LUNDS UNIVERSITET
Ekonomihögskolan

Institutionen för informatik

DevOps – Inte nytt men långt kvar

En kvalitativ studie om hur företag arbetar med DevOps

Kandidatuppsats 15 hp, kurs SYSK16 i Informationssystem

Författare: Filip Brattse
Måns Larsén

Handledare: Benjamin Weaver

Rättande lärare: Miranda Kajtazi
Paul Pierce

Förord

Vi vill börja med att tacka handledare Benjamin Weaver som har bidragit med bra kritik och fungerat som ett bollplank under hela uppsatsskrivandet. Vi vill också rikta ett stort tack till alla respondenter som ställde upp på intervju.

18 maj 2022

Filip Brattse & Måns Larsén

DevOps – Inte nytt men långt kvar: En kvalitativ studie om hur företag arbetar med DevOps

ENGELSK TITEL: DevOps – Nothing new but ways to go: A qualitative study of how businesses work with DevOps

FÖRFATTARE: Filip Brattse och Måns Larsén

UTGIVARE: Institutionen för informatik, Ekonomihögskolan, Lunds universitet

EXAMINATOR: Osama Mansour, PhD

FRAMLAGD: maj, 2022

DOKUMENTTYP: Kandidatuppsats

ANTAL SIDOR: 100

NYCKELORD: DevOps, Alignment, Misalignment, Implementering, Organisationsstruktur, Företagskultur, Kontinuerlig integration och Kontinuerlig leverans

SAMMANFATTNING (MAX. 200 ORD):
DevOps är ett arbetssätt och arbetsmodell som beskriver hur utvecklare och driftpersonal arbetar tillsammans för att säkerställa optimal utveckling, lansering och drift av ett mjukvarusystem. Detta är något som organisationer världen över försöker att implementera för att ligga i framkant. Däremot då DevOps inte omfattas av en universell struktur ser det väldigt olika från företag till företag. Syftet med denna uppsats är att undersöka genom en kvalitativ intervjumetod hur företag arbetar med implementering av DevOps ur ett organisations- och ledningsperspektiv. Uppsatsens empiriska resultat består av fyra intervjuer med personer som besitter en betydelsefull roll inom ämnesområdet. Resultatet av vår studie pekar på att det skiljer väldigt mycket från företag till företag hur man definierar DevOps och hur man väljer att strukturera organisationen, samarbetet, kommunikationen, företagskulturen och kvalitetssäkringsarbetet. Ett företag menar på att utvecklare (Dev) är allt som kommer finnas kvar i framtiden medan ett annant företag menar på att det är en samexisterande variant av både utvecklare (Dev) och driftpersonal (Ops). En gemensam nämnare är att DevOps rör sig mot ännu mer automatisering inom testning av mjukvara och kvalitetssäkringsarbete.

Innehåll

Figurer

Tabeller

1 Introduktion

I detta avsnitt presenteras bakgrund, problemområde, frågeställning, syfte och avgränsningar för att generera en tydlig bild av uppsatsens innehåll till läsaren.

1.1 Bakgrund

DevOps är ett arbetssätt som med principer baserat på samarbete och multidisciplinär organisationsstruktur har ett mål att kontinuerligt möta leveranser av mjukvara på ett korrekt och pålitligt vis (Christensen, 2016). DevOps har sina rötter i det agila arbetssättet (Christensen, 2016). Humble och Farley (2010) redogör att arbeta med DevOps är en förlängning på det agila arbetssättet som innebär korta iterationer med input från kunden mellan dessa. Detta arbetssätt har visat sig framgångsrikt men det kan ibland uppstå problem när mjukvaran ska byta händer och utveckling ska gå över till produktion och drift då detta innefattar många manuella aktiviteter som ofta medför problem (Humble & Farley, 2010). Historiskt har utvecklare tagit fram mjukvara och sedan lämnat över den till driftpersonal (Christensen, 2016). Utveckling och drift har varit två separata avdelningar där driftpersonalen fått förlita sig på dokumentation från utvecklarna för att lösa de problem som uppstår (Christensen, 2016).

Ur ett organisationsperspektiv föreslår DevOps-rörelsen ett närmare samarbete mellan development och operations, därav namnet (Christensen, 2016). Separata silos kan fortfarande existera inom organisationen men samarbetet och sammankopplingen ökar mellan dessa (Christensen, 2016). DevOps breddar scope och ökar hastigheten som mjukvara kan levereras, genom att bygga en bro mellan development och operations. Detta kan göras på olika sätt men målet är att utvecklare och driftpersonal ska närma sig varandra, både fysiskt och kompetensmässigt (Leite, Pinto, Kon & Merielles, 2021; Wiedemann, Wiesche, Gewald & Krcmar, 2020)

Wiedemann et al. (2020) redogör att innan DevOps uppkom hade dessa silos sina egna processer, verktyg och kunskapsbank och de interagerade ofta via ett ticket-system. Med detta system var det vanligt att utvecklingsavdelningen ville lansera nya uppdateringar medan driftavdelningen ville förhindra dessa för att bibehålla stabilitet i mjukvaran (Wiedemann et al. 2020). Detta traditionella arbetssätt resulterade i bra stabilitet men få uppdateringar med lång tid mellan dessa och att de olika silos i organisationen skyller problem på varandra (Wiedemann et al. 2020). Denna konflikt mellan utvecklare och driftpersonal ledde till långa ledtider och mycket frustration, de började samarbeta inom organisationen för att närma sig varandra och 2007 föddes termen ”DevOps” för att beskriva detta (Debois, 2008).

Med mål att effektivisera och förbättra deras leveransprocesser av mjukvara implementerar företag i stor utsträckning DevOps (Chen, 2015).

1.2 Problemområde

Att implementera och övergå till DevOps är lättare sagt än gjort, då det inte är ett system utan ett arbetssätt och filosofi. Lägger man också till att det inte finns någon konsensus, varken i branschen eller bland forskare angående exakt hur DevOps ser ut eller bör se ut i praktiken blir det ännu svårare (Erich, Amrit, & Daneva, 2017). Risken finns att en organisation hävdar att de använder DevOps och i stort sett fortsätter som de alltid har gjort (Hemon, Lyonnet, Rowe & Fitzgerald, 2020).

Leite, Rocha, Kon, Milojicic och Mejrelles (2019) menar på att det inte finns någon tydlig och hårdragen definition för hur en organisation ska struktureras men det finns tre vägar man kan ta. 1; bevara separata avdelningar men uppmana och underlätta samarbete mellan dessa. Detta kräver näst intill inga förändringar i organisationen. 2; Skapa och bygga cross-functional teams. Detta kan innebära omorganisation eller att utvalda personer rör sig mellan, alternativt har en plats i, flera avdelningar. 3; skapa och implementera DevOps- eller plattformsteams. De två senare alternativen kräver omorganisation för att göra det nya arbetssättet möjligt (Leite et al. 2019).

Det finns som sagt många svårigheter med att byta arbetssätt till DevOps och dessa problem blir större ju större organisationen är (Dörnenburg, 2018). Hur man ska adoptera det nya arbetssättet råder det skilda meningar om och ett beslut med tydliga riktlinjer måste fattas (Leite et al. 2019). Tidigare forskning visar på att DevOps problematik hos stora företag kan bero på omställningen av organisationen i form av nya sammansättningar av arbetsgrupper och arbetsprocesser som kräver nya former av samarbete (Dörnenburg, 2018).

Kompetensen hos de anställda måste utvecklas och breddas, framför allt ju mer renodlat man vill implementera DevOps (Leite et al. 2019). Införs cross-functional- eller DevOps-arbetslag måste utvecklare lära sig av drifttekniker och vice versa då deras nya arbetsuppgifter omfattar uppgifter som tidigare tillhörde en annan avdelning (Leite et al. 2019). Att utvärdera framgången kan också vara svårt. Att använda sig av ett top-down KPI-baserat utvärderingssystem kan vara svårt då det inte längre finns tydliga gränser mellan avdelningarna och de olika rollerna (Leite et al. 2019).

Leite et al. (2019) redogör att även något så trivialt som titlar och arbetsbeskrivning kan vara svårt då tidigare tydliga linjer suddas ut och många roller utvecklas och breddas. Detta påverkar både anställningsprocesser, löner och kompetensutveckling inom organisationen (Leite et al. 2019).

Kulturen kommer med största sannolikhet påverkas när DevOps implementeras. Det kommer inte vara lika lätt att skylla ett misslyckande på en annan avdelning och det kommer inte vara lika lätt att ge en avdelning beröm för ett lyckat projekt (Leite et al. 2019). Management måste sträva efter att skapa en kultur där laget går före jaget och där det är tillåtet att misslyckas för att sedan förbättra processerna (Leite et al. 2019).

1.3 Frågeställning

Utifrån ovanstående problematik och den bakgrund som presenterats gällande DevOps- implementeringar och olika definitioner av hur man faktiskt jobbar med DevOps mynnade vi ut i frågeställningen:

Hur arbetar företag med implementering och vidareutveckling av DevOps utifrån ett organisations- och ledningsperspektiv för att nå sina mål och hantera utmaningar?

1.4 Syfte

Syftet med denna studie är att genom en kvalitativ intervjustudie undersöka hur företag arbetar med DevOps ur ett organisations- och ledningsperspektiv i förhållande till ansatsen implementering av DevOps. Studien ämnar belysa både eventuella utmaningar och svårigheter, samt målsättningar och utvecklingsområden inom ämnesområdet. Studien ämnar också reflektera kring faktorer likt arbetskultur och organisationsstruktur influerar arbetet med DevOps.

1.5 Avgränsningar

Denna studie kommer inte aktivt utreda DevOps ur ett utvecklarperspektiv eller användarperspektiv. Därmed inte sagt kan olika typer av utvecklararbetsflöden, arbetsprocesser, användarfunktioner, definitioner och tekniker förekomma då dess relevans för studien kräver det. Studiens primära fokusområde är något större företag då detta ger upphov till att observera mer framhävande och komplexa faktorer inom DevOps på en större skala. Med detta sagt förekommer det även mindre företag i studien.

2 Litteraturgenomgång

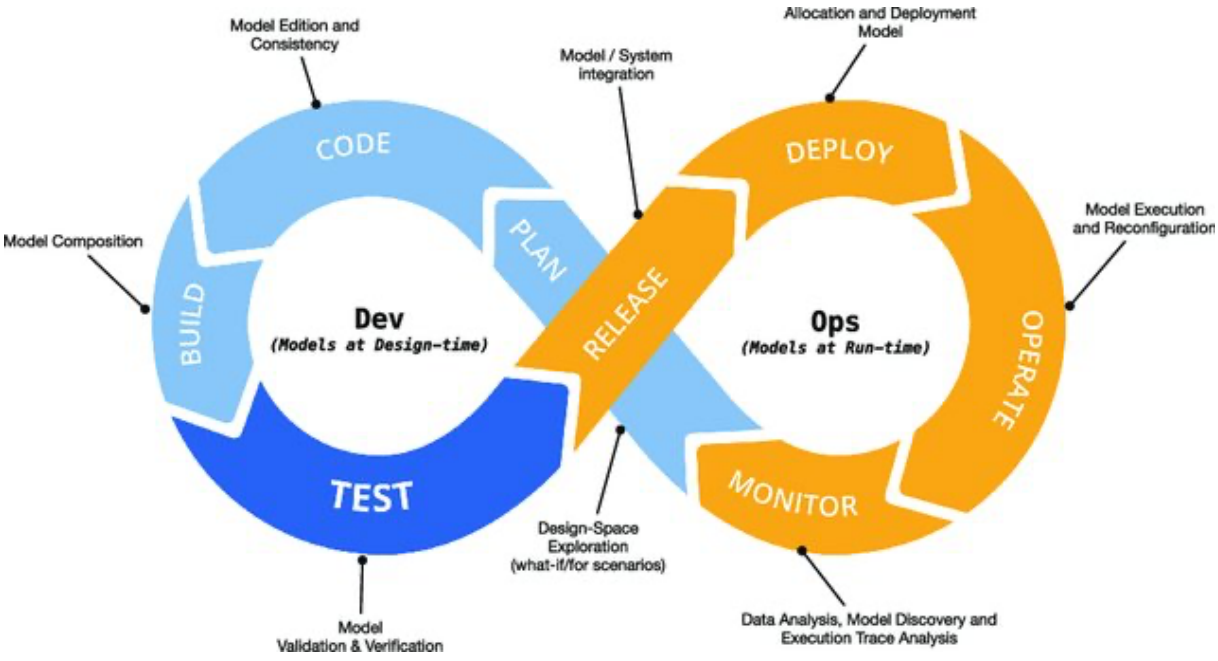
I detta avsnitt definieras begreppet DevOps och underliggande modeller likt kommunikation, samarbete och kontinuerlig integration och kvalitetssäkring. Begrepp likt alignment, organisationsstruktur, företagskultur och utmaningar och förbättringsområden tas också upp i förhållande till dess relevans till ämnesområdet. I avsnitt 2.10 redogörs vårt undersökningsramverk som vuxit fram utifrån en korrelation mellan nyckelord och begrepp som identifierats som viktiga teoretiska spår för studien med tillhörande litteratur utifrån ovanstående rubriker.

2.1 DevOps

Historien om DevOps kan spåras tillbaka till år 2007 då DevOps-konceptet växte fram som en förlängning av dåtidens agila arbetsmetoder och som har sitt ursprung i mjukvaruutveckling ur en praktisk synvinkel (Chen, Kazman, Haziyev & Kropov. 2015; Farroha, & Farroha, 2018; Wettinger, Andrikopoulos & Leymann, 2015). Sedan dess har en rad olika definitioner av DevOps cirkulerat runt och nya varianter likt DevSecOps har vuxit fram på senare tid (Hemon, Lyonnet, Rowe, & Fitzgerald, 2020). Jabbari, Bin, Ali, Petersen och Tanveer (2016) har valt att forska och fokusera på att hitta en allmän vedertagen definition av DevOps för gemensamt bruk genom att undersöka den gemensamma nämnaren i befintliga definitioner av DevOps. Jabbari et al. (2016) menar på att ett nära samarbete mellan komponenterna developers (Dev) på svenska blir detta utvecklare och operations (Ops), vilket blir driftpersonal på svenska, är den centrala grundpelaren i termen och det som framkom mest frekvent i deras forskning. Jabbari et al. (2016) menar på att god kommunikation och teamwork samt en minskning av gapet mellan Dev och Ops i yrkeslivet också spelar en central roll i definitionen av DevOps. Utifrån dessa komponenter definierar Jabbari et al. (2016) DevOps på följande vis:

> "DevOps is a development methodology aimed at bridging the gap between Development (Dev) and Operations (Ops), emphasizing communication and collaboration, continuous integration, quality assurance and delivery with automated deployment utilizing a set of development practices" (Jabbari et al. 2016, s.8).

DevOps-arbetssättet kan beskrivas som en kontinuerlig loop av återkoppling och leverans av arbetsprocesserna (Mohammed, 2017). Mohammed (2017) menar på att denna loop innebär att utvecklingsfasen går snabbare då samarbetet mellan utvecklare och organisation sker på en mer integrerad nivå. Stadier i utvecklingsfasen likt planering, utveckling och sjösättning av produkten sammankopplas med stadier i verksamhetsfasen likt testning, användande, övervakning och återkoppling (Mohammed, 2017). Detta illustreras visuellt i figur 1**: A-model-based-DevOps-approach** (Bobrov, Bucchiarone, Capozucca, Guelfi, Mazzara & Masiagin, 2020).



Figur 1: A-model-based-DevOps-approach Bobrov, Bucchiarone, Capozucca, Guelfi, Mazzara, Masiagin, 2020, (s. 86).

2.2 Organisationsstruktur

Organisationsstrukturen på ett företag har en stor påverkan på hur ett företag kan arbeta med utveckling av mjukvara (Leite et al. 2021). Leite et al. (2021) redogör för att olika organisationsstrukturer lämpar sig olika bra beroende på hur man vill arbeta, och vilken ansvarsfördelning man vill ha och vilka olika roller de anställda ska ha. Eftersom DevOps inte har fasta regler om hur strukturen ska vara finns det olika varianter. Nedan följer en redogörelse för fyra olika organisationsstrukturer samt deras för- och nackdelar (Leite et al. 2021).

2.2.1 Silos

Silos är den traditionella strukturen och viljan att undvika denna var det som gjorde att DevOps växte fram i slutet på 00-talet (Chen, Kazman, Haziyev & Kropov, 2015; Farroha & Farroha, 2018; Wettinger, Andrikopoulos & Leymann, 2015). Avdelningarna är separata och kommunikationen mellan dessa är minimal (Senapathi, Buchan & Osman, 2018). De har ofta olika mål och misalignment är väldigt vanligt (Leite et al. 2021). Denna struktur leder ofta till långa ledtider både för lansering och problemlösning (Leite et al. 2021). Senapathi, Buchan och Osman (2018) redogör att det dock finns några fördelar, exempelvis roller och ansvarsområden är glasklara och det är väldigt lätt att skapa och mäta KPI:er för avdelningar och ner på individnivå. Trots detta väger det negativa över. Silostrukturen gör att varje avdelning arbetar i ett vakuum utan hänsyn till resterande värdekedja vilket orsakar stora problem (Leite et al. 2021). Misalignment för mål och olika arbetssätt bidrar till ett ineffektivt arbetssätt som helst bör undvikas (Leite et al. 2021; Wiedemann et al. 2020).

2.2.2 Klassisk DevOps

Denna struktur fokuserar på samarbete mellan utvecklare och driftspersonal. Strukturen lyckas sällan eliminera alla konflikter men lyckas generellt skapa en bättre miljö där samarbete och delning av kompetens och erfarenhet främjas (Leite et al. 2021). Istället för att ha helt separata avdelningar suddas gränserna ut, exempelvis genom att driftpersonalen kan ha ansvar för underhåll av en databas medan utvecklarna har ansvar för att skriva och uppdatera schemat. Detta sker i samråd med varandra och ansvaret för databasen som helhet delas mellan de tidigare separata avdelningarna (Leite et al. 2021). Detta bidrar till bättre förståelse för helhetsbilden och bättre alignment och samarbete mellan de tidigare separata avdelningarna (Senapathi, Buchan & Osman, 2018). Även om linjer mellan avdelningar suddas ut och samarbete ökar är rollerna tydligt definierade. Alla är numera del av samma team och kommunikationen och samarbetet har ökat men tidigare utvecklare fokuserar fortfarande på utveckling även om de numera är del av hela värdekedjan (Leite et al. 2021). Det ökade samarbetet bidrar i högsta grad till alignment inom organisationen (Wiedemann et al. 2020). Utvecklare kan numera ta hjälp av driftspersonal för att till exempel förändra infrastrukturen så den passar utveckling av mjukvara på bästa sätt (Leite et al. 2021). Vid separata silos har detta inte varit möjligt då de olika avdelningarna har olika mål och deras prestationsmått inte är desamma (Leite et al. 2021).

2.2.3 Cross-functional teams

Denna struktur fungerar precis som klassisk DevOps men går lite längre. Återigen är samarbete och delande av kompetens i centrum. Teamet består av anställda med kompetens både i utveckling, drift och underhållande av infrastruktur (Leite et al. 2021). Detta innebär att driften av ett system landar på samma team som utvecklat det (Leite et al. 2021). Ansvaret ligger på samma team och ansvaret byter aldrig team eller avdelning. Till skillnad från klassisk DevOps är alla team helt självständiga. Detta uppmanar till kreativitet och nya idéer men kan leda till misalignment mellan de olika teamen - kanske kan någon viktig funktion förbises eller så kan två team arbeta på liknande projekt utan att veta om detta, och kommunikation blir återigen en viktig del av denna struktur (Wiedemann et al. 2020). Det kan också vara svårt att erhålla rätt kompetens i alla team. När man tar ansvar för hela värdekedjan måste alla kompetenser finnas i alla självständiga team (Leite et al. 2021). Denna struktur hittas ofta i små organisationer där det inte finns så många team då detta blir ett mer naturligt val i små organisationer, dock är det vanligt att större organisationer försöker lyckas med denna struktur då det är just denna “start-up känsla” man är ute efter (Leite et al. 2021).

2.2.4 Plattformsteams

Denna struktur är lite annorlunda än de tidigare nämnda. Denna struktur är helt och hållet produktorienterad. Ett specifikt team har ansvar för infrastrukturen. Ett team som dock är uppbyggt som alla de andra produktteamen (Leite et al. 2021). Deras uppgift är att tillgodogöra en plattform som utvecklare kan använda för att utveckla och lansera produkter (Leite et al. 2021). Istället för att detta team sysslar med support fungerar det som vilket annat produktteam som helst och plattformen och infrastrukturen är deras produkt (Leite et al. 2021). I detta fall behöver infrastrukturs-teamet utvecklingskunskap och utvecklarna i det övriga produktteamen behöver sköta driften av deras produkter (Leite et al. 2021).

Produktteamen är fullt ansvariga om deras produkt inte fungerar som den ska och det är de som får samtalet om något går fel, inte infrastruktursteamet (Leite et al. 2021). För att undvika en ny silo mellan produktteam och infrastrukturs-teamet måste kontinuerlig kommunikation mellan dessa ske (Leite et al. 2021). Till skillnad från den traditionella silostrukturen så sköts drift och support av produktteamen, inte av infrastruktursteamet (Leite et al. 2021). Fördelen med denna struktur är att infrastrukturen betraktas som en produkt som hela tiden ska förbättras istället för bara underhållas. Detta gynnar alla de olika teamen då utveckling och innovation är i fokus för alla i organisationen (Leite et al. 2021).

Oavsett vilken struktur man väljer är det viktigaste att arbetet börjar från grunden (Leite et al. 2021). Att försöka implementera DevOps på befintlig struktur utan att göra förändringar i organisationen kan ibland leda till fler problem än lösningar (Leite et al. 2021). Det finns organisationer som haft framgång med alla strukturer som lyfts i detta stycke men vad de har gemensamt är att de kartlade vad de ville uppnå och sen gjorde om strukturen från grunden för att passa det nya arbetssättet. I detta ingår även anställning av ny personal om de ansåg att viss kompetens saknades (Leite et al. 2021).

2.3 Kommunikation

Diel, Marczak och Cruzes (2016) redogör att en av nyckelfunktionerna med DevOps är kommunikationen och sammanstrålningen av idéer och förslag mellan olika arbetslag. Kommunikationen kan vara en avgörande faktor för om ett DevOps-projekt lyckas eller ej (Diel, Marczak & Cruzes, 2016). Enligt Diel, Marczak och Cruzes (2016) kan kommunikationen brytas ner i fyra grundpelare som behöver fungera likvärdigt. Den första grundpelaren är *frekvens* vilket innebär hur ofta kommunikation sker och även längden av en konversation (Diel, Marczak & Cruzes, 2016). För låg frekvens av kommunikation kan innebära att detaljer av exempelvis uppgiftsbeskrivningar förloras vilket leder till försämrad kvalité på produkten (Diel, Marczak & Cruzes, 2016; Erich, Amrit & Daneva, 2017). Det handlar om att hitta en balansgång mellan den minimala mängden kontakt nödvändig för att säkerställa adekvat samordning och för mycket kontakt vilket kan överbelasta en av parterna och få dysfunktionella konsekvenser (Diel, Marczak & Cruzes, 2016).

Den andra grundpelaren inom kommunikation handlar om *riktningen* av kommunikationen och innebär att kommunikationen antingen har en vertikal eller horisontell riktning inom organisationshierarkin (Diel, Marczak & Cruzes, 2016). Detta illustreras ofta inom organisationer via endera en top-down approach eller en bottom-up approach vilket innebär inom top-down att kommunikationen distribueras uppifrån makthavarna nedåt i organisationen medan bottom-up är motsatsen (Crespi, Galstyan & Lerman, 2008; Diel, Marczak & Cruzes, 2016). En sak att tänka på är att kommunikationen från utvecklingsteamet vanligtvis är mer informativ, exempelvis rapporter om systemets funktionalitet, medan kommunikation från driftteamet vanligtvis är avsedd att förmedla information om ett specifikt ämne, vanligtvis ett rop på support eller hjälp, eller att informera utvecklaren om kundfeedback på produkten (Diel, Marczak & Cruzes, 2016).

Den tredje grundpelaren är *modalitet* vilket inom detta sammanhang innebär vilken typ av metod som används för att skicka och mottaga information (Diel, Marczak & Cruzes, 2016). Ett sätt att kategorisera modalitet inom organisationer är att kategorisera vilket verktyg som används för det ändamålet, exempelvis: ansikte mot ansikte kommunikation, skriven kommunikation i form av papper eller epost, telefon i form av samtal eller sms och chattjänster likt

Slack,Whatsapp, Messenger etcetera (Diel, Marczak & Cruzes, 2016). Arbetslag rapporterar om att de föredrar en skriven kommunikation som alla kan se och som sedan delas via ett internt kommunikationsverktyg (Diel, Marczak & Cruzes, 2016). Ett annat sätt att kategorisera modalitet är att kategorisera innehållet utifrån parametrarna överförandet av "rik" information och mindre "rik" information. "Rik" information kan anses vara feedback, ansiktssignaler, språkvariation och anpassning av kommunikation medan motparten är motsatsen eller en svagare variant av ovanstående (Diel, Marczak & Cruzes, 2016).

Den fjärde och sista grundpelaren inom kommunikation är *innehållet* vilket kan refereras till som det förmedlade budskapet av meddelandet eller vad som anges i meddelandet. Innehållsanalys av kommunikationsinteraktioner kan göras med hjälp av förbestämda kategorier (Diel, Marczak & Cruzes, 2016). Detta kan förklaras med direkt information av prioritet likt förfrågningar, rekommendationer, överklagande, skyldigheter osv och indirekt information likt informationsutbyte, diskussioner osv. Detta resulterar i olika påverkansstrategier inom innehållskategorin av kommunikation (Diel, Marczak & Cruzes, 2016). Detta kan illustreras i en organisation på följande vis: en viktig mötesbokning gällande en uppdatering av system som ska levereras till en kund, gentemot ett ansikte mot ansikte samtal mellan två kollegor som diskuterar ifall de ska äta lunch tillsammans senare, där det förstnämnda exemplet klassificeras som värdefull och "rik" information mellan två parter medan det andra exemplet klassificeras som mindre värdefull information (Diel, Marczak & Cruzes, 2016).

2.4 Samarbete och lagarbete

Effektivt lagarbete och samarbete mellan medarbetare är en bidragande framgångsfaktor att ta i beaktning när det kommer till hur framgångsrikt ett företag är (Hashmi, Ishak & Hassan, 2018). Samarbete och lagarbete kan definieras på olika sätt beroende på vilken situation eller bakgrund som begreppet syftar att användas inom (Gibson & Zellmer-Bruhn, 2001).

Inom DevOps kan det definieras som; de ömsesidigt beroende prestandakomponenter som behövs för att organisera flera personers prestanda på ett effektivt sätt (Hermawan & Manik, 2021; Salas, Cooke & Rosen, 2008). Detta medför att den kollektiva arbetsinsatsen där varje medarbetare samarbetar med varandra väger tyngre gentemot den enskildes presentation (Hermawan & Manik, 2021; Oh, Lee & Zo, 2019).

Till en viss grad går det att säga att DevOps-samarbete är förekommande i alla organisationer som anställer utvecklare och driftpersonal. Däremot vad som skiljer dessa organisationer åt är andelen interaktion och utbyte av information mellan personalen inom den särskilda organisationen (Erich, Amrit, & Daneva, 2017). Trots att samarbete generellt sett uppmuntras inom alla sektorer i näringslivet så skiljer sig det också mellan företag utifrån hur omfattande detta samarbete är (Diel, Marczak & Cruzes, 2016). Inom vissa organisationer, framför allt inom teknikföretag och IT-industrin, förekommer det exempelvis att driftpersonal inte inkluderas i vissa projekt på grund av bristande tekniska kunskaper enligt utvecklare och i grund och botten kan detta bero på bristande samarbete och kommunikation mellan de olika arbetslagen (Diel, Marczak & Cruzes, 2016; Hoegl & Gemuenden, 2001).

I vissa organisationer kan olika funktioner som utveckling och drift delas upp i föreställningarna att de genom ökad specialisering och standardisering innebär en kostnadseffektiv, säker och hanterbar process (Hoegl & Gemuenden, 2001). Det kan dock istället leda till problem som konkurrerande mål, konflikter och en otydlig ansvarsfördelning mellan jobb som hellre

skyller på varandra än att samarbeta för att lösa dem (Carlsson & Langsager, 2021; Comstedt, 2019).

Ett sätt att förbättra teamwork och kommunikation mellan utvecklare och driftpersonal är att minska distansen både fysiskt och digitalt mellan de olika aktörerna (Hermawan & Manik, 2021). Detta kan genomföras genom att dela fysisk arbetsyta vilket exempelvis möjliggör att enkla frågor kan ställas direkt och mer kommunikation förekommer naturligt (Hermawan, Manik, 2021). Digitala verktyg kan också användas på så sätt att både utvecklare och driftpersonal delar sin arbetsprocess på samma plattform så att båda parter har tillgång till varandras utveckling (Hermawan & Manik, 2021). Detta hjälper till att bygga en bro mellan utvecklare och driftpersonal och på sikt kan detta öka samarbetet, kunskaperna, stödet och motivationen inom ett team när tillfälle ges (Gupta, Venkatachalapathy & Jeberla, 2019). Företag som arbetar med distansarbete primärt kan komplettera med fysiska träffar vid fasta tidsintervall för att höja gemenskapen (Gupta, Venkatachalapathy & Jeberla, 2019).

Fysiska möten och träffar kan hjälpa till att reducera och minska problem med hjälp av enklare kommunikationsmöjligheter mellan de olika arbetslagen och därmed förbättra samarbetet (Diel, Marczak & Cruzes, 2016; Hermawan & Manik, 2021). Fysiska träffar och möten kan hjälpa till på så sätt att utbytet av feedback sker på ett naturligare och bättre sätt än digitalt. Detta ger också en möjlighet att lära sig om företagskulturen i de andra arbetslagen vilket kan underlätta arbetsprocesserna i framtiden (Diel, Marczak & Cruzes, 2016). Däremot behöver det finnas en grundläggande förståelse för samarbete och kommunikation mellan de olika arbetslagen oavsett om kommunikationen sker fysiskt eller digitalt (Nybom, Smeds & Porres, 2016; Wiedemann et al. 2020).

2.5 Kontinuerlig integration, kontinuerlig leverans och kvalitetssäkring

Kontinuerlig integration inom DevOps är en strategi för utveckling av mjukvara där utvecklare ansvarar för integration av koduppdateringar i en gemensam databas (Yarlagadda, 2018). Efter sammanslagningen av sådana uppdateringar körs automatiska kontroller och byggen. Det huvudsakliga målet med denna strategi är att hitta och rätta till buggar i ett tidigt skede av produktutvecklingslivscykeln, vilket kommer att minska den tid som krävs för att lansera programvara (Humble & Farley, 2010; Yarlagadda, 2018). Efter incheckningen av koduppdateringar i ett register, testas var och en av dessa incheckningar med ett automatiskt inbyggt verktyg så att utvecklarna snabbt kan korrigera och respondera på problem (Yarlagadda, 2018).

Kontinuerlig leverans är en strategi för utveckling av mjukvara där koduppdateringar införlivas i en delad repository, sedan utvecklas och granskas de innan de publiceras i produktionsmiljön (Waller, Ehmke, Hasselbring, 2015; Yarlagadda, 2018). Förbättringar av koden utformas, kontrolleras och paketeras automatiskt innan de släpps ut i produktion. Målet är att detta skall ge utvecklarna varningar i ett tidigt skede och på ett pålitligt sätt. Detta åstadkoms genom att automatiserade uppdateringsprocesser och automatiserad testning av mjukvaran sker så att nya projekt kan släppas med ett enda klick (Waller, Ehmke & Hasselbring, 2015). På det stora hela är detta en process som tar kontinuerlig integration till nästa steg genom att införa uppdateringar av kod i produktionsmiljön (Waller, Ehmke & Hasselbring, 2015). I allmänhet är kontinuerlig leverans ett ramverk som hjälper utvecklare att släppa programvaran

och få feedback snabbare (Humble & Farley, 2010). Det är en process som gör det möjligt för företag att ändra applikationer för att möta kraven från användare och konkurrens på marknaden. (Waller, Ehmke & Hasselbring, 2015). Kontinuerlig integration och kontinuerlig leverans är sammankopplade där kontinuerlig leverans är en fortsättning och vidareutveckling av kontinuerlig integration (Waller, Ehmke & Hasselbring, 2015). Kontinuerlig leverans ser helt enkelt till att alla uppdateringar släpps i tid (Waller, Ehmke & Hasselbring, 2015).

Kvalitetssäkring inom DevOps handlar framför allt om att replikera och efterlikna användares användningsmönster och på så sätt minimera upptäckten av obehagliga överraskningar inom mjukvaran (Humble & Farley, 2010). Ett kvalitetsäkringsteam arbetar ofta med att förutse, hitta och leverera reproducerbara scenarier som kan leda till problem för användaren (Roche, 2013). Detta kan illustreras genom att exempelvis skriva in fel lösenord på en inloggningssida för många gånger i rad eller att köra flera program samtidigt för att se hur originalprogramvaran klarar den påfrestningen (Roche, 2013).

2.6 Alignment

IT alignment och IT funktioner inom en organisation har traditionellt varit uppdelade i enskilda enheter för utveckling och drift, dessa har ofta styrts med top-down planering och sekventiell implementering (Wiedemann, Wiesche, Gewald & Krcmar, 2020). Detta innebär att avdelningarna ej samarbetar utan utvecklarna lämnar över ansvar när de anses klara med sin uppgift, driftteknikerna tar därmed över ansvaret när mjukvara är installerad och är ansvariga för att lösa potentiella problem (Wiedemann et al. 2020). Många företag vill frångå dessa metoder och gå mot agila metoder så som DevOps för att öka öppenheten i projekt, möjliggöra skapandet av produkter som fungerar mellan stora lanseringar och kunna snabbt och effektivt möta kundens förändrade kravbild (Wiedemann et al. 2020).

För att göra detta måste verksamheten vara linjerad. IT-linjering definieras som “the degree to which the needs, demands, goals, objectives, and/or structures of one component are consistent with the needs, demands, goals, objectives, and/or structures of another component” (Nadler & Tushman, 1993, s. 119). Målet är att inga enheters mål ska krocka med andra enheters mål (Nadler & Tushman, 1993).

2.6.1 Misalignment

Misalignment kan ha sin grund i många faktorer, några av dessa kan vara; mål, processer, kompetenser och interoperabilitet (Wiedemann et al. 2020). Utvecklare har ofta som mål att vara innovativa och utveckla nya funktioner och system som tar verksamheten framåt medan drifttekniker prioriterar stabilitet och fokuserar på den dagliga verksamheten (Khan, Khan, Khan, Khan & Whangbo, 2022; Wiedemann et al. 2020). Utvecklare arbetar ofta efter formella utvecklingsprocesser och följer diverse metoder för att utföra sitt jobb medan drifttekniker ofta arbetar mer ad hoc och reaktivt (Khan et al. 2022). Ur ett kompetensperspektiv har de olika bakgrund och färdigheter (Khan et al. 2022). Utvecklare är ofta duktiga på att förstå strategiska mål och lämpar sig bra för att utveckla passande lösningar på ett problem medan drifttekniker är duktiga på att lösa problem på studs (Khan et al. 2022). Från ett interoperabilitets perspektiv arbetar utvecklare ofta preventivt för att undvika problem för att nå affärsmålen medan driftteknikerna ofta arbetar reaktivt när problem uppstår. (Wiedemann et al. 2020)

Alla dessa faktorer bidrar till att det blir misalignment inom organisationen och att de olika enheterna inte drar åt samma håll (Comstedt, 2019).

För att motverka dessa faktorer och uppnå alignment har Wiedemann et al. (2020) identifierat tre faktorer; individuell komponentisering, samlat ansvar och tvärvetenskaplig kunskap vilket redogörs nedan.

2.6.2 Individuell komponentisering

För individuell komponentisering finns det tre faktorer att sträva efter; tysta lanseringar, containisering och konvertibel infrastruktur (Wiedemann et al. 2020).

Tysta lanseringar ger utvecklare möjlighet att uppdatera funktionalitet i mjukvaran utan att systemet ligger nere (Wiedemann et al. 2020). Traditionellt har minsta uppdatering krävt att ett system tas offline och detta resulterar i att andra avdelningar blir starkt påverkade i deras uppgifter (Wiedemann et al. 2020). Gör man sig av med detta problem kan utvecklarna fokusera på utveckling och förbättring utan att ta hänsyn till andra avdelningar och deras arbetsuppgifter (Wiedemann et al. 2020).

Containerisering ger möjlighet att arbeta med utveckling på en liten del av ett heltäckande operativsystem (Wiedemann et al. 2020). Ett team kan utveckla på en del av systemet utan att påverka helheten, system är uppdelat i olika containrar (Humble & Farley, 2010). Ett vanligt problem är att ett operativsystems olika delar är beroende av varandra och tas en del ur drift tas hela systemet ur drift. Detta möjliggör små och kontinuerliga förbättring under en DevOps cykel utan att systemet tas ur drift eller att det påverkar driftteknikerna (Wiedemann et al. 2020).

Den tredje faktorn för att uppnå individuell komponentisering är konvertibel infrastruktur. Detta innebär flexibla IT-system som är skräddarsydda till och administreras av DevOps-teamet (Wiedemann et al. 2020). Detta arbetslag löser problem som härleds till gammal IT-arkitektur och gör det möjligt för andra avdelningar att arbeta med denna infrastruktur (Wiedemann et al. 2020). Detta gör att avdelningarna delar ett flexibelt system vilket påskyndar beslut genom att bredda de agila metoderna mellan avdelningar (Wiedemann et al. 2020).

2.6.3 Samlat ansvar

Samlat ansvar är det tillvägagångssätt då ett DevOps-team skapar ansvarighet för hela mjukvarucykeln (Hermawan, Manik, 2021). Detta görs via tre områden; utökande av de agila metoderna, automatisering av processer och produktorientering (Wiedemann et al. 2020).

Att utöka de agila metoderna innebär att man skräddarsyr dessa för att passa den egna verksamheten. Processer, tidsspann och metoder anpassas för att nå maximal flexibilitet (Wiedemann et al. 2020). Istället för att implementera Scrum rakt av som har en iterationscykel på 2–4 veckor kan man anpassa denna till den egna verksamheten om till exempel 1 vecka passar bättre. Att automatisera processer är ett bra sätt att uppnå alignment. Detta både effektiviserar verksamheten och säkerställer att processen utförs på önskat vis varje gång (Wiedemann et al. 2020). I en traditionell organisation kan det ta lång tid att ta beslut mellan enheter. DevOps-enheten gör att enheterna närmar sig varandra. Linjerna suddas ut och automatisering är en viktig komponent i detta. Att gå från projektorientering till produktorientering gör att fokus

läggs på produkten snarare än projektet (Mohammad, 2016). Projekt har ofta ett tidsspann och en budget medan en produkt är klar när den är klar. Traditionellt har utvecklarna stått för första delen av ett projekt och driftteknikerna tar över vid lansering. Med DevOps har detta arbetslag ansvar för hela produktkedjan vilket möjliggör snabba beslut då ansvaret för ett projekt inte är uppdelat utan ansvaret för produkten är koncentrerat i ett arbetslag (Wiedemann et al. 2020).

2.6.4 Tvärvetenskaplig kunskap

Tvärvetenskaplig kunskap innebär att DevOps-teamet utvecklar och samlar den nödvändiga kunskap som krävs för att uträtta mjukvarubaserade aktiviteter. Återigen finns det tre komponenter för att lyckas med detta; breddande av kompetens, problemägande och distribution av kompetens (Wiedemann et al. 2020).

Traditionellt specialiserar sig en anställd antingen inom utveckling eller drift (Comstedt, 2019). Inom DevOps krävs det att man har breddad kunskap inom båda områdena (Comstedt, 2019). Teammedlemmar med utvecklarbakgrund måste lära sig om driften för att kunna fixa buggar medan driftexperter måste lära sig att utveckla för att kunna delta i processer och programmera (Comstedt, 2019; Wiedemann et al. 2020). Detta hjälper till med att alla anställda får ett gemensamt mål och är inte uppdelade beroende på vilken arbetsuppgift de har (Comstedt, 2019; Wiedemann et al. 2020). Problemägande innebär att alla teammedlemmar har en skyldighet att lösa ett problem oavsett vilket sorts problem det är (Wiedemann et al. 2020). Traditionellt har avdelningar skickat problem mellan sig vilket lett till långa ledtider för att lösa dessa. Har alla ansvar för alla problem kommer de lösas snabbare (Wiedemann et al. 2020). Distribution av kompetens innebär att alla inom DevOps-teamet delar med sig av sina kunskaper (Wiedemann et al. 2020). Traditionellt har specialister suttit med sin kunskap och gjort sin del av projektet och sedan skickat vidare det, målet med denna distribution är att alla ska kunna bidra i alla faser av produktutvecklandet (Wiedemann et al. 2020).

2.7 Företagskultur

Att implementera DevOps i en organisation innebär många förändringar och företagskultur är inget undantag. För att framgångsrikt kunna arbeta med DevOps måste kulturen förändras, vissa förändringar kommer vara enklare tack vare till exempel organisationsförändringar medan andra måste man arbeta aktivt med för att få igenom (Mohammed, 2017). Företagskultur anses ofta vara en kritisk framgångsfaktor oavsett projekt och detta har aldrig varit mer sant än när DevOps ska implementeras (Khan et al. 2022). Eftersom DevOps varken är ett system eller har precisa regler och ramverk kan det tolkas lite olika, denna faktor gör att kulturen spelar större roll än vanligt (Khan et al. 2022; Mohammed, 2017). Kärnan av DevOps är att kulturen förändras (Khan et al. 2022). Silos ska bort och samarbete ska främjas och lyckas inte detta kommer DevOps inte vara någon succé (Khan et al. 2022).

Om man ska sammanfatta den kultur man vill ha i en DevOps organisation kan man göra det med orden; samarbete, självständighet, kommunikation och delande (Gall & Pigni, 2022).

Samarbete innebär att man river ner murarna mellan de tidigare silos som funnits i organisationen, omdefinierande av roller och team i både utvecklings- och driftsaktiviteter (Gall & Pigni, 2022). Både utvecklare och driftpersonal samarbetar för att lösa problem och leverera

kvalitativ mjukvara. Den viktigaste aspekten i denna kulturförändring är att acceptera delat ansvar för hela värdekedjan, de traditionella silos ska bort och det är inte längre en möjlighet att lägga över ansvar på en annan avdelning (Gall & Pigni, 2022).

Självständighet gör att ett DevOps team kan tillsammans inom teamet komma fram till och definiera mål som möter krav från både utvecklare och driftpersonal (Gall & Pigni, 2022). Detta gör också att teamet kan själva bestämma när och hur de ska utföra arbetsuppgifterna (Gall & Pigni, 2022). Detta ger en frihet som inte bara är härlig att arbeta med utan varje team kan komma fram till det arbetssätt som passar sina styrkor bäst (Khan et al. 2022). Varje team tar dessutom ansvar för infrastruktur från början av ett projekt till lansering. Detta främjar även innovation inom teamet eftersom man kan experimentera med nya processer och verktyg vilket gör att varje DevOps team börjar likna ett eget start-up företag där de testar, utvärderar och hela tiden utvecklas (Gall & Pigni, 2022). Detta bidrar till snabbhet och innovation inom organisationen och det traditionella arbetssättet som var väldigt trögrörligt börjar försvinna (Gall & Pigni, 2022).

För att göra övergången till DevOps så smidig som möjligt och för att minska avståndet mellan utvecklare och driftpersonal är kommunikation ett av de viktigaste verktygen (Gall & Pigni, 2022). För att främja samarbete inom teamet är öppen och konstruktiv kommunikationskultur något att sträva efter (Gall & Pigni, 2022). Kommunikationen bör vara rak och gamla ticket-system mellan olika avdelningar bör undvikas. Face-to-face är det bästa sättet att kommunicera på men verktyg som Slack, Teams eller Zoom uppmuntras att användas när detta inte är möjligt (Gall & Pigni, 2022).

Delande är det mest centrala i DevOps-kulturen, det är detta som är målet. Delande av kunskap, verktyg, infrastruktur, feedback, framgång och framför allt ansvar är det som för samman organisationen och gör det möjligt att arbeta mer effektivt, agilt och med närmare samarbete (Gall & Pigni, 2022). Detta delande gör det möjligt att samarbeta och utvecklas tillsammans som ett team (Gall & Pigni, 2022). Feedback-loopar är kritiska inom DevOps (Mohammed, 2017). Till exempel kan utvecklare få näst intill direkt feedback på sin kod istället för att vänta på att problem uppstår efter lansering då problemet har landat hos en drifttekniker (Gall & Pigni, 2022; Mohammed, 2017).

Kultur är inget man implementerar en gång och sedan är klar med. Det är ett konstant arbete som hela tiden förändras med företaget (Gall & Pigni, 2022). Det ska vara ett kontinuerligt arbete som hela tiden prioriterar lösningar i ett så tidigt skede som möjligt i livscykeln av mjukvara (Gall & Pigni, 2022). Denna kultur är strukturerad för att lösa problem så tidigt som möjligt med hjälp av samarbete, självständighet, kommunikation och delande (Gall & Pigni, 2022). Detta för att kunna möta kundernas krav som är i konstant förändring (Gall & Pigni, 2022).

2.8 Problemområden och utmaningar

Många stora företag använder sig fortfarande av en traditionell vattenfallsmetod när det kommer till utveckling av deras operativa system (Sebastian, Ross, Beath, Mocker, Moloney & Fonstad, 2017). Däremot genomgår många stora företag en evolution mot en snabbare variant av vattenfallsmodellen som innebär ett närmare samarbete mellan de olika utvecklingsteamen på företaget (Sebastian et al. 2017). Detta är dock inte tillräckligt för att hänga med i den digitala omställningen då mindre företag föds in i en DevOps-kultur och det är en naturlig del

av deras digitala tjänster och plattformar, med hjälp av små multifunktionella team som verkar agilt för att bygga och testa deras produkter (Sebastian et al. 2017).

Företag vill alltid bli bättre och effektivare och det kan finnas många orsaker till att de väljer att implementera DevOps (Sebastian et al. 2017). Några av de vanligaste anledningarna är förbättra förmågan att lösa problem, ledtider från början av ett projekt till avslut, underlätta automation av processer, korta ned releasetiden, öka kvalitet och hastigheten på den kod som produceras samt frigöra resurser för att arbeta på ny funktionalitet (Erich, Amrit & Daneva, 2017; Sebastian et al. 2017). Dessa är några av de vanligaste målen med DevOps och (Erich, Amrit & Daneva, 2017) märkte att många av dessa områden förbättrades efter man gick över till DevOps (Erich, Amrit & Daneva, 2017).

Att gå över till DevOps är inget man gör över en natt. Det är en process som kräver hårt arbete och en del förändringar. Den största förändringen sker för de anställda (Erich, Amrit, & Daneva, 2017). De måste nu bredda sin kompetens och sina färdigheter (Sebastian et al. 2017). En utvecklare kommer inte endast att skriva kod numera utan är involverad i hela kedjan, detta kräver nya kompetenser och bättre kommunikation mellan medarbetare (Sebastian et al. 2017). Infrastrukturen för att stötta det nya arbetssättet måste också förändras, nya verktyg så som versionshantering vilket möjliggör tysta releaser bör införskaffas. (Erich, Amrit, & Daneva, 2017)

Eftersom DevOps inte har någon precis definition och företag gör lite som de vill och anser passa dem bäst kan det uppstå diskussioner om hur man själv ska göra det. Delade åsikter och stridigheter mellan gamla separata avdelningar kan uppstå och måste tas itu med (Erich, Amrit, & Daneva, 2017). Här är ett tydligt top-down ledarskap mycket viktigt. Utvecklare är ofta oroliga för delandet av ansvar och utsuddandet av gränser mellan avdelningar (Erich, Amrit, & Daneva, 2017). Ökat ansvar och att vara tillgänglig när problem uppstår är något de inte är vana vid (Sebastian et al. 2017). Utvecklarna är också oroliga över processerna de nu ska dela med driftpersonalen då utvecklarna anser att driftpersonalen ofta arbetar ad hoc och kaotiskt vilket är en stor kontrast från deras disciplinära tillvägagångssätt (Sebastian et al. 2017). Driftpersonalen i sin tur är ofta oroliga för att de inte kommer kunna fokusera på förbättring av processer när de nu måste lägga mer fokus på förbättring av produkt (Sebastian et al. 2017). Detta ökade samarbete mellan tidigare separata silos kan också göra managements arbete svårare (Sebastian et al. 2017). Ju närmre de arbetar och samarbetar desto svårare blir det att utvärdera individuella anställda och processer, att hela verksamheten blir mer sammankopplad kan alltså ha sina nackdelar också (Erich, Amrit, & Daneva, 2017; Sebastian et al. 2017).

Många av dessa svårigheter kan härledas till en oklar definition av DevOps. Det är inget system som ska implementeras och sedan användas och det finns inga universella regler för exakt hur det ska implementeras och fungera i praktiken (Senapathi, Buchan, Osman, 2018). Har medarbetare olika bild av vad DevOps är och hur det ska implementeras ökar risken för misalignment, missförstånd och uteblivna fördelar (Senapathi, Buchan & Osman, 2018). En tydlig målbild som alla inom organisationen delar är av yttersta vikt för att lyckas med övergången till DevOps (Senapathi, Buchan & Osman, 2018).

2.9 Möjligheter och förbättringsområden

Trots olika definitioner och utformningar av DevOps beroende på vilken organisation man tittar på finns det många och stora likheter. Tekniska och kulturella barriärer bryts ner, kunskap delas, nya team skapas där kunskap delas, allt för att förbättra processer och göra dessa billigare, bättre och snabbare (Leite et al. 2021). Fortsatt utveckling av detta har ett mål och det är automation. Så mycket som möjligt ska automatiseras och detta kommer göra att företag kan snabbt växa och anpassa sig till förändrat affärsklimat (Mohammad, 2016). För att bibehålla sin konkurrenskraft måste företag erbjuda kvalitativ mjukvara och DevOps ger de rätta förutsättningarna för detta (Mohammad, 2016).

Trenden att automatisera så mycket som möjligt av arbetsflödet har bara börjat. Effekten av automation blir att utvecklare kan lägga mer tid och energi på att utveckla appar och funktioner på ett effektivare sätt (Mohammad, 2016). Testning och drift ska helst automatiseras bort vilket frigör resurser som istället kan läggas på utveckling. Innan automation sågs driften och testning av kod som separata delar från utveckling, nu kan man istället skriva kod som sköter dessa uppgifter (Mohammad, 2016).

Automatiserade tester gör att utvecklare får feedback på sin kod i realtid. Istället för att skriva kod för en hel applikation och sedan skicka över det till testarna kan de testa varje rad kod automatiskt och få feedback direkt (Mohammad, 2016). Detta gör att ingen felaktig kod lämnar utvecklingsmiljön och eventuella problem kan åtgärdas direkt och av samma person som skrev koden. Detta bidrar till en mycket effektivare utvecklingsprocess (Mohammad, 2016).

Automatiserad drift eller drift som kod har uppenbara fördelar, framför allt kostnad. Desto färre som arbetar med drift desto billigare blir det (Mohammad, 2016). DevOps är här för att stanna. Det har lyckats integrera utvecklare, driftpersonal, testpersonal och användare till ett och samma arbetsflöde som endast fokuserar på kundens begäran och krav (Mohammad, 2016). Mycket tyder på att framtiden kommer innehålla ännu mer automation än idag vilket gör att mer och mer fokus kan läggas på utveckling, skapande och att möta kundens krav (Mohammad, 2016). Slutmålet för många organisationer verkar vara att allt ska bli Dev så småningom och alla de andra delarna av varukedjan är automatiserade. (Mohammad, 2016)

2.10 Undersökningsramverk

Efter teorikapitlet avslutats togs ett undersökningsramverk fram. Detta ramverk är baserat på de områden och teman som kom fram under undersökningsområdet av befintlig litteratur inom forskningsområdet. De valda undersökningsområdena har stor betydelse för implementation av DevOps och därför har de valts ut i denna studie. I begreppkolumnen förklaras området och dess betydelse samt teorier och begrepp från litteraturen lyfts fram. Detta görs för att kunna skapa ett undersökningsramverk som sedan kommer mynna ut i intervjufrågor som används i kommande empirikapitel. Detta ramverk kommer även ligga till grund för analyskapitlet och är ryggraden i denna studie.

Tabell 1: Undersökningsramverk

Undersöknings-område	Begrepp & faktorer	Relaterad Litteratur
Organisations-struktur	Det finns olika strukturer som används av företag. Den klassiska silos är ofta den man vill komma i från. Vid implementation av DevOps finns det fyra huvudstrukturer att välja mellan: • Silo-struktur • Klassisk DevOps • Cross-functional teams • Plattformsteam	(Leite, Pinto, Kon & Meirelles, 2021) (Sebastian et al. 2017) (Farroha & Farroha 2018) Chen, Kazman, Haziyev & Kropov. 2015 (Wettinger, Andriko-poulos & Leymann, 2015) (Senapathi, Buchan & Osman, 2018)
Kommunikation	Då nya avdelningar och arbetslag kan uppstå blir kommunikationen viktigare. Huvudområden inom kommunikation består av: • Riktning • Modalitet • Frekvens • Innehåll	(Diel, Marczak & Cru-zes, 2016) (Crespi, Galstyan & Lerman, 2008) (Gall & Pigni, 2022) (Erich, Amrit & Da-neva, 2017)
Samarbete och lagarbete	Samarbete blir viktigare då tidigare separata roller närmar sig. Här undersöks: • Minskad distans mellan olika roller • Inkludering av hela organisationen • Olika teamstrukturer • Feedbackloopar	(Hashmi, Ishak & Hassan, 2018) (Gibson & Zellmer-Bruhn, 2001) (Salas, Cooke & Rosen, 2008) (Oh, Lee & Zo, 2019) (Hermawan & Manik, 2021). (Erich, Amrit, & Daneva, 2017)

		(Hoegl & Gemuenden, 2001) (Comstedt, 2019) (Carlsson & Langsager, 2021) (Gupta, Venkatachalapathy & Jeberla, 2019) (Wiedemann, 2018) (Nybom, Smeds & Porres, 2016)
Kontinuerlig integration, leverans och kvalitetssäkring	Utvecklare tar större ansvar i produktkedjan och följer med in i ett driftperspektiv när det gäller uppdateringar och driftstörningar. Att kunna automatisera stora delar av denna kedja med hjälp av testmiljöer är viktigt för kvalitetssäkringsarbetet.	(Yarlagadda, 2018) (Humble & Farley, 2010) (Waller, Ehmke & Hasselbring, 2015) (Roche, 2013)
Alignment	Att linjera verksamheten för att sträva efter samma mål. Ansvar får inte lämnas över utan en utvecklare måste ta ansvar för sin kod i hela värdekedjan. I detta avsnitt lyfts: • Misalignment • Individuell komponentisering • Samlat ansvar • Tvärvetenskaplig kunskap	(Wiedemann, Wiesche, Gewald & Krcmar, 2020) (Nadler & Tushman, 1993) (Khan, Khan, Khan, Khan & Whangbo, 2022) (Comstedt, 2019) (Humble & Farley, 2010) (Hermawan & Manik, 2021) (Mohammad, 2016)

Företagskultur	Här lyfts DevOps påverkan på befintlig kultur och vad organisationer strävar efter. De centrala teman i detta avsnitt är: • Samarbete • Självständighet • Kommunikation • Delande	(Gall & Pigni, 2022) (Mohammed, 2017) (Khan, Khan, Khan, Khan & Whangbo, 2022)
Utmaningar och problemområden	Vilka de största utmaningarna med DevOps är. Det kan vara definition av DevOps, kulturella förändringar, få med de anställda på resan samt arkitektur som används	(Erich, Amrit, & Daneva, 2017) (Sebastian, Ross, Beath, Mocker, Moloney & Fonstad, 2017). (Senapathi, Buchan & Osman, 2018) (Dörnenburg, 2018)
Möjligheter och förbättringsområden	Vilka möjligheter ger DevOps för framtida förbättringar. Automation är ett stort tema inom detta avsnitt.	(Mohammad, 2016) (Leite et al. 2021)

3 Tillvägagångssätt

3.1 Litteraturgenomgång

Efter att ha beslutat oss för att fördjupa oss i ämnesområdet DevOps var nästa naturliga steg att inleda arbetet med att undersöka och granska tidigare forskning och litteratur inom DevOps, organisationsstruktur, företagskultur och alignment för att bygga en kunskapsbas och hitta en nisch inom ämnet för vår studie. Genom att fördjupa oss i journaler, konferenser och vetenskapliga artiklar lyckades vi identifiera användbara artiklar, teorier och nyckelpersoner inom ämnet.

Dörnenburg (2018) beskriver hur större organisationer ofta upplever svårigheter med att implementera DevOps på grund av organisationens storlek, företagskultur och större organisationers trögrörlighet. Av denna anledning började vi undersöka organisationsstruktur och företagskultur kopplat till DevOps för att se om samband existerar.

Relevant litteratur som använts för studien inhämtas primärt via sökmotorerna Google Scholar, Ais eLibrary och Lunds universitets egen sökmotor LUBsearch, detta rekommenderas även av Oates, Griffiths och McLean (2022). Genom att använda oss av referensförteckningen i publikationerna kunde vi även hitta fler underliggande studier som var relevant för vår studie. Vi valde att utvärdera forskningsmaterialet som vi lokaliserat och säkerställa att befintliga källor var välrenommerade. Vi tittade bland annat på citationer, återkommande författare, var publikationerna publicerats samt vilka underliggande källor som artikeln baserats på eller inspirerats av. Vi tog även på oss våra källkritiska glasögon då det finns en risk med källorna att information är missvisande eller vinklat på ett visst sätt (Oates, Griffiths & McLean, 2022). Nedan redogörs de söktermer som använts och hur de har kombinerats med varandra.

Till höger om nyckelordet är antalet träffar i de olika sökmotorerna.

DevOps - 30200 (Google Scholar), 287 (AIS eLibrary), 4206 (LUBsearch)
DevOps Culture - 6850 (Google Scholar), 135 (AIS eLibrary), 143 (LUBsearch)
DevOps Collaboration - 1240 (Google Scholar), 23 (AIS eLibrary), 1 (LUBsearch)
“DevOps” - Continuous integration and quality assurance 9060 (Google Scholar), 3 (AIS eLibrary), 8 (LUBsearch)
DevOps Automation - 13100 (Google Scholar), 168 (AIS eLibrary), 845 (LUBsearch)
Alignment - 5,200,000 (Google Scholar), 14471 (AIS eLibrary), 566,186 (LUBsearch)
Misalignment - 680,000 (Google Scholar), 1276 (AIS eLibrary), 51.214 (LUBsearch)
DevOps Communication - 15300 (Google Scholar), 240 (AIS eLibrary), 620 (LUBsearch)
DevOps Challenges - 15400 (Google Scholar), 256 (AIS eLibrary), 454 (LUBsearch)
DevOps future and opportunities - 7520 (Google Scholar), 159 (AIS eLibrary), 20 (LUBsearch)
Organizational structure and culture - 3,390,000 (Google Scholar), 25719 (LUBsearch), 14610 (AIS eLibrary)

3.2 Metodval

Efter att ha studerat ämnesområdet DevOps utifrån tidigare forskning, litteratur och studiens parametrar som beskrivs i tidigare stycken mynnade vi ut i vår forskningsfråga:

Hur arbetar företag med implementering och vidareutveckling av DevOps utifrån ett organisations- och ledningsperspektiv för att nå sina mål och hantera utmaningar?

För att besvara ovanstående forskningsfrågan behöver vi således en forskningsmetod som ger oss möjlighet att förstå hur implementeringen och vidareutveckling av DevOps fungerar och arbetas med ur ett organisatoriskt perspektiv med insyn både i de operativa delar och med insyn i vissa delar ur utvecklarperspektiv. Begreppet DevOps har en komplexitet kring sig då det skiljer sig från organisation till organisation och kan ha olika definitioner eller betydelser beroende på sammanhang (Hemon et al. 2020).

Utifrån dessa parametrar kunde vi utesluta en kvantitativ metod och valde istället en kvalitativ intervjumetod. För att kunna besvara vår forskningsfråga behövde vi få ett helhetsperspektiv över hur verksamhetens DevOps team arbetar med implementering, ifall problemområden existerade och hur ambition med DevOps ser ut i framtiden. Intervjuer möjliggör för oss att samla in en grundlig förståelse för hur respondenten arbetar och dess kunskap inom ämnet. Intervjuer öppnar också upp för mer komplexa frågor och öppna frågor som kan leda till nya upptäckter (Oates, Griffiths & McLean, 2022; Trinczek, 2009).

Intervjufrågorna kan komma att behövas finjusteras utifrån respondentens roll i organisationen eller dess kunskapsbas inom ämnesområdet DevOps. Det skall dock poängteras att våra intervjupersoner ej forskat inom ämnet eller titulerar sig som experter inom ämnet utan arbetar med DevOps utifrån en praktisk best practice modell som fungerar i deras organisation och erfarenheter och uppfattningar kan variera mellan respondenterna.

Den kvalitativa intervjubaserade metoden hjälper till att få en ökad förståelse för respondenternas erfarenheter och tankesätt (Oates, Griffiths & McLean, 2022; Trinczek, 2009). Förberedelserna inför intervjuerna genomfördes genom att förbereda intervjufrågor baserat på litteraturgenomgångens stora fynd och egna instick från vår sida som är intressant för studien. Däremot lämnades det utrymme för följdfrågor, sidospår och instick från respondenten och vår sida vilket genererar nya dimensioner och aspekter av ämnet vilket skapar en intressant diskussion. Enligt Oates, Griffiths och McLean (2022) är det en gyllene medelväg där man har en semistrukturerad intervju och detta passade vår studie bra. Respondenten ges även möjlighet att vädra egna åsikter och tankar som de anser kan ha betydande roll för studien. Detta resulterar i en detaljrik och omfattande empirisk data (Yung, 1995).

3.3 Urval

3.3.1 Val av företag

Valet av företag skedde utifrån några parametrar. En av dessa parametrar var organisationens storlek, enligt Dörnenburg (2018) har större organisationer svårare med implementering av DevOps gentemot nya och mindre organisationer och företag som på ett annat sätts föds in i DevOps kulturen. Detta ledde till att vi primärt valde att fokusera på större organisationer då

vi även vill belysa problemområden och utmaning inom DevOps i vår studie. Däremot var detta inte en avgörande faktor utan mindre och medelstora företag utvärderas också utifrån kompetens och respondents roll inom DevOps.

Vi var även intresserade att se hur tidsschemat för hur övergången till DevOps hade gått till eller går till just nu. Ifall implementeringsprocessen nyligen hade ägt rum var detta en fördel enligt oss. Detta resulterade att vi letade efter relativt stora organisationer/företag som nyligen varit med om en DevOps implementering eller att respondenten i fråga innehaft sin roll en längre period och varit med om övergången.

3.3.2 Urval av respondenter

Eftersom vi var ute efter ett helhetsperspektiv på implementering och vidareutveckling av DevOps tyckte vi att respondenten bör ha gedigen kunskap inom DevOps från alla perspektiv, vilket resulterade att vi strävade efter att hitta personer som besitter en någorlunda hög hierarkisk position inom deras verksamhet. Titlar likt DevOps manager, Lead DevOps Engineer, Architecture DevOps, CTO, CIO, CDO, HTO, CISO och Lead DevOps officer har varit en del av filterinställningar som använts för att filtrera vårt urval. Däremot uteslöts inte andra roller ifall respondenten visade på kunskap inom ämnesområdet.

Vi använde oss av Linkedin, vårt eget kontaktnät och email för att kontakta våra intervjupersoner. Totalt kontaktades 15 personer varav 10 via Linkedin, 2 via kontaktnät och 3 via email. Via Linkedin fick vi tag i 6 personer som var intresserade av att ställa upp på intervju och 4 personer avböjde eller valde att inte svara. Vårt kontaktnät resulterade i en intervju och en som avböjde på grund av tidsbrist. Email resulterade i en intervju och 2 personer valde att inte svara. Av de 6 personer som var intresserade att ställa upp på intervju från Linkedin så valde 3 personer att hoppa av då de kände att detta inte riktigt var deras ämnesområde eller att det inte kände sig tillräckligt kompetenta eller hade jobbat tillräckligt länge med detta för att kunna ställa upp på intervju. En person hade tyvärr inte tid inom vårt tidsschema för att ställa upp. Detta resulterade i två intervjuer via Linkedin. Detta resulterade i totalt 4 intervjuer med kompetenta personer inom olika branscher och olika titlar vilket vi ansåg vara tillräckligt för denna studie.
Nedan i tabell 2: **Respondenter, bransch och intervjuinformation** kan intervjupersoner, bransch, titel, typ av intervju och annan nödvändig information observeras.

Tabell 2: Respondenter, bransch och intervjuinformation

Respondent	Bransch	Titel	Typ av intervju	Längd på intervju	Datum	Appendix
R1	Globalt Telekombolag	DevOps Manager	Teams-möte	ca 30 min	25/4-2022	A
R2	Konsultbolag Stratiteq Sweden AB	Head of technology	Fysiskt möte på kontor	ca 1h	29/4-2022	B

R3	It-företaget Iver, Tech	DevOps Manager	Teams-möte	ca 45 min	2 /5-2022	C
R4	Multinationellt Digitaliserings företag	Chief information security officer / DevOps	Teams-möte	ca 30 min	5/5-2022	D

Utifrån tabell 2: **Respondenter, bransch och intervjuinformation** går det att observera att det finns variation i branscher och titlar mellan våra respondenter. R1 arbetar som DevOps manager på ett av världens största globala telekombolag vilket innebär att se till DevOps arbetet vidareutvecklades och att alla parter inom produktkedjan var tillfredsställda. Detta involverar utvecklare och driftpersonal.

R2 arbetar på konsultbolag Stratiteq Sweden AB som Head of technology vilket innebär att R2 var involverad inom flera olika DevOps projekt från början till slut, där valde vi att fokusera på ett projekt som involverade aktören Skånetrafiken. Vi fick insyn inom både det egna inhouse arbetet med DevOps hos dem och hur kunder valde att tackla sina utmaningar med DevOps och arbeta agilt.

R3 jobbar på ett IT-bolag med en stor kundbas. R3 innehar en roll som gruppchef/DevOps manager för de team som arbetar med utveckling. Hens arbetsuppgifter är att rekrytera, allokera resurser och hjälpa till med nuvarande arbetsprocesser utan att detaljstyra.

R4 arbetar som CISO (Chief information security officer) och DevOps manager på ett digitaliserings företag som hjälper kunder bli pappersfria. Hens arbetsuppgifter består av att tillhandahålla och upprätthålla en infrastruktur så att DevOps arbete kan fortlöpa på ett smidigt sätt. Hen har också ett säkerhetsfokus involverat i de agila arbetetsprocesserna.

3.4 Intervjuer

Tre av fyra intervjuer genomfördes på distans via digitala verktyg i form av videosamtal. Detta skede på grund av en rad olika faktorer, där en av faktorerna var att vi inte blev begränsade till en geografisk plats utan kunde utföra intervjuer med personer som inte var bosatta i Malmö-Lund området, vilket i sin tur gav upphov till ett större spektrum av intervjukandidater. Den andra faktorn till att vi genomförde intervjuerna digitalt var på grund av önskan från respondenterna som annars inte hade haft möjlighet att ställa upp på intervju och avvara tid åt vår studie. Den sista anledningen till att vi genomförde tre intervjuer på distans var att mötesplatsen för intervjutillfället inte existerade då många respondenter jobbar hemma efter omstruktureringar av Covid-19 pandemin och inte hade tillgång till ett konferensrum.

En intervju genomfördes fysiskt via ett möte på respondents kontor. Enligt Jacobsen (2002) är fysiska intervjuer ansiktet mot ansiktet bättre då intervjupersonen ofta lättare öppnar upp sig och svarar mer direkt på intervjufrågorna. Fysiska intervjuer möjliggör också för att lättare läsa av kroppsspråk, gestikuleringar och subtila signaler hos intervjupersoner (Jacobsen, 2002; Oates, Griffiths & McLean, 2022). Telefonintervjuer gör det lättare för respondenten att slingra sig från frågor och ljuga (Jacobsen, 2002; Oates, Griffiths & McLean, 2022). Däremot minskas dessa effekter av ett videosamtal vilket validerade vårt val av att genomföra tre intervjuer på det sättet.

3.4.1 Pilotintervju

För att förbereda oss för intervjuerna valde vi att genomföra en pilotintervju. Denna intervjus syfte var att vi skulle känna oss bekväma i rollen som intervjuledare och strukturen på intervjun. Pilotintervjun användes också för att kontrollera och testa det digitala verktyg som skulle användas under intervjun Microsoft teams och inspelningsverktyget OBS.app vilket möjliggör för inspelning att lagras lokalt till den egna enheten men även uppkopplat till molnet ifall teknikstrul skulle inträffa och orsaka dataförlust under intervjuns gång.

3.4.2 Intervjumall

Vår intervjumall som återfinns i tabell 3: **Intervjumall** användes som stöd vid intervjuerna och har utformats med hjälp av vårt undersökningsramverk som redovisas i tabell 1: **Undersökningsramverk**. Intervjumallen består av en rad olika steg och procedurer där det första steget är vi ger en introduktion av oss själva och en snabb övergripande beskrivning av vår studie och forskningsfråga. Efter det diskuteras formaliteter, teknikaliteter och etiska aspekter av intervjun som på förhand redan var bestämda men vi valde att dubbelkontrollera att respondenten exempelvis ger godkännande till inspelning av intervjun och ifall respondenten ger godkännande till användandet av hens namn i uppsatsen. Det tredje steget består av bakgrund och inledande frågor inom ämnesområdet DevOps och allmän information om respondenten vilket hjälper till och möjliggör för mer djupgående frågor. Enligt Oates, Griffiths och McLean (2022) är de inledande frågorna viktiga för att få intervjupersonen att slappna av och känna sig bekväm i sammanhanget, samtidigt som det sätter ton för resten av intervjun. Oates, Griffiths och McLean, (2022) beskriver också att frågorna skall vara lättförståeliga då alla som intervjuas kanske inte besitter samma kompetens eller att frågorna är för akademiska. Därför säkerställde vi att våra frågor var lättförståeliga och att begrepp och termer förklarades.

Det nästkommande steget är frågor direkt kopplade till vårt undersökningsramverk där vi ställer frågor konkret kopplade till de olika undersökningsområdena i tur och ordning exempelvis diskuteras ämnet *kommunikation inom DevOps* grundligt innan nästa område påbörjas. Det sista steget i vår intervjumall är att ge respondenten frihet att gå tillbaka till ett ämnesområde som hen vill dela med sig mer av eller ta upp ett nytt ämnesområde som inte var med i vårt undersökningsramverk och som hen tror kan bidra till vår studie. Sedan bör det tilläggas att detta är en intervjumall och instick, följdfrågor och sidospår är något som förekom under intervjuerna.

Tabell 3: Intervjumall

Översiktsplan	**Exempelfrågor & givande resonemang**
Introduktion och presentation	• Inledande presentation av oss, vår studie och vår forskningsfråga

Formaliteter, teknikaliteter och etiska aspekter	• Godkännande av inspelning • Behov av anonymitet • Tillgång till det transkriberade materialet och den slutgiltiga uppsatsen
Bakgrund	• Roll/Titel • Arbetsuppgifter • Hur länge har du jobbat på företaget? • Företaget bakgrundsinformation • Respondents bakgrund
Inledande frågor	• Hur arbetar ni med DevOps i nuläget? • Hur länge har ni arbetat på med DevOps?
Djupgående frågor - kopplat till vårt undersökningsramverk	Se tabell 4: **Intervjufrågor utifrån teoretiskt ramverk**
Övriga punkter	• Är det någonting annat du vill ta upp som vi inte har berört ännu? • Är det någonting du undrar om oss eller om studien generellt?
Avslut	• Tack för att du har medverkat med din kunskap och kompetens

	• Kan vi kontakta dig ifall det dyker upp något nytt vi kommer på under studiens gång?

Tabell 4: Intervjufrågor utifrån teoretiskt ramverk

Undersökningsområde	**Intervjufrågor**	**Relaterad Litteratur**
Organisationsstruktur	• Vilka förändringar behövde göras i organisationen för att implementera DevOps? • Har ni behövt göra omstrukturering i form av nya team eller avdelningar? • Var det ett top-down eller bottom-up beslut?	(Leite, Pinto, Kon & Meirelles, 2021) (Sebastian et al. 2017) (Farroha & Farroha 2018) Chen, Kazman, Haziyev & Kropov. 2015 (Wettinger, Andrikopoulos & Leymann, 2015) (Senapathi, Buchan & Osman, 2018)

DevOps - Kommunikation	• Vad gör ni för att förbättra kommunikationen mellan utvecklare och driftpersonal? • Hur kommunicerar utvecklare och driftpersonal? • Används gemensamma plattformar och digitala verktyg för att se varandras resultat och process? och kommunikation • Ansikte mot ansikte eller digitalt?	(Diel, Marczak & Cruzes, 2016) (Crespi, Galstyan & Lerman, 2008) (Gall & Pigni, 2022) (Erich, Amrit & Daneva, 2017)
Samarbete och teamwork	• Vad fungerar bra respektive mindre bra utifrån ett samarbetsperspektiv inom DevOps teamet? • Hur arbetar de olika teamen i nuläget? • Ovanstående frågor inom kommunikation hänger även ihop med samarbete och teamwork.	(Hashmi, Ishak & Hassan, 2018) (Gibson & Zellmer-Bruhn, 2001) (Salas, Cooke & Rosen, 2008) (Oh, Lee & Zo, 2019) (Hermawan & Manik, 2021). (Erich, Amrit, & Daneva, 2017) (Hoegl & Gemuenden, 2001) (Comstedt, 2019) (Carlsson & Langsager, 2021) (Gupta, Venkatachalapathy & Jeberla, 2019)

		(Wiedemann, 2018) (Nybom, Smeds & Porres, 2016)
Kontinuerlig integration, leverans och kvalitetssäkring	• Jobbar ni med kontinuerlig integration? • Är stora delar av er verksamhet automatiserade inom detta område? • Hur hittas och hanteras buggar och andra problem? • Hur kvalitetssäkrar ni era produkter?	(Yarlagadda, 2018) (Humble & Farley, 2010) (Waller, Ehmke & Hasselbring, 2015) (Roche, 2013)
Alignment	• Har organisationen blivit mer aligned med DevOps? • Varför valde ni att implementera DevOps?	(Wiedemann, Wiesche, Gewald & Krcmar, 2020) (Nadler & Tushman, 1993) (Khan, Khan, Khan, Khan & Whangbo, 2022) (Comstedt, 2019) (Humble & Farley, 2010) (Hermawan & Manik, 2021) (Mohammad, 2016)
Företagskultur	• Har det skett några kulturella förändringar med införandet av DevOps?	(Gall & Pigni, 2022) (Mohammed, 2017) (Khan, Khan, Khan, Khan & Whangbo, 2022)
Utmaningar och problemområden	• Finns det några nackdelar med att arbeta med DevOps?	(Erich, Amrit, & Daneva, 2017)

	• Vad är den största utmaningen med DevOps?	(Sebastian, Ross, Beath, Mocker, Moloney & Fonstad, 2017). (Senapathi, Buchan & Osman, 2018) (Dörnenburg, 2018) • Kan relatera till alla områden
Möjligheter och förbättringsområden	• Har önskad effekt av DevOps uppnåtts? • Arbetar ni med vidareutveckling eller anser ni att DevOps är "på plats"? • Vad är enligt dig några successfactors för att lyckas med DevOps?	(Mohammad, 2016) (Leite et al. 2021) • Kan relatera till alla områden

3.5 Bearbetning av intervjuer

3.5.1 Transkribering

Genom att transkribera inspelningarna av intervjuerna möjliggjorde detta att vi fick en tydligare överblick över vår empiriska data och vilka användningsområden varje enskild del i intervjun har. Transkriberingen av intervjuerna skedde ordagrant men korrigerades utifrån relevans av svar och grammatik. Detta involverar upprepningar, läten, stakningar och onödig information. Detta hjälpte till att bygga ett bättre empirikapitel strukturmässigt. Eftersom några av våra respondenter begärde anonymitet var vi tvungna att minska detaljnivån i deras svar i några av utskrifterna så att de inte kunde identifieras direkt eller indirekt (Jacobsen, 2002; Oates, Griffiths & McLean 2022). För att säkerställa att inget missades eller slarv hade inträffat under transkriberingen valde vi att dubbeltranskribera den empiriska data, vilket är en kvalitetssäkring enligt Jacobsen (2002). Transkriberingarna sparades sedan i en separat mapp som endast vi författare har tillgång till och döptes utifrån datum och nummer på intervju exempelvis "Intervju 1".

3.5.2 Dataanalys

Dataanalysen bygger på att kategorisera texten utifrån de parametrar som författarna anser vara relevanta för den empiriska studien. Detta arbete inleddes med en genomläsning av de

transkriberade intervjuerna och efter antecknades intressanta ämnesområden och citat utifrån relevans. Oates, Griffiths och McLean (2022) menar på att det finns olika sätt att märka texten exempelvis färgkoda. Efter överlagt de olika alternativen kom vi fram till att för vår studie passade att anteckna och markera i ett separat dokument bäst. Eftersom vi redan på förhand ville diskutera och jämföra litteratur redovisad i tabell 1: **Undersökningsramverk** bestämde vi oss för att empirikapitlet och diskussionskapitlet skulle vara strukturerat efter samma rubriker som redovisas i undersökningsramverket. Detta resulterade att vi kunde identifiera begreppsområden och citat inom transkribering från intervjufrågorna och svaren av respondenterna till en viss begreppsrubrik inom empirikapitlet. Mönster från respondenterna och citat plockades ut från transkribering för att sedan paras ihop med en begreppsrubrik i empirikapitlet och för att sedan diskuteras i diskussionskapitlet. Detta ansåg vi var en bra metod då det gjorde det lättare att koppla ihop fynden från transkriberingen till vår forskningsfråga. Oates, Griffiths och McLean (2022) beskriver att presentera de mönster man ser i data bygger på kategorisering och ger ett sammanhang om vad mönstret innebär.

3.6 Undersökningskvalitet

3.6.1 Reliabilitet

Reliabilitet är en viktig aspekt för att utvärdera en uppsats eftersom den avgör om avhandlingen är tillförlitlig eller ej (Thanasegaran, 2009). Enligt Burton, Conway och Holgate (2000) kan man testa huruvida tillförlitlig en uppsats eller avhandling är utifrån en rad olika faktorer. En faktor är instrumentell reliabilitet vilket innebär den utrustningen och tillvägagångssättet som använts för att samla in empiriska data (Burton, Conway & Holgate, 2000). För att beskriva detta kan man fråga sig själv: “Kan denna metod replikeras med samma utfall med samma instrument?” (Oates, Griffiths & McLean, 2022). Detta som kan komma att påverka eller influera trovärdigheten av denna faktor är exempelvis att det mätinstrument som används för att samla in data inte återger den exakta data utan innehåller fel eller en felmarginal som i sådana fall måste tas med i beräkningen (Burton, Conway & Holgate, 2000). I vårt fall där vi använder oss av en intervjumetod via digitalt verktyg kan detta förekomma i form av tekniska problem men instrumentell mätning kvalar inte in som ett problem här, däremot hade vi det i åtanke under arbetets gång.

En annan faktor handlar om effekten av hur intervjuaren agerar både i val av utseende, klädsel och tal under intervjun också kallad intervjueffekten (Burton, Conway & Holgate 2000). Detta är något vi hade med oss i bakhuvudet under intervjuerna och försökte konsekvent att utstråla samma helhetsintryck inför varje intervju genom att bland ha liknande kläder och ett neutralt och öppet förhållningssätt till intervjupersonen. Även i vårt fall med en semistrukturerad intervjumetod spelar det roll hur intervjufrågorna ställs för reliabiliteten (Oates, Griffiths & McLean, 2022).

En annan faktor inom reliabilitet handlar om den mänskliga faktorn (Burton, Conway & Holgate 2000; Jacobsen 2002). Detta kan illustreras på en rad olika sätt, det kan handla om slarv vid hantering av den empiriska data exempelvis transkriberingsfelaktigheter. För att säkerställa att den faktorn påverkade vår studie så lite som möjligt valde vi att spela in intervjuerna som genomfördes och sedan dubbeltranskribera dem. Detta möjliggjorde att vi hittade fel eller slarv som den andre hade missat under sin transkribering. Att vi också valde att kategorisera den empiriska data med citat och begreppsunderlag säkerställde att vi inte missade några

relevanta bitar från intervjuerna. Eftersom vi genomförde dataanalysen grundligt vilket är en viktig aspekt enligt (Oates, Griffiths & McLean, 2022), anser vi att vi har täckt denna faktor av reliabilitet.

3.6.2 Validitet

Validitet är ett begrepp som hänvisar till trovärdigheten för uppsatsen och svaren på den forskningsfråga som undersöks (Oates, Griffiths & McLean, 2022). Man ifrågasätter helt enkelt relevansen av den insamlade data samt metod för insamlande av denna data för att kunna säkerställa att resultatet av undersökningen är pålitligt. Validitet är därmed viktigt när man formar sig egna undersökning samt när man bedömer någon annans (Oates, Griffiths & McLean, 2022). Opålitlig forskning bör undvikas, både från eget håll och att citera andras (Jacobsen, 2002).

Validitet delas in i två kategorier, extern validitet och intern validitet (Oates, Griffiths & McLean, 2022). Extern validitet handlar om möjligheten att generalisera kring det som uppdagats i undersökningen (Oates, Griffiths & McLean, 2022). Intern validitet handlar om att säkerställa att insamlingen av empirin är vad som ansågs och att den är giltig (Jacobsen, 2002; Oates, Griffiths & McLean, 2022). För att uppnå hög validitet är det viktigt med metodval, val av litteratur att använda, utformning av undersökning, urval av intervjuobjekt osv. Det går till exempel inte att börja med intervjuer för att sedan skriva litteraturkapitlet, risken är då väldigt hög att man är ”färgad” av resultaten av intervjun och endast letar efter litteratur som bekräftar det man tror är sant efter intervjuerna, detta bidrar till låg validitet (Oates, Griffiths & McLean, 2022).

För att uppnå extern och intern validitet grundades undersökningen på befintlig litteratur inom området DevOps. Urvalet av litteratur baserades på en kombination av mängden citeringar, författarnas trovärdighet, publikationens status samt hur ny artikeln är. Litteraturen låg helt till grund till de intervjufrågor som sedan formulerades och ställdes till intervjuobjekten, frågorna var det sista som konstruerades innan empirin samlades in för att säkerställa validitet och undvika partiskhet i konstruktionen av frågorna. Vidare ville vi säkerställa att de svar vi får speglar industrin och kan användas för att generalisera och i alla fall kan tolkas som en indikation och fingervisning för företag som inte deltog i undersökningen. Denna externa validitet uppnås genom att noggrant välja ut intervjuobjekt. Fokus lades på personer med lång erfarenhet och högt uppsatt position inom en organisation som under de senaste åren valt att implementera DevOps. Personer som endast arbetar inom en DevOps organisation undveks då de befarades ta en för personlig vy och inte ha ett helikopterperspektiv på hela organisationen.

Den externa validiteten blir tyvärr lite lidande då endast fyra intervjuobjekt intervjuades. Detta berodde dels på tidsbrist, dels svårigheter att hitta rätt sorts intervjuobjekt som var villiga att ställa upp.

3.7 Etik

En vetenskaplig studie kan ibland stöta på etiska dilemman och deltagarna som deltar i studien bör behandlas med respekt och bör inte utsättas för någon psykisk, fysisk, social eller ekonomisk skada, och detsamma bör gälla för de företag / organisationer där de arbetar (Jacobsen, 2017; Oates, Griffiths & McLean, 2022). Graden av dessa kan vara varierande, till

exempel stöter vi inte på några stora etiska dilemman såsom testning av produkter på djur eller människor, trots detta är denna studie inte helt skonad från att åtminstone ha etik i åtanke när studien genomförs. Tyvärr finns det ej något ramverk eller tydliga regler om vad som är rätt och fel vilket gör att fokus bör läggas på vilken etisk utgångspunkt man som forskare har (Jacobsen, 2017). Det är fullt möjligt att konstruera en studie som inte bryter mot några lagar men som däremot inte är etiskt korrekt (Oates, Griffiths & McLean, 2022), därmed måste vissa avväganden göras innan studien genomförs.

Det finns tre krav som Jacobsen (2002) menar på absolut måste uppfyllas inom den forskningsetik som råder och dessa används som utgångspunkt för denna studie och att uppfylla dessa har varit en strävan. Dessa krav och vilka rättigheter man som deltagare har är krav på privatliv, krav på att bli korrekt återgiven och informerat samtycke (Jacobsen, 2002; Oates, Griffiths & McLean, 2022).

3.7.1 Krav på privatliv

I undersökningar så som denna finns det alltid en risk att någon form av känslig information kommer upp. Detta kan vara information om organisationen en individ arbetar i eller om dennes privatliv, värdering av hur känslig informationen är måste göras och sedan måste steg för att anonymisera data tas om detta anses nödvändigt (Oates, Griffiths & McLean, 2022). Information som kan användas för att identifiera en individ som bett om anonymisering måste handskas extra varsamt. Anses detta vara en risk ska data anonymiseras (Jacobsen, 2002; Oates, Griffiths & McLean, 2022). Exempel på hur detta görs kan vara att ta bort information om ålder, kön och boendeort. Andra taktiker kan vara att inte beskriva en respondent så detaljerat, användning av fraser som ”lång erfarenhet i branschen” istället för ”Har arbetat på företag X i 8 år”. Även organisationers namn ska hållas dolda om man inte explicit fått tillåtelse att nämna dem vid namn (Oates, Griffiths & McLean, 2022).

3.7.2 Krav på att bli korrekt återgiven

Att återge insamlade data är på ett korrekt sätt är något av det viktigaste vid en undersökning (Oates, Griffiths & McLean, 2022). Tyvärr är det väldigt svårt att uppnå fullständig återgivning. Att analysera data innebär att detaljer reduceras och kanske att olika datapunkter får olika mycket fokus i analysen. Med det sagt ska en intervju återges i sin helhet och riktiga sammanhang (Jacobsen, 2002; Oates, Griffiths & McLean, 2022).

3.7.3 Informerat samtycke

Det absolut viktigaste är att en person som deltar i en undersökning gör det frivilligt på eget bevåg, personen ska ha blivit informerad om vilka eventuella vinster och risker som är associerade med medverkande (Jacobsen, 2002; Oates, Griffiths & McLean, 2022). Detta kan visa sig svårt i verkligheten. Att lämna all bakgrund angående en studie är ofta farligt, det kan resultera i att medverkan blir för tidskrävande samt att informationsutträttning uppstår hos den medverkande. Detta kan göra att personen väljer att tacka nej till medverkande eller ännu värre, medvetet anpassar sina svar för att passa eller inte passa undersökningen (Oates, Griffiths & McLean, 2022). ”Tillräcklig information” måste dock förmedlas i förväg. Detta innefattar studiens syfte och hur och till vad resultaten kommer användas (Oates, Griffiths & McLean, 2022).

3.7.4 Tillvägagångssätt

För att hitta intervjuobjekt och deltagare till studien använde vi oss av de digitala plattformarna vi ansåg passade bäst för att hitta och identifiera kandidater vi ansåg mötte våra kriterier. Primärt använde vi oss av LinkedIn för att identifiera kandidater och e-mail för att kontakta dem. Vi skrev ett kort meddelande som innehöll en presentation av oss, vårt program samt studien vi utförde. Vi inkluderade även information om intervjun, vilken typ av frågor samt hur genomförandet av denna skulle gå till.

Var personen i fråga intresserad av att delta bestämde vi en träff på tid och datum som passade alla involverade parter över Zoom eller Microsoft-Teams alternativt en fysisk träff som var fallet för respondent 2.

Varje intervju började med de formella artigheterna, presentation av varje person och information om hur upplägget skulle vara. Vi presenterade vår forskningsfråga och förklarade syftet med vår undersökning. Detta gjorde vi för att repetera premisserna samt se till att inga missförstånd hade skett och att personen förstod vad det innebar att vara med i undersökningen. Innan intervjun satte i gång på riktigt bad vi om tillåtelse att spela in intervjun i syfte av transkribering samt frågade om personen och dess organisation ville bli anonymiserad eller om vi kunde använda namn etcetera. Samtliga intervjuobjekt tillät inspelning av ljud men några valde att bli anonymiserade i studien. Vi erbjöd även intervjuobjektet en chans att ta del av transkriberingen för att säkerställa att vi återger den korrekt men ingen tog upp oss på detta erbjudande. Efter detta startade vi inspelningen och utförde intervjun. De som ville bli anonymiserade blev detta i transkriberingen och genom hela studien.

Vi anser inte att vår studie vidrört något känsligt material för vare sig individ eller organisation men förstår självklart att man vill vara anonym ändå. Att ett objekt är anonymt kan också vara en fördel då vi märkte att de som ville förbli anonyma uttryckte sig mer fritt och i många fall gav väldigt givande och intressanta svar. Särskilt när det kom till frågor som berörde svårigheter, utmaningar och nackdelar relaterade till ämnet.

Efter intervjuerna transkriberades dessa ordagrant och sammanfattades och analyserades sedan. Transkriberingarna finns med i studien som en bilaga så det inte råder några tvivel om att intervjuerna blivit korrekt återgivna.

4 Empiriska resultat

I detta kapitel presenteras resultaten från vår empiriska undersökning. Fynd och sammanfattningar lyfts fram under rubriker som härstammar från vårt undersökningsramverk och från våra undersökningsområden under intervjuerna. Kapitlet är uppdelat så att respondenternas svar hålls isär från varandras och respondenterna benämns utifrån tabell 2: **Respondenter, bransch och intervjuinformation** och tabellerna som återfinns i appendix *med identifikation R1-R4. Transkriberingar återfinns i Appendix A-D.*

4.1 Organisationsstruktur

R1 beskriver att det har förekommit en hel del förändringar efter introduktionen av DevOps när det gäller organisationsstrukturen, R1 beskriver detta enligt följande:

> "Idag implementerar vi safe, en scaled agile setup och den hjälper. Vi Cross-functional teams och det är dem som är den viktigaste komponenten i hela DevOps. Du har teams som består av arkitekter, utvecklare och operations-folk, alltid i samma team. Vi bygger teamen på ren kompetens, medlemmarna kan också flyttas runt till andra teams om det behövs och detta görs ofta för att optimera organisationen och den kompetens vi besitter" (Appendix A, r26).

R1 redogör att det skett stora organisatoriska förändringar sedan de tog beslutet att implementera DevOps. Tidigare använde man sig av en klassisk silo-struktur. Utveckling var en avdelning och drift en annan. Utvecklarna utvecklade och lämnade sedan över driften. Idag arbetar man med Cross-functional teams, de är den viktigaste komponenten i deras DevOps-struktur. Det finns cirka 50 olika teams, alla dessa består av en blandning av arkitekter, utvecklare och driftpersonal. Nu blandas rollerna inom ett team istället för att skicka ansvaret vidare till de andra avdelningarna och organisationen som helhet har gått från att vara avdelningsbaserad till att vara team-orienterad. All personal kan förflyttas mellan team om det skulle behövas, beroende på om resurser behöver allokeras om. Alla team koordineras med ett releasetåg och det är viktigt att alla jobbar åt samma håll. De arbetar med kontinuerliga releaser vilket underlättas när avståndet mellan utveckling och drift har minskat och båda rollerna finns i samma team.

R1 fortsätter med att beskriva om hur beslutet togs att gå över till den här typen av organisationsstruktur och arbetssätt:

> "Jag tror en hel del kom från ledningen, de bestämde att vi ville ligga i framkant och vara tidiga med DevOps, vi ville experimentera och vara väldigt tidiga med det här sättet att jobba och bygga vidare på de agila metoderna" (Appendix A, r8).

R1 beskriver att det är viktigt att hela organisationen är involverad i en förändringsprocess och att man verkligen förklarar och förankrar beslutet i hela organisationen. R1 menar på att detta är en högt prioriterad vision inom ledningen av hans företag och därför har de lyckats genomföra denna implementering av DevOps.

Efter att ha frågat R2 om hur deras organisationsstruktur ser ut svarar R2 följande:

> "Vi har tagit ett agilt arbetssätt med hjälp av Scaled agile framework och på så vis jobbar vi med DevOps. Man har länge jobbat med sprintar och sedan tweakat det utifrån behoven med DevOps på vårt företag eftersom vi jobbar med både strategi och teknik. Därför måste projektmodellen också kunna hantera kravhanteringen från båda hållen" (Appendix B, r18).

R2 beskriver sin organisationsstruktur som en traditionell hierarki då företaget härstammar därifrån och är inte ett renodlat teknikföretag utan arbetar mycket med strategi också. Däremot har de stora projekten på företaget anammat en mer flytande organisationsstruktur där projektteamen är fyllda med en variation av olika kompetenser som samarbetar på ett jämnt spelfält för att uppnå sina mål. Man genomförde denna förändring primärt genom att skapa en egen modell av Scaled agile framework och sedan tweaka och utveckla ett eget arbetssätt med DevOps utifrån det. Däremot finns det ändå en struktur med en projektledare inom varje projekt vilket R2 benämner som en nödvändighet för att klara kravbilder och avrapportering.

Här får vi en konkret beskrivning av R3 där hen förklarar hur organisationen är uppbyggd och vart ansvar ligger i organisationen. Väldigt intressant är att om man tar det hen säger och läser lite mellan raderna så är hen inte helt nöjd och känner i vissa lägen att företaget inte givit DevOps de bästa förutsättningarna att lyckas. R3 förklarar att:

> ”Sen har vi delat upp våra medarbetare i olika produkt- eller leveransteam och där är ju alltså hela syftet med de agila tankarna eller ett av det stora syftet med de agila tankarna är att försöka få självorganiserande team det vill säga jag eller ingen annan för den delen ska behöva gå in och micro managera utan teamet ska ha den kompetensen och det förtroendet så att man kan lösa sina dagliga arbetsrelaterade problem” (Appendix C, r2).

R3 fortsätter och fyller i med:

> ”Nu har vi gemensam driftavdelning och gemensam sälj vilket har inneburit förändring och det är bra förändringar men vårt DevOps arbete blir lite lidande tyvärr. I denna fråga lyssnade de inte på mig” (Appendix C, r14).

Medarbetarna är uppdelade i olika produktteam som är kopplade till vilket affärsområde de utvecklar för. De använder sig med andra ord av silos runt varje affärsområde. Medarbetarna byter sällan team utan håller sig i sin silo. Varje team består av en produktchef, leveranschef och utvecklare varken personer med drift- eller arkitekturkompetens ingår i teamet. De som har driftkunskap finns på en helt annan avdelning och de med arkitekturkompetens finns i ett separat team vars produkt är just arkitekturen och deras roll är att supporta övriga team. De använder sig således av silostruktur med ett litet inslag av plattformsteam när det kommer till arkitekturen. Silostrukturen är R3 inte helt nöjd med, den skapar problem med alignment, kommunikation och kompetensdelande, många problem uppstår på grund av denna struktur. Det största problemet är dock att driftpersonalen inte ingår i produktteamen. Det är detta som är kärnan i DevOps och det går man miste om med den valda strukturen. Vidare beskriver R3 detta som:

> ”Vi har en intern gruppering som vi kallar för “DevOps-intra” vars huvudsyfte var att vara med så att man får just den här driftsperspektivet för annars faller hela DevOps konceptet. Från och med måndag så kommer en ny organisationsstruktur och då är

> tyvärr den gruppen bortflyttad från oss så Ops delen av DevOps brottas vi med just nu tyvärr" (Appendix C, r7).

R4 benämner organisationsstruktur på följande vis:

> "And you know, apart from business culture, like in a multinational organization, between countries, there are cultures that are different, that reflect the way businesses are structured" (Appendix D, r21).

R4 beskriver sin organisationsstruktur och den faller snabbt in i kategorin Plattforms-team

> "We have various dev teams that are responsible for their own products and their own kind of specialties that they work on. Then the DevOps team is responsible for stuff, like infrastructure operations, and looking after CI" (Appendix D, r5)

Varje team är ansvarig för en produkt. Sedan finns det ett specifikt team som de kallar "DevOps-team". Detta team ansvarar för infrastruktur och drift. Detta ska inte ses som endast support utan deras produkt är infrastrukturen som de andra produkterna körs på. Således är driftpersonalen samlad i "DevOps-teamet" och övriga produktteam består av en produktansvarig och utvecklare.

R4 menar på att organisationsstrukturen reflekteras av kulturen som finns i det landet och reflekteras i hur företagen de arbetar med är strukturerade, R4 fortsätter med att tillägga:

> "The way business hierarchies are how power is shared or delegated within organizations. So, I don't really like to put any kind of value judgment on what's better or worse, just as long as it's not abusing people. And it's working for that organization, like different structures" (Appendix D, r21).

R4 vill inte lägga någon värdering eller favorisera någon organisationsstruktur över en annan utan menar på att så länge ingen i organisationen utnyttjas och allt fungerar så är det bra. R4 nämner dock att det är viktigt att ha en tydlig struktur och framför allt en tydlig vision med sitt DevOps arbete för att lyckas och även involvera alla avdelningar som berörs.

4.2 Kommunikation

Efter att R1 mottog en fråga gällande om hens företag använder sig av några gemensamma plattformar eller verktyg för kommunikationen mellan utvecklare och driftpersonal hos dem svarade R1 följande:

> "Vi är multinationatienella och vi har haft Covid senaste två åren. Vi har Microsoft Teams ju som vi sitter på nu, vi har gemensam infrastruktur med login och diverse verktyg, jag ska inte säga vad de heter för då kan man lista ut vilket företag jag arbetar på. Men det är serviceverktyg där vi också kan kommunicera med varandra, så ja det finns gemensamma verktyg för kommunikation" (Appendix A, r10).

R1 klargör att kommunikation är extremt viktigt. Kommunikationen ska helst vara direkt. Ett ticket-system används för att håll koll på ärenden och kommunikation och är essentiellt då det är en väldigt stor organisation men kommunikationen ska helst ske direkt då detta ger bättre

och snabbare resultat. Face-to-face sker inte så ofta då det är en multinationell organisation och det har varit Covid de senaste åren, digitala verktyg används för den direkta kommunikationen istället, några av dessa är Slack, Teams samt en egenutvecklad produkt. Respondenten menar dock att inget av detta är revolutionerande utan den huvudsakliga fördelen med DevOps inom kommunikation är att utvecklare och driftpersonal nu ingår i samma arbetslag. Tidigare har de varit separerade och endast kommunicerat genom ticket-systemet, utvecklarna ”Kastade färdig kod över till andra sidan staketet” (Appendix A, r29) och sedan var det upp till driftpersonalen att sköta driften, hade de frågor fick de öppna ett ärende i ticket-systemet. Nu ingår driftpersonalen i de tvärfunktionella arbetslagen redan från start vilket gör att de är med på hela resan och det är huvudsakligen inom dessa arbetslag den direkta och kontinuerliga kommunikationen äger rum.

Efter att R2 fått frågan: “Hur kommunicerar DevOps-teamen? Används gemensamma plattformar för att se resultat och framsteg? Kommunikationsverktyg? (Appendix B, r27) Svarade R2 enligt följande:

> “Alla kan se allt via Microsofts DevOps Azure molntjänst och sedan använder vi också Slack för snabb kommunikation. Alla i teamet har access till det här verktyget. Det här verktyget styr också kvalitetsgranskning. Vi har också automatiserat hela uppsättning av infrastrukturen i Azure. Vi har också kopplat notiser från DevOps verktyget in i slack exempelvis för kvalitetssäkring så får jourpersonalen en notis om det” (Appendix B, r28).

R2 redogör för att kommunikation är en viktig komponent inom DevOps och nämner att de använder sig av Microsoft Azure DevOps som ett verktyg för att kommunicera mellan utvecklare till utvecklare men också utvecklare till verksamhet. Via detta verktyg har stora delar av den interna kommunikationen automatiserats exempelvis när det gäller larm om buggar eller kritiska fel i produktkedjan dyker detta upp som en notifiering i Azure DevOps och vidarebefordras senare till deras interna kommunikationskanal via Slack där problemet hanteras av den som får notifieringen.

R2 berättar också att kommunikationen sker till största del digitalt via dessa två verktyg och hela företaget har inblick i dessa system. Dessa verktyg har höjt produktiviteten, då det möjliggör för utvecklare att se varandras framsteg men även för andra intressenter som lättare kan stämma av. Däremot menar R2 på att innovation och kreativitet är något som bäst kommuniceras ansikte mot ansikte. R2 beskriver att det är svårt att förmedla kreativa lösningar digitalt och spontanitet vid kaffemaskinen är en viktig faktor inom brainstorming.

R3 var direkt väldigt tydlig med vikten av kommunikation när man arbetar med DevOps, både inom teamen men framför allt lyfts många intressanta tankar angående kommunikation mellan team och hur man kan förbättra detta och framför allt möjliggöra kompetensdelning mellan personal som arbetar på olika projekt och i olika affärsområden. R3 beskriver detta fenomen på följande vis:

> ”vi kör ju SCRUM så vi har det här med sprintdemos allting sådant som är inbyggd i det arbetssättet och vi jobbar mycket med workshops och annat tillsammans med verksamheten för att försöka sprida kunskap och erfarenheter men det är en work in progress för oss hur vi ska kommunicera. Den stora utmaningen är hur man kommunicerar mellan team. Eftersom vi är indelade i affärsområden där personalen inte byter område så bildas silos runt dessa områden och silos är aldrig bra. Vi försöker motverka

> detta genom att alla dessa silos står på en grund där vi försöker samla kunskap. Denna plattform kallar vi Steam och i denna plattform har vi våra gemensamma bibliotek, ramverk, kod-standarder, säkerhetsprotokoll som alla måste förhålla sig till så det är ett sätt att kommunicera de tekniska kraven i alla fall men vi har svårt att hitta ett bra forum för kommunikation utvecklare till utvecklare mellan teamen" (Appendix C, r9).

R3 redogör för att kommunikation inom team och vidare till linjechef fungerar bra och skapar inte några problem för arbetet. De plattformar och verktyg som används gör det lätt att ta del av framsteg, planering och backlog, och det är lätt att få en överblick av ett projekt, här har SCRUM och de agila metoderna varit till stor hjälp och det är en märkbar skillnad på kommunikation inom teamen sedan dessa infördes. Trots detta uppstår problem i kommunikation mellan team och kunskapsdelningen fungerar inte på ett optimalt sätt. R3 förklarar att deras organisationsstruktur är en stor faktor i detta. Eftersom de har silos runt varje affärsområde blir kommunikationen svår. Utvecklare förflyttas inte mellan silos utan man har sin silo och sitt team. För att försöka få till kunskapsdelning och affärsutveckling har många modeller testats, workshops, veckomöten, planering bland andra. R3 är medveten om detta problem och de testar sig fram för att hitta en modell som kan fungera för dem men inser att de är långt ifrån att ha en lösning eller koncept på plats som fungerar och arbetar lite i motvind med tanke på hur organisationen är strukturerad. I deras fall ger inte deras silos de bästa förutsättningarna för kunskapsdelande och effektiv kommunikation.

R4 beskriver kommunikationen på deras företag fungerar enligt följande:

> "So usually, communication, we try to do it asynchronously. Or just a lot of it happens asynchronously because people are in different locations. If you want other people to be able to see the history of your discussion later and then you need to have a discussion about it in either a public channel like public within the organization, not necessarily public to the world. But like in a chat channel in whichever chat program you use, or you need to record your conversation in that issue tracking software that you use. So where people can attack that and get a notification that this issue they're working on and has your feedback in it, they can go and address it. If you're just working on things that are kind of just figuring things out, then you can just have either direct messaging in a chat program, or to get on a call with somebody if you really have to have real time communication. So real time communication is the fastest for getting certain things done" (Appendix D, r11).

R4 berättar att på grund av hen arbetar på ett multinationellt företag med kontor över hela världen är det viktigt att kommunikation sker skriftligt på många sätt. Detta menar R4 hjälper mot att saker och ting glöms bort och inte hanteras. Men även för att människor som arbetar i olika tidszoner ska kunna vid ett senare tillfälle kunna gå in och läsa vad som har skrivits och diskuterats. R4 berättar att det finns kommunikationsverktyg som fungerar som stora chattforum där utvecklare kan se varandras framsteg och även problemområden man behöver hjälp med. Däremot ifall man har ett akut problem är det bättre att ta ett videosamtal eller skicka ett direktmeddelande och menar på att realtidshantering är att föredra i den typen av situationer. Däremot menar R4 på att diskutera idéer och nya projekt i uppstartningsfasen inte fungerar speciellt bra på videolänk vilket hen beskriver som:

> "But for working on a project, for example, like early stage fleshing out, new system it doesn't go as well to have a video call, you need to sit there and think about it and

> write your ideas out and then like, give the next person provide their feedback as well" (Appendix D, r11)

R4 menar på att detta kräver mer direkt kommunikation så att feedback och motargument enklare kan förmedlas.

4.3 Samarbete och lagarbete

När R1 fick blev tillfrågad hur samarbetet inom DevOps sker på hens företag svarade R1:

> "Vi har DevOps när vi utvecklar men det blir lite olika när man har många teams. Vi har väldigt många teams, det finns Dev-folk men också Ops-folk som är med i utveckling för att få bättre förståelse för produkten" (Appendix A, r14).

R1 svarar också enligt följande gällande driftpersonalens involvering i utvecklingen:

> "Är inte driftpersonalen med i utvecklingen kan det bli väldigt dyrt att drifta ett system" (Appendix A, r12).

R1 såg detta område som ett av de viktigaste för sin organisations framgång med DevOps. Syftet med DevOps var från första början att minska distansen mellan utveckling och drift. Driftpersonalen har svårt att förstå systemet om de inte varit med end-to-end. Målsättningen i organisationen är att allting tillslut ska bli Dev. Drift vill man i största utsträckning automatisera och därmed förvandla det till development snarare än operations. Att driftpersonal ingår i samma SCRUM-teams som utvecklare från dag 1 i ett projekt har varit nyckeln för att minska avståndet mellan utveckling och drift. En separat supportavdelning finns kvar men operations närmar sig mer och mer development och är med på hela resan från start till mål. Detta möjliggör även snabb och direkt feedback mellan de traditionellt separata avdelningarna och rollerna. En drifttekniker som sitter med under utvecklingen kan numera komma med feedback direkt angående kod som kan bli dyr i drift i ett senare tillfälle. Detta har resulterat i ett effektivare arbetsflöde med bättre resultat än tidigare.

R2 beskriver att samarbete och lagarbete är A och O och beskriver det enligt följande som svar på frågan "hur samarbete och lagarbete ser ut för dem i förhållande till DevOps?" (Appendix B, r19).

> "Vi gör faktiskt ingen differentiering på utvecklare och driftpersonal. Vi jobbar inte med ren förvaltning" (Appendix B, r20).

R2 beskriver att samarbete inom DevOps är viktig men nämner också att de gör ingen skillnad på utvecklare och driftpersonal utan har mer anammat en struktur där det är en och samma person/personer som tar ansvar för hela produktkedjan. I övrigt består deras team av en projektledare, en systemarkitekt, en scrummaster, utvecklare och en person som sysslar med UI-design. R2 berättar att hela teamet lär sig hur systemen är uppbyggda från grunden vilket innebär att alla i teamet kan ta ansvar för buggar och problem under projektets gång. R2 säger också att det finns ett underteam som har fokus på en viss funktion och dessa funktionsgrupper har specialkompetens inom just det området.

Efter att R2 fått frågan om hur drift av system sköts beskriver R2 att detta fungerar ungefär som en jourverksamhet där det finns ett rullande schema för utvecklarna att vara stand-by för driftstörningar under veckas tid dygnet runt. Kritiska problem hanteras direkt och kräver en allokering av resurser till problemet direkt och genomförs utan att produkten tas ur bruk medan mindre kritiska problem läggs i backloggen och åtgärdas vid nästa lansering av nuvarande sprint. Detta innebär att det finns ett stort samarbete inom teamet på alla fronter och R2 säger att kompetens är en väldigt viktig komponent för att samarbete skall fungera så smidigt som möjligt. R2 beskriver att man inte vill åta sig konsultuppdrag utan att hela teamet är involverat för att kunna nyttja all kompetens och samarbetsvilja.

R3 redogör för att detta var en av huvudanledningarna till att DevOps och de agila metoderna infördes. Att skapa team där de tillsammans kunde ta sig an uppgifter var en stor drivkraft i att implementera DevOps. R3 berättar hur de arbetssättet såg ut innan DevOps;

> ” Det var faktiskt så här att i praktiken skrev ägaren varje morgon en lista med typ 10 grejer, det här vill jag ha utvecklat nu och så gick hen med den till min företrädare hon som var gruppchefer för det för den här gruppen innan. Hen lämnade listan och sen var det bara att börja. Det är inte så mycket pull i detta arbetssätt, man tar inget eget ansvar utan man blir tilldelad exakt vad man ska göra varje dag” (Appendix C, r16).

Föregående arbetssätt var väldigt ad hoc. De hade en ägare som var inne och petade och utövade full kontroll ända ner på funktionsnivå och prioriteringslista. Utvecklarna fick individuella arbetsuppgifter där samarbete knappt uppmuntrades, självklart kunde man ta hjälp av kollegor men det fanns ingen struktur eller forum för att underlätta detta. Här har konstruktionen av arbetslag och de agila metoderna gjort underverk. Nu tillhör alla ett team som har ett gemensamt mål som de tillsammans ska uppnå. De kan dela upp arbetsuppgifter beroende på vilka kompetenser de olika medarbetarna besitter vilket gör att de också kan ta lärdom av varandras kompetenser och erfarenheter. Det säger sig självt att lagarbetet förbättras när man tillhör ett lag men det tåls att betona ändå. Återigen har de en lång väg att gå, framför allt mellan de olika lagen och återigen arbetar organisationsstrukturen och silosen emot detta. De hade även velat få till ett närmare samarbete mellan utvecklare och driftpersonal. Det finns ingen driftpersonal i arbetslagen och detta gör att DevOps-konceptet faller lite. De har kommit en bra bit med de agila metoderna men ren och skär DevOps har de inte lyckats uppnå märker vi ju mer R3 berättar om företaget. Målet för framtiden är att hitta en struktur där det blir möjligt för utvecklare och driftpersonal att samarbeta närmare. Återigen går detta problem tillbaka till organisationsstrukturen och deras silos.

R4 beskriver samarbete och lagarbete i förhållande till DevOps enligt följande:

> “Yeah, so the way we have it is kind of very general, which is a way that I quite like because we have various dev teams that are responsible for their own products and their own kind of specialties that they work on. Then the DevOps team is responsible for stuff, like infrastructure operations, and looking after CI and stuff, we look after on one hand, operations like shared infrastructure” (Appendix D, r5).

Vad som framkommer när R4 berättar om samarbete och lagarbete på hens företag är att det finns en tydlig team-struktur där det finns några utvecklar team som arbetar med speciella funktioner och fokuserar primärt på det. Sedan finns det ett universellt DevOps team som är ansvarig för drift och den gemensamma infrastrukturen. R4 beskriver också att utvecklarna tar

ett större ansvar i produktkedjan av utvecklingen av nya system, detta beskrivs på följande vis:

> "It's about moving the responsibility of infrastructure towards the development of the developers and also moving the responsibility of the developers of looking after their own stuff" (Appendix D, r7).

4.4 Kontinuerlig integration, leverans och kvalitetssäkring

R1 beskriver deras kontinuerlig integration, leverans och kvalitetssäkringsprocess enligt följande:

> "Vi har en hel testarkitektur, vi vill ha så mycket information som möjligt från vår CI-kedja och vi vill se direkt när något går fel för att kunna fixa det så snabbt som möjligt. Vi arbetar väldigt mycket automatiserade tester på enhetsnivå, systemnivå och hela vägen upp. Man ska inte ta en hel miljö åt gången utan man bryter ned testerna på komponentnivå, man ska äga den kod man skriver och man ska se till att det funkar. Men som sagt, väldigt mycket stöd av testning på alla nivåer" (Appendix A, r22).

R1 säger att kontinuerlig integration, leverans och kvalitetssäkring är hörnstenar när det kommer till utveckling och DevOps. Väldigt mycket fokus läggs på detta och målet är att automatisera så mycket som möjligt. Det finns en testarkitektur på plats och dessa bryts ner på komponentnivå. Tester och kvalitetssäkring görs inom samma SCRUM-team som äger hela processen end-to-end vilket har gjort att utvecklarna nu kan tänka sig att automatisera tester istället för att lämna över sin kod till driftpersonal som tidigare stod för testerna. Krav och regler finns självklart på plats för detta men organisationens ledord är built-in-quality. Man ska äga sin kod från början till slut och här gör den snabba och direkta feedbacken ett bra jobb då saker kan åtgärdas tidigare i processen än tidigare.

R2 beskriver att arbetet med kontinuerlig integration, leverans och kvalitetssäkring är viktigt för dem men ser det inte som ett eget område utan snarare som en integrerad del av deras arbetssätt. Efter frågan hur deras arbetsprocess med kontinuerlig integration och kvalitetssäkring fungerar svarar R2 enligt följande:

> "Generellt har vi två veckors sprintar och lanserar efter det för att uppnå kontinuerlig integration och kontinuerlig leverans. Dessa leveranser går sedan vidare till en testmiljö där vår kund och våra utvecklare har möjlighet att testa själva hur det fungerar och vi får feedback på detta. För att det sedan skall gå vidare till slutgiltlig lansering behövs det tre personer samtidigt en testare, en kvalitetsgranskare och en utvecklare för att godkänna releasen sedan när kunden också har godkänt så lanseras produkten live. Det tar ungefär 3–4 veckor från att vi börjar koda till vi har något på marknaden live" (Appendix B, r24).

R2 fortsätter och beskriver att deras kvalitetsarbete är till stor del automatiserat med hjälp av API:er. R2 beskriver även att deras två kommunikationsverktyg Microsoft Azure DevOps och Slack är integrerade med larm om buggar eller brister i testmiljön vilket underlättar deras kvalitetssäkringsprocess.

Efter att frågat R3 hur de jobbar med kontinuerlig integration, leverans och kvalitetssäkring svarar R3:

> "Vi lägger mycket fokus på automatisering till exempel de ekonomiska flödena kan vi ju skära bort otroligt mycket manuell handpåläggning och skapa dokumentation för hur detta fungerar. De olika produktägarna håller reda på vad som ligger i backloggen tillsammans med processägaren och väljer vad som ska prioriteras" (Appendix C, r4).

R3 fortsätter med att beskriva processen med testning såhär:

> "Sen jobbar vi mycket med testning och intern kvalitetssäkring och så där innan kunden tar del av lösningen" (Appendix C, r5).

R3 är noga med att poängtera att de inte är klara med sitt automatiseringsarbete utan att de har ambitionen att förbättra det de kommande åren. Det viktigaste för R3 är att automatisera bort onödiga manuella processer som också kräver manuell handledning. R3 menar på att automatisering är en bra funktion men vill samtidigt inte automatisera bort allt utan exempelvis ha kvar manuella tester för kvalitetssäkring och kvalitetshantering.

R4 beskriver deras automatiseringsprocess och deras arbete med kontinuerlig integration och leverans enligt följande efter att ha mottagit en fråga om hur det arbetar med kvalitetssäkring, kontinuerlig integration och leverans:

> "In general, there are automated tests and there are manual tests. So, I think one of the most important things that's happened in the last like 10 years is this rise in tooling around. So automated pipelines for moving code specifically, from the Devs through to the production environment and taking it through automated testing" (Appendix D, r15).

R4 fortsätter och säger:

> "You can go to some people who want to go all in and completely automate all the way to deployment. And that's fine depending on the particular risk profile of the product and the business that's doing it" (Appendix D, r15).

R4 belyser att det är väldigt viktigt för dem att ha både automatiska tester av deras mjukvara och system men även lägga till manuella tester som fungerar som ett extra lager av säkerhet för att säkerställa kvalité utan buggar och bekymmer. R4 berättar även generellt att detta varierar från företag till företag enligt hen så fungerar full automation hos vissa företag medan hos andra fungerar det ej. Detta beror på hur riskmedveten man är och bekväm med automation generellt.

4.5 Alignment

R1 menar att det stora underliggande temat för alla förändringar som gjorts till följd av implementationen av DevOps har och göra med att alignment. Organisationsstrukturen har ändrats, arkitektur och arbetssätt har bytts ut och förändrats och roller och avdelningar har närmat sig varandra för att möjliggöra att hela organisationen jobbar mot gemensamma mål. R1 beskriver strukturen enligt följande:

> "Absolut! Det som är viktigt i dessa är lineup. Alignment på SAFE. Vi har ett releasetåg som går hela vägen upp till solutionnivå. Så du kan ha ett arc som består av kanske 12 SCRUM-teams, vi har kanske 5 teams till VM-solution som har hand om full lösningsnivån och de här ska så klart synka med varandra, både mellan och inom teamen" (Appendix A, r28).

R1 säger att misalignment är organisationens största fiende. Målet med DevOps är att alla drar åt samma håll för att uppnå gemensamma mål. Delat ansvar mellan Dev och Ops är en stor del i detta, har man ett end-to-end ansvar för en produkt eller funktion kommer man se till att den fungerar bättre. Har en utvecklare inget ansvar för driften kommer denna inte att bry sig så mycket om driftkostnader. Det arbetar hela tiden med att se till att infrastrukturen stöttar verksamheten, infrastrukturen förbättras konstant för att utveckla DevOps inom organisationen. Detta innefattar kontinuerliga releaser utan att systemet ligger nere samt automatiserade tester. De nya Cross-functional teams är också en stor framgångsfaktor. I dessa delas ansvar och kompetens och med hjälp av feedback blir arbetslagen hela tiden bättre och drar nytta av varandras kompetens och erfarenheter.

R2 säger att organisationen var ute efter att linjera verksamheten. Tidigare hade de köpt in system som externa parter utvecklat, detta resulterade i väldigt höga driftkostnader samt begränsad flexibilitet. IT-avdelningen la nästan alla resurser på drift och ingen på innovation trots att ambitioner och idéer för förbättringar fanns.

R2 implementerade DevOps för att linjera verksamheten. Första steget var att ta hem utvecklingen från externa parter. När väl detta var gjort kunde arbetet med att jobba mot gemensamma mål börja. Första steget var att varje team tar ansvar från ax till limpa. Varje team ansvarar för att det man bygger fungerar. R2 beskriver detta fenomen enligt följande:

> "Vi ligger inte på 20% utveckling 80% drift utan snarare 60–70% utveckling och sedan 30% drift" (Appendix B, r40).

> *"*Eftersom man är involverad från början till slut vill man utvecklare ta ett större ansvar eftersom man får ta hand om sin egen skit*"* (Appendix B, r40)

Detta menar R2 har bidragit till bättre linjering. De arbetar också med flexibel infrastruktur, kommer det ny funktionalitet är man snabba på bollen och utnyttjar denna. De jobbar även med containerisering, i början flyter ett projekt på ganska bra men ju mer komplext ett system blir desto svårare blir det att uppdatera. De försöker därför bryta ner systemet på komponentnivå vilket gör att de kan jobba på en avskärmad del av systemet utan att störa de andra delarna. De nya arbetslagen har också varit till stor hjälp, driftpersonal är med från steg ett vilket R2 säger har varit den största framgångsfaktorn för deras DevOps-arbete.

R3 nämner inte alignment vid namn men vi märker att de tidigare upplevt problem med mål och processer som inte är linjerade med varandra. De har börjat adressera detta men inte kommit ända in i mål och några konkreta arbetssätt för att motverka detta har inte sjösatts. Vi uppfattar några områden där konkreta ändringar hade varit till hjälp.

> "Man vill vara produktiv och effektiv som utvecklare. Man vill vara oberoende av andra team och kunna utföra sina leveranser när man är färdig, man vill inte behöva checka och synka som man behöver om man inte fått rätt på skärningarna. Den

> vardagen jag har idag är att jag måste deploya allt samtidigt, vi har inte fått till kontinuerliga leveransen" (Appendix C, r21).

Här stöter vi på misalignment. Ett team kan vara klar med sitt projekt och redo för deployment men kan inte göra detta för att de till exempel väntar på testning av kod eller att ett annat team ska vara redo för deployment eftersom de måste göra det samtidigt. Ett team vill gå vidare men kan inte eftersom de är beroende av ett annat. En annan faktor som bidrar till misalignment är även att Ops-personalen är en beställningsfunktion från en annan avdelning. Detta betyder att Ops-personalen kan behöva prioritera driften åt kunder kontra driftperspektiv till utvecklingsteamen. Det finns en risk att de olika avdelningarna inte kommer arbeta utefter samma mål och på så sätt bidra till misalignment.

R4 beskriver eller pratar inte exakt om termen alignment men tar upp delar av det vid frågan om hens arbetsuppgifter och företaget struktur. Detta beskrivs enligt följande av R4:

> "So shared infrastructure, we look after that. We look after things that are specific tools for the developers to be able to go through a continuous integration pipeline and integrate their code in with what's already running" (Appendix D, r5).

R4 menar på att huvudmålet med deras DevOps-arbete är att hela organisationen ska arbeta åt ett och samma håll, både utvecklare och driftpersonal. För att kunna göra det menar R4 på att en gemensam infrastruktur med integration genom hela utvecklingsfasen är avgörande. De arbetar och prioriterar att infrastrukturen förbättras konstant för att fortsätta utveckla deras DevOps arbete inom organisationen. Detta innefattar kontinuerliga releaser utan att systemet ligger nere samt automatiserade tester. R4 fortsätter med att berätta om det delade ansvaret mellan utvecklare och driftpersonal:

> "It is like a shared responsibility there's some stuff that developers are never going to touch, like network staff and routers and you know, physical infrastructure in a data center, but then you want to provide an abstracted layer where developers can then come and run their own their own things, and then manage them" (Appendix D, r7).

Detta delade ansvar beskriver R4 som viktigt men att det ändå finns begränsningar där utvecklare exempelvis inte vill ta hand problem likt nätverksproblem och servrar som driftpersonal tar bättre hand om.

4.6 Företagskultur

Efter att ha fått frågan "Har det skett några kulturella ändringar med införandet av DevOps?" (Appendix A, r28) svarade R1:

> "Ja absolut! Vi har blivit mer moderna skulle jag säga. Vi använder den absolut senaste teknik, chatops och automatisering" (Appendix A, r29).

R1 fortsätter med att beskriva att de inspireras av andra företag och tittar på andra företag hur de jobbar och hur deras kultur ser ut:

> "Vi tittar på hur Facebook och Google jobbar, vi vill kunna göra många kontinuerliga och små releaser ofta, jag tror Netflix gör 2000 releaser om dagen, man ska releasa ofta och man måste ha DevOps för att göra detta" (Appendix A, r29).

R1 säger att kulturen har absolut förändrats med DevOps, Samarbeten mellan olika roller har ökat och framför allt sker mycket mer kommunikation när ett och samma team tar ett end-to-end ansvar. Man lämnar inte längre över ansvar till en annan avdelning. När de bestämde sig för att införa DevOps var en av anledningarna att bli modernare, att utforska de nya agila metoderna delvis för att effektivisera men också för att attrahera talang. De praktiska fördelarna med detta arbetssätt har också ändrat kulturen till det bättre. När utvecklare involverar sig i automatisering av testning och drift blir även kulturen mer öppen, barriärer bryts ner samt att organisationen som helhet blir modernare vilket också kan göra att de ses som en attraktiv arbetsplats. Snabb feedback ger bättre resultat men förbättrar också samarbetskulturen då det finns mer och bättre feedback att tillgå.

Efter att R2 mottagit en fråga om hur deras företagskultur hade förändrats svarar R2:

> "Absolut, vi är ju ett traditionellt bolag i vissa av våra projekt om vi tittar utanför Skånetrafiken. Det projekten är ju lite vattenfall, vi kommer in i projektet och får en kravbild och jobbar med den och levererar efter det och sen är vi färdiga. De större projekten har vi som tur är en agil arbetsmetod och den typen av företagskultur som hör till det" (Appendix B, r38).

R2 beskriver att kulturella förändringar var något som inträffade ganska naturligt i deras större projekt med kunder där man prioriterar starkt att jobba agilt med DevOps och med den företagskultur som följer med både med kunden och internt. Däremot beskriver R2 att Stratiteq är ett traditionellt företag vilket innebär att de fortfarande åtar sig projekt som har en viss kravbild och leveransförväntningar vilket innebär att då tenderar det att bli mer av vattenfallsmodellen som också har en viss företagskultur förankrad till sig. R2 beskriver att förväntningshantering är annorlunda när man jobbar agilt med DevOps. Det är svårt att sätta en tydlig deadline på när arbete är klart till skillnad från vattenfallsmodellen då projektet kontinuerligt förbättras och arbetas med i sprintar. På grund av detta beskriver R2 att hela teamet är med i hela processen vilket skapar ett djupt samarbete med nära relationer och bra företagskultur.

R3 lyfter att kulturen genomgått en stor förändring sedan de nya arbetssätten infördes. Mycket handlar om förändringar och det har inte alltid varit lätt att få med alla på resan. Alla förstår principen men ibland är det svårt att lära en gammal hund att sitta. R3 fortsätter med att beskriva fenomenet på följande sätt:

> "Är man skolad med att få dagens uppgifter kl. 09.00 så är det svårt att anpassa sig till det nya arbetssättet. Självklart finns det seniora utvecklare som tycker det nya är skitkul också så jag ska inte dra alla över samma kam men det finns individer som tycker det är jättejobbigt med förändringar" (Appendix C, r18).

Kulturen har blivit väldigt påverkad av att DevOps innebär en stor förändring för företaget. Självklart är alla för bättre samarbete, kommunikation och delande av kunskap. Större ansvar från ax till limpa är inte lika självklart. Många av de seniora på företaget har varit med i många år och var ganska vana och bekväma med att bli tilldelade en uppgift och sedan lämna ifrån sig den när man var kvar. En organisation är bara så stark som den svagaste länken och

det gäller att få med alla på förändringsarbetet. Trots att det tagit lite tid är R3 hoppfull om att det går åt rätt håll och säger:

> ”Det roliga med förändringsarbete är att sakta men säkert så hör man mindre och mindre gnäll och till slut erkänner nästan alla att det var en bra grej (Appendix C, r18).

Efter att R4 mottagit en fråga om hur företagskulturen har förändrats efter man har börjat arbeta med DevOps svarar R4 såhär:

> “But it's hard to say what's better or worse, because different companies have, you know, different requirements and different cultures. And you know, apart from business culture, like in a multinational organization, between countries, there are cultures that are different, that reflect the way businesses are structured” (Appendix D, r21).

R4 beskriver att det ser olika ut beroende på vilket företag de arbetar mot i nuläget. R4 nämner att de själva är ett multinationellt företag vilket betyder att personer från hela världen jobbar där och den lokala företagskulturen tar plats i deras gemensamma företagskultur så länders olika företagskultur influerar den kultur som finns.

4.7 Utmaningar och problemområden

Den enda utmaningen som tas upp av R1 i deras implementation är att utvecklarna inte vill arbeta med drift. R1 anser att statusen för driftpersonal är lägre än för utvecklare, därav vill de inte närma sig den delen av verksamheten. R1 beskriver detta fenomen enligt följande:

> “Utvecklare vill utveckla och vara en del av the creative process, inte drifta saker, egentligen så tar vår utveckling ansvar end-to-end men ibland vill de inte blanda sig i produktionen. De tycker det verkar läskigt men jag kommer inte på något” (Appendix A, r33).

För att motverka detta problem menar R1 att det är viktigt att företaget arbetar med automation och end-to-end ansvar. Det slutgiltiga målet är att operations försvinner och är helt automatiserat, detta är enda sättet att få utvecklare att närma sig driften, kan de koda driften och se det som ett problem som måste lösas blir de intresserade. En allmän svårighet som lyfts är att det inte finns någon definition eller regler man kan vända sig när man bygger sin DevOps organisation. Det viktigaste är att man gör det helhjärtat. Respondent 1 säger att implementera DevOps på legacysystem och med samma silostruktur kommer inte lyckas. Man måste ha arkitekturen, strukturen och arbetssättet på plats om man vill lyckas med denna förändring.

R2 har inga stora brister eller svårigheter att lyfta fram. Det enda som nämns är svårigheten att hitta kompetent personal och svårigheten att uppskatta hur lång tid ett projekt kommer att ta. Kompetensbrist verkar vara ett generellt problem för hela branschen och har inte så mycket med DevOps att göra men R2 kan ibland bli lite frustrerad av att det är svårt att uppskatta längden på ett projekt. R2 beskriver detta på följande vis:

> “Det finns ju nackdelar med alla arbetssätt. Men en sak är att det går inte riktigt att predicera hur långt tid det tar att förverkliga vissa ambitioner då nya saker kan uppstå

> längs vägen. Vi kan inte frysa allt och sätta oss ner i 3 månader och bara bryta ner kraven utan det är en mer flytande process” (Appendix B, r48).

R2 säger att de agila metoderna väger definitivt upp detta med effektivitet och bättre resultat men arbetar man med sprintar så tittar man bara 2 veckor framåt åt gången (i detta fall). Detta gör det extremt svårt att uppskatta slutdatum och detta är ett recept för frustrerande samtal med kunder. Trots detta skulle hen aldrig vilja gå ifrån DevOps.

Under intervjun tog R3 upp ett antal problemområden. Vi har redan redogjort för några av dem i tidigare avsnitt men vi tänkte sammanfatta dem här. Kommunikation och kompetensdelning mellan team, alignment och koordination mellan team, förändringsarbetet inom kulturen och få med de anställda på tåget, samt organisationsstrukturen som skulle kunna ses som grunden och orsaken till många av problemen. Utöver dessa som vi redogjort för tidigare lyfts även svårigheter med att attrahera senior kompetens samt känslan av att man aldrig blir klar med utvecklingen av DevOps som arbetssätt.

> ”Sen har vi även den externa faktorn att det blir svårt att bygga team när det är näst intill omöjligt att få tag på seniora utvecklare. Vi söker med lykta men det är verkligen en flaskhals som håller tillbaka oss” (Appendix C, r29).

Kompetensbristen verkar vara ett problem som hela branschen står inför och självklart blir det svårt att utveckla ett effektivt arbetssätt för mjukvaruutveckling om man inte har kompetent personal. En bra organisationsstruktur är inte värd mycket om man inte har utvecklare i teamen till exempel.

> ”Men ett lite frustrationsmoment är att man aldrig blir klar, det är en konstant förändringsprocess och som utvecklare i botten är man van vid att leveransen är slutet på ett projekt med DevOps-arbetet blir man aldrig klar” (Appendix C, r29).

Här har DevOps som koncept stor påverkan. Ramverk och guidelines finns det gott om men någon klar definition att jämföra sig med finns inte. Det kommer således vara omöjligt att veta när man är klar eller mäta hur långt på vägen man kommit. Detta kan självklart bli ett frustrationsmoment då man som människa gärna vill få bekräftelse på sitt arbete.

R4 beskriver de största nackdelarna med DevOps och att arbeta agilt enligt följande:

> “And the disadvantage is that it's much more difficult to hire people because you need to have a much wider range of experience in order to do this type of work” (Appendix D, r15).

R4 fortsätter med att beskriva att det krävs en högre nivå av erfarenhet och färdigheter som inte är speciellt vanligt nuförtiden:

> “And that's it, yeah, with a wide range of skill sets. It's like the basic levels for DevOps and infrastructure hiring are pretty much like you're looking at people who 15 years ago, I like senior engineers. And in order to kind of be starting off, it's really hard to find people” (Appendix D, r15).

R4 menar på att det största nackdelen med DevOps är att den generella kompetensnivån har höjts vilket gör det betydligt mycket svårare att hitta kompetent personal att rekrytera. R4

beskriver detta som utmaningsområde för i nuläget hade man behövt kombinera nyrekryteringar med seniora utvecklare som har 15 års erfarenhet för att få till en bra komposition. Det svåra är att motivera någon med 15 års erfarenhet att ansluta sig till just ditt företag och att det råder en brist på den typen av kompetens i nuläget.

4.8 Möjligheter, förbättringsmöjligheter & framtidsutsikter

R1 anser att de kommit relativt långt i sitt DevOps-arbete. Alla processer har blivit effektivare och företaget är på en bättre plats idag än för bara ett år sedan. Vägen framåt handlar endast om automation. Operations och kvalitetskontroll kan automatiseras till den gränsen att det inte finns som arbetsroll, istället är det utvecklare som utvecklar dessa processer som kod. Då kommer teamen bli ännu mer streamlineade och allt fokus kan läggas på utveckling. Däremot menar R1 på att utvecklingen av driftprocesser kommer att finnas kvar men integrerat i molnlösningar istället för egen driftpersonal. Efter en fråga ifall Ops personal kommer vara en nödvändighet i framtiden svarar R1:

> "Nej precis! Om man tittar på cloud så får man väldigt mycket gratis, man sitter inte och driftar servern. Allting går mot cloud så den biten försvinner av naturliga skäl. Sen har vi din lösning och är den tillräckligt bra gjord så kommer du inte behöva göra särskilt mycket. Så Ops kommer minska och förhoppningsvis försvinna och kvar finns bara en organisation som är DevOps, det måste vara ambitionen tror jag" (Appendix A, r43).

R2 menar att deras arbetssätt med DevOps nog inte kommer förändras så mycket i framtiden. Modellen med DevOps-team, sprintar och agil arbetsmetod kommer att bestå. Det som kan förändras är infrastrukturen som ger utrymme för mer och ny funktionalitet. R2 svarar på frågan: "Hur ser ni på er beroende roll av Microsoft Azure DevOps?" (Appendix B, r41) på följande vis:

> ”Självklart är vi beroende av deras kapacitet gentemot ett traditionellt IT-bolag men vi har också många fördelar, vi kan lansera produkter i rusningstrafik utan problem tack vare det och det tror jag inte många andra IT-bolag har kapacitet till ifall de inte är ledande inom branschen" (Appendix B, r42).

R2 fortsätter att berätta att framtiden kommer gå mot automation. Kan man automatisera delar av verksamheten sparar man pengar. De har kommit en bit när det gäller testning men lanseringar och drift är fortfarande relativt manuell. De har fortfarande personal som ansvarar för detta men de gör allt för att ligga i framkant när det gäller automation.

R3 hade inte så mycket spaningar om framtiden för deras DevOps-arbete. R3 tryckte mer på att organisationer måste bli mer flexibla och anpassningsbara för ändrade omständigheter och förutsättningar. R3 fortsatte med att hen tror att de agila metoderna kommer fortsätta att bre ut sig inom alla delar av näringslivet då det är de metoder som ger organisationer bäst förutsättningar att vara snabbfotade och kunna byta riktning och ställa om ifall detta skulle behövas.

> ”Det agila arbetssättet är det enda arbetssätt som kommer att fungera i framtiden och då tänker inte jag bara systemutveckling och det jag tänker egentligen igenom ett helt företag" (Appendix C, r25).

> ”Oavsett bransch måste man snabbt kunna ändra riktning och det måste man ha en modell för att hantera förändrade krav och därför gillar jag de olika agila ramverken” (Appendix C, r25).

När det gäller DevOps hävdar R3 att framtiden ligger i automation. Det har så stora och uppenbara fördelar att det skulle vara en stor miss att inte försöka hoppa på det tåget menar R3. För varje sak man lyckas automatisera frigör man resurser som kan läggas på annat. Manuell handläggning kan i framtiden helt försvinna och organisationen kan då lägga mer och mer resurser på problemlösning och innovation. R3 berättar att de inte kommit så långt i sin egen automatiseringsresa, de har fullt upp med att effektivisera de andra delarna men detta är något som kommer få allt mer fokus och resurser i framtiden.

> ”Går vi inte på DevOps nivå så är det stora absolut automation, det tror jag inte är någon hemlighet och ni har säkert stött på detta svar tidigare” (Appendix C, r25).

> ”Att automatisera bort saker som är liksom rutin har ju enorma fördelar för verksamheten både ekonomiska och om man tänker på effektivitet. Att dessutom få in kvalitetssäkring och säkerhet i denna automatisering har enorma fördelar för utveckling” (Appendix C, r25).

R4 besvarar frågan om framtidsmöjligheter för DevOps och de största fördelarna med DevOps på följande vis:

> “I think that the biggest advantage of this kind of model is that you can release updates to a product, like new iterations, a lot faster with higher reliability than if you're doing it in a very segregated way. So, I guess what that means is how quickly you can change your product is very valuable” (Appendix D, r15).

R4 fortsätter med att säga att DevOps är en av de viktigaste konkurrensfördelarna de har att jobba med. Ett välfungerande DevOps-arbete ger väldigt bra resultat gentemot en föråldrad metodologi. R4 beskriver detta på följande vis:

> “Competitiveness of your product over time, for being able to react to changes in the market changes in the direction you want things to go in. And if you have a kind of a pre-DevOps era methodology where you have to have different teams’ kind of in lockstep, moving things along through a process through deployment and very slow feedback loops” (Appendix D, r15).

R4:s framtidsutsikter gällande DevOps bygger på att hen gärna vill se mer DevSecOps implementerat på ett mer formaliserat sätt. Detta innebär att säkerhetsaspekter tar större plats och är mer integrerade. R4 hävdar också att det alltid kommer finnas ett behov för driftpersonal då infrastrukturen kräver det.

5 Diskussion

I detta kapitel jämför vi och ställer våra empiriska resultat gentemot de teoretiska fynd som presenteras i kapitel 2. I jämförelse med tidigare forskning kommer vi att betona skillnader, likheter och andra viktiga aspekter av våra empiriska data. Den empiriska studiens respondenter hänvisas till med hjälp av tidigare förkortningar: R1-R4

5.1 Organisationsstruktur

I detta avsnitt hände något väldigt intressant. Vi hade lyft fyra olika organisationsstrukturer som Leite et al. (2021) beskriver som de fyra vanligaste och de fyra olika vägarna man kan gå. Det var väldigt intressant att vi lyckades hitta fyra olika företag där alla hade valt olika strukturer och detta underlättar nu i jämförandet av dessa strukturer. De har inte kopierat strukturerna rakt av utan självklart finns det skillnader från de strukturer som lyftes i teorikapitlet men det blev ganska tydligt att våra respondenter valt olika vägar när det kommer till hur de strukturerar sina organisationer.

R1 arbetar med Cross-functional teams och dessa påminner om de cross-functional teams som Leite et al. (2021) beskriver. Cross-functional teams med blandad kompetens inom utveckling, drift och arkitektur där teamet tar ett end-to-end ansvar för allt de utvecklar. De anställda kan också byta team när som helst vilket vi tror bidrar till distribution av kunskap vilket bidrar till tvärvetenskaplig kunskap och samlat ansvar hos de anställda inom organisationen (Wiedemann et al. 2020). Vi anser att R1 var det företag som kommit längst i sitt DevOps-arbete och vi tror den valda organisationsstrukturen har stor påverkan på detta. De har verkligen börjat om från början och skapat en struktur där all nödvändig kompetens struktureras på ett sätt som gör att de får ut så mycket av sitt DevOps-arbete som möjligt.

R2 har valt att använda strukturen klassisk DevOps (Leite et al. 2021) med några skillnader. De är ett konsultbolag och arbetar med mycket leveranser till kund där kunden sedan sköter driften samt att de även arbetar väldigt mycket med molnlösningar, dessa två faktorer tillsammans gör att de inte behöver fokusera lika mycket på driften eftersom det är kunden eller molnet som kommer stå för den. Varje Team består av en projektledare, utvecklare och även personer med driftperspektiv. De gör ingen differentiering på utvecklare och driftpersonal men självklart är det personer som har mer erfarenhet med drift med i teamet. Här går strukturen lite ifrån ett klassiskt DevOps team (Leite et al. 2021), vi tror att detta beror på den faktorn att de inte har så stort ansvar för driften av systemen de skapar då den tas över av kunden vid leverans alternativt är outsourcat till molnet. Att de inte differentierar bidrar dock till att rollerna har närmat sig varandra, något som vi och Comstedt (2019) är väldigt positivt. Alla utvecklare måste då ha driftperspektivet i varje rad kod de skriver, oavsett om systemet ska driftas av Microsoft i molnet eller av kunden. Vi anser att denna modell fungerar väldigt bra för företaget, de har absolut en fördel mot de andra då de inte behöver fokusera lika mycket på drift men vi anser att molnlösningar är ett bra sätt att utifrån ett eget perspektiv automatisera eller i alla fall outsourca driften. Den enda nackdelen med detta är att man sitter i knäna på ens molnleverantör. Det är i detta fall Microsoft som dikterar vilka lösningar som blir möjliga att skapa då driften kommer skötas av dem i molnet.

Vi skulle säga att R3 är den respondent som har längst kvar med sitt DevOps arbete. De har valt en silostruktur för sin organisation, Leite et al. (2021) var väldigt tydlig med att det ofta är denna struktur man vill gå ifrån då den kan skapa stora problem inom övriga undersökningsområden (Leite et al. 2021; Wiedemann et al. 2020). Silosen är baserade på affärsområden där varje affärsområde har ett DevOps-team. Dessa team förblir intakta och de anställda rör sig sällan mellan silosen. När man pratar om silos ur ett DevOps-perspektiv menar man ofta mellan utvecklare och driftpersonal (Leite et al. 2021) och inte mellan affärsområden, denna struktur hade kunnat fungera effektivt om de olika kompetenserna funnits i varje team men tyvärr finns de silos även mellan rollerna då R3 berättar att driftpersonalen är en annan avdelning och de ej ingår i teamen. Man skulle kunna hävda att de på grund av detta inte kommit någonstans med DevOps och i praktiken inte ens utövar det, då utvecklare och driftpersonal inte närmat sig och inte ens är del av samma avdelning.

Detta skapar stora problem i de övriga undersökningsområdena då ingen delning av kompetens uppstår och leveranser måste ske samtidigt och koordineras vilket bidrar till misalignment (Wiedemann et al. 2020). Statiska teams inom silos gör att kommunikationen och samarbetet blir lidande då de olika teamen sällan interagerar vilket bidrar till sämre prestation för företaget som helhet (Hermawan, Manik, 2021; Oh, Lee, & Zo, 2019). På det stora hela har R3 långt kvar att gå och vi anser att silostrukturen är det sämsta valet av de olika strukturerna. Det är ofta denna struktur man vill komma ifrån då den skapar många av de problem som DevOps försöker lösa (Leite et al. 2021). En stor faktor i detta tror vi är att de inte började om från grunden. De försökte ta befintlig struktur och implementera DevOps på den utan att behöva göra stora förändringar. Vi tror att man måste börja om från början och kartlägga de kompetenser som behövs i vilka team för att DevOps ska fungera så bra som möjligt. Det går inte att ta befintlig struktur och tro att ett DevOps-tänk kommer lösa problemen. Fel struktur är grunden till många problem och måste prioriteras vid implementation av DevOps.

R4 har valt att implementera en form av plattforms team (Leite et al. 2021). R4 berättar att de har olika team som är ansvariga för sina egna produkter och ett så kallat “DevOps-team” som är ansvariga för arkitektur, infrastruktur samt drift. Servers och infrastrukturen har många olika produkter som körs på dem och “DevOps-teamet” är ansvariga för att dessa ska fungera. Målet med denna struktur är att infrastrukturen och driften av denna ska ses som en produkt i sig och “DevOps-teamet” ska inte fungera som support (Leite et al. 2021). Detta berättar R4 att de inte riktigt fått till. Istället har det nästan bildats en silo runt det så kallade ”DevOps-teamet” som fungerar som plattformsteamet. De har med andra ord försökt uppnå en struktur men tyvärr hamnat lite i en silostruktur, precis det som försökte undvikas med andra ord. Att samla all infrastruktur- och driftkompetens i ett team har inte varit framgångsrikt och vi anser att det hade varit bättre att sprida ut denna kompetens i produktteamen och på så sätt skapa cross-functional teams (Leite et al. 2021).

Efter att ha kunnat jämföra fyra olika företag som valt fyra olika organisationsstrukturen anser vi att detta är det viktigaste undersökningsområdet inom DevOps och där man ska börja om man bestämmer sig för att arbeta på detta vis. Många av de problem som våra respondenter upplever härstammar från hur de är organiserade, framför allt anser vi att det går att koppla en bra och DevOps-anpassad struktur till alignment och det blir tydligt att misalignment ofta har med organisationsstrukturen att göra då den kan ses som grunden och förutsättningarna för att nå alignment. På samma sätt kan många av de vinningar som gjorts härledas till samma ursprung. Det är med andra ord i organisationsstrukturen som allt startar, både problem och förbättringar.

När det kommer till vilken struktur och hur man ska organisera sig är vi ödmjuka för att alla har sina fördelar och alla kan fungera på ett mycket bra sätt (Leite et al. 2021). Skulle vi rekommendera en så skulle det bli Cross-functional-teams. Vi anser att R1 var den respondent som kommit längst med sitt DevOps arbete, det var även hen som lyfte minst problemområden samt fokuserade mest på framtiden och de möjligheter som finns inom automation. För oss är detta klara indikatorer för att de har en väloljad maskin och har uppnått störst alignment genom att blanda kompetenser i sina team. De har helt eliminerat silos och cross-functional teams är i våra ögon det bästa sättet att göra detta på. En faktor som bidragit till R1s framgång är att de började om från grunden när de ville implementera DevOps. De försökte inte retroanpassa DevOps på befintlig struktur utan började om från grunden, skapade nya avdelningar, nya team, omorganiserade kompetens och införskaffade den kompetens som saknades. Detta tror vi är den största faktorn för deras framgång och vill en organisation implementera DevOps är det exakt detta tillvägagångssätt man måste ha.

5.2 Kommunikation

5.2.1 Frekvens

Under intervjuerna förekom det fall när intervjupersonen benämnde och talade om hur frekvensen av kommunikation påverkade deras DevOps-arbete vilket är en grundpelare av kommunikation enligt Diel, Marczak och Cruzes (2016). R1 beskriver att kommunikationsfrekvensen har ökat sedan utvecklare och driftpersonal sitter i samma tvärfunktionella arbetslag, tidigare användes ett ticket-system som det kommunikationsverktyget. Att använda sig av mer en direkt kommunikation mellan två parter är något som ökar frekvensen, Diel, Marczak och Cruzes (2016) och Erich, Amrit och Daneva (2017) menar på att detta hjälper till att minska risken för att detaljer kring arbetsbeskrivningar förloras i kommunikationen och leder till en försämrad slutprodukt. Diel, Marczak och Cruzes (2016) beskriver att det finns en balansgång mellan den minimala mängden kontakt nödvändig för att säkerställa adekvat samordning och för mycket kontakt vilket kan överbelasta en av parterna och få dysfunktionella konsekvenser. Utifrån vad Diel, Marczak och Cruzes (2016) definierar som en balansgång mellan låg och hög frekvens går det att argumentera för att R1 befinner på relativt balanserad frekvensnivå i förhållande till de andra intervjupersonerna och litteraturens definition. Med det sagt finns det utrymme för att öka frekvens utan att överbelasta någon av kommunikationspartnerna gällande R1.

I jämförelse kan man placera R2 på en högre placering av frekvens gällande kommunikation då det framkommer från intervjun att R2:s företag använder sig av flera olika kommunikationskanaler för att kommunicera gentemot R1, exempelvis kommunikationsverktyg likt Slack och integrerade molnlösningar, innovationsmöten och regelbundna avstämningar. R2 beskriver företagets interna kommunikation som transparent där alla kan se alla. Detta menar Diel, Marczak och Cruzes, (2016) på är positivt så länge kommunikationsfrekvensen ej blir för hög och överbelastar en av parterna.

R3 och R4 beskriver i intervjuerna att det finns svårigheter med deras kommunikation. R3 beskriver att några av driftpersonalen är externa konsulter och detta försvårar kommunikationen vilket resulterar i en låg frekvens. R4 berättar att på grund av att deras företag är ett multinationellt företag etablerat över hela världen så sker kommunikationen asynkront vilket innebär att kommunikationen inte sker i realtid utan exempelvis skickas ett meddelande till en

medarbetare mottages och responderas det 5 timmar sen. Detta är motsatsen till den direkt kommunikation som R1 beskriver och därför anser vi detta är en indikation på en låg nivå av frekvens. Utifrån vad Diel, Marczak och Cruzes, (2016) poängterar gällande frekvens finns det möjlighet för både R3 och R4 att höja frekvensen av kommunikation för att förbättra den inom DevOps.

5.2.2 Riktning

Samtliga respondenter redogör för att riktningen av kommunikation är en viktig aspekt i det generella kommunikationsarbetet. Crespi, Galstyan och Lerman (2008) menar på att det vanligaste är en top-down approach när det kommer till större beslut och utvecklingsområden. Samtliga av respondenterna redogör att det är via en top-down approach som majoriteten av kommunikationen på deras organisationen riktas. Det som skiljer respondenterna åt är hur de berättar om riktningen inom exempelvis ett DevOps-team där R2 menar på att det går att ställa vilken fråga till som helst till vem som helst och att organisationen är platt inom teamen medan R3 hellre har en mer klassisk struktur där kommunikationen följer den hierarkiska ordningen inom organisationen.

Diel, Marczak och Cruzes (2016) beskriver att kommunikation från driftteamet vanligtvis är ett rop på support eller hjälp, eller att informera utvecklaren om kundfeedback på produkten medan kommunikationen från utvecklingsteamet vanligtvis är mer informativ via rapporter om funktionalitet. Detta fenomen är något som respondenterna också beskriver i intervjuerna och framför allt R2 som beskriver att när det rör sig om en bugg eller systemfel larmas detta akut som ett rop på hjälp från supporten medan utvecklare använder sig av längre briefingar eller rapporter för att kommunicera. Detta kan resultera i att olika kommunikationsgrupperingar inom organisationen växer fram då man kommunicerar på olika sätt vilket minskar den enhetliga kommunikationsstrukturen.

5.2.3 Modalitet

Fynden från intervjuerna visar att samtliga intervjupersoner använder sig av någon form av internt kommunikationsverktyg som är integrerat i hur utvecklare och driftpersonal kommunicerar och arbetar med varandra. Diel, Marczak och Cruzes (2016) redogör för att det finns "rik" information vilket kan vara feedback och arbetsbeskrivningar. Detta ligger i linje med vad R1, R2 och R4 beskriver som möjligheten att se varandras framsteg och arbeta med problemlösning digitalt där dessa respondenter använder sig av molnlösningar likt Microsoft Azure DevOps, AWS eller en intern lösning för att utvecklare och driftpersonal ska kunna följa utvecklingen i arbetet och ge förslag och inputs på det. R3 beskriver att det finns brister inom detta område för dem där hen säger

> "men vi har svårt att hitta ett bra forum för kommunikation utvecklare till utvecklare mellan teamen" (Appendix C, r9).

Diel, Marczak och Cruzes (2016) menar på att termen modalitet bygger på att användandet av ett gemensamt kommunikationsverktyg kan vara avgörande för att lyckas med DevOps-arbetet. Arbetslag rapporterar om att de föredrar en skriven kommunikation som alla kan se och som sedan delas via ett internt kommunikationsverktyg (Diel, Marczak & Cruzes, 2016). Denna typ av modalitet är något vi anser att 3 av 4 intervjupersoner har koll på i deras

DevOps-arbete medan R3 behöver jobba vidare med användandet av interna kommunikationskanaler för att låsa upp en större potential inom DevOps.

5.2.4 Innehåll

Att kategorisera innehållet inom kommunikationen mellan utvecklare och utvecklare men även utvecklare och driftpersonal är viktigt, Diel, Marczak och Cruzes (2016) menar på att en innehållsanalys av kommunikationsinteraktioner kan göras med hjälp av förbestämda kategorier. Dessa kategorier innebär att de två parterna inom en konversation inser ifall innehållet i en konversation är högt prioriterat likt förfrågningar, överklagande och skyldigheter gentemot lägre prioriterade samtalsämne som en diskussion vid kaffeautomaten om helgplaner (Diel, Marczak & Cruzes, 2016).

Trots att dessa kategorier kan verka som en självklarhet för många människor har det en påverkan på hur kommunikation sker på en organisation och att ha ett informationsmöte/workshop om detta kan förbättra den generella kommunikationsnivån på företaget. Ingen av respondenterna gick in på detalj ifall de använde sig av ett prioriteringssystem för innehållet i en konversation. Däremot anser vi att detta är något som lätt glöms bort och kan ha en positiv påverkan på kommunikationen och effektivitet inom en organisation och är något som hade kunnat underlätta för respondenterna.

5.3 Samarbete och lagarbete

Samarbete och lagarbete är en viktig aspekt i DevOps-arbete menar Hashmi, Ishak och Hassan (2018). Det är en avgörande faktor för ifall en implementering av DevOps är framgångsrik eller ej (Hashmi, Ishak & Hassan 2018).

5.3.1 Teamstruktur, en avgörande faktor?

Våra intervjupersoner var något delade i sina svar gällande samarbete, även om de alla svarade att samarbete och lagarbete var viktigt. Intervjupersonerna hade olika teamstrukturer exempelvis gör R2 ingen skillnad på utvecklare (Dev) och driftpersonal (Ops) och ingår i ett och samma team medan R3 beskriver att de har anställda utvecklare och sedan hyr de in externa konsulter som driftpersonal. R1 berättar om att det finns cross-functional teams som består av utvecklare och driftpersonal i samma team, medan de även finns renodlade utvecklingsteam för utveckling av specialfunktioner. R2 berättar att de personer som håller på med specialfunktioner hos dem är integrerade i det stora teamet. Självklart kan storleken på organisationens roll spela in här där R1 jobbar på ett stort globalt multinationellt företag gentemot R2 som jobbar på ett konsultbolag endast verksamma i Sverige. R4 beskriver att de jobbar ett liknande sätt som R1 med flera utvecklingsteam.

Hoegl och Gemuenden (2001) redogör att funktionalitet kan delas upp i olika team utifrån expertis vilket kan leda till en ökad nivå av specialisering och standardisering som i sin tur innebär en mer kostnadseffektiv arbetsfördelning. Däremot menar Comstedt (2019) och Carlsson och Langsager (2021) att detta kan orsaka konkurrerande mål, konflikter och en otydlig ansvarsfördelning mellan team och på så sätt ett minskat samarbete mellan de olika aktörerna inblandade. Det går att konstatera att trots specialisering av funktionalitet är samarbete

välfungerande hos R2, vilket kan bero på den integrering av personal inom ett och samma team vilket möjliggör samarbete på ett bättre sätt. Samma sak går att observera med R1:s cross-functional teams, utan den sammanslagningen av utvecklare och driftpersonal hade nog inte samarbetet fungerat lika bra. Detta är något som också bevittnas i R3:s svar där samarbetet mellan utvecklare och den externa driftpersonalen var bristfälligt. Detta kan innebära att integrering av utvecklare och driftpersonal är en nödvändighet för att få ett välfungerande samarbete och bygger mycket på teamstrukturen hos en organisation. För mycket segregation mellan teamen kan också bidra till att vissa parter inte inkluderas i viktiga arbetsprocesser menar Diel, Marczak, Cruzes, (2016) och Hoegl, Gemuenden (2001) på.

5.3.2 Distans eller fysiska möten?

Hermawan och Manik, (2021) redogör att ett sätt att förbättra teamwork och kommunikation mellan utvecklare och operatörer är att minska distansen både fysiskt och digitalt mellan de olika aktörerna. Detta kan genomföras genom att dela fysisk arbetsyta vilket exempelvis möjliggör att enkla frågor kan ställas direkt och mer kommunikation förekommer naturligt (Hermawan, Manik, 2021). Däremot behöver det finnas en grundläggande förståelse av samarbete och kommunikation mellan de olika arbetslagen oavsett ifall kommunikationen sker fysiskt eller digitalt anser Nybom, Smeds och Porres (2016).

Intervjupersonerna är eniga i att majoriteten av arbetet sker på distans nuförtiden, delvis på grund av pandemin som skapat en omställning i hela samhället men även på grund att effektiviteten har gått upp. R2 redogör för att effektiviteten och samarbetet mellan utvecklare har ökat i utvecklingsprocessen. Däremot beskriver R2 att innovationsprocessen blir lidande där utvecklare och driftpersonal behöver samarbeta för att vara kreativa och hitta lösningar. R2 menar på att det inte finns tillräckligt bra verktyg för att göra detta digitalt. Detta ligger i linje med vad Gupta, Venkatachalapathy och Jeberla, (2019) redogör för att fysiska möten hjälper till att bygga en bro mellan utvecklare och driftpersonal och på sikt kan detta öka samarbetet, kunskaper, stödet och motivationen inom ett team. R3 beskriver att fysiska möten skulle hjälpa till generellt och tror på en hybrid variant av distans och fysiska möten. Detta är något som styrks av Diel, Marczak och Cruzes (2016) som menar på att fysiska möten kan reducera och minska problem med hjälp av enklare kommunikationsmöjligheter mellan de olika arbetslagen generellt. R1 och R4 redogör för att på grund av deras företag är multinationellt är distans det enda värdefulla alternativet i nuläget då fördelarna väger upp nackdelarna menar de.

5.4 Kontinuerlig integration, leverans och kvalitetssäkring

Yarlagadda (2018) redogör för att kontinuerlig integration och leverans är två termer som inträffar sammankopplat. Innebörden kan beskrivas som att utvecklare tar ansvar för integration av koduppdateringar i en gemensam databas och att alla uppdateringar släpps i tid och löpande (Waller, Ehmke & Hasselbring, 2015; Yarlagadda, 2018).

Respondenterna beskriver att det är en viktig del av deras DevOps-arbete. R1 beskriver att hens företag applicerar en built-in design när det kommer till kontinuerlig integration och leverans att system hjälper dem att hålla leveranstider och ansvarsfördelning. Detta är något som R2 och R4 också arbetar med men har en tydligare struktur och arbetar med sprintar på två veckor och med tydligare feedback-loopar gentemot R1. Waller, Ehmke och Hasselbring (2015) menar på att ha en tydlig arbetsstruktur för allt utvecklingsarbete inklusive driftarbete

ökar chansen att hitta buggar och fel i ett tidigt skede. R3 beskriver att de befinner sig längre bak i denna process, de har inte hittat ett gemensamt förhållningssätt för att lyckas med att utvecklare tar ett större ansvar. R3 beskriver att en av anledningar bakom detta är kompetens och personal, det är svårt att rekrytera då de söker efter erfarna utvecklare som är öppna för ett nytt arbetssätt.

5.4.1 Kan man automatisera bort driftpersonal?

Automatisering är en stor del av kvalitetsarbetet (Humble & Farley, 2010; Yarlagadda, 2018). Waller, Ehmke och Hasselbring (2015) redogör att genom att automatisera processer i produktkedjan likt testning av mjukvara och automatiserade uppdateringsprocesser minskar man de obehagliga överraskningar som utvecklare och driftpersonal behöver hantera exempelvis buggar och systemfel.

Roche (2013) beskriver dock kvalitetssäkringsarbetet som att efterlikna användares användningsmönster och på så sätt minimera upptäckten. Detta är något som Roche (2013) beskriver som ett arbete som kräver både manuella och automatiserade. Automatiserade tester kan användas för att se hur ett system eller mjukvara klarar överanvändning medan manuella tester bör användas för att memika mänskligt beteende (Roche, 2013). Waller, Ehmke och Hasselbring (2015) och Yarlagadda (2018) menar dock på att med hjälp av AI kan även den användarbeetende hanteras automatiskt.

Att automation behövs är något samtliga respondenter är överens om och att det är en del av framtiden inom DevOps råder det inga tvivel om. Däremot skiljer sig åsikterna kring hur mycket som ska automatiseras. R1 och R2 menar på att så mycket automation som möjligt är det bästa utfallet, R1 uttrycker sig på följande sätt: ”Endgame är att allt blir Dev” (Appendix A, r43), vilket är en indikation på att hens företag ser på automation som ett sätt att ta bort Ops som arbetsroll helt och hållet. R3 och R4 beskriver att automation handlar mer om att förenkla DevOps-arbetet med testning och kvalitetssäkring för både utvecklare och driftpersonal vilket är mer i linje med vad Roche (2013) beskriver som behovet av både manuella och automatiserade tester och kvalitetssäkringar.

5.5 Alignment

Samtliga respondenter upplevde någon form av misalignment tidigare och detta var i stor utsträckning en bidragande faktor till att alla valde att implementera DevOps. Många av misalignment faktorerna som Wiedemann et al. (2020) lyfter har upplevts. Dels så var inte de olika avdelningarnas mål linjerade vilket gjorde att driftpersonal och utvecklare ofta hamnade i konflikt, detta visade sig tydligast när utvecklare villa releasa en uppdatering men de var tvungna att vänta på andra team för att inte behöva stänga ner hela systemet fler gånger än nödvändigt, detta är tydliga bevis på att man inte arbetar med individuell komponentisering (Wiedemann et al. 2020).

R1 är den respondent som kommit längst med sitt alignment-arbete. De har infört konkreta ändringar för att uppnå alignment. För att uppnå samlat ansvar (Wiedemann et al. 2020) har de utökat de agila metoderna och skräddarsytt dessa till verksamheten. De har skapat ett releasetåg som går hela vägen upp på ledningsnivå för att koordinera både inom och mellan team. Varje team är produktorienterad istället för projektorienterad och detta kan endast uppnås när

teamen kan arbeta självständigt. Självständiga team uppnår de genom individuell komponentisering (Wiedemann et al. 2020). Varje team kan arbeta och utveckla i samma system utan att de påverkar någon annan, detta uppnås genom att dela upp systemet i olika containrar (Humble och Farley, 2010). Detta ger även möjlighet till tysta lanseringar (Wiedemann et al. 2020) vilket betyder att man kan lansera ny kod utan att ta systemet ur drift, även arkitekturen är flexibel då denna kompetens finns i varje cross-functional team. Behövs någon ändring i arkitekturen finns denna kompetens inom teamet och de slipper öppna en ticket med ett annat team eller annan avdelning. Den sista faktorn som motverkar misalignment hos R1 är Tvärvetenskaplig kunskap (Wiedemann et al. 2020). Detta uppnås med hjälp av cross-functional teams. Eftersom teamen består av blandad kompetens inom områdena utveckling, drift och arkitektur så kan denna kunskap spridas och på så sätt närmar sig de traditionellt separata rollerna varandra, utvecklarna får lite mer av ett driftperspektiv när de har teammedlemmar som kommer med den sortens input.

R2 arbetar också aktivt med alignment. Inte lika hårt som R1 och vissa av de faktorer vi lyft i litteraturen glöms bort. Tidigare köpte de in system som externa parter utvecklat vilket resulterade i dålig innovation och höga driftskostnader. Nu utvecklar de sina egna system och bara det har gjort verksamheten mer aligned. Tvärvetenskaplig kunskap Wiedemann et al. 2020 menar på att det uppnår de genom sammansättningen av deras klassiska DevOps-team, som vi nämnt tidigare differentierar de ej mellan utvecklare och driftpersonal vilket gör att alla är ansvariga från ax till limpa och de lär sig av varandras kompetenser och erfarenheter. Detta bidrar även positivt för att uppnå samlat ansvar (Wiedemann et al. 2020). När det inte finns olika roller och någon överlämning ej sker så samlas end-to-end ansvaret i varje team, detta upplever R2 har varit den största och mest framgångsrika faktorn att uppnå alignment. När det kommer till Individuell komponentisering (Wiedemann et al. 2020) säger R2 att de arbetar med flexibel infrastruktur (Wiedemann et al. 2020).

Detta stämmer men vi är inte villiga att hålla med till 100%. Den är flexibel på så sätt att Microsoft hela tiden uppdaterar den och kommer med nya funktioner men vi anser att den inte är flexibel på så sätt att de själva har ingen input i vad som ska ändras och uppdateras. Outsourcar man hela infrastrukturen sitter man i knäet på sin leverantör. Det kan komma en dag då de saknar något och då är de maktlösa och måste vänta på att Microsoft möter deras kravbild. Containisering (Wiedemann et al. 2020) används också för att ge möjlighet för olika team att arbeta i samma system samtidigt medan tysta lanseringar är inte är något de arbetar med. De releasar alla uppdateringar en gång i veckan. R2 berättar att detta är för att stresstesta systemet men vi är lite oroliga att ju större system de arbetar med desto mer kommer de behöva införa tysta lanseringar.

R3 berättar att de har långt kvar för att uppnå alignment. Många av dessa problem är resultatet av att de inte uppnått Individuell komponentisering (Wiedemann et al. 2020). Teamen är väldigt beroende av varandras schema och en lansering är inte alltid möjlig då de måste vänta in andra lanseringar. Detta gör att teamet blir stillastående och ineffektivt. Samlat ansvar (Wiedemann et al. 2020) och tvärvetenskaplig kunskap (Wiedemann et al. 2020) har inte heller uppnåtts. Detta tror vi kan härledas till organisationsstrukturen de har. Eftersom de arbetar med silos (Leite et al. 2021) och att driftpersonalen tillhör en annan avdelning blir det svårt att få till End-to-end ansvar samt att kunskap delas mellan utvecklare och driftpersonal. Detta är faktorer som måste adresseras och vi tror att detta kommer vara svårt med den organisationsstruktur som företaget har idag.

R4 har en klar och tydlig bild över att alignment är viktig faktor för att lyckas. Hen har också arbetat hårt för att få konkreta arbetssätt på plats för att underlätta detta arbete. Huvudsyftet ligger i samlat ansvar och hen delar den bild som (Wiedemann et al. 2020) målade upp. R4 berättar att utvecklare ska känna frihet att skapa och drifta sina egna produkter sen kommer det alltid finnas saker de inte rör, till exempel fysisk infrastruktur. Mycket fokus läggs på individuell komponentisering och R4s bild över detta stämmer bra överens med (Wiedemann et al. 2020). Mycket fokus läggs på infrastrukturen då de har ett specialiserat plattformsteam som sköter detta, deras uppgift är att hålla den så flexibel som möjligt vilken möjliggör containerisering och tysta lanseringar för övriga team. Detta är en modell som fungerar väl för att uppnå dessa två faktorer men vi anser att man går miste om den tvärvetenskapliga kunskap som Wiedemann et al. (2020) beskriver. När all personal som sysslar med arkitektur, infrastruktur och drift sitter i plattformsteamet finns det en risk att man inte kan dra lärdom av varandra. Utvecklare och driftpersonal kommer då inte närma sig varandra utan det kommer alltid finns ett gap mellan dessa två roller.

Alla respondenter är införstådda med vilka problem misalignment skapar i en organisation, trots detta når inte alla hela vägen i sitt arbete mot detta. Vi anser att det beror på att man inte skapar de rätta förutsättningarna för detta arbete. Många av de problem som fortfarande finns i de olika organisationerna kan härledas till den organisationsstruktur som finns. Till exempel kommer silos alltid arbeta mot tvärvetenskaplig kunskap, precis på samma sätt som att inte blanda kompetens inom teamen gör. Är infrastrukturen hos en annan avdelning kommer man inte kunna ha den flexibilitet som utvecklarna kräver för att vara konkurrenskraftiga i dagens affärsklimat.

5.6 Företagskultur

Företagskultur kopplat till DevOps som Gall och Pigni (2022) benämner det som, handlar om en integrerad företag där delande av kunskap, verktyg, infrastruktur, feedback, framgång och ansvar är gemensamt för både utvecklare och driftpersonal samt organisationen som helhet (Gall & Pigni, 2022).

Under intervjuerna med respondenterna framkommer det från samtliga respondenter att företagskulturen har förändrats sedan implementeringen av DevOps. R1 beskriver att kulturen har förändrats så att de har blivit mer moderna på så sätt det finns ett annat laserfokus på att saker och ting ska ske så smidigt och enkelt som möjligt med ett tight samarbete. R1 säger att de tittar mycket på hur man gör saker hos de stora Tech-jättarna i Silicon valley. Detta är något vi tror är smart men samtidigt behöver man hitta en strategi som fungerar för det egna företaget då det kan skilja sig mycket mellan olika företag. Samtidigt kan man inte bara ta ett fungerande koncept och implementera och sedan vara klar enligt Gall och Pigni (2022). Det är ett konstant arbeta som hela tiden förändras med företaget (Gall & Pigni, 2022).

R2 nämner också att det har skett kulturella förändringar i form av att de gick från en vattenfallsmodell och den företagskulturen som hör till och nu har de anammat en mer DevOps-kultur trots att det har en traditionell företagsstruktur. R2 beskriver också att det finns specialistfunktionalitet inom teamet som arbetar mer enskilt, detta är något som Khan et al. (2022) menar är bra för kulturen då det främjar även innovation inom teamet eftersom man kan experimentera med nya processer och verktyg.

R3 nämner att kulturen är något som långsamt förändras hos dem och den stora utmaningen är personer som jobbat länge på ett visst sätt skall ställa om vilket kan vara en svårighet beskriver R3. R3 beskriver att det bästa upplägget vore att blanda nyexade utvecklare och seniora utvecklare för att uppnå en balans inom företaget och få en kultur där de kan lära sig av varandra. Gall och Pigni (2022) beskriver samarbete som en viktig komponent för en bättre kultur på arbetsplatsen.

Något som skiljer R1 och R4 åt trots att båda jobbar på stora multinationella företag är att R1 beskriver en gemensam företagskultur och R4 menar på att det finns subkulturer från varje land som influerar den gemensamma kulturen. Däremot menar R4 på att kulturen inte spelar så stor roll så länge det inte är en utnyttjande kultur och företagsstruktur och att de andra komponenter i DevOps fungerar. Detta skiljer sig från vad Gall och Pigni (2022) och Mohammed, (2017) hävdar, de menar på att en aktiv företagskultur som ligger i linje med DevOps-arbetet kan vara en avgörande framgångsfaktor för att lyckas med DevOps.

5.7 Utmaningar och problemområden

När det kommer till utmaningar och problemområden hade respondenterna inte så mycket att säga. Detta kan självklart ha att göra med att de är lite partiska. De arbetar med DevOps dagligen och är så klart förespråkare för arbetssättet men några gemensamma utmaningar lyfts. Kompetensbrist lyftes av samtliga respondenter. Detta verkar vara ett generellt problem i branschen som helhet och är inget som vi lyft i tidigare kapitel eller stött på i litteraturen. DevOps har dock ytterligare en dimension i detta problem. När rollerna som utvecklare och driftpersonal ska närma sig varandra ställer detta nya krav på den anställde, deras kompetensområde utökas vilket ställer högre krav på den anställde (Erich, Amrit, & Daneva, 2017; Sebastian et al. 2017). Detta i sin tur gör kompetensbristen ännu värre. Alla företag verkar ha svårt att hitta kompetent utvecklingspersonal och arbetar med DevOps som dessutom kräver att denna utvecklare också har kompetenser inom drift blir de potentiella medarbetarna ännu färre.

Ett annat problem som lyfts av framför allt R2 och R3 är att det är svårt att estimera hur mycket ett projekt kommer kosta samt hur lång tid det kommer ta. Detta tror vi beror på att många fortfarande är vana vid vattenfallsmodellen där man spenderar mycket mer tid på att bryta ner krav innan man börjar utveckla. De agila metoderna fungerar på ett annat sätt och här kan det vara viktigt för ledning att förmedla att vi måste stå ut med denna svårighet för att det gör företaget effektivare. Ett tydligare och starkare ledarskap hade kanske kunnat motverka att känslan att detta är något negativt för medarbetarna. Här tror vi att Top-down management som Erich, Amrit, och Daneva (2017) lyfter är väldigt viktigt.

Den enda svårigheten som lyfter av respondenterna som även lyfts i litteraturen är att det inte finns en klar och tydlig definition av hur man ska implementera och arbeta med DevOps (Erich, Amrit, & Daneva, 2017). Detta hade framför allt R3 och R2 upplevt. Eftersom det ser olika ut på alla företag så var det svårt att veta exakt vilka vägval som skulle göras, detta var aktuellt i flera undersökningsområden.

En utmaning som förvånade oss lite eftersom den inte fanns i litteraturen och som vi inte tänkt på själva heller var synen och statusen på de olika rollerna. R1 berättade att driftpersonal nästan alltid har lägre status än utvecklare. Detta leder till att utvecklare inte vill blanda sig med driften. Lösningen på detta var att det var driftpersonalen som blev insatt tidigare i

utvecklingsprocessen snarare är att en utvecklare stannade längre (Appendix A). Det känns som att detta är ett kulturellt problem som nog kommer bättras ju längre tid som går, desto mer rollerna närmar sig varandra och desto mer kunskapsutbyte som äger rum som i sin tur leder till alignment inom organisationen (Wiedemann et al. 2020).

Något väldigt intressant när vi pratade om utmaningar och problem är att många av lösningarna på dessa utmaningar också lyfts i möjligheter. Med andra ord om man lyckas lösa problemen så har man redan tagit första steget i att förbättra sitt DevOps arbete och ta vara på de framtidsmöjligheter som finns. Till exempel kan lösningen på att utvecklare inte vill arbeta med drift vara att utvecklarna automatiserar bort driften vilket direkt gör att man som organisation tar nästa steg och blir ännu effektivare (Mohammad, 2016). Att det är svårare att estimera projekt tror vi är något man får vänja sig vid, det kommer på köpet med arbetssättet och här gäller det att ha tydligt ledarskap som står fast vid att så är det bara (Erich, Amrit, & Daneva, 2017). Kompetensbristen har vi inte någon lösning på och har vi förstått det rätt kommer den fortsätta vara ett problem för företag i många år framåt, det blir då ännu viktigare att profilera sig som en arbetsplats där man vill arbeta.

5.8 Möjligheter, förbättringsområden och framtidsutsikter

När det kommer till möjligheter, förbättringsområden och framtiden fick vi två typer av svar av våra respondenter. De kan delas in i två kategorier; vidareutveckling av de agila metoderna och DevOps och automation.

Vikten av att vara snabbfotad och snabbt kunna ändra riktning på ett företag när kravbilden och omvärlden ändras lyfts framför allt av R3 och R4. De ser detta som den största konkurrensfördelen ett företag kan ha idag och för detta är DevOps ett arbetssätt som ger väldigt bra förutsättningar (Christensen, 2016). Att arbeta med traditionell vattenfallsmodell där man spenderar månader på att bara bygga en kravbild kommer innebära att kraven ändrats redan innan man hunnit börja utveckla, detta är en bild som R2 helt delar.

Framtidsspaningar och profetior är alltid luriga och bör tas med en nypa salt då fantasin ibland kan skena iväg. Med det sagt finns det inga tvivel om att automation är framtiden inom mjukvaruutveckling samt att det är detta som fokus kommer läggas på inom branschen (Mohammad, 2016). R1 drar fram konkreta exempel på automation och de har kommit igång med arbetet men har såklart en bra bit kvar.

Det finns två områden där detta arbete redan är i full gång, dessa är testning och drift (Mohammad, 2016). Automatiserade tester frigör resurser som nu kan läggas på utveckling och gör att felaktig kod aldrig lämnar utvecklingsmiljön med hjälp av feedback och testning i realtid (Mohammad, 2016) R2 upplever att detta har gjort deras utvecklingsteam väldigt mycket effektivare vilket gör att de har fått till kontinuerliga leveranser vilket också bidrar till den alignment som Wiedemann et al. (2020) skrev om när de förespråkade tysta och kontinuerliga releaser. Detta bidrar självklart också till att organisationen blir mer snabbfotad.

Automatiserad drift tror vi kommer bli nästa steg i DevOps. Grundidén var att förenkla driften genom att minska avståndet mellan utvecklare driftpersonal (Christensen, 2016). Framtiden skulle kunna innebära att driften är helt automatiserad och att den arbetsbeskrivningen inte finns längre, driften sköts av kod som utvecklare står för med andra ord (Mohammad, 2016). Detta är väldigt spännande och framför allt R1 var inne på detta när hen sa:

”Endgame är att allt blir Dev” (Appendix A, r43)

Kan detta uppnås blir resultatet att resurserna som går till utveckling blir betydligt större vilket kommer starta en positiv loop inom innovation som kommer göra att ännu mer automation blir möjligt. Vi har en liten känsla av det kan gå väldigt snabbt på denna front och det gäller verkligen för företag att fokusera och hänga med på denna resa. Vilket R1 beskriver som:

“När allting går mot cloud så sitter man inte och driftar servrar” (Appendix A, r43).

Utvecklingen av de stora plattformarna blir avgörande för hur långt företag kan ta detta och så småningom pratar man kanske inte om DevOps då Ops-biten helt fallit till glömska eftersom den sköts av kod som en ambitiös utvecklare skrivit.

6 Slutsats

Författarnas slutsatser i forskningsfrågan presenteras i detta kapitel, följt av uppsatsens bidrag till intressenter, och slutligen diskuteras studiens begränsning och framtida forskning.

6.1 Slutsatser på forskningsfråga

Baserat på vår forskningsfråga: "*Hur arbetar organisationer och företag med implementering och vidareutveckling av DevOps utifrån ett organisations- och ledningsperspektiv för att nå sina mål och hantera utmaningar?*" kan vi dra följande slutsatser:

DevOps är ett arbetssätt som verkar enkelt att applicera och implementera i teorin, har du ett bra samarbete och kommunikation mellan utvecklare och driftpersonal och arbetar agilt så får du DevOps. Tyvärr är det inte så enkelt i praktiken. Ett av de största problemen med DevOps är att alla organisationer och företag har olika uppfattningar om vad DevOps innebär och arbetsmodellens effekt och påverkan på företaget. Detta resulterar i att vissa företag beskriver att de har genomfört en implementering av DevOps men jobbar egentligen på samma sätt som förut med några små ändringar i titlar och struktur.

Efter genomgång av befintlig litteratur och egna efterforskningar har vi kommit fram till att organisationsstruktur är en av om inte den viktigaste faktorn som avgör hur framgångsrik en organisation blir vid implementeringen av DevOps. En omorganisation för implementering är högst nödvändig, alla våra respondenter har genomgått stora förändringar vid implementation av DevOps. Ska DevOps fungera måste driftpersonal och utvecklare närma sig, både fysiskt och kompetensmässigt, detta kommer endast bli möjligt med en ny struktur där silos bör undvikas. Vi anser att den bästa modellen är Cross-functional teams, där bred kompetens samlas i samma team och detta team tar ett end-to-end ansvar för den mjukvara som utvecklas. Denna struktur är den som ger bäst förutsättningar att uppnå alignment i organisationen. Ansvar samlas och man lämnar inte ifrån sig mjukvara så fort den är utvecklad, utvecklare kan lättare få ett driftperspektiv om de ingår i samma team och vice versa och fokus på containerisering och flexibel arkitektur blir lättare och hamnar mer i fokus när end-to-end ansvar finns inom ett och samma team. Vår studie har visat att många av de problem och svårigheter som kan uppstå kan härledas till den organisationsstruktur man väljer, det är därför av yttersta vikt att välja en struktur som ger de rätta förutsättningarna för övriga undersökningsområden. Det är också extremt viktigt att inte fega, man ska inte vara rädd att riva befintlig struktur ner till grunden. Att försöka bygga om DevOps på befintlig struktur eller göra en halvdan omorganisation kommer endast leda till problem.

För att lyckas med DevOps har vi kommit fram till att det behövs en all-in approach, hela företaget måste vara med om förändringen från ledning till utvecklare och det måste vara förankrat i hela organisationen. Ett misstag som vi har observerat under studiens gång är att man anser att vissa saker likt kommunikation, samarbete och företagskultur kommer som en biprodukt efter att man har implementerat DevOps. Här anser vi att detta är områden som snarare behöver distribuering av resurser, tid och kunskap för att själva DevOps-arbetet skall lyckas. Alla hävdar att kommunikation och samarbete är viktigt men det finns ingen struktur på plats för att underlätta detta, ofta görs en omorganisation och sedan hoppas organisationerna att dessa faktorer löser sig själva.

Det råder inga tvivel om att automatisering är framtiden för DevOps. Grundidén är att utvecklare ska ta ett större driftperspektiv för att sänka driftkostnader. Framtiden för detta är att drift som arbetsroll försvinner helt. Många framsteg har redan gjorts inom testning av mjukvara samt releaser och lanseringar. Organisationer idag är på god väg och inser att desto mer de kan automatisera desto effektivare blir de och desto mer resurser kan läggas på utveckling och innovation. Detta kommer i sin tur leda till att utvecklarna kan närma sig affärsutvecklingen och på så sätt vara med och driva affärsmodellen framåt mot något slags "BisDev" koncept vilket innebär att Business och Development går ihop till (BisDev). Även säkerhet kommer spela stor roll i automatisering framöver, ett nytt koncept som heter DevSecOps har på senare tid blivit populärt och går ut på att systemsäkerhet integreras i de kontinuerliga leveranserna och kvalitetssäkringsarbetet.

6.2 Bidrag till forskningen, begränsningar och framtida forskning

6.2.1 Bidrag till forskningen

Vår studie kan förhoppningsvis hjälpa att belysa viktiga komponenter på hur företag arbetar med DevOps i praktiken och dess problem som finns och att trots att DevOps har funnits länge finns det mycket kvar att göra inom ämnet för att hitta en gemensam best practice utifrån hur många olika sätt det finns att arbeta med DevOps. Vi hoppas kunna bidra och med viktiga poänger och aspekter för människor som redan arbetar med DevOps samt människor och organisationer som planerar att implementera DevOps inom en snar framtid.

6.2.2 Begränsningar och limiteringar

Det finns begränsningar med vår studie. På grund av studiens storlek och den mån av tid som gavs utrymme för, har endast 4 respondenter intervjuats och då arbetssättet kring detta ämne skiljer sig så mycket mellan organisationer så blir det svårt att dra konkreta slutsatser för alla organisationer som arbetar med DevOps. En annan begränsning kan vara att vi under intervjuerna ställde väldigt öppna frågor som var fria för tolkning, detta gjorde vi för att vi ville ha öppna svar och ge respondenterna möjlighet att "sväva" iväg i sina resonemang. Studien hade kunnat kompletteras med ett kvantitativt perspektiv i form av enkäter för att ge en mer konkret bild över fältet.

6.2.3 Framtida forskning

Vi rekommenderar framtida forskning inom området med fokus på automatisering. Hur skapar man de bästa förutsättningarna för att lyckas med en implementering och vidareutveckling av DevOps med hjälp av automation? Går det att hitta någon best practice som funkar för de flesta företagen trots att alla företag arbetar på olika sätt? Framtida forskning inom BisDev och DevSecOps är också något som kan vara intressant. En annan aspekt för framtida forskning är att undersöka implementering av DevOps från fler intressenter för att kunna bidra med instick från en större mängd av företag som resulterar i en större mängd kvalitativa data. Rekommenderar även att intervjua flera personer från samma organisation då detta ger än större förståelse för hur företagen arbetar med DevOps.

Appendix A

Tabell 5: Transkribering av intervju 1 med respondent 1, förkortning R1

Rad	Personer	Information
1	Filip	Varför valde din organisation att implementera eller gå över till DevOps?
2	Respondent	Det är lite svårt för jag är ganska ny men det jag har hört och lärt mig så var anledningen när man arbetar i en stor organisation och utvecklar system så måste det också driftas och steget mellan utveckling och operations är ganska stort och om operations inte är med på utvecklingsresan kan det bli ganska jobbigt att underhålla och operera ett sådant system. När man kommer närmare varandra så minskar man detta avstånd mellan dev och operations och det finns ganska många vinster till det, mer skalbart, närmare samarbete så jag tror det som var anledningen. Även flexibilitet och ansvar end-to-end. Att när man utvecklar en funktion så har man ansvar för den hela vägen till operations. Det vi gör är att vi har så kallade Purple teams, dessa är medlemmar i våra SCRUM-teams som faktiskt operera lösningen i slutändan, vi är en stor organisation och har kanske 50 SCRUM-teams så det är stor skala vi pratar om och vi har en egen Ops-avdelning som sitter i ett helt annat land och jobbar med supporten men när man jobbar tillsammans så får vi en bättre lösning överlag till slutkund vi lyckas få bättre kvalité då. Att Ops-personalen verkligen vet vad lösningen betyder eftersom man har varit med i utvecklingen från början. Man har varit med på kraven och vilken typ av lösning som förväntas och man förstår vad som händer i systemet. Det finns många fördelar att jobba med DevOps. Självklart måste man ha arkitekturen för att supporta DevOps, man måste ha microservices man ska ha service based architecture och det gjorde vi ju då när vi gick över. Den arkitektur som vi har tillämpad matchar de kraven som DevOps ställer. Det går inte att göra DevOps som en ”wannabe”, det blir väldigt jobbigt.
3	Måns	Det var vår nästa fråga, om ni använder er av något ramverk eller arkitektur för att bygga upp det? Skapade ni ert eget eller använde ni ett befintligt?
4	Respondent	Lite både och då, vi har tillämpat de DevOps modeller som fanns tillgängliga för oss. Vi jobbar väldigt mycket med infrastruktur så GitOps är ett bra sätt att automatisera via teraform och infrastructure as code så vi har en hel vertikal då som vi kallar för en techservice och ett team äger hela den servicen så det kan vara login till exempel och man äger hela vertikalen då, man har all kunskap och man deployar i den miljön och så och man kan drifta den. Det är så vi har tillämpat det.
5	Måns	Okej! Har varje team en egen techservice eller finns det en gemensam för alla team?

6	Respon-dent	Nej varje team har en egen techservice en så kallad microservice med olika tjänster. Så ett SCRUM-team äger också en techservice.
7	Måns	När beslutet togs, vet du om det var från ledningen nedåt eller var det något som växte fram naturligt och var ett mer bottom-up beslut som växte fram för att man kände att det här behövs för att vi ska kunna jobba på ett mer naturligt sätt?
8	Respon-dent	Jag tror en hel del kom från ledningen, dem bestämde att vi ville ligga i framkant och vara tidiga med DevOps, vi ville experimentera och vara väldigt tidiga med det här sättet att jobba och bygga vidare på de agila metoderna.
9	Måns	Inom DevOps pratas det mycket om kommunikation och samarbete, Hur fungerar detta mellan developers och operators? Använder ni er av gemensamma plattformar för att kunna se resultat eller är det mycket face-to-face möten eller hur fungerar det?
10	Respon-dent	Inte face-to-face eftersom vi är multinationatienella och vi har haft Covid senaste två åren. Vi har Microsoft Teams ju som vi sitter på nu, vi har gemensam infrastruktur med login och diverse verktyg, jag ska inte säga vad dem heter för då kan man lista ut vilket företag jag arbetar på. Men det är serviceverktyg där vi också kan kommunicera med varandra, så ja det finns gemensamma verktyg för kommunikation.
11	Filip	Du nämnde också att personer som ska sköta operations är en del av SCRUM-teamet redan från utvecklingsfasen så jag gissar att en del kommunikation sker inom teamet.
12	Respon-dent	Det finns ren Ops också men vi har dem här Purple-team som har inblandade Ops-människor. Dem börjar med att jobba i samma SCRUM-team som utvecklarna för att det är det bästa sättet att lära sig en lösning på djupet, vad det är och hur den verkligen fungerar då kommer man förstå väldigt mycket bättre när man väl tar över och ska drifta den. Dokumentation är inte ett tillräckligt bra sätt att lära sig, det gäller att jobba hands-on för att verkligen förstå.
13	Filip	Fungerar det på samma sätt åt andra hållet? Att vissa utvecklare följer med till operationsspåret eller detta är med en ensidig överlappning?
14	Respon-dent	Nej, det är mer Ops-folk som närmar sig utvecklare för att få bättre förståelse och bredda sin kompetens. Utvecklare gillar att utveckla och är inte så förtjusta i att vara on-call 24 timmar om dygnet. Vi är ett ganska stort företag som har specifikt Ops-folk så övergången från dev till Ops sker inte i samma utsträckning. Vi är ett stort företag men har man bara ett SCRUM-team så är alla Dev-Ops, då gör alla allt. Får man en serverkrasch mitt i natten får alla hjälpa till att fixa den. Men när man har arkitekter och sånt så ser folk helt annorlunda på det, det finns olika skalor av DevOps beroende på företagets storlek. Vi har DevOps när vi utvecklar men det blir lite olika när man har många teams. Vi har väldigt

		många teams, det finns dev-folk men också Ops-folk som är med i utveckling för att få bättre förståelse för produkten.
15	Filip	Skulle du säga att kommunikationen är direkt även om den inte är face-to-face? Eller är det endast ett ticket-system som används?
16	Respondent	Det finns ett ticket-system också, det måste vi ha för att hålla koll på saker men absolut, det är mycket direktkontakt mellan de anställda
17	Måns	Är det någonting du kan se som hade kunnat förbättras med kommunikationen, något annat verktyg eller arbetssätt?
18	Respondent	Inget jag kan komma på nu eller tänkt på tidigare.
19	Filip	Men det är framför allt att Ops är med tidigare i varukedjan som gör att man får önskade effekter?
20	Respondent	Ja, absolut! Och att man kan se att koden är levererad. Vi jobbar med public cloud så kostnaden prioriteras, om något görs på fel sätt så måste de uppmärksammas direkt eftersom det annars blir väldigt dyrt i produktion. Dem måste få den feedbacken direkt. Att det kanske är dålig kod och den kommer kosta för mycket att drifta. Feedback är extremt viktigt och att man får den direkt.
21	Måns	Nästa område vi har handlar om automatisering och kvalitetssäkring och testning för att till exempel hitta buggar och liknande, är det något som integreras i era system eller hur arbetar ni med detta?
22	Respondent	Absolut! Vi har en hel testarkitektur, vi vill ha så mycket information som möjligt från vår CI-kedja och vi vill se direkt när något går fel för att kunna fixa det så snabbt som möjligt. Vi arbetar väldigt mycket automatiserade tester på enhetsnivå, systemnivå och hela vägen upp. Man ska inte ta en hel miljö åt gången utan man bryter ned testerna på komponentnivå, man ska äga den kod man skriver och man ska se till att det funkar. Men som sagt, väldigt mycket stöd av testning på alla nivåer.
23	Måns	Arbetar ni på något annat sätt med kvalitetssäkring eller det är framför allt testning?
24	Respondent	Det finns hur mycket som helst! Vi har olika krav och regler man ska förhålla sig till. Organisationsriktlinjer osv. Hur mycket som helst. Mitt företag har väldigt mycket fokus på built-in-quality
25	Filip	Vilka förändringar i organisationsstrukturen behövde göras för att göra DevOps möjligt? Nya avdelningar? Nya team? Ny personal? Hur såg det ut innan jämfört med hur det ser ut idag?
26	Respondent	Jag var inte här innan men jag antar att det liknade mer silostruktur innan. Idag implementar vi safe, en scaled agile setup och den hjälper. Vi Cross-functional teams och det är dem som är den viktigaste komponenten i hela DevOps. Du har teams som består av arkitekter, utvecklare och

		operations-folk, alltid i samma team. Vi bygger teamen på ren kompetens, medlemmarna kan också flyttas runt till andra teams om det behövs och detta görs ofta för att optimera organisationen och den kompetens vi besitter. Nya avdelningar vet jag inte riktigt om vi införskaffade men teams är viktiga, vi är mer team-baserade än avdelningsorienterade. Det ska vara bra samarbete inom teamet, tydliga roller, tydliga mål, tydlig ansvarsfördelning för att linjera hela verksamheten.
27	Filip	Så kanske mer fokus på teams istället för avdelningar?
28	Respondent	Absolut! Det som är viktigt i dessa är lineup. Alignment på SAFE. Vi har ett releasetåg som går hela vägen upp till solutionnivå. Så du kan ha ett arc som består av kanske 12 SCRUM-teams, vi har kanske 5 teams till VM-solution som har hand om full lösningsnivån och dem här ska så klart synka med varandra, både mellan och inom teamen. Det är inte bra om alla jobbar åt olika håll. Alla måste gå åt samma håll och dem taktas med PI, program increments så alla releasar samtidigt. Vi har kontinuerliga releaser men man release on demand så att säga, man taktar tillsammans och alla går åt samma håll. Säger vi att nu ska vi implementera funktion X så ska alla gå åt samma håll. Så SAFE är ett väldigt bra sätt för oss att arbeta, det supportar DevOps, att SCRUM-teams jobbar med Dev-Ops. För mig är DevOps att man tar end-to-end ansvar. Man kommer alltid ha first-line, second-line och third-line och det kommer finnas separata Ops-avdelningar men utveckling och operations jobbar tillsammans för en lösning är sättet vi arbetar på.
28	Måns	Har det skett några kulturella ändringar med införandet av DevOps?
29	Respondent	Ja absolut! Vi har blivit mer moderna skulle ja säga. Vi använder den absolut senaste teknik, chatops och automatisering. Man får laserfokus på att lösningen ska vara så lätt som möjligt att drifta, underhålla och monitorerna. Allting ska vara så automatiserat som möjligt allting constructed som code. Det här med att kasta över en massa kod över staketet till Ops-folket finns inte längre, det ska vara tajt samarbete och man ska kunna göra AB-releaser och kunna få feedback i realtid från olika kunder och kunna utvärdera vilken lösning som är bäst. Vi tittar på hur facebook och google jobbar, vi vill kunna göra många kontinuerliga och små releaser ofta, jag tror Netflix gör 2000 releaser om dagen, man ska releasa ofta och man måste ha DevOps för att göra detta. Det går inte att göra dem här big-bang releases en gång i månaden som fuckar upp allting, det går inte längre. Man måste ha små releaser ofta och DevOps får det att funka.
30	Måns	Ja vi läste att Amazon uppdaterar sin hemsida var sjunde sekund
31	Respondent	Jävlar!! Haha, det funkar kanske inte om man opererar ett kärnkraftverk men det finns något mitt emellan. Jobbar man med telekom eller gamla banksystem det finns en stor mängd branscher och företag där man kan bli bättre och får man testa i produktion så ökar effektiviteten.

32	Filip	Vad tycker du är svårast med att arbeta med DevOps? Är det något som hade kunnat förbättras? Finns det återkommande problem som ni stöter på kontinuerligt?
33	Respondent	Utvecklare vill utveckla och vara en del av the creative process, inte drifta saker, egentligen så tar vår utveckling ansvar end-to-end men ibland vill dem inte blanda sig i produktionen. Dem tycker det verkar läskigt men ja kommer inte på något. Det skulle i så fall vara dålig implementation av DevOps, det finns ju ingen direkt definition av vad det är. Det kan se ut på ett sätt hos oss och sen vara helt annorlunda på ett annat. Det finns ingen definition utan är mer ett buzz-ord
34	Filip	Vad skulle du säga är det vanligaste felet i en dålig implementation av DevOps? Vart är det vanligast att man går bort sig?
35	Respondent	Att inte göra det fullt ut! Nr.1 att inte organisera om som vi pratat om och nr.2 att inte ha en arkitektur som inte supportar DevOps. Att försöka tillämpa sina leacy-system rakt av på DevOps ska man inte göra, det går inte. Den arkitektur sättet man driftar den kommer inte vara kompatibel. Ska man göra DevOps så ska man göra det från grunden. Man ska inte försöka retrofita befintliga system. Det är doomed to fail.
36	Filip	Är det lättare för Ops-personerna att involvera sig i utveckling kontra att få utvecklarna att drifta?
37	Respondent	Ja!
38	Filip	Vad tror du det beror på?
39	Respondent	Jag tror att Ops ses ner på lite. Man ska jobba mitt i natten och alltid vara tillgänglig och se till att systemet snurrar. Jag tror ju mer tid som går kommer man se Ops som en del av utveckling, det kommer bara vara en grej och då försvinner Ops. Egentligen försöker man automatisera bort Ops helt och hållet. När ni kör gmail, hur ofta kontaktar du support liksom?
40	Måns	Nej, det är aldrig
41	Respondent	Precis! Så det finns liksom ingen first-line osv. De har ju massa bug-engineers osv. Tittar ni på google SRE:s så har dem väldigt hög status, dem är med på hela resan och ses som högsta tjänst inom utveckling och jag tror att det går åt det hållet. Att man förstår att reliability och Ops är en del av utvecklingen. Så i den klassiska bemärkningen tror jag man ser ner lite på Ops tyvärr. Utvecklare gillar att utveckla, dem vill inte sitta och operera, dem brinner inte för det, dem vill vara kreativa och brinner inte för drift. Men när man ser Ops som något man kan koda och automatisera så blir det lite mer intressant tror jag.
42	Filip	Så endgame är att ingen kommer ha Ops i sin titel eller arbetsbeskrivning?

43	Respon-dent	Nej precis! Om man tittar på cloud så får man väldigt mycket gratis, man sitter inte och driftar servern. Allting går mot cloud så den biten försvinner av naturliga skäl. Sen har vi din lösning och är den tillräckligt bra gjord så kommer du inte behöva göra särskilt mycket. Så Ops kommer minska och förhoppningsvis försvinna och kvar finns bara en organisation som är DevOps, det måste vara ambitionen tror jag. Endgame är att allt blir Dev.
44	Måns	Hur ser du på framtiden? Både inom ditt företag och allmänt, vad tror du vi går mot? Mer automatisering? Bort med Ops helt?
45	Respon-dent	Det hetaste ämnet är definitivt automation, vår chef pratar om det på varje möte, så mycket att det nästan blir tjatigt. Dem här stora komplexa lösningarna har blivit väldigt dyra i drift och business-caset blir dåligt väldigt fort på grund av detta. 20/80 regeln, att det kostar 20% att utveckla och 80% att drifta det går inte längre så man måste fokusera på automatisering och kanske använda lite AI, det kommer mycket med AI från Ops sidan också, man kanske kan gå åt det hållet. Att kunna se anomalier eller säkerhetsproblem med AI, då kan man också automatisera Ops. Jag tror allting kommer bli dev i framtiden och att Ops i så stor utsträckning som möjligt är automatiserat eller åtminstone standardiserat.
46	Filip	Arbetar ni aktivt med DevOps idag? I form av organisationsstruktur, arkitektur och infrastruktur, förbättras och förändras det hela tiden eller känner ni att det är ”på plats” och sköter sig själv nu?
47	Respon-dent	Det var en bra fråga! Nej vi har det här continuous improvement så det är alltid förbättring. Vi har retrospect efter varje sprint, vi utvärderar alla projekt och vi försöker alltid bli bättre och det är en del av DevOps, det är en del av all utveckling, man ska systematiskt förbättra varje del av verksamheten.
48	Filip	Så ni ser på DevOps som en slags levande organism?
49	Respon-dent	Ja! Bara titta på alla ramverk och verktyg som finns, dem blir bättre hela tiden och vi måste förändras med tiderna. Kommer det en ny cloudtjänst som är bättre måste vi använda den för att vara konkurrenskraftiga. Så vi försöker alltid förbättra de tekniska aspekterna, ramverk och arbetssätt. Som jag sagt finns det ingen standard att förhålla sig till inom DevOps, det finns ingen maturitymodel för DevOps, det borde det kanske göra.
50	Måns	Det var alla frågor och vi tackar så mycket för din medverkan!
51	Respon-dent	Tack själva grabbar! Ha det bra!

Appendix B

Tabell 6: Transkribering av intervju 2 med respondent 2 - Fredrik Töörn, förkortning R2

Rad	Personer	Information
1	Respondent	Hej, trevligt och träffas, välkomna hit.
2	Måns	Tack så mycket, kul och vara här!
3	Filip	Ja verkligen!
4	Måns	Går det bra att vi spelar in den här intervjun?
5	Filip	Och vill du vara anonym i både transkribering och uppsatsen i helhet?
6	Respondent	Jag behöver absolut inte vara anonym och det går bra att spela in intervjun
7	Måns	Om vi börjar med lite bakgrund från din sida och projektet Skånetrafiken som vi ska prata om idag, varför ville Skånetrafiken gå över till DevOps och anlita er som konsultbolag?
8	Respondent	Bakgrunden är att för 6 år sedan så började vi jobba med Skånetrafiken och för att förtydliga vi här ju både med strategi och teknik. De tog primärt in oss för strategiarbete, de ville börja jobba mer agilt. De var ett väldigt traditionellt IT-bolag om man tittar på IT-avdelningen med vattenfallsmodellen. Vi höll en del kurser och föreläsningar för management om hur man jobbar agilt. Under dessa möten framkommer det att Skånetrafiken har biljettsystem som dem inte är nöjda med och de vill bygga ett eget biljettsystem. Det skulle innebära att de skulle äga biljettsystemet och i sin tur ha frihet att göra vad dem vill med det. För de ansåg att nuvarande system satt dem fast med.
9	Filip	Var det ett externt system dem hade?
10	Respondent	Ja det var ett externt system som också hade en extremt hög driftkostnad. Detta resulterade att man tog beslutet att bygga ett eget system och att man ville bygga system på ett agilt sätt och vi ska göra det molnbaserat. Kostnadsmässigt driftmässigt innebär det här en 1/10 av nuvarande kostnad däremot finns det ju en utvecklingskostnad. Det är den huvudsaklig anledningen varför de kontaktade oss. Sedan satte man igång med upphandling då det är offentligt projekt och den vann vi och vi har jobbat med dem sedan dess. De hade en väldigt tydlig bild av vad de ville åstadkomma med en ordentlig backlog. Efter 2 år hade vi gjort ungefär 30% av backloggen som ersatte deras arbete dem redan hade gjort på projektet så det skrotades och vi tog över.

11	Filip	Vet du vad anledning var att Skånetrafiken ville börja arbeta mer agilt med DevOps? Hur såg det ut innan?
12	Respondent	Vattenfallsmodellen. Förut hade dem en IT-organisation och en verksamhet. En klassisk uppsättning. Kommunikationen mellan dessa avdelningar var inte speciellt bra och då hade dem dessutom en tredje part som levererade systemet till dem. Så när de väl ville göra nya idéer med sitt system så gick det genom lång kedja av steg och tog 1–2 år att förverkliga.
13	Filip	Vad var de huvudsakliga effekterna som man ville uppnå?
14	Respondent	Jag vet inte om ni minns deras tidigare app, kundupplevelsen var hemsk. De ville förändra det och jobba mer digitalt. Ett mer digitalt sätt att hantera biljetter på. Man ville lyfta fram en app mycket mer.
15	Filip	Hade ni en klar och tydlig bild av hur det nya arbetssättet skulle se ut?
16	Respondent	Vi har sedan länge en projektmodell som vi kallar det och Skånetrafiken tog en kopia på den och ändrade lite på den men använder nu den.
17	Filip	Skulle du kunna gå in lite på hur eran projektmodell ser ut? Är modellen baserad på er erfarenhet eller något ramverk?
18	Respondent	Vi har tagit ett agilt arbetssätt Scaled agile framework. Man har länge jobbat med sprintar och sedan tweakat det utifrån behoven med DevOps på vårt företag eftersom vi jobbar med både strategi och teknik. Därför måste projektmodellen också kunna hantera kravhanteringen från båda hållen. Så vad projektmodellen handlar om är att vi vill gärna vara med väldigt tidigt i projektet gärna i idéstadiet. Det är där vi ser att vi att vi är konkurrenskraftiga och kan utmana ordentligt och sedan även vara med i slutet av projektet, hela produktkedjan. Man vill också se en effekt av arbetet och det vill även Skånetrafiken därför är detta viktigt. Sedan har vi själva implementeringens delen den här delen mitt i kedjan. Det är här vi arbetar med DevOps och arbetar agilt med backlog och sprintar.
19	Filip	Hur ser sammansättningen av teamet ut hos er och Skånetrafiken under implementeringsfasen? Är det både utvecklare och driftpersonal involverat?
20	Respondent	Vi gör faktiskt ingen differentiering på utvecklare och driftpersonal. Vi jobbar inte med ren förvaltning.
21	Måns	Innebär det att utvecklare har mer ansvar och ser till att produkten körs hela tiden?
22	Respondent	Absolut. Teamet består av vi har en projektledare naturligtvis och en deltids scrummaster. Sedan har systemarkitekt, utvecklare, UI-design. Sedan har vi delat in utvecklare i funktionsgrupper som tittar på funktionsområden. Men alla i teamet lär sig hur system eller produkten är uppbyggd och kan ta ansvar för haveri. Vi kör detta via Microsoft Cloud Azure. Så blir det problem med de generella funktionerna så kan alla

		hjälpa till. Sedan finns det såklart specifika funktioner som endast vissa har kunskap om. Så vad vi egentligen har är att vi har ett rullande schema på "jourtjänst" på drift momentet där alla då och då är på alerten och hjälper till med det operativa. Vi ser att de flesta operativa problem inte är kritiska och då lägger vi dem i backloggen inför nästa uppdatering. Kritiska problem ger sig jourpersonalen in och löser dem direkt. Dygnet runt har vi personal. Detta innebär att vi då allokerar om resurser till problemet vilket och märks av i sprintarna som vi jobbar med. Men vi har också jobbat med Kanban tidigare. Kanban är egentligen lättare för då prioriteras det som ligger överst i backloggen istället för att planera en sprint på två veckor för att få in problemet och lösa det. Vi använder Microsofts verktyg Azure DevOps molnlösning och hela teamet arbetar där. Vilket är ett toll-set som hanterar allting från idé till slutförande. Vi använder också det för test-miljö och test lanserar våra produkter via det. Arbetar även med kvalitetsgranskning via detta program.
23	Måns	Ni kör då lanseringar via det här verktyget så fort det är redo?
24	Respondent	Ja det gör vi men generellt har vi två veckors sprintar och lanserar efter det för att uppnå kontinuerlig integration och kontinuerlig leverans. Dessa leveranser går sedan vidare till en testmiljö där vår kund har möjlighet att testa själva och vi får feedback på detta. För att det sedan skall gå vidare till slutgiltlig lansering behövs det tre personer inklusive en testare, en kvalitetsgranskare och en utvecklare godkänna det samtidigt sedan när kunden också har godkänt som lanseras produkten live. Det tar ungefär 3–4 veckor från att vi börjar koda till vi har något på marknaden live.
25	Filip	Sker testerna manuellt eller är dem automatiserat?
26	Respondent	Testerna är automatiserade. Vi jobbar mest med APIer vilket underlättar den processen då appen konsumerar den.
27	Måns	Hur kommunicerar DevOps teamen? Används gemensamma plattformar för att se resultat och framsteg? Kommunikationsverktyg?
28	Respondent	Alla kan se allt via Microsofts DevOps Azure molntjänst och sedan använder vi också slack för snabb kommunikation. Alla i teamet har access till det här verktyget. Det här verktyget styr också kvalitetsgranskning. Vi har också automatiserat hela uppsättning av infrastrukturen i Azure. Vi har också kopplat notiser från DevOps verktyget in i slack exempelvis för kvalitetssäkring så får jourpersonalen en notis om det. Vi får alla våra larm och ser till att det ringer i någonstans vid problem. Speciellt viktig är slack nu på grund av pandemin när folk har suttit hemma osv.
29	Måns	Funkar det lika bra med kommunikationen nu efter pandemin folk jobbar hemma?

30	Respon-dent	Ren utveckling har vi fått upp effektiviteten och kommunikationen och delning av kod har blivit bättre av distansarbete. Men för att ta fram en backlog och vara kreativ med idéer och skapa någonting är mycket svårare med distansarbete gentemot face to face.
31	Filip	Så själva sprinten har blivit mer effektiv men planering av sprinten har blivit mindre effektiv?
32	Respon-dent	Ja absolut. Planering går hyfsat men innovationen går sämre. Att sitta och kommunicera såhär som vi gör nu face to face är ju en annan sak gentemot att sitta i videosamtal det är svårare att måla upp idér och liknande. Det finns inga riktigt bra verktyg för att kunna göra det digitalt och spontanitet blir också svårare.
33	Filip	Vad behövde Skånetrafiken göra med deras organisationsstruktur för att få DevOps att fungera? Var det nya avdelningar / nya team?
34	Respon-dent	De gjorde faktiskt en revolutionerade sak tycker jag personligen. De hade en IT-avdelning och verksamhet men när de beslutade sig för att genomföra det här nya projektet så valde dem att sätta ihop en ny utvecklingsavdelning som bildade ett lab som endast jobbade med detta nya projekt. Detta nya lab ägs av verksamheten och inte IT-avdelningen. Sedan finns delar av IT-avdelningen lite där också men det körde ett helt nytt team och avdelning för att få igång skapande och innovation utan andra processer som integrerade. Första året var det bara nyskapande men desto mer tiden gick desto mer integrerades IT-avdelningen i det här nya projektet. Men det mesta är ju realiserat i molnet vilket gör att du inte har samma traditionella behov av den här IT-avdelningen.
35	Måns	Nya system och infrastruktur?
36	Respon-dent	Det är Microsoft Azure DevOps. Hela systembiten är kopplat till det. Så vad man kan se är Skånetrafiken har hela sitt biljettsystem i molnet men deras tidtabeller är lokala servrar vilket gör att ibland så kan du köpa en biljett men inte se när din buss går då två olika system har ansvar. De har ju råkat ut för lite DDOS attacker och det mäktar inte deras eget system med det men Microsoft kan ju styra om trafiken så molnet klarar av sådana saker.
37	Filip	Har det skett några kulturella förändringar vid införandet av DevOps?
38	Respon-dent	Absolut, vi är ju ett traditionellt bolag i vissa av våra projekt om vi tittar utanför Skånetrafiken. Det projekten är ju lite vattenfall, vi kommer in i projektet och får en kravbild och jobbar med den och levererar efter det och sen är vi färdiga. De större projekten har vi som tur är en agil arbetsmetod och den typen av företagskultur som hör till det. Förväntningshanteringen är ju annorlunda i ett agilt arbetssätt, det är svårt att sätta en tydlig deadline på när arbetet är klart då projektet kontinuerligt förbättras och arbetas med i sprintar efter vi inte lovar något. Vi brukar kalla det ett managed team vi utlovar kompetens. Vi ser till att vi har en

		kapacitet att ta hand om dem problem som finns. Vi har också en ekonomi med i DevOps teamet faktiskt.
39	Filip	Hur har utvecklarna tagit den nya rollen av att ta mer ansvar för hela processen? Istället för vattenfall där utvecklarna gör klart sin kod och sedan tar deras ansvar slut?
40	Respondent	Vi ligger inte på 20% utveckling 80% drift utan snarare 60–70% utveckling och sedan 30% drift. Eftersom man är involverad från början till slut så vill man som utvecklare ta ett större ansvar och se till att man skriver bra kod och som har låga driftkostnader eftersom man får ta hand om sin egen skit. Detta innebär att som utvecklare så tänker man sig för vad man skapar och tar då hellre ett ägande av sin egen produkt. Implementerar mer kvalitetskrav för att slippa onödiga buggar vid ett senare tillfälle. Ägandeskap är något alla sätter ära i. Det här ska vara långsiktig och minska tekniska skulder och ta hänsyn uppdateringar hos program man arbetar med som också uppdateras. Allting har blivit mer långsiktigt vilket är mer attraktivt. Den fysiska miljön är inte nödvändig med Azure, då vi kommer från mjukvaru hållet och inte är ett traditionellt IT-bolag.
41	Måns	Hur ser ni på er beroende roll av Microsoft Azure DevOps?
42	Respondent	Självklart är vi beroende av deras kapacitet gentemot ett traditionellt IT-bolag men vi har också många fördelar, vi kan lansera produkter i rusningstrafik utan problem tack vare det och det tror jag inte många andra IT-bolag har kapacitet till ifall de inte är ledande inom branschen. Du kommer behöva en serverteknik ifall du vill göra denna typ av leveranser annars och det slipper vi. Bara arkitekturen av verktyget ger möjlighet att skapa Ops delen inom DevOps.
43	Måns	Hur ser du på framtiden med DevOps? Ännu mer automatiserat?
44	Respondent	Vi har ju fördelen att vi inte har så mycket interface och det sparar oss otroligt mycket tid. Vilket också ger oss möjlighet att fokusera mer på kvalitén. Sen pågår det diskussioner med andra entiteter som också vill ta del av vårt DevOps arbete med Skånetrafiken så där har vi också en framtidsutsikt. Det var inte vad vi tänkte på när vi designade system så där kommer vi på nog behöva göra någon förändring arkitekturellt men sedan arbetsmässigt tror jag inte vi kommer ändra speciellt mycket. Vi kommer behöva fler specialister på flera funktionsområdet men målet är att vi ska arbete med DevOps och ha fler som kan vara inne systemet och ha koll på Ops och funktionalitet. Det här DevOps system som vi har satt upp funkar verkligen riktigt bra. Sedan vet vi inte vad framtiden har på gång med hela den här Web 3.0 grejen och blockchains. Distribuerade infrastruktur. Vi är ändå öppna ifall infrastrukturen ger nya möjligheter.
45	Filip	Den här arbetsmodellen är inte huggen i sten?
46	Respondent	Ja den kan komma att förändras men jag tror modellen med sprintar, en agil arbetsmetod, den tror jag kommer vara kvar. Vi är inte intresserade

		av att jobba i en vattenfallsmodell, det är skitsvårt att sätta sig ner och bryta ner alla krav i ett stort system och sedan leverera en stor leverans på slutet av projektet. Plus att vi att när vi har gjort en liten påväg så kommer det nya idéer därför är det bättre att jobba med backlog och sprintar. Sen om du kör sprintar eller Kanban spelar inte så stor roll. Upplägget där vi har DevOps där utvecklarna också är med i det operationella är bra och jag ser ingen anledning till att byta det. Det kanske beror lite på hur komplext det blir i framtiden och hur mycket som går att automatisera.
47	Filip	Största nackdelarna med att jobba med DevOps?
48	Respondent	Det finns ju nackdelar med alla arbetssätt. Men en sak är att det går inte riktigt att predicera hur långt tid det tar att förverkliga vissa ambitioner då nya saker kan uppstå längs vägen. Vi kan inte frysa allt och sätta oss ner i 3 månader och bara bryta ner kraven utan det är en mer flytande process. Men för den egna utvecklingen tror jag inte det är så många nackdelar med DevOps. Jag skulle vilja se att vi fick chansen att öka på innovationsarbetet. Det finns många idér som inte riktigt kommer med i nuläget så ännu bättre samarbete mellan verksamhet och DevOps utvecklare hade varit optimalt så att man tillsammans kan spåna fram saker. Mer BuisDev.
49	Filip	Har ni har uppnått de förändringar och önskad effekt med Skånetrafiken?
50	Respondent	Ja det tycker jag absolut. Sen blir ju systemet mer komplext och då tar det längre tid att uppdatera gentemot i uppstartningsfasen. Men det jobbar vi med så att man kan uppdatera ett område utan att stanna systemet i det huvudsakliga verksamhetsområdet.
51	Filip	Vad anser du är de viktigaste successfactors för att jobba med DevOps? Vad måste man få rätt?
52	Respondent	Det är framför allt kompetensen hos människorna och viljan hos dem personer. Det handlar om en införsäljningsfaktor exempelvis att jobba med förvaltning är inte roligt men som vi ser det om du jobbar med förvaltning i en kontinuerlig förändring så då blir det bättre. Men kompetensen är A och O. Sedan tror jag det är viktig att vi vill vara med tidig i produktkedjan i idéfasen. Vi var med väldigt tidigt i Skånetrafiken fallet exempelvis så fanns det 1000 olika biljettyper innan men vi hjälpte till att bolla fram att det endast skulle vara 3 olika zoner av biljetter vilket underlättade jättemycket för byggandet av biljettsystemet.
53	Filip	Är det därför du tror att utvecklarna tycker om att ta ägandeskap för produkter för att ni är med så tidigt i idéstadiet?
54	Respondent	Ja det tror jag. Vi är inblandade väldigt tidigt i dialogen gällande krav. För mig och de som jobbar här tycker det är viktigt att vara med och påverka och sen slutligen se resultatet av arbete man har gjort. Det är roligt att skapa något. Förr i tiden satte man upp en kursplan och följde den nu

		sätter man upp en miljö som man kan vara del av på ett annat sätt. Det är därför vi kallar det manage teams, vi vill inte hyra ut en resurs, vi vill att det kommer ett team som kommer och hjälper kunderna med problem. I det här teamet skapar man en gemenskap och möjlighet och tillsammans kan man växa.
55	Måns	Okej, jag tror vi har fått med allt vi behöver här så vi tackar så mycket för din medverkan.
56	Filip	Ja verkligen stort tack för att du ställde upp på intervju.
57	Respon-dent	Tack själva, det ska bli spännande att se vad ni skriver sen och funderar ni på något är det bara att slänga iväg ett mail.

Appendix C

Tabell 7: Transkribering av intervju 3 med respondent 3, förkortning R3

Rad	Personer	Information
1	Måns	Ska vi se då var vi igång med och första saken som jag tänkte att vi kunde ta upp lite är bakgrund till hur ni började arbeta med DevOps, när gick ni över och hur länge har du jobbat med den rollen du har idag?
2	Respon-dent	Jag själv har varit i den här rollen i knappt ett år nu. Jag hade lite andra grupp-chefsuppdrag på företaget innan jag kom in här och vi har kört DevOps och agilt sedan förra våren så det blir cirka ett år. Historiskt sett så har det varit så att det en stor grupp utvecklare där det fortfarande har 24 anställda och 13 konsulter. Förr var det liksom en klump med utvecklare och så var det en person som fick in 70 000 olika förfrågningar och jag överdriver lite men en person som fick in alla förfrågningar och som sedan mer eller mindre delade ut dessa till alla utvecklare. Detta funkade inte ju mer vi växte. Vi är nu uppe i 10 olika bolag som är uppköpta och den utvecklar gruppen jag är chef för just nu många av dem kommer ifrån den tiden när man var ett mycket mindre företag och då funkar det gamla sättet mycket bättre för då kände alla i företaget varandra och man satt på samma kontor då. Nu är vi sextonhundra medarbetare utspridda över hela Sverige och då funkar det liksom inte att man skickar ett mail till någon om att man vill ha någonting utvecklat och så ger den personen den uppgiften till någon. Där är lite bakgrunden till varför vi ville gå åt det agila och DevOps och det var absolut då skiftet skedde. Jag har min roll som chef för utvecklarteam och är linjechef för dessa. De rapporterar till mig, jag står för rekrytering och allting kring lönesamtal samt performance reviews, allting kopplat kring personen. Sen har vi delat upp våra medarbetare i olika produkt eller leveransteam och där är ju alltså hela syftet med de agila tankarna eller ett av det stora syftet med de agila tankarna är att försöka få självorganiserande team det vill säga jag eller ingen annan för den delen ska behöva gå in och micro managera utan teamet ska ha den kompetensen och det förtroendet så att man kan lösa sina dagliga arbetsrelaterade problem så jag slipper gå in och peta i någon specifik utan de sköter sitt jobb. Vi har delat upp teamen i 8 olika leverans delar efter vår EUS interna affärsprocesser så till exempel har vi som jobbar mot billing och ett mot finans osv. Varje team har en produktägare och produktägaren tillsammans med representation från verksamheten som vi på iver kallar för processägare bestämmer tillsammans vad som ska göras och försöker vara lyhörda för vad vår verksamhet behöver så man inte fastnar i att bara utveckla coola nya funktioner. Vi vill inte utveckla funktioner som inte gör någon nytta i organisationen. I vårt fallet är vi är en intern stödfunktion åt resten av företaget. Vi är uppdelat

		så att det är ett utvecklarteam, sen så har vi verksamhet och så var det lite olika med produkt-nisch där och på det på det viset ja men precis så vi har ju nästan överallt utpekade så kallade processägare då ifrån vår verksamhet som vet vad som ligger i pipen och vilka stora initiativ bör vi göra men även liksom småfixa.
3	Måns	Jobbar ni automatisering inom tester eller hur funkar den biten?
4	Respon-dent	Vi lägger mycket fokus på automatisering till exempel de ekonomiska flödena kan vi ju skära bort otroligt mycket manuell handpåläggning och skapa dokumentation för hur detta fungerar. De olika produktägarna håller reda på vad som ligger i backloggen tillsammans med processägaren och väljer vad som ska prioriteras. Detta planeras och läggs in i sprintar, vi kör sprintar på 2 veckor. Vi försöker få med så mycket som möjligt på dessa två veckor men det kan vara en utmaning att estimera och förstå hur lång tid något tar när man är van vid den gamla vattenfallsmodellen. I början är man inte jättepricksäker och oftast så tar man kanske på sig lite för mycket i början och det här är liksom en del av hela den här agila tanken men med ständiga förbättringar så att man också liksom lär sig att estimera så att man blir så pricksäker som möjligt så man hela tiden förbättrar sin planering. Sen kan det hända oväntade saker som förstör en sprint men vi arbetar med att bli bättre på att planera sprintarna så att det som ska göras kan gå till release efter 2 veckor.
5	Respon-dent	Sen jobbar vi mycket med testning och intern kvalitetssäkring och så där innan kunden tar del av lösningen. Där har vi har lite utmaningar för den systemet vi jobbar i började alltså utvecklas 1998 så det har några år på nacken vilket gör att vi har en källkods-bas på kanske beroende på om man räknar men 10 miljoner rader kod. Nu vet jag inte hur mycket ni själva utvecklat men 1998 var till exempel clyde server arkitektur skit-coolt. Idag är det inte alls coolt så ibland arbetar vi med kod som är 10, 15 till och med 20 år gammal och då var inte heller testning så coolt, så vi försöker retroaktivt få inte automatiserad testning för ett väldigt gammalt system. Däremot den nya kod vi skriver har vi kommit en liten bit på vägen, bara de senaste 2 månaderna har vi arbetat mycket med test-driven utveckling och automation av denna. Vi vill få in detta i vårt continuous integration flöde och kunna göra till exempel regression och all kod ska dessutom dubbelkollas av någon annan, så vi är på väg med automation och säkerhetsaspekter när det kommer till tester men vi jobbar med många svårigheter kopplade till vårt arv. Systemet är internt utvecklat, så har vi har nästan inga inköpta system för allt ifrån finans fakturahantering till tidrapportering till cdb och allting sånt är egenutvecklat.
6	Filip	Kul! jag tänkte på det här med sammansättningen av teamen, om jag förstod det rätt så var teamen baserade på produkten alltså vilken inriktning den hade, där fanns en processägare och en produktägare. Består resterande personal av utvecklare? är där även driftpersonal som är med från början av utvecklingen eller de kopplas in i ett senare skede och bara tar över driften?

7	Respondent	Nu har vi precis gjort en omorganisering här. Jag står ju ganska mycket för Dev-delarna i DevOps flödet. Vi har en intern gruppering som vi kallar för DevOps-intra vars huvudsyfte var att vara med så att man får just den här driftsperspektivet för annars faller hela DevOps konceptet. Från och med måndag så kommer en ny organisationsstruktur och då är tyvärr den gruppen bortflyttad från oss så Ops delen av DevOps brottas vi med just nu tyvärr. Jag vet inte hur mycket ni vet om oss med vår huvudsyssla är inte att utveckla system som min grupp gör utan vi säljer genererade IT tjänster till andra bolag, man kan ha sina servrar hos oss, backuper, nätverk. Det är det som gör att vi brottas lite med Ops delen just nu. Vi är tillsagda att vi internt ska beställa den infrastrukturen och de miljöer som vi behöver av organisationen som samtidigt jobbar med att sälja och drifta åt kund och ni kan ju tänka er vart vi utvecklare hamnar på den prioriteringslistan. Så det blir ett problem när vi inte är ett renodlat utvecklingsföretag och dessutom säljer drifttjänster, de interna grejerna blir tyvärr inte prioriterade och detta försvårar vårt DevOps arbete. Det är svårt att ha en bra Ops del och dessutom försöka få med den på utvecklingsresan med den nya organisationsstrukturen. Vi håller på att lyfta den frågan du vi inte kommer ha intern Ops-kompetens inom min grupp, de är alla upptagna med kundprojekt och vi måste beställa kompetensen från den avdelningen alternativt ta in konsulter så Ops delen är lite up in the air just nu.
8	Måns	En annan sak vi kollar lite på är kommunikationen och då lite både mellan Dev och Ops men även inom teamet. Utifrån vad vi har sett tidigare så ser vi det som en ganska viktig parameter där hur fungerar det hos er det har ni något gemensamt verktyg där man kan se varandras kod och hur det går med projektet?
9	Respondent	Allting vi gör är helt visuellt för alla vi kör DevOps med våra tavlor som är öppna för alla, jag som inte jobbar aktivt har ändå access att titta på allt, jag kan inte ändra någonting men jag kan se allt och medlemmar i teamet man sitter i kan givetvis flytta på kort och lägga till och skriva uppdateringar och sånt så den delen är helt visuell. Vad gäller produkt teamet till affärsverksamheten så är så är det lite inbyggt i den modellen vi har tagit fram, vi kör ju SCRUM så vi har det här med sprintdemos allting sånt som är inbyggd i det arbetssättet och vi jobbar mycket med workshops och annat tillsammans med verksamheten för att försöka sprida kunskap och erfarenheter men det är en work in progress för oss hur vi ska kommunicera. Den stora utmaningen är hur man kommunicerar mellan team. Eftersom vi är indelade i affärsområden där personalen inte byter område så bildas silos runt dessa områden och silos är aldrig bra. Vi försöker motverka detta genom att alla dessa silos står på en grund där vi försöker samla kunskap. Denna plattform kallar vi Steam och i denna plattform har vi våra gemensamma bibliotek, ramverk, kodstandarder, säkerhetsprotokoll som alla måste förhålla sig till så det är ett sätt att kommunicera de tekniska kraven i alla fall men vi har svårt att hitta ett bra forum för kommunikation utvecklare till utvecklare mellan teamen. Våra utvecklare vill utveckla och vill inte sitta med i en massa möten. Det är nog med möten för dem i och med den agila processen.

		Där får vi grooming möten, perspektiv, planering och retrospectives. Det är mycket möten med krav och hela den biten redan. Vi har försökt med lite olika spår, jag och min chef håller i möten osv men det blir med som information, ingen riktig dialog tyvärr. Det är självklart jätteviktigt att information och kunskap delas i organisationen men vi har inte riktigt hittat en bra modell för detta, framför allt inte för kommunikation mellan team, inom teamet löser sig själv men våra silos runt affärsområdena arbetar mot oss på denna front. Corona har dessutom gjort detta ännu svårare. Att utveckla passar perfekt med hemmajobb, man behöver tid att sitta själv och fokusera men de obligatoriska mötena i SCRUM-processen har blivit lidande och vi vill i framtiden lägga dessa på plats så man kan få det där kreativa utbytet när man träffar någon vid kaffeautomaten eller man hör att någon arbetar på ett problem som man själv sitter med så kan man diskutera detta direkt.
10	Måns	Ja det är många andra som säger det som vi också har intervjuat just det här med spontanitet och innovations delar
11	Respondent	Precis! Planering och sånt löser man men att sitta och överhöra varandra som man sitter vid så hör man någon snacka om något som låter intressant eller det jobbar jag ungefär med samma problem och det tappar man och vi försöker sakta men säkert liksom få tillbaka folk in på kontoret och vi sagt att en dag i veckan börjar vi med så tror vi att man kommer upptäcka fördelar med att faktiskt vara på kontoret så att det blir mer av en frivillig sak än att jag behöver säga ja men företaget har som policy att det ska vara 3 dagar i veckan det är bättre att få in folk frivilligt än att man tvingar in. Men med det sagt så har faktiskt produktiviteten gått upp med hemmajobb men vi har som sagt tappat samarbetet, både inom det egna teamet och samarbetet mellan team arbetar vi på en modell som kan fungera men inget som används än.
12	Respondent	Filip frågade om andra kompetenser innan…utöver produktägare och processägare är det endast systemutvecklare, vi har ingen nätverkstekniker utan han finns i vårt DevOps-infra grupp som är ett eget team som ser till att infrastrukturen för alla team fungerar. Där emellan har samarbetet fungerat väldigt bra och det tror jag beror på att de ses som ett av utvecklarteamen, de fokuserar på oss och har ingen kundkontakt, tyvärr flyttas ju Ops-teamet bort från mig och vi ska se hur det kommer fungera i framtiden. Samarbetet med DevOps-infra gruppen har fungerat bra eftersom man sluppit skapa tickets osv utan de har bara fokuserat på att teamen ska fungera bra, nu går vi till beställningsfunktion från Ops-delen och det kan bli problematiskt om vi inte är prioriterade.
13	Filip	Vart har den här omorganisationen kommit ifrån? Nu vill jag inte lägga ord i din mun men det låter inte som att du är jättenöjd med vissa av förändringarna?
14	Respondent	I stort det var jättenöjd med den omorganiseringen för det som vi har gjort är att, förut var varje region sin egen till exempel Väst öst syd och Norge vara en egen region. Varje region har sin säljorganisation sin

		driftsorganisation alla skulle sälja samma tjänster men de levererades på lite olika sätt. Nu har vi har tagit bort regionerna och så har vi delat in det mycket mer efter kompetensområden och kundsegment så vi har en avdelning som jobbar med Enterprise kunderna, så stora kunder som Mölnlycke som är en kund till oss, de är jättestora så har vi satt upp en organisation lite mer kring vad som den typen av kund behöver för stöd av oss. De kanske behöver lite mer hjälp med att vara innovativa deras digitaliseringsresa. Nu har vi satt upp en organisation anpassad efter våra kunder och deras behov. Då blev det så att vi har en avdelning som jobbar mer med små-medium kunder som kanske har mer behov av ett eget kundteam, samma person man alltid kontaktar. Så nu är organisationen uppbyggs utefter våra kunder och det funkar väldigt bra overall men tyvärr som jag nämnt tidigare så flyttas driftavdelningen från mig och ut till de nya områdena som är uppdelade efter kund. Nu har vi gemensam driftavdelning och gemensam sälj vilket har inneburit förändring och det är bra förändringar men vårt DevOps arbete blir lite lidande tyvärr. I denna fråga lyssnade dem inte på mig och jag har lyft frågan, vi kommer självklart lösa detta på något sätt men idag vet jag inte hur.
15	Filip	Hur såg arbetssättet ut innan ni började med DevOps? Du nämnde något om att ni bara var en klump utvecklare innan som blev tilldelade funktioner att utveckla.
16	Respondent	Ja precis! Han som ägde bolaget innan var väldigt involverad i utvecklingsprocessen. Det var faktiskt så här att i praktiken skrev ägaren varje morgon en lista med typ 10 grejer, det här vill jag ha utvecklat nu och så gick han med den till min företrädare hon som var gruppchefer för det för den här gruppen innan. Han lämnade listan och sen var det bara att börja. Det är inte så mycket pull i detta arbetssätt, man tar inget eget ansvar utan man blir tilldelad exakt vad man ska göra varje dag. Idag är en viktig del i vårt arbetssätt att vi vill att varje utvecklare tar ett eget ansvar för vad som ska utvecklas. Man tar den funktion som är högst prioriterad och tar sedan ansvar för att denna görs. Det är en resa att ändra på folks mindset, från att få en lista på morgonen med 10 punkter till att som team vara ansvarig för en hel leverans med allt från kravinsamling, utveckling, test och leverans. Vi är på god väg men detta tar tid, det är inget man får till på en vecka i praktiken. Det kan ni tänka på när ni börjar arbeta att vill man jobba agilt och DevOps så ska man inte börja den resan med att prata verktyg och plattformar, vi skulle kunna köra det i notepad. Agilt arbetssätt sitter i huvudet och är ett mindset. Det finns säkert 100 olika verktyg men den svåra förändringen sitter i huvudet. Det är en förändring i processer snarare en teknisk förändring.
17	Filip	Hur har dina kollegor anpassat sig till de kulturella förändringarna som kommit med DevOps? Har det varit lätt eller har vissa stretat mot?
18	Respondent	För vissa är det ju lätt, här blir det ju lite upp och ned vänt. Tar jag in en ny-exad utvecklare så är de bekanta med detta, de läser om agilt och DevOps i skolan och anpassar sig snabbt. Det är värre med dem som kanske har jobbat på företaget sedan 2000 då de var med och tog fram

		klientserver lösningen. Det var ju state of the art på den tiden. Så är det ju… om 20 år så kommer vi sitta och skratta åt och containerdrift och sådant som är modernt idag. Syntaxmässigt och kodmässigt är det sällan några problem men är man skolad med att få dagens uppgifter kl. 09.00 så är det svårt att anpassa sig till det nya arbetssättet. Självklart finns det seniora utvecklare som tycker det nya är skitkul också så jag ska inte dra alla över samma kam men det finns individer som tycker det är jättejobbigt med förändringar. Men det roliga med förändringsarbete är att sakta men säkert så hör man mindre och mindre gnäll och till slut erkänner nästan alla att det var en bra grej. Det tar olika lång tid för alla och det ska man ha respekt för och försöka fokusera på att stötta och hjälpa dessa individer.
19	Filip	Vilka skulle du säga är det största utmaningarna och problemområdena med DevOps och vad är viktigast för att lyckas?
20	Respondent	Viktigast för att lyckas tror jag är just det att du får med dig Ops biten. Att man tar ansvar för hela kedjan från krav till deployment och att du automatiserar bort de processer som går genom automation. Du behöver bygga in kvalité säkerhet och testning i automationen och det finns en massa principer kring hur detta ska göras som konstant uppdateras. Att tester utförs innan produktmiljö utan att en riktig person manuellt behöver kolla koden är en extrem fördel. Den största utmaningen så som jag ser det är att få till organisationsstrukturen för att stötta DevOps. Våra DevOps teams är ju indelade efter affärsområde och det har blivit lite silostruktur på det hela som vi pratat om tidigare. Det är otroligt svårt att bestämma sig för vart skärningarna mellan två affärsområden ska gå, vem har ansvar för gränsområdet osv… att hitta vad som definierar en produkt, vart den börjar och sedan tar slut. Detta blir viktigt när man sedan pratar deployment för har man inte får skärningarna rätt kan man bli påverkad av andra teams leveranser och planering.
21	Respondent	Man vill vara produktiv och effektiv som utvecklare. Man vill vara oberoende av andra team och kunna utföra sina leveranser när man är färdig, man vill inte behöva checka och synka som man behöver om man inte fått rätt på skärningarna. Den vardagen jag har idag är att jag måste deploya allt samtidigt, vi har inte fått till kontinuerliga leveransen. Allt hänger ihop i bakkant och vi har inte fått till skärningarna tillräckligt bra för att släppa på detta. därför blir gränserna så viktiga i ett stort bolag som dessutom inte har mjukvaruutveckling som main business, vi är egentligen bara en supportfunktion för resten av verksamheten. Sen går det att fundera på om vår organisationsstruktur gör oss en tjänst när det kommer till DevOps arbetet. Den är väldigt bra för företaget som helhet men om jag ska vara lite självisk och endast tänka på mig och mina utvecklingsteam så hade jag undvikit silos, det blir aldrig bra och jag hade självklart tagit tillbaka mitt Ops folk men jag förstår att vi inte är en prioritering. Ni får säga till om jag svävar iväg eller babblar för mycket

22	Måns	Inte alls! Det är superintressant så det är bara att köra på, vi har snart varit inne och touchat på alla våra områden.
23	Respondent	Ahh men vad skönt och höra! Det sista som vi måste få till som har varit väldigt svårt för oss är att hitta de här crossfunktionella forumen mellan teamen när man fått till skärningen och sina silos, pandemin har ju inte varit till hjälp på denna punkt. även om vi hade velat så har vi inte kunnat testa så många upplägg men vi testar oss fram och är ödmjuka för resultatet och förhoppningsvis kan vi bli bättre, det finns ingen silver bullet lösning som kommer lösa alla problem. Det är så viktigt att ha med sig när man pratar SCRUM och agilt att varje företag gör på sitt sätt, vi gör på vårt sätt men kollar man på till exempel Volvo så gör dem säkert på sitt sätt. Olika strukturer ger olika möjligheter osv. jag gissar ni har järnkoll på det redan men vill ändå lyfta det. haha
24	Måns	Hur ser du på framtiden och utvecklingen inom DevOps? Vad kommer bli viktigt, vad kommer ni försöka göra?
25	Respondent	Först och främst så tror jag att det agila arbetssättet är det enda arbetssätt som kommer att fungera i framtiden och då tänker inte jag bara systemutveckling och det jag tänker egentligen igenom ett helt företag. det händer så jäkla mycket, det har man ju sagt i 20 år men det gör det liksom och där som vi står om man bor i Västsverige som jag gör så är man ju ganska nära bilindustri till exempel och lastbilar och sådant. Det är jätte förändringar som den industrin står inför, att svänga om till en elproduktion som tankesätt ifrån fossila drivmedel till någonting annat. Bara en sådan grej som idag är ett måste för att överleva för de olika bilbolagen. Oavsett bransch måste man snabbt kunna ändra riktning och det måste man ha en modell för att hantera förändrade krav och därför gillar jag de olika agila ramverken. Jag tycker verkligen dem hjälper och är lätta att förstå. Man kan komma rätt långt med sitt agila arbetssätt på rätt kort tid med hjälp av dessa. Att ha en inbyggs förmåga att snabbt kunna byta och ändra riktning kommer vara en extremt stor konkurrensfördel i framtiden. Går vi inte på DevOps nivå så är det stora absolut automation, det tror jag inte är någon hemlighet och ni har säkert stött på detta svar tidigare. Jag brukar säga att en lat utvecklare är den bästa utvecklaren för han kommer hitta sätt automatisera så mycket som möjlig av det som är tråkigt och det ligger någonting i det. Att automatisera bort saker som är liksom rutin har ju enorma fördelar för verksamheten både ekonomiska och om man tänker på effektivitet. Att dessutom få in kvalitetssäkring och säkerhet i denna automatisering har enorma fördelar för utveckling. Här finns möjlighet att dessutom väva in ramverk och externa regelverk och det finns jättefina verktyg för detta idag. Får jag vara lite fräck och fråga vad andra har svarat på denna fråga?
26	Måns	Haha absolut, det är nästan 100% automation som gäller för framtiden. Alla fokuserar på detta

27	Respon-dent	Jag misstänkte nästan det haha!
28	Filip	Jag tror vi täckt allt som vi ville täcka så vi tänkte avsluta med; Har ni uppnått de effekter ni ville när ni gick till DevOps?
29	Respon-dent	Vi har absolut en lång väg kvar. Vi har blivit lite bättre på det mesta och som helhet funkar det bra men vi är långt ifrån klara och jag tror att det finns utrymme för stora förbättringar. Man kan hela tiden tweaka saker för att underlätta arbetet, som sagt tidigare hade jag gärna haft en annan struktur på organisationen om man bara tar mitt perspektiv osv. Men ett lite frustrationsmoment är att man aldrig blir klar, det är en konstant förändringsprocess och som utvecklare i botten är man van vid att leveransen är slutet på ett projekt med DevOps-arbetet blir man aldrig klar. Det infinner sig aldrig en känsla av att gå i mål utan det finns hela tiden saker att förfina. Men för att svara på din fråga så har vi inte ens kommit till DevOps version 0 än. Vi har långt kvar, framför allt med Ops-delen och kommunikation, det är dem två stora för oss. Sen har vi även den externa faktorn att det blir svårt att bygga team när det är näst intill omöjligt att få tag på seniora utvecklare. Vi söker med lykta men det är verkligen en flaskhals som håller tillbaka oss. För även om jag gillar att anställa nyexade just för de har rätt mindset så kan jag inte sätta ihop ett team med endast nyexade och förvänta mig att dem fixar att ta ansvar från ax till limpa, det är mixen av juniora och seniora jag vill ha. Så DevOps är fantastiskt men vi har en lång väg att gå!
30	Måns	Då tackar vi så mycket för att du ställde upp på intervjun.
31	Filip	Ja, tusen tack för din medverkan!
32	Respon-dent	Det är jag som tackar grabbar och jag ser fram emot att läsa er uppsats!

Appendix D

Tabell 8: Transkribering av intervju 4 med respondent 4 - förkortning R4

Rad	Personer	Information
1	Måns	Alright. So, what we want to discuss today is basically how your company transitioned into DevOps and how you guys worked before and how you see the future regarding DevOps and working with these types of things. So, if you could potentially start by explaining what is your job title and what your job description is, basically.
2	Respondent	So, my job title is a DevOps manager, and I don't really like DevOps as a job title because I think of DevOps more as an idea of the way that you would like to organize responsibility and work rather than as a title. You know, like, the actual work that I do is infrastructure engineering. So that's IT- infrastructure, like servers and AWS stuffs to infrastructure engineering, coding and automation. And looking after developers that are, you know, doing infrastructure coding. So, my work intersects with development and operations. So that's why we use DevOps as a term but I'm not a fan of trying to hire people as DevOps engineers, because it really doesn't say much about what you do and what specific qualifications that you have.
3	Respondent	So, I guess you know, the most important thing is to define what you mean by DevOps. Like, for me, it's, it's just about the intersection between development and automation and operations work. But my background was as a system administrator, so like the old school, Linux sysadmin, where you would have system administrators working on the servers, and then you would have developers completely separate. And, you know, the developers would just manage their own applications. Or they would give them to the operators to run and then the operators would just look after that and make sure that things were running and that's it. If you want to automate things that are on a bigger scale than the people doing operations, you know, you need to have development skill sets as well. So, I think the way that I see it is that DevOps is about operations still looking after the servers and providing an environment but working alongside the developers in order to provide better tooling, like, I guess one thing in terms of a definition for DevOps that I see so often is it's about tools, DevOps tools. So, I guess Yeah, I don't really like that. But yeah, providing tools for developers to be able to do stuff themselves
4	Måns	That's interesting. How would you say that the team structure is aligned at your company? Do you have a developer's team and one operations team? Or do you have a team where both Dev and Ops personnel work

		together in a single team? Or how does the team structure look like for you guys?
5	Respon-dent	Yeah, so the way we have it been kind of very general, which is a way that I quite like because we have various dev teams that are responsible for their own products and their own kind of specialties that they work on. Then the DevOps team is responsible for stuff, like infrastructure operations, and looking after CI and stuff, we look after on one hand operations like shared infrastructure. So that's a thing, like servers that can have multiple services running on them. multiple products, things like that. So shared infrastructure, we look after that. We look after things that are specific tools for the developers to be able to go through a continuous integration pipeline and integrate their code in with what's already running. And occasionally, we directly help developers with problems that they face, especially if it relates to how they're going to make things work when it gets to the production environment. So, like when I bring people into the team, and I'm interviewing people, I quite often ask a question along the lines of if you were going to sit down with a developer and the developer has a brand new service that they want to onboard, but they haven't, they're just starting to develop it, what's all the information that you would want to get about that new service? So that you could start designing infrastructure for it so then it can have a nice, automated pipeline for deployment and build the whole service lifecycle for that service?
6	Måns	Would you say that your developers take more responsibility for the product and system that they developed in terms of how they develop it? And then do they follow along the product chain into testing environments and stuff like that?
7	Respon-dent	I think it's in two parts. It's about moving the responsibility of infrastructure towards the development of the developers and also moving the responsibility of the developers of looking after their own stuff. All the way through to afterwards running in production on that provided infrastructure so the end result of a service that's running and has customers using it. It is like a shared responsibility there's some stuff that developers are never going to touch, like network staff and routers and you know, physical infrastructure in a data center, but then you want to provide an abstracted layer where developers can then come and run their own their own things, and then manage them. So, there's various abstractions that are used, I guess the biggest one that people use are Kubernetes, but there are plenty of alternatives. So, it's not just you got to use one specific thing that makes you use DevOps or something like that.
8	Måns	Cool, alright we are also investigating and discussing communication in our study mainly between Dev and Ops. And as you mentioned before you guys use some infrastructure. Are you able to see each other's progress and track development through this program/infrastructure? From both sides or how does it work?

9	Respon-dent	Yeah. So, in terms of philosophy around it, for me as a distributed company, we have offices around the world, so we try to write everything in writing as much as possible. In a similar way, I guess, a company that's very, very open about how they do this and talk about it is GitLab. They are also a very distributed company and to everything buyer via text, more or less. So, yeah, you need to have a kind of a source of truth where you can track the status of different work. So, you need to have, obviously, source code repository, whichever software you wish to use for that kind of VCs and issue tracking system. There are plenty of them, right? For example, JIRA Atlassian is a pretty, big one. And documentation system where you can send documents and there's dozens available as well. And if you've got those and your kind of, make it a core principle that you document and hand over things in writing, then you can fall into that easily.
10	Måns	That's cool. And are any communication tools used? How do meetings look between developers and developers, but also developers and operations? Are you mostly doing video calls? Like this or are you meeting face to face, or do you have any tools to provide, if you have an idea about the code or want to discuss something? How does that work?
11	Respon-dent	So usually, usually communication, we try to do it asynchronously. Or just a lot of it happens asynchronously because people are in different locations. If you want other people to be able to see the history of your discussion later and then you need to discuss it in a public channel like public within the organization, not necessarily public to the world. But like in a chat channel in whichever chat program you use, or you need to record your conversation in that issue tracking software that you use. So, where people can attack that and get a notification that this issue they're working on and has your feedback in it, they can go and address it. If you're just working on things that are kind of just figuring things out, then you can just have either direct messaging in a chat program, or to get on a call with somebody if you really must have real time communication. So real time communication is the fastest for getting certain things done. Like if you want to troubleshoot something, pretty much all you need to do is real time communication, like a phone call, or a video call. But for working on a on a project, for example, like early stage fleshing out, new system it doesn't go as well to have a video called like, you need to sit there and think about it and write your ideas out and then like, give the next person provide their feedback as well. So yeah, I don't really like to advocate any specific tools. Because it doesn't really matter at the end of the day. What matters is the function that particular tool fulfills for you.
12	Måns	All right makes sense, what would you say works well, with the DevOps style that you exercise and what would you say works less well?
13	Respon-dent	What do you mean could you expand on that a little bit?

14	Måns	Yeah sure, regarding teamwork and collaboration on the way you work with DevOps in your company and DevOps in general, what would you say are the biggest benefits and what would you say are the biggest downgrades/flaws compared to working with a different type of work method like the waterfall method?
15	Respon-dent	Yeah. So, I think that the biggest advantage of this kind of model is that you can release updates to a product, like new iterations, a lot faster with higher reliability than if you're doing it in a very segregated way. So, I guess what that means is how quickly you can change your product is very valuable. Competitiveness of your product over time, for being able to react to changes in the market changes in the direction you want things to go in. And if you have a kind of a pre-DevOps era methodology where you have to have different teams' kind of in lockstep, moving things along through a process through deployment and very slow feedback loops. It can be very difficult to make change happen quickly and also reliably. Like if there are problems that are discovered later on. It takes a very long time to fix things. So, I guess it comes down to agility being the main advantage. And the disadvantage is that it's much more difficult to hire people because you need to have a much wider range of experience in order to do this type of work. Welcome. You can't just be a Linux system administrator and just learn Linux command line stuff and file systems or processes. And just be in your box and then a developer just, you know, doing Java or something like that and just be there and the two don't ever have to meet. Like you need to have a wider dev seem to have an understanding of the infrastructure, at least the abstract running on and the people looking after the infrastructure and operations need to have also an understanding of the requirements of the code and like, have a handle on that side as well. In order for things to work well. And that's it, yeah, with a wide range of skill sets. It's like the basic levels for DevOps and infrastructure hiring are pretty much like you're looking at people who 15 years ago, I like senior engineers. And in order to kind of be starting off, it's really hard to find people. The only way that seems to work is to get people like whatever you can the best you can and just to spend some time and train them like it would take a couple of years. So that's the biggest challenge that I see.
16	Måns	Yeah, we heard that from other people that we also interviewed for this study, they were mentioning that it's really hard to recruit new people. Just because of this lack of experience within this field. You were mentioning before testing and quality assurance. Could you elaborate on how you work with those functions like continuous integration, continuous delivery and also how you quality test your products?
17	Respon-dent	In general, there are automated tests and there are manual tests. So, I think one of the most important things that's happened in the last like 10 years is this rise in tooling around. So automated pipelines for moving code specifically, from the Devs through to the production environment and taking it through automated testing. Automated Testing pipelines where you do unit tests, end to end test integration tests, test database,

		migrations, everything like that. Before you even consider manually testing things. You can go to some people who want to go all in and completely automate all the way to deployment. And that's fine depending on the particular risk profile of the product and the business that's doing it. And different companies may wish for it. I think, to have more hands on checking before something goes live. But you know, it's like on a product-by-product basis exactly how much automation you are comfortable with, depending on the particular risks of each software, right like their information security risks, and also other business risks of downtime and stuff like that. Yeah, I guess that's kind of really taken off in the last 10 years. Just kind of automation around pipelines. Yeah, was there a second part of the question?
18	Måns	No, I think you answered it completely right, so I think I got the answer. Are you familiar with the term DevSecOps, which includes security?
19	Respondent	Yes, I am, so I really like the idea of, like, infrastructure as code, right. So being able to apply the same version control that you use for like that, that's been around for a long time for code for the code itself, like application code, and applying that to infrastructure. And it's not very well developed yet, but I really liked the idea of applying those same principles. So, version control and like, treating things as code and generating processes. And, you know, testing through pipelines. Like that, but other business processes. So, it seems to me like dev SEC Ops is the business process where you would try and apply those same principles into security as well. I really like it, but I haven't found one yet. A meaningful way to do it. Like you can definitely say that you're working in dev SEC ops. But I don't know what that means. Is that the real world like in terms of concretely what you actually do, like are you know, managing security appliances as code because that's really just infrastructure. As code like, to me, it seems like in order to be doing security in that way, you have to somehow formally define some of your business partners to us, and I haven't seen language or methodology for formalizing business processes that way yet. I'd love to though.
20	Måns	It’s a cool field for sure. Another thing that we are also investigating is the organizational structure that you already mentioned a little bit, but also the company culture. How would you say that has changed when you started working with DevOps in general?
21	Respondent	Um, it's definitely changed over my career in terms of cultures and organizational structures. But it's hard to say what's better or worse, because different companies have, you know, different requirements and different cultures. And you know, apart from business culture, like in a multinational organization, between countries, there are cultures that are different, that reflect the way businesses are structured. The way business hierarchies are how power is shared or delegated within organizations. So, I don't really like to put any kind of value judgment on what's better or worse, just as long as it's not abusing people. And it's working for that organization, like different structures. You know, at the end of

		the day, the ones that are working will succeed and the ones that don't work will fail. I'm not very prescriptive. about it as to what people should do.
22	Måns	How do you see the future with DevOps if we are moving forward? What's the next step in this?
23	Respon-dent	Because there is still actual infrastructure. There's going to be some kind of need for operations with kind of, increase of, you know, platforms, as a service, like AWS to Azure and all that. I could see that for many or-ganizations, the need to have dedicated operations staff is going to de-crease so that you can kind of run with a bare minimum. But then, like, even if you were a completely AWS company today, you would still probably need to have one person looking after shared infrastructure. At least for cost savings, because you can just have the Devs go and create their own stuff. If you have shared infrastructure, you're going to save a lot of money. I don't see a future without operation stuff. But I do think that with continued abstraction of the infrastructure like that there may be less involvement between operations and DevOps and DevOps may just end up being about maintaining like a really, really well-defined set of like shared infrastructure, like abstractions like where you, you may just be managing Kubernetes and some successor of Kubernetes and then your Devs you know, you just have a culture where they are com-pletely comfortable with just using that. No, I don't know. I mean, there's lots of different ways it could go.
24	Måns	Cool, interesting point of view, though. I think that's basically all the questions I want to ask today. Is there anything else you'd like to add or discuss more?
25	Respon-dent	No, that's pretty much it. Yeah, I'm not sure how mainstream my views are. So, I don't know if my perspective is very different from other peo-ple that you talked to about this kind of stuff.
26	Måns	It's a little different, but yeah, you kind of have the same basic views and stuff.
27	Respon-dent	Yeah, yeah. I, I guess the most interesting thing for me is what you touched on with DevSecOps like I that I really hope in the future to see formalization in the same way as infrastructure as code has formalized infrastructure in other business processes and documents and things like that, this that they can be manipulated and passed using tooling. Instead of by the eye like I can define. I can define a complete infrastructure stack from you know, almost the hardware all the way to the software that's running on it. Then, you know, my next task of the day is going to be to review a business policy about something where I'm just sitting there reading a word document. I could come to completely separate worlds. And as a technically minded person, like I would, I'd like to bring them together.

28	Måns	That will be a cool future. Once again thank you for participating in this interview.
29	Respon-dent	Thank you, it was fun.

Referenser

Bobrov, E., Bucchiarone, A., Capozucca, A., Guelfi, N., Mazzara, M., & Masyagin, S. (2019). Teaching DevOps in academia and industry: reflections and vision. In *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment* (pp. 1-14). Springer, Cham. Available Online: https://link.springer.com/chapter/10.1007/978-3-030-39306-9_1 [Accessed 5 April 2022].

Carlsson, L., & Langsager, J. (2021). Implementering av DevOps: En fallstudie på ett IT-konsultföretag. Available Online: https://www.diva-portal.org/smash/get/diva2:1561007/FULLTEXT01.pdf [Accessed 12 April 2022].

Chen, L. (2015). Continuous delivery: Huge benefits, but challenges too. *IEEE software*, *32*(2), 50-54.Available Online: https://ieeexplore.ieee.org/abstract/document/7006384 [Accessed 13 April 2022].

Chen, H. M., Kazman, R., Haziyev, S., Kropov, V., & Chtchourov, D. (2015). Architectural support for DevOps in a neo-metropolis BDaaS platform. In *2015 IEEE 34th symposium on reliable distributed systems workshop (SRDSW)* (pp. 25-30). IEEE. Available Online: https://ieeexplore.ieee.org/abstract/document/7371437 [Accessed 13 April 2022].

Christensen, H.B. (2016). Teaching DevOps and cloud computing using a cognitive apprenticeship and story-telling approach. In *Proceedings of the 2016 ACM conference on innovation and technology in computer science education* (pp. 174-179). Available Online: https://dl.acm.org/doi/abs/10.1145/2899415.2899426 [Accessed 5 April 2022].

Crespi, V., Galstyan, A. and Lerman, K. (2008). Top-down vs bottom-up methodologies in multi-agent system design. *Autonomous Robots*, *24*(3), pp.303-313. Available Online: https://link.springer.com/article/10.1007/s10514-007-9080-5 [Accessed 10 April 2022].

Debois, P. (2008). Agile infrastructure and operations: how infra-gile are you?. In *Agile 2008 Conference* (pp. 202-207). IEEE. Available Online: http://www.jedi.be/presentations/agile-infrastructure-agile-2008.pdf [Accessed 22 Mars 2022].

Diel, E., Marczak, S. and Cruzes, D.S. (2016). Communication challenges and strategies in distributed DevOps. In *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)* (pp. 24-28). IEEE. Available Online: https://ieeexplore.ieee.org/abstract/document/7577415 [Accessed 4 April 2022].

Dörnenburg, E. (2018). The path to devops. *IEEE Software*, *35*(5), pp.71-75. Available Online: https://ieeexplore.ieee.org/abstract/document/8409919 [Accessed 30 Mars 2022].

Erich, F.M., Amrit, C. and Daneva, M. (2017). A qualitative study of DevOps usage in practice. *Journal of Software: Evolution and Process*, *29*(6), p.e1885. Available Online: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1885 [Accessed 10 April 2022].

Gall, M. and Pigni, F. (2022). Taking DevOps mainstream: a critical review and conceptual framework. *European Journal of Information Systems*, pp.1-20. Available Online: https://www.tandfonline.com/doi/abs/10.1080/0960085X.2021.1997100 [Accessed 10 April 2022].

Gibson, C.B. and Zellmer-Bruhn, M.E. (2001). Metaphors and meaning: An intercultural analysis of the concept of teamwork. *Administrative science quarterly*, *46*(2), pp.274-303. Available Online: https://journals.sagepub.com/doi/abs/10.2307/2667088 [Accessed 15 April 2022].

Gupta, R.K., Venkatachalapathy, M. and Jeberla, F.K. (2019). Challenges in adopting continuous delivery and devops in a globally distributed product team: a case study of a healthcare organization. In *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)* (pp. 30-34). IEEE. Available Online: https://ieeexplore.ieee.org/abstract/document/8807506 [Accessed 15 April 2022].

Hashmi, A., Ishak, S. and Hassan, H.B. (2018). Role of team size as a contextual variable for the relationship of transformational leadership and teamwork quality. *Asian Journal of Multidisciplinary Studies*, *6*(5), pp.76-81. Available Online: https://www.researchgate.net/profile/Shahibudin-Ishak/publication/326146424_Role_of_team_size_as_a_contextual_variable_for_the_relationship_of_transformational_leadership_and_teamwork_quality/links/5b3b12dfaca2720785052faf/Role-of-team-size-as-a-contextual-variable-for-the-relationship-of-transformational-leadership-and-teamwork-quality.pdf [Accessed 12 April 2022].

Hemon, A., Lyonnet, B., Rowe, F. and Fitzgerald, B. (2020). From agile to DevOps: Smart skills and collaborations. *Information Systems Frontiers*, *22*(4), pp.927-945. Available Online: https://link.springer.com/article/10.1007/s10796-019-09905-1 [Accessed 12 April 2022].

Hermawan, A. and Manik, L.P. (2021). The Effect of DevOps Implementation on Teamwork Quality in Software Development. *Journal of Information Systems Engineering and Business Intelligence*, *7*(1), pp.84-90. Available Online: https://www.e-journal.unair.ac.id/JISEBI/article/view/25904
[Accessed 12 April 2022].

Humble, J. and Farley, D. (2010). Continuous delivery: reliable software releases through build, test, and deployment automation. *Pearson Education*, pp.24-34 Available Online: https://ptgmedia.pearsoncmg.com/images/9780321601919/samplepages/0321601912.pdf
[Accessed 22 Mars 2022].

Hoegl, M. and Gemuenden, H.G., (2001). Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence. *Organization science*, *12*(4), pp.435-449. Available Online: https://www.sciencedirect.com/science/article/abs/pii/S0007681304001120 [Accessed 12 April 2022].

Jacobsen, D. I. (2002). Vad, Hur Och Varför? Om Metodval i Företagsekonomi Och Andra Samhällsvetenskapliga Ämnen, edited by G. Sundin, Lund: Studentlitteratur AB.

Jacobsen, D. I. (2017). Hur genomför man undersökningar: introduktion till samhällsvetenskapliga metoder, edited by S. Andersson, Lund: Studentlitteratur AB.

Khan, M. S., Khan, A. W., Khan, F., Khan, M. A., & Whangbo, T. K. (2022). Critical Challenges to Adopt DevOps Culture in Software Organizations: A Systematic Review. *IEEE Access*, *10*, 14339-14349. Available Online: https://ieeexplore.ieee.org/document/9690862 [Accessed 10 April 2022].

Leite, L., Pinto, G., Kon, F. and Meirelles, P. (2021). The organization of software teams in the quest for continuous delivery: A grounded theory approach. *Information and Software Technology*, *139*, p.106672. Available Online: https://www.sciencedirect.com/science/article/abs/pii/S0950584921001324 [Accessed 14 April 2022].

Leite, L., Rocha, C., Kon, F., Milojicic, D. and Meirelles, P. (2019). A survey of DevOps concepts and challenges. *ACM Computing Surveys (CSUR)*, *52*(6), pp.1-35. Available Online: https://dl.acm.org/doi/abs/10.1145/3359981 [Accessed 14 April 2022].

Lindsjørn, Y., Sjøberg, D.I., Dingsøyr, T., Bergersen, G.R. and Dybå, T. (2016). Teamwork quality and project success in software development: A survey of agile development teams. *Journal of Systems and Software*, *122*, pp.274-286. Available Online:

https://www.sciencedirect.com/science/article/pii/S016412121630187X [Accessed 7 April 2022].

Mohammad, S. M. (2016). An exploratory study of DevOps and it's future in the United States. *International Journal of Creative Research Thoughts (IJCRT), ISSN*, 2320-2882. Available Online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3668439 [Accessed 18 April 2022].

Muñoz, M. and Rodríguez, M.N. (2021). A guidance to implement or reinforce a DevOps approach in organizations: A case study. *Journal of Software: Evolution and Process*, p.e2342. Available Online: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2342 [Accessed 5 April 2022].

Nadler, D.A. (1983). A general diagnostic model for organizational behavior: Applying a congruence perspective. *Perspectives on behavior in organizations*, pp.112-124. [Accessed 12 April 2022].

Senapathi, M., Buchan, J. and 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* (pp. 57-67). Available Online: https://dl.acm.org/doi/abs/10.1145/3210459.3210465 [Accessed 15 April 2022].

Nybom, K., Smeds, J. and Porres, I. (2016). On the impact of mixing responsibilities between devs and ops. In *International Conference on Agile Software Development* (pp. 131-143). Springer, Cham. Available Online: https://link.springer.com/chapter/10.1007/978-3-319-33515-5_11 [Accessed 12 April 2022].

Oates, B. J., Griffiths, & McLean, R. (2022). Researching Information Systems and Computing, Second edition., Thousand Oaks: SAGE Publications Ltd.

Oh, J., Lee, H., & Zo, H. (2019). The effect of leadership and teamwork on ISD project success. *Journal of Computer Information Systems*. Available Online: https://www.tandfonline.com/doi/10.1080/08874417.2019.1566804 [Accessed 12 April 2022].

Roche, J. (2013). Adopting DevOps practices in quality assurance. *Communications of the ACM*, *56*(11), pp.38-43. Available Online: https://dl.acm.org/doi/abs/10.1145/2524713.2524721 [Accessed 13 April 2022].

Salas, E., Cooke, N. J., & Rosen, M. A. (2008). On teams, teamwork, and team performance: Discoveries and developments. *Human factors*, *50*(3), 540-547. Available Online: https://journals.sagepub.com/doi/abs/10.1518/001872008X288457 [Accessed 10 April 2022].

Sebastian, I.M., Ross, J.W., Beath, C., Mocker, M., Moloney, K.G. and Fonstad, N.O. (2017). How Big Old Companies Navigate Digital Transformation. *MIS Quarterly Executive*, *16*(3). Available Online: https://publikationen.reutlingen-university.de/frontdoor/deliver/index/docId/1501/file/1501.pdf [Accessed 7 April 2022].

Trinczek, R. (2009). How to interview managers? Methodical and methodological aspects of expert interviews as a qualitative method in empirical social research. In *Interviewing experts* (pp. 203-216). Palgrave Macmillan, London. Available Online: https://link.springer.com/chapter/10.1057/9780230244276_10 [Accessed 19 April 2022].

Waller, J., Ehmke, N.C. and Hasselbring, W. (2015). Including performance benchmarks into continuous integration to enable DevOps. *ACM SIGSOFT Software Engineering Notes*, *40*(2), pp.1-4. Available Online: https://dl.acm.org/doi/abs/10.1145/2735399.2735416 [Accessed 11 April 2022].

Wiedemann, A., Wiesche, M., Gewald, H. and Krcmar, H. (2020). Understanding how DevOps aligns development and operations: a tripartite model of intra-IT alignment. *European Journal of Information Systems*, *29*(5), pp.458-473. Available Online: https://www.tandfonline.com/doi/full/10.1080/0960085X.2020.1782277 [Accessed 10 April 2022].

Yarlagadda, R.T. (2018). Understanding DevOps & bridging the gap from continuous integration to continuous delivery. *Understanding DevOps & Bridging the Gap from Continuous Integration to Continuous Delivery', International Journal of Emerging Technologies and Innovative Research (www. jetir. org), ISSN*, pp.2349-5162. Available Online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3807611 [Accessed 10 April 2022].

Yeung, H.W.C. (1995). Qualitative personal interviews in international business research: some lessons from a study of Hong Kong transnational corporations. *International Business Review*, *4*(3), pp.313-339. Available Online: https://www.sciencedirect.com/science/article/abs/pii/096959319500012O [Accessed 19 April 2022].