig, vil alle data utenom sluttresultatet bli slettet.

Hva gir oss rett til å behandle personopplysninger om deg?
Vi behandler opplysninger om deg basert på ditt samtykke.

På oppdrag fra NTNU har Personverntjenester vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.

Dine rettigheter
Så lenge du kan identifiseres i datamaterialet, har du rett til:

- innsyn i hvilke opplysninger vi behandler om deg, og å få utlevert en kopi av opplysningene
- å få rettet opplysninger om deg som er feil eller misvisende
- å få slettet personopplysninger om deg
- å sende klage til Datatilsynet om behandlingen av dine personopplysninger

Hvis du har spørsmål til studien, eller ønsker å vite mer om eller benytte deg av dine rettigheter, ta kontakt med:

- Knut Arne Strand ved NTNU. Epost: knut.a.strand@ntnu.no .
- Vårt personvernombud: Thomas Helgesen. Epost: thomas.helgesen@ntnu.no .

Hvis du har spørsmål knyttet til Personverntjenester sin vurdering av prosjektet, kan du ta kontakt med:

- Personverntjenester på epost (personverntjenester@sikt.no) eller på telefon: 53 21 15 00.

Med vennlig hilsen

Jean-Pierre Valenzuela Strand og Torbjørn Steine Sæle/ Knut Arne Strand
(Forsker/veileder)

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet *DevOps i Norden*, og har fått anledning til å stille spørsmål. Jeg samtykker til:

å delta i intervju.
å delta i observasjoner.

Jeg samtykker til at mine opplysninger behandles frem til prosjektet er avsluttet

---

Slett "å delta i intervju" eller "å delta i observasjoner" dersom du ikke ønsker å ta del i en av disse aktivitetene. Send så det som står igjen mellom de stiplede linjene med ditt svar via bedriftsepost til: torbjoss@stud.ntnu.no eller jpstrand@stud.ntnu.no .

Vedlegg 3: Godkjennelse fra NSD

NSD NORSK SENTER FOR FORSKNINGSDATA

Vurdering

Referansenummer

636269

Prosjekttittel

DevOps i Norden

Behandlingsansvarlig institusjon

Norges teknisk-naturvitenskapelige universitet / Fakultet for informasjonsteknologi og elektroteknikk (IE) / Institutt for datateknologi og informatikk

Prosjektansvarlig (vitenskapelig ansatt/veileder eller stipendiat)

Knut Arne Strand, knut.a.strand@ntnu.no, tlf: 73559563

Type prosjekt

Studentprosjekt, masterstudium

Kontaktinformasjon, student

Torbjørn Steine Sæle, torbjoss@ntnu.no, tlf: 91574829

Prosjektperiode

17.01.2022 - 20.05.2022

Vurdering (1)

01.02.2022 - Vurdert

Det er vår vurdering at behandlingen av personopplysninger i prosjektet vil være i samsvar med personvernlovgivningen så fremt den gjennomføres i tråd med det som er dokumentert i meldeskjemaet med vedlegg, og eventuelt i meldingsdialogen mellom innmelder og Personverntjenester. Behandlingen kan starte.

TYPE OPPLYSNINGER OG VARIGHET
Prosjektet vil behandle alminnelige kategorier av personopplysninger frem til den datoen som er oppgitt i meldeskjemaet.

LOVLIG GRUNNLAG
Prosjektet vil innhente samtykke fra de registrerte til behandlingen av personopplysninger. Vår vurdering er at prosjektet legger opp til et samtykke i samsvar med kravene i art. 4 og 7, ved at det er en frivillig, spesifikk, informert og utvetydig bekreftelse som kan dokumenteres, og som den registrerte kan trekke tilbake.

Lovlig grunnlag for behandlingen vil dermed være den registrertes samtykke, jf. personvernforordningen art. 6 nr. 1 bokstav a.

PERSONVERNPRINSIPPER
Personverntjenester vurderer at den planlagte behandlingen av personopplysninger vil følge prinsippene i personvernforordningen om:

• lovlighet, rettferdighet og åpenhet (art. 5.1 a), ved at de registrerte får tilfredsstillende informasjon om og samtykker til behandlingen
• formålsbegrensning (art. 5.1 b), ved at personopplysninger samles inn for spesifikke, uttrykkelig angitte og berettigede formål, og ikke behandles til nye, uforenlige formål
• dataminimering (art. 5.1 c), ved at det kun behandles opplysninger som er adekvate, relevante og nødvendige for formålet med prosjektet
• lagringsbegrensning (art. 5.1 e), ved at personopplysningene ikke lagres lengre enn nødvendig for å oppfylle formålet

DE REGISTRERTES RETTIGHETER
Så lenge de registrerte kan identifiseres i datamaterialet vil de ha følgende rettigheter: innsyn (art. 15), retting (art. 16), sletting (art. 17), begrensning (art. 18), og dataportabilitet (art. 20).

Personverntjenester vurderer at informasjonen om behandlingen som de registrerte vil motta oppfyller lovens krav til form og innhold, jf. art. 12.1 og art. 13.

Vi minner om at hvis en registrert tar kontakt om sine rettigheter, har behandlingsansvarlig institusjon plikt til å svare innen en måned.

FØLG DIN INSTITUSJONS RETNINGSLINJER
Personverntjenester legger til grunn at behandlingen oppfyller kravene i personvernforordningen om riktighet (art. 5.1 d), integritet og konfidensialitet (art. 5.1. f) og sikkerhet (art. 32).

For å forsikre dere om at kravene oppfylles, må dere følge interne retningslinjer og/eller rådføre dere med behandlingsansvarlig institusjon.

MELD VESENTLIGE ENDRINGER
Dersom det skjer vesentlige endringer i behandlingen av personopplysninger, kan det være nødvendig å melde dette til oss ved å oppdatere meldeskjemaet. Før du melder inn en endring, oppfordrer vi deg til å lese om hvilke type endringer det er nødvendig å melde: https://www.nsd.no/personverntjenester/fylle-ut-meldeskjema-for-personopplysninger/melde-endringer-i-meldeskjema

Du må vente på svar fra oss før endringen gjennomføres.

OPPFØLGING AV PROSJEKTET
Personverntjenester vil følge opp ved planlagt avslutning for å avklare om behandlingen av personopplysningene er avsluttet.

Lykke til med prosjektet!

Vedlegg 4: Koder

Ulemper-utfordringer

Avhengig av nøkkelpersoner med nøkkelkunnskap

Tilgang og krever mye av tjenestene som brukes

Det tar mye arbeidstid

Trenger mer tilbakemeldinger
Kan bli mye jobb for visse ansatte
Pålegg fra myndigheter
Nytt system tregere enn forventet
Det er vanskelig å bryte gamle rollemønster
Må fremdeles ha en operations-avdeling
Ting som ikke er arbeidsoppgaven blir brukt tid på

Fordeler

Bedre kodekvalitet
Får mer innsikt i hele prosessen
Mindre stress rundt release
Man kan forme egen arbeidshverdag
Prosesser blir automatisert
Trenger ikke vente lenge på små endringer
Bedre flyt i levering av produkt
Bedre oversikt over produktene
Spredning av kunnskap
Fritt valg av verktøy blant utviklere
Gjør det lettere å ha oversikt over produktet

Mening

Jeg liker DevOps
Best om alle satt sammen
Godt å kunne gå til andre
Mangler oppmuntring fra ledelsen
DevOps funker bra
Samlokalisering ikke nødvendig
Jeg liker automatisering
DevOps var litt skremmende, men ikke nå lengre
DevOps har alltid hørt ut som en god ide

DevOps kommer til å bli standard

DevOps beskriver et teoretisk scenario

Min mening om DevOps har endret seg

Gjelder ikke meg

Motstand

Liker ikke det nye systemet

Ingen motstand

Å gi tilbakemeldinger føles ubehagelig

Lite motstand fordi det ikke ble påtvunget

Passiv motstand

Skeptisk i begynnelsen

Kultur

Deling av ansvar

Ikke overlevering av ansvar

Det er vanskelig å få utviklerne til å ha ansvar for produktet helt til release

Negativ endring

Fokus på avdelingen en selv jobber i

Trenger forståelse for at avdelinger jobber på forskjellig måte

Kjenner andre personlig

Snakke direkte med andre i stedet for å bruke systemet

Ingen endringer i kultur

Er bare interessert i det en selv jobber med

Redde for nytt system

Fysisk avstand ikke et problem

Nye og unge folk åpne til DevOps

Vanskelig å komme bort fra gamle rollemønster

Alle må være med

Trenger kultur der en er villig til å gjøre endringer eller kaste bort gammelt arbeid

Vilje til å lære nye ting er viktig

Pandemien har endret holdninger

Implementering

Det er ikke nødvendig å implementere hele DevOps

Hønen eller DevOps?

Flytende implementering

Må ha støtte fra de ansatte

Trenger mer involvering fra ledelsen

Trenger mer involvering fra de ansatte

Implementeringen har gått bra

Ingen endringer på kort sikt

God dialog er nødvendig

Trenger støtte fra de som bruker produktet

Misforståelser

Leveransedelen er ikke veldig DevOps

Er villig til å implementere mer DevOps

Vanskelig å automatisere gamle prosesser

Målet var-er ikke DevOps

Legacy-system

Kan ikke implementere fullstendig DevOps

Må være villig til å endre på ting

Definisjon av DevOps

DevOps er vagt

DevOps er deling av ansvar

DevOps er kontinuerlig utvikling

DevOps er tett samarbeid

DevOps er tett kobling mellom release-infrastruktur og koding

DevOps er release hver 14. dag, ca

DevOps er en egen avdeling

Hva det betyr er ikke så viktig for de utenfor utvikling

DevOps er bruk av skyløsning

DevOps betyr at utvikling er med på hele prosessen

Verktøy

Få mer ned skriftelig

Automatisering av verktøy

Skyløsning

Verktøyene vi har funker fint

Raskere å fysisk snakke med noen enn å bruke digitale kommunikasjonsverktøy

Store endringer i verktøy

Intuitivt

Alle kan bruke backlogg

Master-branch

Ingen store endringer i verktøy

Ingen automatisert release

Deploy-pipeline er nødvendig

Sammenfallende oppdatering

Trenger testing

Statusmøter

Seminarer

Oppdeling av programmet

Enkelt å sette seg inn i nytt system

Valgfri bruk av utviklerverktøy

Jean-Pierre V. Strand og Torbjørn Steine Sæle

DevOps i Norden

NTNU
Kunnskap for en bedre verden