LUNDS UNIVERSITET
Ekonomihögskolan
Institutionen för informatik

DevOps - Boon or Bust

Problematik och Framgång i CAMS

Kandidatuppsats 15 hp, kurs SYSK16 i Informatik

Författare: Hampus Sandell
Johannes Ångman

Handledare: Magnus Wärja

Rättande lärare: Markus Lahtinen
Ahmad Ghazawneh

DevOps - Boon or Bust: Problematik och Framgång i CAMS

ENGELSK TITEL: DevOps - Boon or Bust: The problems and successes in CAMS

FÖRFATTARE: Hampus Sandell och Johannes Ångman

UTGIVARE: Institutionen för informatik, Ekonomihögskolan, Lund Universitet

EXAMINATOR: Odd Steen, Docent, Fil Dr

FRAMLAGD: Maj, 2019

DOKUMENTTYP: Kandidatuppsats

ANTAL SIDOR: 68

NYCKELORD: DevOps, Kultur, Automation, Mätning, Delning

SAMMANFATTNING:
DevOps är ett relativt nytt förhållningssätt inom världen för informationssystem som saknar en tydlig definition, men ändå jobbar många företag med det. Vi har därför undersökt hur DevOps påverkat tidigare existerande integrationsproblematik inom IT/IS-branschen. Vi har intervjuat ett antal företag som implementerat DevOps och ställt frågor utformade efter CAMS för att avgöra om de implementerat DevOps och upplevt mindre problematik till följd för att avgöra om det varit en framgång. Avsaknaden av tydlig definition av begreppet DevOps och vad som faktiskt ska göras i DevOps, öppnar upp för frågan om DevOps bara är en trend och ifall det faktiskt är ett koncept som är här för att stanna. Vi upptäckte att företagen har olika uppfattning om vad som är DevOps, och att det finns flera faktorer som har påverkat deras definitioner och implementationer, där avsaknaden av tydlighet kan ha varit en styrka.

Förkortningar:
IT - Informationsteknologi
IS - Informationssystem
ERP - Enterprise Resource Planning
BPM - Business Process Management

Ordförklaring:
Repository - En delad mapp för versionshantering av kod mellan utvecklare.
Technical Debt - En framtida tidsåtgång som krävs för att åtgärda dålig kod och implementation.
DevOps - Teleskopord av Development (Dev) och Operations (Ops)
CAMS - Akronym av Culture, Automation, Measurements, och Sharing.
Hype Train - Ett fiktivt tåg fyllt av fantaster. Skanderar ofta “This is the next big thing!”
Legacy system - Äldre hårdvaru- och mjukvarusystem.
Buzzword - Trend; Slagord; Modeord.
Downtime - Dödtid, systemet är nere.

Innehåll

Figurer

1. Inledning

IT och teknologiska framsteg tvingar idag fram förändringar inom organisationer vare sig de vill eller inte (Broberg, 2013). Enligt Burnes (2014) går det inte att undkomma förändring, det är en evigt närvarande aspekt av företagsamhet vilket genomsyrar samtliga nivåer, allt från det strategiska till den vardagliga verksamheten. Hur de sedan hanterar förändringarna som krävs för att nå dit blir även de en viktig del att förhålla sig till (By, 2005).

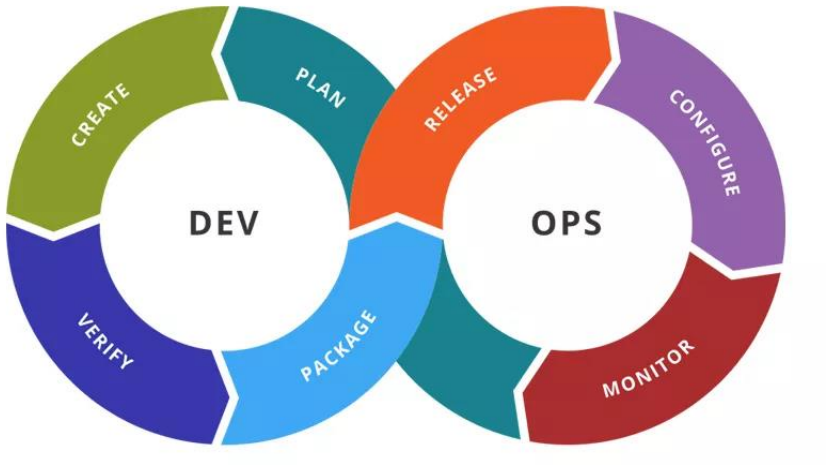
En av dessa förändringar är DevOps, och det är en ny trend inom IT-sektorn som ämnar korta ner leveranstider i informationssystem genom bättre kooperation mellan arbetslag för IT-drift och utveckling (Trubiani et.al, 2019). Termen "DevOps" är ett vedertaget begrepp, men få kan säga specifikt vad det är då ingen unik förklaring av begreppet etablerats och accepterats (Hütterman, 2012; Fitzgerald, 2015). Frånvaron av en vedertagen och accepterad förklaring av vad DevOps verkligen utgör kan leda till svårigheter vid implementation av DevOps och leda till frågor om det som faktiskt har implementerats är DevOps.

Så vad är DevOps? Det ett förhållningssätt som främjar ett nära och öppet samarbete mellan utvecklare och drifttekniker, som ämnar korta ner leveranstider och höja kvalitén på mjukvara genom att i löpande takt släppa uppdateringar till arbetsmiljön.

1.1 Bakgrund

Historien kring DevOps framtagande börjar med att företag och organisationer runt 2005 började uppleva en ökande mängd technical debt, vilket förlängde utveckling- och integrationscykler (Werner, 2015). År 2009 så myntas begreppet "DevOps" av Patrick Debois som ett svar på det växande missnöjet inom branschen, och som en reaktion mot dåvarande praxis (Mueller, 2019; Mezak, 2018).

Traditionellt sett har utvecklare och IT-driftpersonal haft separata och ofta konkurrerande mål, ledning och kriterier de blir utvärderade efter (Atlassian, 2019). Vilket tveklöst lett till friktion avdelningarna emellan och i sin tur att resurser gått till spillo. Hütterman (2012) sammanfatta det som att utvecklare oftast vill släppa sina förändringar så snabbt som möjligt medans IT-driften förespråkar stabilitet och därmed långsammare uppdateringar. För att adressera denna problematik mellan arbetsgrupperna och förbättra arbetsprocessen skapades DevOps.



Figur 1.1: DevOps process (Techtrainees, 2018)

Framgången av DevOps rörelsen är dess integration mellan utveckling och drift, där målsättning och incitament för arbetsgrupperna likställs, där det tidigare existerat en fellinjering av mål och incitament för funktionerna. En närmare integration mellan arbetslagen ska tjäna till att leverera fungerande mjukvara och hårdvara så snabbt som möjligt, då båda grupper vinner på att arbeta mot gemensamma mål.

Hütterman och Cuppett menar att DevOps handlar först och främst om personer, inte tekniker, vilket gör en implementation av DevOps diffust (Hütterman, 2012; Cuppett, 2016). Det är en aning motsägelsefullt då en stor del av DevOps bygger på automation av uppgifter, bland annat testning av mjukvara och mätning av system, men det är inte DevOps om inget samarbete mellan människorna adresseras först. Damon Edwards och John Willis anser att DevOps bygger på fyra grundprinciper som tillsammans bildar akronymet CAMS för Culture, Automation, Measurements, Sharing, begrepp som myntats 2010 (Willis, 2010). Sen Edwards och Willis etablerade CAMS som grundprinciper har de itererats i flera artiklar, böcker, och bloggar, som de facto principerna och kärnan i DevOps (Hütterman, 2012; Cuppett, 2016; Wiedemann, 2017; Kartman, 2019; Agarwal et.al 2018).

I CAMS definieras kulturen i DevOps som att där bör finnas en hög grad av acceptans för gemensamt ansvar i arbetsgrupper för att kunna leverera högkvalitativ mjukvara, arbetsgrupperna skall vara öppna och transparenta för att kunna lära sig från varandras kompetenser och erfarenheter, både misslyckanden och framgångar (Wiedemann, 2017; Hütterman, 2012).

Automatisering definieras enligt CAMS i DevOps som att arbetsmoment såsom testning, konfigurering av miljöer och leveranser bör vara automatiserade till en väldigt hög grad för att bidra med snabb återkoppling till utvecklare och arbetsledare, minska ledtider, sänka felfrekvens och höja kvalitén för användare (Wiedemann, 2017; Kartman, 2019). En hög grad av automatisering kräver dock kompetens och är kortsiktigt väldigt tidskrävande, men långsiktigt effektivt då framtida tidsbesparingar uppstår (Hütterman, 2012).

Mätning enligt CAMS handlar om att faktiskt se resultatet från utförda förändringar, och att inte bara gå på den upplevda känslan. Effektiva mätetal som underlättar inte enbart möjligheten att se

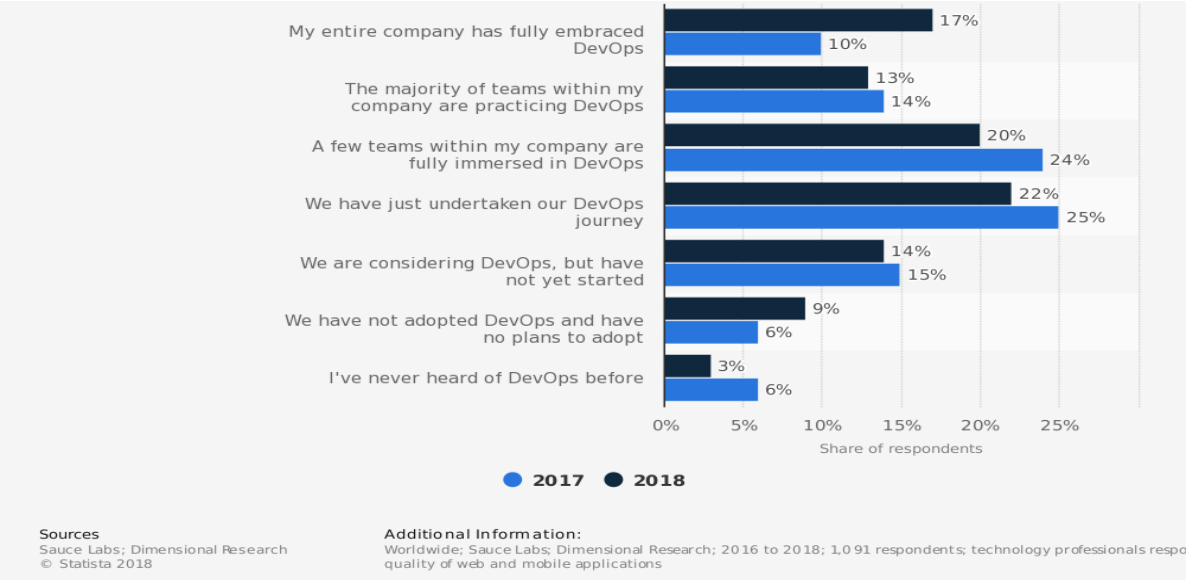
förändringarnas effekter, utan hjälper även till att målsätta och motivera arbetsgrupper. Genom att mäta prestanda på system och arbetslag så blir förändringar tydliga, men mätetal bör komplettera varandra, så att en förändring mäts i flera dimensioner (Kartman, 2019).

Delning handlar om att sammanföra utveckling och drift, där individer och grupper delar med sig av erfarenhet och kompetens mellan arbetsgrupper, men även verktyg, framgång och misslyckanden. Delning knyter ann till kulturen och principerna bygger på varandra, där en DevOps arbetsgrupp förmodligen inte har endast kulturen, eller delning.

DevOps är fortfarande ett ungt förhållningssätt, som har ett hippt namn och låter allt mer som ett modeord för varje dag som går. Detta faktum kombinerat med lovande förbättringar gör DevOps till ett väldigt intressant förhållningssätt för ivriga företag, men detta gör också att det finns ett kraftigt behov att utforska dess potentiella fallgropar, om där ens finns några, och huruvida grundpelarna CAMS som etablerades 2010 är lika relevanta idag som de var då.

1.2 Problemområde

Med ett löfte om effektivare IT så implementerar allt fler företag DevOps i sina organisationer, men DevOps är ett relativt outforskat område hittills inom IS i förhållande till ämnen som ERP och BPM, områden som vid sökningar i Google Scholar genererar avsevärt fler träffar än DevOps. Än färre träffar är tillgängliga vid sökningar gällande DevOps framgång än problematik i DevOps. Detta antyder att problematik kring DevOps implementation är aningen mer utforskat område än framgången, tio år efter dess begynnelse och med tillväxten av molnbaserade tjänster har utvecklare och IT-drift varit med om ett stort skift.



Figur 1.2: Level of DevOps adoption in 2018 (Statista, 2018)

Implementationen av DevOps inom organisationer växte explosionsartat som illustreras i figur 1.2, med 7 % mellan 2017 och 2018, och tyder på att det är en aktiv trend med färre företag som står i startgroparna. Med tanke på att giganter som Facebook, Google, Netflix och Amazon implementerat DevOps principer är det förvånansvärt lite litteratur som existerar och detaljerar

problematiken och framgången (Wiedemann et.al, 2018; Babb et.al, 2017). Det indikerar för oss att litteraturen är kritisk mot DevOps och att de förekommande problemen i DevOps står i vägen.

DevOps handlar om kultur, automation, mätning och delning i sin kärna, och det manifesteras i form av CAMS, men vi finner än mindre litteratur som undersöker problematik utefter dessa fyra etablerade pelare. CAMS ämnar vi därmed använda för att dela upp företagens upplevda problematik i de fyra områden som DevOps hävdas bestå av. Vi ser även att mjukvaruutveckling är en föränderlig värld med snabb introduktion av nya tekniker och förhållningssätt som skiftar landskapet, och gör att ny problematik kontinuerligt uppstår till följd, men också att lösningar likt Scrum, Extreme Programming och DevOps dyker upp i rask takt. Vi vill därmed teoretisera kring huruvida problematik som DevOps ämnat lösa faktiskt har påverkats, och gör DevOps till en framgång eller om det är tomma löften från en trend utan substans och bara ytterligare ett så kallat “Hype train”.

1.3 Frågeställning

Givet ovanstående blir vår frågeställning:

På vilket sätt löser DevOps rådande integrationsproblematik inom IS/IT branschen utifrån ett CAMS perspektiv?

1.4 Syfte

Uppsatsen ämnar bidra med teorier gällande DevOps bidrag till att lösa problematik i organisationer.

2 Litteraturgenomgång

I detta kapitel presenterar vi uppsatsens undersökningsmodell samt den litteratur och forskning som ligger till grund för modellen. Vi redogör agila metoder och hur begreppet DevOps passar in samt den upplevda problematik som dyker upp i litteraturen kring DevOps.

2.1 DevOps härkomst

Inom mjukvaruutveckling finner vi Software development life cycle (SDLC), processen för att bygga och underhålla system. Den kartlägger viktiga områden för utvecklarna som planering, analys, design och implementatio. SDLC har gått igenom flera generationer av utvecklare och de olika metoderna som växt fram är många. Inom SDLC kan vi idag finna bland annat vattenfallsmodellen och V-model som kan anses tillhöra den första och mer traditionella generationen. För att sedan göra rum för de mer moderna trenderna i form av agila metoder som lean software development, SCRUM och XP i den andra generationen (Beng et.al 2012). DevOps kan i sin tur ses som en typ av fortsättning eller förlängning av det agila arbetet (Virmani, 2015). DevOps med sitt fokus på kommunikation och samarbete mellan Dev och Ops istället för verktygen och processerna, kan på så vis uppnå agila mål gällande arbetslagets responstid och förlänga agila principer i hela mjukvaruutveckling processen (Farroha & Farroha 2014; Virmani, 2015). Där systemets helhetsbild får ett större fokus genom att brygga kommunikationsklyftan mellan specialiserade silorna och göra slut på ordkastningen funktioner emellan. Hosono (2012) uttrycker även hur de agila arbetssätten kan anses vara möjliggörare i adoptionen av DevOps. Så med detta sagt kan vi likt Eficode (2019) anta att DevOps är som de uttrycker; tredje generationens mjukvaruutvecklings metod, en fortsättning av andra generationens agila metoder där utveckling och drift tar ett steg tillbaka för att se helheten i deras arbeten.

2.2 Varför DevOps?

Varje generation av utvecklare har stött på problem som nästa generation ämnat lösa. Det agila arbetssättet växte fram som svar på en allt mer stel utvecklingstrappa som fick allt svårare att anpassa sig efter en mer dynamisk efterfrågan. Men med nya arbetssätt är det lätt att nya problem identifieras, och DevOps kommer nu som svar på en växande klyfta inom IT-avdelningar och bristen på kvalitetstänk för hela systemutvecklingskedjan. DevOps kom till världen som ett svar på tidigare problematik rörande det traditionella sättet att driva och hantera utveckling och drift inom IT. Nedan kommer en övergripande bild av problematiken redogöras för:

Hütterman (2012) och DeGrandis (2011) nämner några situationer som kan stötas på i de flesta typer av projekt som innehåller utveckling. I IT-projekt är det inte omöjligt att en eller flera utvecklare varken delar med sig eller dokumenterar sitt arbete. Detta skapar en hjältekultur, där enbart skaparen “hjälten” förstår sig på sin kod och kan pusha förändringar. Betoning av titlar är också problematiskt, titlar korrelerar ofta med tiden en varit på företaget, och inte med

kompetensen en har. *Shadow responsibilities,* att undvika ansvar, ex. att fokusera på sekundära och otydliga aktiviteter. Detta skapar en störning mellan rollbeskrivning och det faktiska arbetet vilket i sin tur resulterar i problem med planering och koordination. Sist så nämns *favoring plan over planning* vilket isolerar de avsedda resultaten från själva målen. Revideringar av ursprungsplanen görs sällan eftersom målet ändrats, från att leverera värde till kund, till att följa planen till punkt och pricka istället.

Kulturella och organisatoriska barriärer är en av de mest kritiska utmaningarna som DevOps ämnar bemöta. Den traditionella uppdelningen av utveckling (Dev) och drift (Ops) med sin motsvarande silo mentalitet är ett stort hinder i föreningen av mål och ambitioner i hopp om att leverera mervärde. Dessutom kan avsaknaden av ett gemensamt språk hittas, vilket försvårar kommunikationen ytterligare. Missförstånd, extra arbete och förvirring är några av de vanliga följderna av detta (Hütterman, 2012; Colavita, 2016).

Fellinjeringen av mål och incitament mellan de olika arbetsgrupperna nämner flera forskare som ett problemområde, och beskriver DevOps som att det kan anses vara lösningen (Hütterman, 2012; Hussaini, 2014, Humble & Molesky 2011; DeGrandis 2011). Utifrån det positiva stöd som DevOps får från flertalet författare så ser DevOps onekligen ut som en räddare i nöden, men det bör också ifrågasättas då implementationsbeslut bör grundas utifrån både ris och ros. Den agila metodologin har med sin framfart bidragit till både ökad produktivitet och transparens inom utvecklings- och kvalitetssäkrings områden (Mohammad et.al, 2013), men driften har traditionellt sett fallit i skymundan (DeGrandis, 2011). De agila teamen inom utveckling har som mål att leverera mer funktioner på kortare tid, och verksamheten förlitar sig på att de svarar snabbt på skiftande trender och buggar i mjukvara (Hussaini, 2014). Å andra sidan så förlitas driftsättning leverera funktionerna vidare till produktionsmiljön med ytterligare krav rörande stabilitet som mycket väl kan försakas vid förändringar i den redan existerande miljön. Något som ofta motstrider sig leveranskraven (Hütterman, 2012).

Som tidigare nämnts så anammar agil utvecklingsmetodologi små iterativa utvecklingsprocesser, medans agilt förespråkar samarbete och koordination så faller driftsättning trots allt utanför. Den nu ökade takten som utvecklarna levererar funktioner i, tjänar här till att pressa arbetsgrupper i driftsättning ytterligare och för de olika funktionerna längre ifrån varandra (DeGrandis, 2011). Hütterman (2012 s.23) sammanfatta de som att fördelarna med agila processer ofta nollställs på grund av "*the obstacles to collaboration, processes, and tools that are built up in front of operations*". Medans mjukvaran kontinuerligt släpps i testmiljöerna så hamnar den på hög till produktion, eftersom operations traditionellt sett är emot att leverera i så hög takt. Denna obalans i färdiga funktioner och korresponderande leveransdatum påverkas ytterligare av *horisontell optimering* där operations prioritet ligger på att optimera infrastrukturen istället för att göra förändringar i en stabil produktionsmiljö (Hütterman, 2012).

Bristen på feedback i realtid uppmärksammas även som ett problem. Bland grundprinciperna för Lean utveckling så säger en "*eliminate waste*" (Poppendieck, 2002). Det är vidare förstått att funktioner som kunden inte behöver för att utföra sin uppgift inte heller behöver implementeras (Poppendieck & Poppendieck, 2007). Vetskapen om huruvida funktioner ska gå vidare eller inte kräver god kommunikation mellan utvecklare och kunder, så kallad *feedbackloop* (Highsmith & Cockburn, 2001). Saknad av denna kommunikation menar Humble & Molesky (2011) står till svars för de långa ledtiderna ofta sedda hos icke-DevOps organisationer.

Humble & Molesky (2011) tar slutligen upp den överväldigande komplexiteten kring systemen vilket försvårar spårningen av funktionalitet och vilka system som är vitala kontra redundanta. Komplexiteten i systemen höjer sannolikheten att råka avveckla ett system som fortfarande har en aktiv roll i maskineriet, vilket kan medföra en minskning i affärsverksamhet. Men om ingen handling utförs så upplevs ineffektivitet genom att ha alla system kvar i drift och på så vis bibehålla driftkostnaderna med redundant funktionalitet.

Hütterman (2012) och Cuppettt (2016) nämner fyra dimensioner: Culture, Automation, Measurement och Sharing. De anser dessa fyra områden bildar grunden för att korrekt tolka vad DevOps omfattar. Dimensionerna kan tas i beakta av organisationer, både under implementationen men även för det kontinuerliga arbetet med DevOps, för att se till vilken grad implementationen görs i de olika områdena. Förblir man medveten om dessa, kommer sannolikheten för ett framgångsrikt DevOps arbete öka. CAMS fyra dimensioner angriper tillsammans de traditionellt existerande problem som nämnts i det tidigare stycket, med avsikt att rikta in alla intressenter, framförallt de hos utvecklarna och driftpersonalen. Av de olika ramverken som existerar idag som stöd till att förstå DevOps, kan CAMS-modellen anses vara en av de mer relevanta (Gain, 2018).

2.3 Kultur

Organisationskultur är väldigt brett och holistiskt, och det är väldigt svårt att definiera en organisationskultur till följd av att det involverar en grupp av individer och bygger på dess traditioner, värderingar och bakgrund, vilket gör att det mycket väl kan skilja sig mellan avdelningar och arbetslag (Alvesson & Svenningsson, 2016). Men en kultur är något som varje organisation har och lever i, och kulturen är det som står till grund och vägleder handlingarna inom organisationer snarare än själva handlingen (Alvesson & Svenningsson, 2016). Kulturen i en organisation har även liknats med dess kärna, eftersom den beskriver hur ett företag bedriver sin verksamhet, arbetsflöden och hur relationen ser ut mellan de anställda (Adeniji et.al, 2015).

Men hur ser kulturen ut i DevOps då? Kulturen i DevOps bygger på att det finns en acceptans för delat ansvar i arbetsgrupper som hjälper till att leverera högkvalitativ mjukvara till kunden, för en lyckad DevOps implementation krävs därmed ett främjande av kultur som bygger på egenskaper som samarbete, lärande och förtroende under en längre tid (Wiedemann, 2017)

Det viktigaste i en organisation är den mänskliga resursen, kan anses vara den mest kritiska i en organisations konkurrenskraft och framgång (DiVanna & Rogers, 2005). Med den mänskliga resursen som den mest kritiska faktorn till framgång, så är DevOps ett steg i rätt riktning för företag, för DevOps handlar först och främst om människor, inte tekniker (Hütterman, 2012). Men det är också den svåraste då kultur som tidigare nämnt gäller en grupps värderingar och bygger på traditioner och bakgrund, vilket gör det väldigt svårt att genomföra en meningsfull förändring som är märkbar på kort tid (Alvesson & Svensson, 2016). En DevOps implementation är något som växer fram gradvis under en längre tid genom att främja ett beteende som linjerar sig allt mer med den önskade förändringen. För att då kunna ändra sättet man arbetar på, blir det viktigt att man får med sig samtliga berörda parter i förändringsprocessen (Wiedemann et.al 2018).

En stark företagskultur kan innebära en styrka, men kan vid förändringar vara problematiskt, mer specifikt kan en stark kultur försvåra implementationen av DevOps-tänket, då förändringar i kultur inte uppstår genom ett knapptryck, utan måste byggas upp över tid. Kalliosaari et.al (2016) spekulerar att potentiell problematik kan uppstå då en stark kultur motsätter sig förändringar för att de är oflexibla till följd av etablerade praxis och roller. I förhållande till kulturell förändring lägger Dawson & Andriopoulos (2017) till att etablerade kulturer också kan vara oflexibla då redan etablerade mål och processer inte förändras och ställs i linje med DevOps, vilket kompletterar Kalliosaari et.al (2016).

Det finns även problem om en organisation är svag, där stöd och intresse från ledande roller saknas vilket blir en indirekt typ av motstånd (Kalliosaari et.al, 2016), och det underlättas inte av möjligheten som erbjuds genom att två arbetsfunktioner skall samarbeta, där Smeds et.al (2015) fann att bristande intresse bortförklaras av båda sidor genom att skylla på den andra. Bristen av intresse i svaga organisationer är problematiskt då resurser som tid och pengar spenderas på projekt som inte har något hållfast gehör, men problematiken i att en svag organisation ej tar till sig DevOps kan ses som ett symptom på ett möjligt större problem. Både Kalliosaari et.al (2016) och Smeds et.al (2015) nämner DevOps avsaknad av tydlig definition som ett problem, men drar inte slutsatsen att både motstånd i starka kulturer och bristande intresse i svaga kan vara tecken på att definitionen av DevOps inte var etablerad och att en förståelse för ämnet inte tydligt kunnat förmedlas.

Hur allvarliga problem ansågs vara inom organisationer var beroende på dess storlek. Där oftast större företag ansåg sig värre drabbade än mindre organisationer, då mindre organisationer anammar en nonchalant attityd om att allt kommer lösa sig (Kalliosaari, 2016; Rajpal, 2016). Det har även uppmärksammats att hierarkiska beslutsled måste försvinna och makten ska hamna hos de autonoma teamen, för att facilitera de snabba kontinuerliga arbetet (Wiedemann et.al 2018).

2.4 Automation

DevOps kan med hjälp av automation uppfylla många av de kriterier agilt arbete medför (Kim, 2016). Automationen i relationen till DevOps bygger på dess implementation för att kunna skapa ett effektivt arbetsflöde, där nybyggda lösningar automatiskt testas, vilket ger förtroende i koden om testerna är godkända och därmed kräver mindre manuellt arbete. Det kan leda till effektivisering av en stor del processer i främst utvecklingsfaserna där den mänskliga faktorn tagits bort. Tester och implementationer etc. utförs på ett pålitligt, konsekvent, effektivt och snabbt sätt. Mänskliga faktorns frånvaro underlättar inte bara för testerna utan tillåter även människan att inta en observerande istället för agerande roll. Vilket minskar påfrestningen på individen och gör det möjligt att allokera resurser på annat håll (Haight, 2015). Valet av verktyg problematiseras också av Smeds et.al (2015) och Ghantous & Gill (2017) som en aspekt att ta hänsyn till när arbetssättet nu delvis förändrats.

Enligt Humble och Molesky (2011) så är målet med automation att minska ledtider och skapa snabb återkoppling. De sammanfattar ovanstående del i att arbetet börjar med att kompilering, drift och regressionstester automatiseras och att en *Infrastructure as Code* (IaC) skapas. Det går vidare med att säga att en leveranspipeline även behöver implementeras. Vad pipelinen gör är att

skapa en övergripande bild av alla processer i värdekedjan, den modellerar hela processen från att utveckla, bygga, testa och sist leverera slutgiltig produkt (Bass et.al, 2015; Humble & Farley, 2010). Med allt som automation uttrycks tillföra och förbättra så lyfter Smeds et.al (2015) att externt motstånd kan komma att upplevas på grund av bristande IT-mognad, där DevOps processerna inte är lämpliga för implementation och kunden är vid behov av mer traditionellt sätt strikta procedurer.

Systemens kompatibilitet är också en viktig del i etableringen av bland annat pipelines och andra automatiska flöden. Problematik kring äldre utdaterade system, monolitiska byggen och de tekniska begränsningar de medför är därför ett faktum (Smeds et.al, 2015; Kalliosaari et.al, 2016; Ghantous & Gill, 2017).

2.5 Mätning

Mätning av organisationers processer är av stor vikt då det visar påverkan som förändringar har. Mätning ska utföras på processer som itereras ofta och kan pålitligt reproduceras för att generera nyckeltal som avgör prestanda och effektivitet (Hütterman, 2012). Det är av stor vikt för att se hur väl en avdelning eller organisation presterar, men också vilken effekt som en genomförd förändring haft. Betydelsen av mätning är stor av flera anledningar, främst av dessa är att om ingen mätning sker och förändringar genomförs så kan resultatet inte jämföras objektivt, utan enbart subjektiva.

Men vid mätning måste mycket omsorg läggas vid valet av nyckeltal, och om det som mäts är viktigt i DevOps. Vilka nyckeltal som mäts är av stor vikt i DevOps, då de ska spegla de gemensamma målen som arbetsgrupperna arbetar efter och värdet som skapas skall reflekteras genom nyckeltalet. Men trots att mätning väger tungt i DevOps så är det inte glasklart vad som bör mätas enligt Ravichandran et.al (2016), och de spekulerar att det är till följd av att DevOps inte har ett formellt etablerat ramverk. Hütterman (2012) varnar gällande fallgropar som uppstår i mätning, där kvantifierade tal förblindar arbetsledare och blir ett fokus, men talen kan mycket väl sakna substans och ger då en illusion av kontroll. Ravichandran et.al (2016) föreslår att de redan existerande mätetalen utvärderas, även om de bestrider DevOps-konceptet. Problematiken som lett till DevOps introduktion var att utvecklare och driftpersonal hade olika målsättningar och bedömningskriterier, så mätningen under ett DevOps arbetssätt måste frångå vad avdelningarna traditionellt fokuserat på.

Humble & Molesky (2011) nämner även hur mätning påverkar hur folk arbetar. Visualiseringen av mätetalen tjänar även till att hålla arbetarna upprättade om hur väl de presterar oavsett skede. Mätetal kan även ha en inverkan på upptäckten av suboptimala delar av processer (Humble & Molesky, 2011) - ett viktigt steg för att bibehålla en lean process. Cuppett (2016) introducerar vad han anser vara konkreta och viktiga mätetal för att åstadkomma en effektiv mjukvaruleverans. (1) *Continuous code integration*, hur effektivt samspelar den nyintroducerad och redan existerande koden. Här mäts antalet fel (felfrekvens), och det genomförs i automatiserade testmiljöer. (2) *Automated testing*, här granskas funktionaliteten av ny eller modifierad kod i relation till förväntan. Regressionstester genomförs på hela eller delar av systemet till följd av buggfixar, omstrukturering och implementationen av nya eller utökad

funktionalitet. (3) *Continuous delivery/deployment,* hur ofta släpps ny kod in i produktionsmiljön och här mäts både release takten samt huruvida ny kod skapar störningar vilket mäts i antalet incidenter.

2.6 Delning

Transparens och effektiv delning mellan arbetslagen är kritiskt för att brygga gapet mellan utvecklings- och driftpersonalen. Enligt Wiedemann (2017) handlar detta om delning av allt från verktyg till kompetens för att effektivt arbeta gränsöverskridande och tvärfunktionellt. Där individer inte skall sitta inne på information som leder till fenomenet hjältekultur, vilket står i stark kontrast till DevOps, där kommunikation med öppenhet och transparens är viktiga ledord och kunskap ska kunna delas mellan alla.

I samma anda kring transparens och kommunikation kan vi återkoppla till DevOps huvudsakliga mål om att brygga gapet mellan Dev och Ops. Här identifieras problematik kring en fortsatt isolerad mentalitet, och itereras av samtliga artiklar rörande DevOps utmaningar och problem (Smeds et.al 2015; Kalliosaari et.al 2016; Ghantous & Gill, 2017; Wiedemann et.al 2018).

Det skall ske ett kontinuerligt informationsutbyte mellan arbetslag och individer. Vilket utöver informationsutbytet även bidrar till att stärka de sociala aspekterna av DevOps. Humble & Molesky (2011) belyser vikten av att dela verktyg och praktiska färdigheter som effektiva men till delning. Men en inte får glömma bort de sociala aspekterna och hur de också väger tungt för både delning men också den genomgående kulturen. Att arbetslagen befinner sig på samma geografiska plats och har möten i person tjänar till att ytterligare stärka de sociala aspekterna av DevOps. Vilket itereras ytterligare av att både Smeds et.al (2015) och Kalliosaari et.al (2016) upptäckt problem relaterat till geografiska barriärer.

Att dev och ops kör samma applikationer och verktyg innebär att de inte längre enbart har delade mål, utan de delar även sina loggar, nu lagrade i ett gemensamt system med ett aktivt språk, för att lättare kunna prioritera arbetet. Fördelen med den nya delade och gemensamma miljön är en starkare error-hantering och spårning av diverse incidenters inverkan på andra delar i arbetet (Kim, 2016).

2.7 Kritik mot CAMS

CAMS kom under kritik av Jez Humble, en välkänd profil inom DevOps fenomenet, som menade på att CAMS saknade ett klart effektivitets fokus, och föreslog tillskottet av Lean, vilket skapade akronymen CALMS (Beyer et.al 2018). Området har valts att presenteras avskilt från CAMS på grund av att problemet främst lyfts av Humble och inte tas upp i annan litteratur berörande CAMS. Även utifrån tidigare moment anses Lean fortfarande vara relevant för arbetet och Lean-tänket är fortfarande en stor del inom systemutveckling överlag då DevOps används i samband med agila utvecklingsmetoder såsom SCRUM. Utöver att agila metoder kompletterar DevOps, så bygger DevOps redan på att korta ner ledtider och leveranser, och är på så vis ett förhållningssätt för att effektivisera arbetet mellan grupper. Likväl så står automatisering och

öppenheten i DevOps kultur för att motverka ineffektiv användning av resurser vilket gör tillägget av Lean till ett överflödigt slagord som är starkt förknippat med gamla tillvägagångssätt. DevOps är nytt och behöver inte knyta an till alla gamla slagord och anor.

I samma mån som automation tillför positiva aspekter så kan även för mycket automation leda till en överkomplicerad infrastruktur. Allt måste nyttjas med måtta och för mycket av något bra kan leda till något negativt istället (Przybyl, 2017). En lean-process strävar efter att presentera sin renaste form utan det lilla extra, vilket oavsett inte har någon betydelse för kund. Målet är att arbeta iterativt och att jobba mindre men snabbare. Det är dock viktigt att notera, en kontinuerlig leveransprocess som denna kräver robust felhantering (Spaven, 2018), och bara för att arbetet sker iterativt ska det inte innebära att arbetet genomförs halvdant (Cohn, 2006).

Poppendieck och Poppendieck (2003) formulerade sju grundprinciper för appliceringen av Lean i mjukvaruutveckling. (1) *Eliminate waste*, här avser waste allt som inte tillför något värde, och därmed kan tas bort utan konsekvenser. (2) *Amplify learning*, ständigt utvärderings- och förbättringsarbete skapar starka möjligheter att lära sig. (3) *Deciding as late as possible*, utifrån ett utvecklingsperspektiv betyder att systemen ska besitta förutsättningarna att kunna ändras och korrigeras. (4) *Deliver as fast as possible*, kommer produkten ut snabbare till kund så gäller likadant för återkopplingen tillbaka, en av aspekterna för en lyckad feedbackloop. (5) *Empower the team*, Vid beslutstagande, främst tekniska, bör de som ansvarar för utförandet vara delaktiga. De korta iterationerna tillåter inte för långa beslutsled och ansvaret bör läggas på de som är närmast arbetet. (6) *Building Integrity in*, kvalitetssäkring i alla delar av produktionsledet säkerställer att rätt saker går vidare och slutligen hamnar i produktionsmiljön. (7) *Seeing the whole*, sammanfatta som “A system is not just the sum of its parts- it is the product of their interactions” (Poppendieck & Poppendieck, 2003, s. 146). Kan tolkas som att system likt organisation måste se till samverkan av alla komponenter och inte bedöma utifrån individuell prestation. Flera av dessa leden är redan aktiva länkar i DevOps tänk.

2.8 Sammanfattning av CAMS-modellen

Sammanfattningsvis så är CAMS grunden för vad DevOps omfattar. Det handlar om att kulturen i en DevOps organisation skall vara öppen och transparent, information och ansvar skall delas av alla involverade för att uppnå en så hög kvalité på produkterna som möjligt. Det handlar även om att bygga ett förtroende för varandras kompetens, genom den kontinuerliga delningen av information och öppenheten så att beslut kan tas av personer som står närmare systemet istället för att vänta på flera godkännande från överordnade. Här kan det förekomma mest problematik i organisationer, då kultur beskrivits som bland det svåraste att förändra om kulturen är en stark sådan enligt Dawson & Andriopoulos (2017), men det bör även vara svårt med en svag sådan då det finns en avsaknad av intresse i förändringsarbetet och även om intresset finns för förändring så lever folk inte alltid som de lär.

Automationen står till grund av DevOps, där de uppgifter som kan automatiseras skall automatiseras för att undvika tidsåtgång långsiktigt, detta gäller allt från testning och uppsättning av nya miljöer. Det skall minska beroendet av andras ingripande i arbetsprocesserna, där utvecklare kan bygga och testa mjukvara med tydliga krav på kvalité, och driftpersonal kan med förtroende för tester och utvecklare implementera uppdateringar utan att försaka systems stabilitet. Men automatisering och kodkvalité är starkt beroende av kompetens, och inom IT/IS så kan landskapet förändras på kort tid. Detta bidrar med problematik i form av avsaknad av kompetens, och till en början mycket tidsåtgång att bygga in automation i processerna som kan fasa ut individers arbetsuppgifter.

Mätning i DevOps anses vara en central del av arbetet. Med hjälp av mätning finns kvantitativa data på hur förändringar i kultur och automation kan ha förändrat effektiviteten av arbetsgrupperna genom att jämföra resultaten före och efter genomförda förändringar. Mätnings effektivitet kan manifesteras i exempelvis systemprestanda, då det speglar de tvärfunktionella arbetsgruppernas möjlighet att samarbeta. Problematik i området har centrerats kring svårigheter att använda mätetal som korresponderar med verklighetsförankrade förbättringar, men även som förmedlar en gemensam sanning för båda arbetsfunktionerna inbegripna i DevOps.

Delning i DevOps sker för att skapa en återkoppling mellan arbetsgrupper, och individer, så att ett kompetensutbyte sker kontinuerligt och erfarenheter kan bytas. Men det skall inte enbart ske ett utbyte av mjuka egenskaper utan även verktyg och resurser så att funktionerna har sina specialitéer, men till följd av kompetensutbytet skall arbetsgrupperna bli mer tvärfunktionella så att de kan höja kompetensen.

2.9 Sammanfattning av identifierad problematik i CAMS

Nedan kommer en sammanställning av tidigare utmaningar som DevOps ämnat bemöta, samt upplevd problematik kring arbetet med DevOps.

CAMS	Författare	Traditionell problematik	Författare	Problematik i DevOps
Culture	DeGrandis, 2011 Hütterman, 2012 Colavita, 2016	Hjältekultur Stöd Mentalitet	Smeds et.al, 2015 Kalliosaari et.al, 2016 Ghantous & Gill, 2017 Wiedemann et.al, 2018	Buzzword Geografiska barriärer Överbelastning Mentalitet Stöd
Automation	DeGrandis, 2011 Hütterman, 2012 Hussaini, 2014	Legacy Överbelastning Teknisk skuld	Smeds et.al, 2015 Kalliosaari et.al, 2016 Ghantous & Gill, 2017	Legacy Mognad Val av verktyg Externt motstånd
Measurement	Humble & Molesky, 2011	Legacy Nyckeltal	Smeds et.al, 2015 Ghantous & Gill, 2017	Legacy Nyckeltal
Sharing	Highsmith & Cockburn, 2001 Poppendieck & Poppendieck, 2007 Hütterman, 2012	Gemensamt språk Kommunikation	Smeds et.al, 2015 Kalliosaari et.al, 2016 Wiedemann, 2018	Organisatorisk struktur Kommunikation

Figur 2.9 Sammanfattning av identifierad problematik i CAMS

3. Metod

3.1 Datainsamling

För att besvara frågeställning gällande problematik i DevOps lämpade sig en kvalitativ undersökning av ämnet. Genomförandet av en kvalitativ undersökning grundar sig i faktumet att DevOps beskrivs av flertalet författare som att det främst handlar om människor, och för att organisationskultur är en dimension som väger väldigt tungt i dess implementation, och vi såg inte hur en kvantitativ undersökning skulle kunna besvara frågan.

3.1.1 Val av undersökningstyp

För att genomföra en kvalitativ undersökning av ämnet valde vi mellan två tillvägagångssätt, enkätundersökning och intervjuundersökning. Då stort fokus läggs på människor och kultur i existerande litteratur ansåg vi det lämpligt att genomföra intervjuer istället för enkätundersökning, även om båda undersökningstyper skulle kunna besvara frågeställningen. Båda tillvägagångssätten hade sina styrkor för hur de skulle komma att bidra med ett resultat. Att undersöka upplevd problematik hade varit möjligt med en enkätundersökning då respondenter besvarat ett fast antal frågor som varit konsekventa mellan alla respondenter och möjliggjort en hög grad av jämförbarhet mellan svaren, likväl hade det varit möjligt att med hjälp av nyckelord tydliggöra frågeställningen för respondenten i enkäten. Men vi valde däremot att inte genomföra enkätundersökning då vi misstänkte en låg svarsgrad. Intervjuer förväntades vara ett mer fördelaktigt medium då det skapade en personlig koppling till respondenter där de kunde se sina intervjuare och besvara frågor väldigt frispråkigt men även be om förtydligande i frågeställningen. Den personliga kopplingen är något som är viktigt i kvalitativ undersökning för att kunna nå den värdefulla data som en respondent är villig att dela (Seidman, 2006).

3.1.2 Urval av intervjuobjekt

Då vår frågeställning riktar sig mot upplevd problematik i DevOps så har vi sökt respondenter som aktivt arbetar i DevOps, och för att identifiera företag och nyckelpersoner som arbetar med DevOps har vi sökt kontakt genom vårt sociala kontaktnät via email och telefon för att samla upp så många respondenter hos varierande företag och efterfrågat intervjuer. Respondenter med direkt erfarenhet av implementation av DevOps eller arbete i DevOps har varit av intresse.

Vi har genomfört en intervju med varje respondent och transkriberat varje samtal. Seidman (2006) rekommenderar tre intervjuer med varje respondent för att bygga ett närmare samarbete och förtroende med respondenten, men också för minska pressen på att intervjuaren måste utvinna värdefulla data vid ett enda tillfälle. För att motverka pressen att första tillfället måste vara bra har vi efterfrågat möjligheten att maila respondenter gällande förtydliganden och följdfrågor om det skulle visa sig nödvändigt.

Verksamhets område	Antal anställda	Befattning	Längd	Typ	Respondent
Privat	10 +	Systemutvecklare	35min	Videolänk	P1
Privat	100 000 +	Teknisk projektledare	44min	I person	P2
Privat	10 000 +	Manager, Solution Architect	40min	Videolänk	P3
Offentlig	1000 +	IT-konsult	50min	Videolänk	P4

Figur 3.1.2 Sammanställning av intervjuobjekt

3.1.3 Intervjuform

Inför intervjuerna har vi skickat ut ett kort sammanfattat underlag (se bilaga 8.6) för våra respondenter att utvärdera innan vi genomförde intervjun för att de skulle ha möjlighet att förbereda och formulera sina tankar. I underlaget använde vi oss av nyckelord i förhållande till CAMS för att rikta in deras tankegångar, snarare än att utförligt beskriva varje enskild punkt. Vi resonerade som så att det vore bättre för våra respondenter att ha en grovhuggen förståelse för vad vi är ute efter och begrepp som är förknippade med vår undersökning, snarare än att de först introduceras under intervjuerna och försöker svara oförberett utan någon vägledning alls. På så vis försökte vi minska pressen på oss som intervjuare, men även respondenten, och höja kvalitén på intervjuerna.

Magnusson & Maracek (2015) rekommenderar en intervjuguide för alla nivåer av intervjuare, och vi valde att följa rekommendationen och investera tid i att ta fram en intervjuguide, framförallt för att ställa frågor som generera data som kan besvara vår frågeställning, men även för att upprätthålla en nivå av professionalism och respekt gentemot respondenten som bidrar med sin värdefulla tid. Leavy (2017) påpekar att även val av språk är av vikt vid intervjun, inte enbart med anledning att få fram data, utan även för att visa respekt och hänsyn mot människor och kulturer. Detta gjorde att vi var noga med de termer vi använde när vi ställde frågor, så att de inte var av känslig natur, och så att de relaterade till ämnet. Detta var viktigt då vi som studenter vid Lunds universitet är representanter för universitetet och det systemvetenskapliga kandidatprogrammet, och upprätthålla professionalism är av prioritet.

Intervjuerna genomfördes på plats med respondenten om möjligt, annars genomfördes en videokonferens över Open-source videotjänsten Jit.si. Varje intervju tog 35–45 minuter för att gå igenom alla frågor som förberetts, samt följdfrågorna som ställdes under intervjuernas gång. Frågorna som ställdes hölls medvetet öppna för att respondenten utförligt skulle kunna reflektera över frågorna.

3.1.4 Intervjustöd

Här följer det en del av det intervjustöd vi använde, där vi fokuserade på att strukturen av CAMS och problematik. Vi fokuserade på att använda stödord för att ställa frågor och våra frågor var därmed ostrukturerade.

Företagskultur: Utbildningar, internt motstånd, omstruktureringar (team, avdelningar etc), nya roller, omfördelning av ansvar (nya ansvarstagare), godkännande, förtroende.

Automation: Tester, verktyg, script, leveranspipeline, testmiljöer.

Mätning: hur mäter ni? Nyckeltal (commits, tickets, antal "leveranser", felfrekvens, felhantering etcetera), målsättningar, processer

Delning: Informationsutbyte, återkoppling och feedback, ta vara på kompetens (arbetslagen).

3.2 Genomförande av analys

För att genomföra analys av den insamlade empirin utfördes kodning i form av en grov strukturering av resultaten, där respondenternas svar sorterades utefter det korresponderande området i CAMS som behandlades under intervjun, för att sedan grupperas efter liknande upplevd problematik i den mån det var möjligt. Då intervjuerna genomfördes som semistrukturerade med grundpelarna som är etablerade i CAMS förekom en struktur i empirin, men då de specifika frågorna som ställdes under intervjuerna inte var ordagrant lika kunde svar ej kodas utefter frågorna, eller intervjuprotokollet, men det har bidragit med en grov struktur.

3.2.1 Transkribering

Vid transkribering så lyssnade vi igenom de inspelade intervjuerna och transkriberade all information som kunde vara av nytta för uppsatsen. Transkriberingen utfördes inte ordagrant, då vi valde att eliminera utfyllnadsord som ofta används vid när en respondent formulerar ett svar. Detta gjorde även att vi fick ett flyt i svaren som annars inte är vanligt förekommande i muntlig konversation. Då båda författarna var närvarande vid intervjutillfällena diskuterade vi svaren som mottogs vid transkriberingen långt om länge för att balansera korrektheten i data och läsbarheten. Vid gränsfall av förståelse mailades en uppföljningsfråga till respondenten, eller så diskuterades svaret utifrån inspelningen och vår gemensamma erfarenhet från intervjun.

Vid intervjuerna som utfördes över videolänk så spelades både ljud och bild in, då detta var standardutförandet i tjänsten som användes. När vi lyssnade tillbaka på dessa ansågs det inte som något viktigt att bild från webbkameror också spelats in, men under intervjun användes detta för att tolka respondenternas reaktioner till frågorna och bidrog även till en bättre upplevelse för de involverade parterna.

3.3 Validitet och reliabilitet

För att våra resultat skall vara giltiga är det viktigt att vårt arbete förblir objektivt och är av hög inre validitet (Jacobsen, 2002). Vi har därmed fokuserat på detta genom arbetets gång, och detta fokus har genomsyrat metodkapitlet, där vi följaktligen beskrivit våra handlingar, och resonemang i detalj. För att uppnå en hög grad av validitet har vi genomfört en deltagarkontroll mot slutet av våra intervjuer i form av dialogisk validering, där vi försökt sammanfatta intervjun och de svar vi mottagit från respondenten så att denne vid intervjutillfället haft möjlighet att korrigera sina svar (Malterud, 2014). Vi har även sett till att samla ihop de för oss viktigaste delarna av varje transkriberad intervju och därefter bett respondenterna kontrollera dessa för att ytterligare validera vårt resultat. Vi samlade ihop de viktigaste delarna av den transkriberade intervjun och sökte validering är för att höja kvalitén av data, och ville genomföra ett så kort utskick som möjligt så att de verkligen läste texten istället för att skumma igenom den och fokuserar på fel punkter som ett helt transkriberat utskick kunde gjort.

I förhållande till teknisk kvalité på intervjuerna så har den varit god då det finns en hög lägsta nivå på teknisk utrustning i dagsläget, vilket bidrar till reliabiliteten av arbetet. Intervjuerna som utfördes i person spelades med hjälp av en smartphone av modell OnePlus 3T. När intervjuer har genomförts över videolänk användes ett Open-sourceverktyg Jitsi Meet som föreslogs utav en av respondenterna, och dessa genomfördes på en HP-laptop med dess inbyggda mikrofon, webbkamera och högtalare.

Kvalitén på intervjuerna har försökt hållas hög genom hela arbetet. För att uppnå det så har vi valt att skicka ut förberedande text till intervjuobjekten som kunnat bilda sig en väldigt grundlig förståelse för vad vår intervju kommer att fokusera på, och de har därmed kunnat formulera sina tankar i förväg. Detta gjorde vi till följd av att vi bara bokade in ett intervjutillfälle med varje intervjuobjekt och vi hade därmed endast en chans att söka data som kunde besvara frågeställningen, men det var också möjligt att komplettera den via mail i efterhand. Vi har även inför intervjuerna försökt bilda oss en djup förståelse för vad DevOps är och vad som tidigare etablerats som problematik i förhållande till de identifierade grundpelarna, detta efter rådgivning från vår handledare som föreslog att vi sköt upp datainsamlingen med ungefär en vecka så att vi kunde bilda oss en klarare bild av vad vi var ute efter.

Härnäst så genomfördes varje intervju vid en tidpunkt som passade respondentens schema i ett försök att minska utomstående press, och externa faktorer försökte uteslutas genom att föra direkt kontakt med respondenterna under förutsättningen att de medverkade av egen fri vilja. Vi förberedde en typ av stöd inför intervjuerna som bestod av stödord och en struktur som skulle kunna följas under intervjuns gång, detta stöd var inte i form av en utförlig intervjuguide eller mall, men lånade idéer som att upprätta en struktur för intervjun i vilken ordning vi ställde frågor med olika fokus, samt stödord som kunde hjälpa oss ställa frågor. Detta gjorde att intervjuerna var semi-strukturerade, men frågorna var inte ordagrant densamma för alla respondenter och hade därmed mer inslag av ett ostrukturerat genomförande.

Vi har observerat att våra erfarenheter under datainsamlingen har påverkat resterande datainsamling, där vi ställt frågor som grundat sig delvis från tidigare erfarenheter, vilket gör att vi onekligen påverkat följsamheten i den insamlade data.

3.4 Etik

Problematik är i sin natur ett känsligt ämne, då det kan ha starka kopplingar till individers eller organisationers oförmåga att prestera och tackla svårigheter som uppstår. Då vi sökte organisationer som arbetade med DevOps hade detta stark koppling till en organisations IT-avdelning, och en svagare koppling till resterande del av organisationen genom det stöd som IT och IS bidrar med. För att respondenternas upplevda problematik inte skall påverka deras nuvarande anställning, eller framtida, så försäkrade vi dom om att de och organisationen förblev anonyma i vårt slutgiltiga arbete. Detta löfte förmedlades muntlig inför intervjun. Innan intervjun påbörjades frågade vi även muntligen om lov att spela in intervjun i syfte att kunna återge data mer korrekt efter transkribering.

Vi valde att hålla alla respondenter, och deras organisationer, anonyma även om de påpekade att detta inte var nödvändigt vid 3 av 4 intervjutillfällen, men vi ville bibehålla en konsekvent behandling av data från respondenterna trots att en majoritet indikerade att de inte ansåg det viktigt. Men då vi var tydliga med vårt tillvägagångssätt från första början, och öppna med att vi anonymiserade intervjuerna, kunde respondenterna känna sig bekväma under intervjun gång och tala fritt. Kontakt med respondenterna utfördes över telefon och email, mot arbetsrelaterade telefonnummer och adresser för att inte inkräkta på deras privatliv. Om en respondent svarade på vår korrespondens under sin fritid var det utan vår vetskap och av deras egen vilja, men vi försökte undvika detta i den mån vi kunde kontrollera.

4. Resultat

4.1 Tolkning

Under genomförandet av intervjuerna har vi undersökt hur respondenterna definierar DevOps, då litteratur har definierat avsaknaden av tydlig definition av begreppet som problematik. Vi har då funnit ett varierat utbud av definitioner för begreppet, vilket gör att förståelsen kring DevOps är spridd, men överlag har dess kärnprinciper om samarbete och automation nått fram. Alla respondenter har tolkat begreppet i den mån de kommer i kontakt med det, vilket var märkbart då tre stycken agerade i liknande arbetsroller och tolkade därmed begreppet snarlikt. Den fjärde hade en arbetsroll och helhetsbild av DevOps som skilde sig från övriga då den arbetat aktivt med integration av alla principer hos ett flertal kunder.

> "*DevOps för mig är en kultur där alla hjälps åt med sina speciella kompetenser och oavsett om man är utvecklare eller drifttekniker så samarbetar man.*" (P4)

Anledningen till att de övriga tres definition skilde sig markant var då de knappt hade kontakt med en tydlig driftsfunktion i organisationen, utan istället använde sig av paketering och molntjänster istället för vad som är traditionell infrastruktur. Detta har då format deras definition.

> "*Utvecklarna tar mer eller mindre hand om allt*" (P2)

4.2 Kultur

Kultur är det mest diffusa området i CAMS och med all rätt, då det är ett holistiskt begrepp som innefattar ett brett område. Området handlar i mångt och mycket om människan och dess värderingar, bakgrund, och traditioner, något som står i kontrast till automation och mätning, vilka individuellt sätt handlar mer om teknik. Vi fann att två av våra respondenter uttryckte sig med att kultur var den största utmaningen som fanns i DevOps.

> *"Mycket kan sammanfattas i människan och för att knyta an till det jag sa i början, så är DevOps kultur för mig och det är den stora utmaningen också"* (P4)

Vi har försökt fastställa ifall problematik som tidigare etablerats i litteraturen har förekommit hos våra respondenter. Våra respondenter har upplevt varierande nivåer av problematik inom kategorierna (1) *Brist på stöd*, (2) *Motstånd*, (3) *Överbelastning* och (4) *Mentalitet.*

4.2.1 Brist på stöd

Problematiken gällande bristande stöd som har yttrats av våra respondenter har delvis grundats i organisatoriska strukturer, där exempelvis koncept såsom automation ifrågasatts ovan i en

organisations hierarki, och hos en annan på strategiskt plan, men det har varit lätt att lösa. Det har även förekommit globala olikheter i organisationsstrukturer som har påverkat stödet för arbetet som förväntats från båda håll, detta har varit efter att ansvaret bland utvecklare ökat till följd av DevOps och därefter skillnaderna i arbetsrätt länder emellan. Problematik rörande brist på stöd noterades ha upplevts hos tre av respondenterna. Detta har även skett på olika nivåer hos de svarande. Problemen har uppfattats finnas på ledningsnivå och teamnivå.

P4 stötte på en ledning som förhöll sig kritiskt gentemot ett väldigt proaktivt arbetssätt, istället för ett reaktivt, genom automatisering och testning och därmed krävde övertygelse.

P2 nämner också hur viktigt det är med stöd från en organisation som förstår sig på kulturella skillnader länder emellan, och garderar sig mot brist på stöd kontraktuellt, och uttryckte att han valt att gå från ett uppdrag till följd av det.

Gällande teamnivå kan problemen handla om något så basalt som tycken gällande nya arbetsmetoder och om dessa inte anammas kan arbetet försvåras:

> *"Vi hade t.ex. en teknisk ledare som inte tyckte om automatiserade tester speciellt mycket, så då blir det extremt svårt att få igenom de eftersom teamet måste vilja göra de för att kunna göra det. Så man måste ha folket med sig[...]"* (P2)

P3 nämner inget konkret huruvida han stött på någon form av bristande stöd, men han nämner i en annan fråga att han upplevt det som att organisationen saknar förståelse för vad som krävs av organisationen för att lyckas med DevOps.

4.2.2 Motstånd

Det område som upplevts som mest problematisk inom arbetsgrupperna är faktiskt motstånd mot testning, vilket vi tolkar som kulturellt då det varit på individnivå som motstånd har mötts. Alla respondenterna upplevde motvilja bland utvecklare att skriva tester och speciellt testdriven programmering trots att det är en central del i DevOps. Angående motstånd så uttrycker sig tre av respondenterna mycket likt gällande vilken typ av motstånd de stött på främst. De har alla sett eller upplevt det kring utvecklares syn på att skriva tester till följd av mer automatisering:

> *"Min känsla är att majoriteten av utvecklarna inte tycker det är speciellt roligt att sitta och koda tester när man kommit förbi enhetstester."* (P4)

Vi noterar även att när P4 talar om motståndet till att skriva tester så refererar han till att utvecklarna vill skriva "riktig kod". Vi får även samma intryck från P3 och P2 att utvecklare överlag, inte anser tester tillhöra deras arbetsbeskrivning.

4.2.3 Överbelastning

Gällande överbelastning så har utvecklarna uttryckt problematik till följd av ett ökat ansvar för sina systems drift i sina roller. Men att de till följd av DevOps har utökad kompetens och ett bättre kvalitetstänk för hur koden ska skrivas, då de nu är så mycket mer involverade i driften.

Problemet uppfattades hos tre av respondenterna i varierande grad. Hos två av dem ansågs det vara till följd av den nya arbetsfördelning där mer arbete hamnat på utvecklarna.

> *"För en utvecklare så kan de bli ganska mycket teknisk skuld om du vill ha en helt automatiserad pipeline, de är väldigt mycket jobb att automatisera[...]"* (P3)

Medans P4 också upplevt liknande börda som P2 och P3 så kan detta varit till följd av omständigheter kopplade till lagkrav som introducerades i den offentliga sektorn, och ett starkt behov av att hantera teknisk skuld som uppstod i legacy system.

Trots att den nya arbetsfördelningen resulterat i mycket större arbetsbörda på utvecklarna, var samtliga respondenter positivt inställda till den nya insikt och kontroll de nu hade i hela arbetsprocessen. I samma anda kunde vi utläsa en gemensam vy kring det nya ansvaret som hamnat hos utvecklarna, vilket resulterat i mer frihet. De hade nu mer frihet gällande metoder, miljöer etc. samt makten att ta sina egna beslut kring features och funktioner att ta vidare.

4.3 Automation

Automationen är den del som våra respondenter hade minst problem med, och näst mest fokus på, då det är en väldigt saklig del av DevOps. Automationen upplevdes som en väldigt viktig del i samtliga av respondenternas arbete. Det var en av de stora byggstenarna i att få igång det effektiva arbetsflöde som eftersträvas i DevOps för att kvalitetssäkra. Samtliga av våra svarande använde bland annat automatiserade tester fast i varierande grad. Något vi ser som resultat av olika mognadsgrader hos de olika respondenterna. Infrastructure as Code (IaC) och leveranspipelines nämns även hos våra respondenter, något som tas upp i litteraturen som viktiga delar inom automationsområdet för DevOps. Vi tolkar det som att problemen kring automation inte varit speciellt avsevärda, där respondenter påpekat att kompetens funnits internt eller tillgängligt i någon form av forum. Den problematik som vi identifierat har vi kategoriserat som (1) *Val av verktyg*, (2) *Legacy system*, (3) *Tidsbrist*.

4.3.1 Val av verktyg

Vi såg att samtliga respondenter hade haft lite olika möten med valet av verktyg. P1 såg problemet som något för framtiden när de väl ska lära sig den nya miljön. De var nämligen i en pågående process att flytta sin verksamhet till AWS:

> "*[...]i AWS så har de extremt mycket verktyg[...]Så utmaningen där är hur man ska navigera AWS-världen.*" P1

P2 sa att det knappt varit någon utmaning att välja verktyg. Hen förklarade det som att deras tidiga intåg på området innebar att det inte fanns speciellt mycket att välja på och de fick ta lite vad som fanns. P3 berättade att de aldrig behövt leta verktyg på grund av sin nära relation till Microsoft. De hade sen tidigare jobbat i Visual Studio Online som sedan blivit Azure DevOps, och vid behov, vände de sig i förstahand till Microsofts portfölj. P4 däremot upplevde lite

svårigheter men inte i själva urvalet utan mer kring bristen på kompatibla verktyg som uttryckt nedan:

> "*[...]det har snarare varit begränsningar i verktyg som gjort att det varit svårt att välja rätt*" (P4)

4.3.2 Legacy system

P1 och P3 uttrycker aldrig något tidigare problem med legacy system eller kod. I P1:s fall kan de bero på att hen tillhört ett mindre företag vilket minimerat problemets slagkraft. För P3 kan vi endast spekulera att det har med deras koppling till Microsoft att göra. P3 säger att de använder huvudsakligen Microsofts egna produkter. Detta kan ha inneburit mindre friktion vid både vidareutveckling av systemen, tillägget av andra och övergången till nya. P2 och P4 hade i liten utsträckning stött på det som ett hinder. Båda hade stött på situationer där det inte kunde göra vad P2 kalla "*Full DevOps*" utan fick göra delvis DevOps lösning:

"*Sen hade vi automatiska tester som vi var manuellt tvungna att starta[...]Så vi kunde inte köra det automatiskt på grund av lite legacy kod*" (P4)

Vad som bör noteras är att de två som nämner problemet, verkar upplevt att legacy bara varit ett mindre hinder till en fullständig körning av DevOps och inte varit ett direkt problem. De har båda löst situationen på liknande vis precis som P2 säger:

"*Man får i de flesta fall försöka göra en så bra DevOps som möjligt utifrån de tekniska förutsättningarna.*" (P2)

4.3.3 Tid

Det har redan upplevts problematiskt kring utvecklarnas attityd mot tester men även att denna typ av arbete anses tidskrävande och att tiden inte räcker till. P4 nämner det flytande i sitt svar gällande problem som upplevts i samband med automation medan P3 uttrycker nedan:

"*[...]att sätta sig ner och koda lite tester.[...]handlar även om att arbetet är tidskrävande[...]*" (P3)

P1 gav oss aldrig någon uppfattning att tidsbrist upplevts på något vis, att deras system inte varit affärskritiska kan ha spelat in i deras något avslappnade attityd kring många av våra frågor. I P2 fall uttrycktes innehavet av mycket god kompetens och att företaget inte uppfattat tid som ett problem. Men P2 nämner även hur viktigt det är med en gemensam definition av ord för att kunna utnyttja tiden effektivt:

"*[...]viktigt att man försöker hitta en gemensam beskrivning för vad begrepp som "klar" innebär.*" P2

Möten och avstämningar uppfattades problematiska när flera individer hävdade att de va klara men fortfarande hade arbete kvar.

4.3.4 Externt motstånd

Två respondenter uttryckte att de stött på kunder som inte haft den mognad som krävs för DevOps och indirekt inneburit ett motstånd. P3 uttryckte sig som följer:

"*Så ibland har vid med oss SCRUM coacher[...]de får i så fall åka ut till kunden och utbilda dem i hur man jobbar inom SCRUM och inom DevOps så vi från företaget kan fortsätta arbeta som vi brukar göra.*" (P3)

De arbetade aktivt med att lägga grunden hos de kunder som inte kunde ackommodera deras arbetssätt. Något som även uppfattas från P4 trots att hen inte uttrycker det i klartext. Men de senaste uppdrag P4 haft handlade om att införa ett DevOps-tänk hos kunder, och etablera en infrastruktur som kunde stödja DevOps principer.

4.4 Mätning

Av våra fyra respondenter var det bara tre som använde sig utav mätning i olika grad, och den fjärde utförde ingen mätning överhuvudtaget. P2 och P3 använde det enbart för att identifiera flaskhalsar i de tekniska systemen och P4 behövde det som underlag för att motivera val till sina kunder.

4.4.1 Legacy

Precis som för automation kan legacy försvåra även arbetet med att mäta. Olik metrik data och svag kompatibilitet systemen emellan, kan innebära en osammanhängande bild av data att förhålla sig till. En av våra respondenter uttryckte sig indirekt att legacy försvårat arbetet att mäta överhuvudtaget.

4.4.2 Nyckeltal

Alla respondenter har på något sätt kommit i kontakt med utmaningar kring valet av nyckeltal hur mycket de brydde sig om mätning var beroende på arbetsgruppers storlek och mognadsgrad. Det var främst prestandamätning av systemen som gällde för utvecklarna medan IT-konsulten behövde data för att kunna motivera val och få igenom beslut.

4.4.3 Ingen mätning på gruppnivå

Hos tre av våra respondenter genomfördes ingen mätning av individer eller arbetsgruppers prestanda, utan de fokuserar på relationerna de byggt upp med sina medarbetare, och den erfarenhet de hade därifrån. Vår fjärde respondent mätte prestandan på funktionerna vid början av sitt uppdrag för att kunna ha en utgångspunkt att jämföra med under uppdragens gång.

4.5 Delning

Området handlar om en transparens och öppenhet grundat i organisationens värderingar, där delningsprinciperna i DevOps är så djupt grundat i det kulturella att området blir svårt att särskilja från kulturen för våra svarande. Vad vi kunde utläsa från våra respondenter var att problematik kring delning varit mer synlig och märkbar tidigare men att det nu var ett mindre återkommande problem som samtliga aktivt försöker arbeta bort. Vad vi trots deras positiva syn kring delning kunde utläsa var att problematiken som faktiskt upplevts kunde sammanfattas i (1) *Människan*, (2) *Arbetsform* och (3) *Organisationen*.

4.5.1 Människan

Problematiken rörande människan var något tre av respondenterna nämnde i samband med parprogrammering, och hur synen kring detta uppfattats hos deras medarbetare.

Både P2 och P3 uttryckte sig gällande problematiken på individnivå, där koncept såsom parprogrammering och närmare samarbete skräddarsyddes efter arbetsgruppens sammansättning, där mer sociala individer kunde ta del av koncepten och mindre sociala arbetade mer på egen hand. Det var inte av intresse för någondera av våra respondenter att förändra ett sådant beteende, utan spelade mer på individens styrkor när det inte motsatte sig arbetsprinciperna på företaget. Beteende som motsatte sig samarbete och transparens i DevOps var något som var viktigt att ändra för det fortsatta arbetet, men väldigt krävande att förändra.

> "*Det svåra där är ju att nå fram till dessa personer och få dem att förstå att de inte ska känna sig hotade bara för att fler kan de du kan. Och det är ju en jätteutmaning för det är ju inget du kan koda bort eller automatisera bort eller så. Det gäller ju att förändra en persons beteende och det är i min erfarenhet bland det svåraste man kan ge sig på och förändra*" - P4

4.5.2 Arbetsform

Arbetsform, specifikt konsultuppdrag, nämns av en respondent som ett märkbart hinder vi delning, huvudsakligen då kompetensdelning. Det ansågs problematiskt då tid är kontraktuellt planerad för specifika arbetsuppgifter. Vilket inneburit mindre utrymme till att ta sig tid till att hjälpa andra. Det ska noteras att det inte handlar om någon typ av motvilja eller okunskap utan enbart begränsningar i uppdragets utförande.

> "*...som konsult så har du ofta som uppdrag att lägga ex antal timmar på olika projekt och då kan det bli svårt att lägga några timmar bara på att lära upp någon...*" -(P2)

4.5.3 Organisationen

Typen av organisation visade sig också vara av problem för delningen i DevOps, där P4 uttryckte att dennes kunder i flera fall hanterade väldigt känslig information som gjorde anställda väldigt försiktiga i förhållande till vilken information de kunde dela i arbetsgrupperna och mellan funktionerna.

> "*... när det gäller att dela med sig så har Org3, Org1 och till viss del Org2 en kulturell barriär i och med att man har väldigt känslig information. [...] Den typen av kultur sprider sig även in i stödverksamheter som IT.*" - (P4)

P4 anmärkte även på hur strukturella barriärer i samband med organisationen kunde hindra kommunikationen mellan utvecklare och slutanvändare.

4.5.4 Mentalitet

Mentaliteten i DevOps har endast varit problematisk för en respondent till följd av naturen av dennes arbetsplats. Våra respondenter har sett en jämnare fördelning av kunnighet bland utvecklarna, och därmed att hjältekultur nästintill har försvunnit. DevOps växte fram ur iden att brygga gapet mellan de två traditionellt skilda avdelningarna. Dessa två "sidor" har uppfattats enbart existera hos P4 som varit den av respondenterna aktiv inom offentliga sektorn.

> *"Vi har haft tydligt, här är utvecklarna och här borta har vi typiskt servertekniker, DBA, övervakningsspecialister och den typen av person[...]"*

I vårt samtal med P4 så ger han oss ett tydligt intryck av att offentliga sektorn är en mycket konservativ arbetsplats och det är lätt att mentalitet som separerar arbetsfunktioner i fack genomsyrar hela organisationen och dess stödverksamheter som IT drabbas. Denna typ av mentalitet, rädsla, ledde i P4:s fall till en bristande förståelse i andra sidan, och murarna där emellan upplevdes endast bli högre.

Det tidigare beskrivna "*hero cult*" eller hjältefenomenet var något som upplevts hos samtliga respondenter, men det ska noteras att samtliga upplevde att problemet minskat avsevärt sen implementationen av DevOps.

4.6 Sammanfattning av problematik hos respondenter

CAMS	Problem/Utmaningar	Referens
Culture	Bristande stöd	P2, P4
	Motstånd	P2, P3, P4
	Överbelastning	P2, P3, P4
Automation	Val av verktyg	P1, P2, P4
	Legacy System	P2, P4
	Tid	P2, P3, P4
	Externt motstånd	P3, P4
Measurement	Legacy	P1
	Nyckeltal	P1, P4
	Ingen mätning på gruppnivå	P1, P2, P4
Sharing	Människan	P2, P3, P4
	Arbetsform	P2
	Organisationen	P4
	Mentalitet	P1, P2, P3, P4

Figur 4.6: Sammanfattning av problematik hos respondenter

4.7 Positiva utfall

Vi fann även att hos respondent P4 som varit ensam med att arbetet utförts i de traditionella Dev och Ops, upplevt ett ökat förtroende till följd av att mer automation införts i de olika arbetsprocesserna.

"*[...]ja absolut det har tagits väl emot och grunden för att bygga det här förtroendet har varit när vi automatiserat saker[...]Så ja absolut detta har byggt mycket förtroende."* P4

P1 fann att kompetensspridningen höjdes över hela dennes arbetsgrupp på ett litet företag genom att börja arbeta med DevOps och SCRUM. Respondenten upplevde utöver en bredare kompetensspridning också att förekomsten av hjältefenomenet, medvetetet och intet, minskade markant. P2 och P3 uttryckte också att kompetensen spred med hjälp av kodgranskning och parprogrammering. De mindre intima teamen gjorde det även möjligt att lättare hugga tag i varandra om de behövde hjälp. P4 nämner aldrig explicit kompetensspridning, men det har också arbetat i mindre team och nyttjat SCRUM.

P1, P2 och P3 upplevde en kraftigt sänkt mängd teknisk skuld i sina utvecklingsroller. Det berodde bland annat på att arbetsgrupperna stod närmare koden och agerade med mycket mer autonomi, eller för att grupperna var mindre hierarkibundna vilket öppnade upp för egna beslut. De kunde därmed fokusera allt mer på att producera kvalité i sina arbeten då det underlättade för automation och testning i framtiden.

P2 och P3 uttrycker tydligt att de driver en testdriven utveckling medan P1 och P4 antyder på att stort fokus på tester, men är inte tydliga med när i processen de sker. De är åtminstone överens om att testning är en stor del av att upprätta automationen, och få ett starkt arbetsflöde.

P4 kunde även påvisa att flera projekt som denne arbetat i varit väldigt framgångsrika vid implementation av DevOps, där arbetsgrupper kunnat släppa flera stycken leveranser under en och samma vecka utan incidenter och på kortare tid, vilket var en tydlig förbättring från möjligheten att göra det endast fyra gånger per år, med en hög sannolikhet för fel.

5. Diskussion

I detta kapitel ämnar vi att analysera resultaten från vår empiriska undersökning och jämföra dem mot de farhågor och problematik som yttrats i tidigare forskning som vi tagit del av. Diskussionen har vi kategoriserat efter CAMS, likt hur resultaten presenterades.

Undersökningen som vi genomförde grundades i att försöka utreda ifall problematik som DevOps ämnat lösa, har lösts, och om problematik i DevOps som kunnat pekas ut dykt upp i organisationerna som vi intervjuat. Vår undersökning har därmed varit inriktad på just problematik, där frågor sökt negativa toner, men fått svar som nyanserats av den framgång som upplevts. Våra respondenter uttryckte problematik, som var både ny och som tidigare identifierats, men också att de upplevde framgångar i sina arbetsroller efter att DevOps implementerats. DevOps stora brister är det höjda kompetenskravet som läggs på utvecklare och driftpersonal. De är specialist roller som tidigare har vuxit fram av en anledning för att de är komplexa tekniker i system. Men för att få ett heltäckande system av kvalité så tillför tvärfunktionell kompetens i alla led stora vinster för att personal har liknande nivåer av branschspråkförståelse som underlättar att nå samma övergripande mål. Det är de DevOps försöker sätta för hela IT-avdelningen, en gemensamhet och kollektivt tillvägagångssätt som angriper arbetet tillsammans.

5.1 Tolkning

Vad vi som författare trodde skulle vara störst problematik var att DevOps saknade tydlig definition, då flera författare noterade att en definition för begreppet saknades och för att det beskrivs som ett problem i den litteratur vi funnit (Hütterman, 2012). Vi kopplade avsaknaden av definition även till andra möjliga problem som att DevOps upplevdes som en trend, eller att det antyddes vara ett “Buzzword”, och därav mötte motstånd (Kalliosaari et.al, 2016; Smeds et.al, 2015). Detta var dock en felaktig uppfattning, då respondenterna uttryckte att deras utvecklare oftast var yngre individer som var väldigt öppna för nya arbetssätt och agil utveckling, och de implementerade kärnprinciperna i DevOps trots avsaknad av tydlig definition. Men varje respondent baserade DevOps definition utifrån den organisation de arbetar i. I dessa fall verkar det som att DevOps avsaknad av tydlig definition gav rum för att själva definiera och implementera DevOps på organisationen, så att organisationen inte gjordes om för DevOps. Detta verkar ha agerat i DevOps fördel, där definitionen om vad som implementerats är DevOps inte behöver tas, utan fokus hamnar på dess kärnvärden.

Hur organisationerna arbetade i DevOps är av stort intresse då det formade respondenternas bild av DevOps. Vi hade tre respondenter som arbetade i utvecklingsroller, och hade den mest enformiga definitionen av DevOps, med svar som var nästintill ordagrant likadana. Detta berodde på att de inte hade någon direkt kontakt med ett arbetslag för drift längre för att de arbetade mot molntjänster istället. Detta var förunderligt, då ingen litteratur som vi stött på innan intervjuernas gång diskuterat detta förekommande och vi fick upp ögonen först efter intervjuerna att fenomenet dubbats som “NoOps” för att DevOps blir mycket mer utvecklingscentrerat och driften outsourcas till molnet. Vi ser på detta som att det är en produkt av just molntjänsternas

ökande popularitet, men också att de tekniska verktyg som har utvecklats för att höja kvalitén och testbarheten av kod börjat fasa ut behovet av specialiserade arbetsroller som sköter driftsättning. Men att de specialiserade rollerna fasas ut är nog inte hela sanningen, då kompetensen som de bidrog med är högst relevant till automation och byggande av system. Vad vi har observerat, och tror har hänt, är snarare att scopet för utvecklarrollen har breddats från att fokusera på att koda och testa, till att slutföra hela processen genom att även paketera och leverera sin kod till molnet så att det kan användas på servrarna. Detta kan antyda på att de som tidigare arbetat uteslutande med driftsättning assimilerats in i utvecklarrollen.

5.2 Kultur

5.2.1 Bristande stöd

Förekomsten av bristande stöd i den form som Kalliosaari et.al (2016) identifierat kunna existera hos starka kulturer och svaga organisationer var låg. Där starka kulturers motstånd mot förändring är en typ av brist på stöd, men även en svag organisations avsaknad av engagemang för förändring är en annan typ av brist på stöd, hamnar båda på motsatta sidor av ett spektrum. Respondenterna hade olika fokus och upplevelser.

Bristen på stöd som kom från ledningen, var i en kostnadsfråga där vår respondent i rollen som IT-konsult behövde rättfärdiga sin position och DevOps fokus där effektivitet bygger på automation av alla aktiviteter som stödjer det för långsiktig effektivitet. Detta kräver en märkbar investering i utvecklingstid kortsiktigt, och eventuellt investeringar i ny arkitektur. Bristen på stöd kan knytas an till tidigare problematik i uppfattningen om att DevOps är en trend, men då respondenten försäkrade oss om att ledning och chefer var ombord och stödde implementationen, så är det inte en korrekt slutsats i detta fall. De ifrågasatte investeringen som de uppfattade som kostsam, och sökte efter en försäkran att det skulle effektivisera arbetet. Det är problematik som funnits sen urminnes tider i projekt och är knappast inom ramen för vad DevOps kan tänkas lösa.

Respondenten från vårt största företag med globala kontor behövde hantera förväntningarna på utvecklarna korrekt i förhållande till arbetsrätt länder emellan. Detta växte fram ur okunnighet, och att DevOps innebar att ett ökat ansvar hamnade på utvecklarna. Problemets omfattning var lågt och var enligt respondenten lätt att lösa. DevOps introducerade problemet genom att utöka ansvaret för utvecklare. Ingen av författarna tog upp ökningen av ansvar som ett möjligt problem, och vi förväntade oss det därav inte, men att internationella skillnader finns förvånar oss knappast, eller att förväntningar måste hanteras på internationell nivå.

5.2.2 Motstånd

Problematiken kring motstånd som identifierats i litteraturen handlade huvudsakligen om att DevOps är en trend “buzzword”, och till viss del att DevOps innebär en förändring i arbetssättet som påverkar både kund och företaget (Smeds et.al, 2015; Kalliosaari et.al, 2016; Ghantous & Gill, 2017; Wiedemann et.al, 2018). Vad vi kunde utläsa så har termen hunnit mogna till den grad att våra respondenter inte haft någon negativ inställning till begreppet, vilket inte heller

uppfattades ha skett hos deras kunder heller. Motståndet vi fick förklarat för oss av respondenterna handlade istället om en motvilja bland utvecklare att skriva tester. Något så basalt, och nästintill barnsligt, som knappt kan antas vara problematik. För arbetet som vi uppfattade det genomförs fortfarande men med en sur min. Det fanns därmed en attityd att tester inte var riktig kod eller tillräckligt fräck för deras kompetens.

5.2.3 Överbelastning

Samtliga respondenter uttrycker att arbetsbördan blivit högre till följd av det nya ansvaret utvecklarna fått. Utöver de nya uppgifterna associerade med driften, som i mångt och mycket hamnat hos utvecklarna bland våra respondenter, så är automation och arbetet därtill mycket tidskrävande och kräver högre kompetens från berörda parter. Något vi upptäckt skiljer sig från Smeds et.al (2015) fynd var en avsaknad på den negativa synen på den nya belastningen. Trots att respondenterna nämner att det är mer arbete så är det ingen negativ klang kring uttalandena. Vi får även höra att med det nya arbetssättet så har besluten trillat ner till utvecklarna själva i stor utsträckning, något som krävs för att kunna upprätta den arbetsform, med continuous integration och continuous deployment, som DevOps eftersträvar, och som nämns av Wiedemann et.al (2018). Detta kan ses som att arbetet trots allt fungerar bra och att något bristande intresse av "andra sidan" inte heller existerar. Tonen som sätts kring arbetet är trots allt positiv och samtliga uttrycker det som att arbetet välkomnas vilket indirekt även inneburit en kompetensutveckling för samtliga.

5.3 Automation

5.3.1 Val av verktyg

Valet av verktyg är viktigt för att så effektivt som möjligt kunna bygga upp den infrastruktur som krävs för att bygga dina pipelines, etablera det kontinuerliga arbetet och i den mån det går skapa de miljöer som krävs för både tester och kvalitetssäkring. Svårigheterna kring verktygen som vi själva ser hos respondenterna är lite annorlunda gentemot litteraturen. Hos våra respondenter, har samtliga kommit i kontakt med problem relaterat till verktyg i varierande grad eller som i ett av fallen, inte alls. Ghantuos och Gill (2017) tar upp det som att mängden verktyg kommer försvåra valet av rätt verktyg för din verksamhet. Vilket hos en av respondenterna upplevts i motsatt anda, där uttrycktes en brist på tillgängliga verktyg och därmed svårigheter att välja. Något vi tror kan ha att göra med att hen varit aktiv inom offentlig sektor och inte privat. Offentlig sektor innebär en ökad byråkrati och hanterade i respondentens fall mycket känsliga data. Vilket kan ha inneburit en rad olika restriktioner vilket i sin tur minskat utbudet av vad som kan ansetts vara godkända verktyg.

Mognadsgraden kan även spela in precis som det gjort hos de två som i dagsläget inte upplevt några svårigheter kring verktyg. Detta grundade sig mycket i deras organisationers tidigare arbete och att de redan haft stark etablerad IT-infrastruktur som faciliterat en lättare adaption av DevOps-verktyg.

5.3.2 Legacy system

Hur väl kommer äldre system, funktioner och monolitiska byggen sätta stopp för det snabba arbetsflödet agila och i förlängning DevOps ämnar åstadkomma? Problemet lyfts som både ett traditionellt men även aktuellt problem just för DevOps. Något som kan tänkas vara svårt att komma ifrån överhuvudtaget. Konstigt nog är det bara två av våra respondenter som tar upp det som ett problem, i detta fall även något som lyckats åtgärdas. Precis som Eficode (2019) redogör för så ska DevOps säkra att legacy system och funktioner tas vara på i den mån det går, precis som våra respondenter gjort. Introduktionen av mikrotjänster hos en av respondenterna tjänade till att stycka upp en tidigare monolit till en mängd mindre tjänster som kunde både utvecklas, köras och underhållas parallellt. I det andra fallet så gjordes en delvis DevOps implementation något som Wiedemann et.al (2018) identifierat som ett sätt att kringgå problemet med system eller produkter som inte är fullständigt kompatibla med varandra.

5.3.3 Tid

För att skapa de flöde som ska finnas i samband med continuous integration och deployment så ansåg våra respondenter att det var en väldigt krävande process att investera tid i automation och tester som utan manuell input förde byggen vidare i arbetsprocessen. De ansåg att det var väldigt krävande på kort sikt, men att de vann mycket effektivitet på lång sikt. De gick därmed miste om viktig utvecklingstid för att automatisera och bygga tester, men ansåg också att det var lättare att bygga testning och automation tidigt i ett projekt än senare. Denna typ av tillvägagångssätt, där utvecklingen är mestadels testdriven, innebär som utlästes från våra respondenter, en fråga om kostnad i form av tid.

En avsaknad av tydliga definitioner och gemensamt språk, problematik som lyfts av både Hütterman (2012) och Colavita (2016), anser vi mynnar ut i en kostnad i form av tid. Missförstånd och extra arbete var något som uppstod hos P2 när individer definierade termen “Klar” olika. Det blev häpna miner och extra arbete som behövde planeras in, av den anledning att leveranser av funktioner var i olika stadier men ansågs färdigställda av sina skapare, och P2 påpekade att till följd blev en del arbete sällan klart vid utsatt tid. Där vi kan peka på att en bidragande faktor till att arbetet går över tiden, berodde att tidsbrister i projekt inte flaggades tidigt nog i processen till följd av varierande uppfattning av vad “Klar” innebar. En utvecklares definition kunde vara att färdigställa funktioner, komplett med enhetstestning och redo för leverans, medan en annan endast genomfört enhetstestning, om ens det, och därför behövde mer tid efter att den färdigställt sitt projekt enligt sin definition.

5.3.4 Externt motstånd

En viss mognad, gällande både organisationen och IT, måste finnas för att skapa en grund att adaptera DevOps på. Även om arbetet fungerar väl för företagen som adapterat DevOps principerna, kan de innebära friktion i relation till externa aktörer. Kunder lyfts av Smeds et.al (2015) som ett motstånd till implementationen av DevOps, eftersom det nya sättet att arbeta på kanske inte är kompatibelt med kundens sätt att arbeta. Problemet stöttes på hos P3 och P4, ena var tydlig med att uttrycka att kundernas mognad för deras egna arbetsprocesser var varierande och kunde liknas med ett lotteri, där det ibland funka och ibland inte. De ska noteras att

problemet effektivt lösts genom att nyttja så kallade coacher, vars uppgift var att utbilda kunderna inför företagets inträde. Ett effektivt sätt att kringgå och indirekt hjälpa kunderna i detta fall att sadla om till ett mer agilt arbetssätt. P4 upplevelse kan liknas med vad coacherna hade för uppgift, hen var nämligen med i team som hade som uppgift att införa DevOps hos kunderna, där både arbetssätten var utdaterade och systemen föråldrade. Motståndet som manifesterade sig var då främst gällande det nya sättet att bedriva utvecklingen, där respondenten visar på motgångar och arbete för att få igenom sina förändringar.

5.4 Mätning

5.4.1 Legacy

P1 uttryckte att det var svårt att mäta deras system till följd av att de hade en stor flora av legacy system som gjorde det väldigt svårt att ta fram några rimliga mätetal på systemens prestanda i sin helhet. Men det var problematik som alltmer försvann allt eftersom de började flytta tjänster till i en molntjänst där de bombarderades med möjliga nyckeltal för att analysera sina applikationer. De upplevde därmed liknande problematik som P2 och P3, gällande systems komplexitet, men kunde inte mäta något som var verklighetsförankrat enbart till följd av deras legacy system, eller av en ovilja att vidare utforska möjligheterna när de ändå skulle emigrera till molntjänster. Men de närmar sig därmed samma problematik som P2 och P3 upplever i sitt utvecklingsarbete där även de arbetar främst mot molntjänster, men har svårt att välja och följa nyckeltal bland alla möjliga som finns tillgängliga.

5.4.2 Nyckeltal

Valet av nyckeltal ansågs svårt av flera respondenter, där P1 hade svårigheter till följd av systemens begränsningar så upplevde P2 och P3 en svårighet att välja bland de mängder med nyckeltal de hade tillgång till, och att mängden system förvärrade detta. Vi förstod det som att de ofta gick på svarstid för sina applikationer som ett mätetal för att avgöra ifall systemet agerade som förväntat och om det kunde gå snabbare. Men i DevOps arbetade de med flera mindre system som kunde påverka varandra och det blev svårare att diagnostisera vart problematiken i systemen var till följd av den ökande komplexiteten. Även om de implementerade kod som lyssnade och slog larm när en komponent i systemet inte svarade som den skulle så kunde det mycket väl bero på ett fel i ett underliggande system, därav kunde fel utvecklare tillkallas för att fixa ett problem. En förändring från den monolitiska arkitekturen, men en helt klar förbättring då endast en liten del av systemen någonsin påverkades.

5.4.3 Ingen mätning på gruppnivå

Nästan alla våra respondenter genomförde någon typ av mätning i DevOps, och flera fokuserade sin mätning på att höja kvalitén på systemet genom prestandamätning av exempelvis responstiderna i de aktiva systemen, men hade inget fokus på arbetsgruppernas prestanda i form av ledtider eller leveranser som de uttryckte det. De respondenterna som inte fokuserade på arbetsgruppens prestation var däremot mycket måna om att de fokuserade på deras relation till

individerna och därav visste vad de kunde förvänta sig från varje individ. Vi ser därmed att inte fokusera på kvantifierade tal över prestanda kan ha varit fördelaktigt hos våra respondenter, då Hütterman (2012) varnade för att inte fokusera på mätetal som kvantifierar arbetsprestanda. Arbetsgruppernas fokus blev istället relationerna människor emellan, och hur det påverkade kvalitén i systemen. Vi utläser det som ett resultat av att arbetsgrupperna inte var större än 9 anställda, men också att respondenterna beskrev avdelningen som väldigt horisontell och hierarkilös, och att de därav fokuserade på sina medmänniskor. Detta kan vi ställa i kontrast till hur P2 reagerade i ett internationellt samarbete i sin organisation och valde att gå ifrån ett uppdrag då samarbetet genomfördes med en mer vertikal organisationskultur. Fokuset som DevOps placerar på mätningen av arbetsgruppers prestation verkar därmed vara grundat från en mer vertikal organisation i åtanke med tydligare chefsroller till skillnad från våra respondenters arbetsplatser i svenska organisationer.

Den respondent som faktiskt mätte arbetsgruppernas prestanda gjorde det i uppdrag som IT-konsult. Det var då möjligt att visa de framsteg som gjordes under ett uppdrag, vilket är varför Hütterman (2012) lägger en väsentlig vikt på mätning av förändringars effekt. Varför just P1, P2 och P3 inte fokuserar på mätning av hur en arbetsgrupp presterar kan även bero på det faktum att de inte sitter på en strategisk nivå där den översikten bidrar med ett högre värde. Men deras avsaknad av nyckeltal för att mäta arbetsgruppers prestanda verkade inte förhindra deras arbete på förbättringar, utan de hade istället fokuset på människan och gick efter upplevd förbättring.

Valet av nyckeltal är svårt när det gäller arbetsgrupper, för det krävs ett eller flera nyckeltal som kompletterar varandra för att avgöra en grupps prestanda och välmående (Hütterman, 2012). Det är även svårt att mäta en individs prestanda av flera anledningar, då vi som människor är så komplexa. Vi bygger upp vanor snabbt på både ont och gott, för dessa vanor kan mycket väl vara fördelaktiga, men är också lätta att störa och kan påverka arbetsprestation. Stress och sömnbrist är lysande exempel som påverkar psykisk och fysisk prestation på en väldigt märkbar nivå, och kan leda till en försämrad arbetsprestation som reflekteras i nyckeltal för en arbetsgrupps prestation. Med ett fokus på nyckeltalen kan sådana mänskliga faktorer förbises i jakten på det kvantifierbara, därav påpekar Ravichandran et.al (2016) att vad som bör mätas i DevOps är oklart. Men med ett fokus på människorna så kan påverkan och orsak vara tydligare. Våra respondenter med fokus på människorna var mer måna om arbetsplatsens upplevda kvalité, och sina medmänniskors välmående, för att avgöra arbetskvalitén än dess prestation i siffror, då de upplevde det som det bättre valet. Att balansen mellan nyckeltal och människor vägde så tungt i människors fördel var förvånansvärt, då vi misstänkte att nyckeltal var ett underliggande stöd för att diagnostisera arbetsgrupperna, men detta var en felaktig uppfattning. Den oklara definitionen som DevOps har av mätning enligt har Ravichandran et.al (2016), agerar i våra respondenters fördel, där de fortfarande kan implementera DevOps men göra det på sitt sätt.

5.5 Delning

Delning visade sig vara svårt att särskilja från kulturen i DevOps vid intervjutillfällena, då den är så djupt rotad i principen och är ämnad för att visa på den iterativa naturen som ska upplevas i DevOps. Vi utläser också att det var problematiskt att förknippa med våra respondenter då flera aktivt arbetade i endast utvecklingsgrupper mot molntjänster, där ingen upplärning av nya

verktyg behövde ske som bryggade drift och utveckling. Molntjänsterna har utan tvekan bidragit till att många arbetsgrupper kunde arbeta som uteslutande utvecklare och inte vara i kontakt med någon driftpersonal där ett kompetensutbyte och språk behövde normaliseras mellan funktionerna, vilket gör att delningsprincipen i CAMS får ett mycket mindre fokus.

5.5.1 Människan

Två respondenter återräknade att det upplevts enstaka fall av individer som inte lämpade sig lika mycket för vissa typer av arbete, exempelvis parprogrammering, och istället föredrog en arbetsplats som var bättre anpassad för egenhändigt arbete. Det var inte av anledning att vara emot någon DevOps princip, då individerna fortfarande var väldigt delaktiga i samarbete för övrigt, men de föredrog att arbeta på egen hand. Wiedemann et.al (2018) noterar att det är viktigt att ha med alla intressenter i ett förändringsarbete, men det borde vara applicerbart även i detta läge, där det är viktigt att alla i arbetsgrupperna kan vara med i det faktiska DevOps arbetet, vilket individerna var, även om de motsatte sig andra introducerade koncept som lätt kunde planera runt av managers.

5.5.2 Arbetsform

Rörande delning och kompetensutbyte så anmärkte P2 att arbetsformen, specifikt konsultrollen, kan ha en betydelse för hur väl delning genomförs anställda emellan. Inget som uppmärksammats i tidigare litteratur men som vi ansåg vara värt att notera. Rollens natur i att tid kontraktuellt planeras för kräver en förståelse för vad detta kan innebära om det inte hanteras väl. Även om företag har riktlinjer eller ledord som förespråkar exempelvis att alla ska hjälpa varandra, så betyder det inte att de garanterat kommer ske i praktiken. En god översikt av vilka typer av anställningsformer som finns i teamet kan därmed vara av vikt för att framtida planering och allokering av resurser ska främja delningen ytterligare.

5.5.3 Organisationen

Typ av organisation har inte identifierats som ett problem i tidigare litteratur men dök upp hos en av våra respondenter. Problemet som vi tolkat de har att göra med att organisationerna som respondenten arbetat hos varit aktiva i offentliga sektorn. Det har uppstått kulturella och strukturella hinder till följd av verksamhetens natur och den känsliga information som hanteras. Möjligheten att etablera en så kallad feedback-loop (Highsmith & Cockburn, 2001) mellan utvecklarna och slutanvändarna har därmed inte varit möjligt. Respondenten anmärkte på att slutanvändarna enbart kunde kontakta någon typ av systemförvaltare och att utvecklarna inte kunde få den informationen direkt, vilket Humble & Molesky (2011) menar på leder till långa ledtider. P4 uttryckte att problemet löste sig då de hade en gruppmedlem som var starkt etablerad i verksamheten och därmed indirekt kunde få den information de behövde från slutanvändarna. Men problemet är en verklighet och feedback är en viktig del för att säkerställa att scopet för utvecklingen är korrekt kalibrerat och att arbetet inte avviker från de ursprungliga kravet och behovet. Det är väldigt lätt att bli blind för sitt eget arbete, och användares vägledning och återkoppling är en viktig del då de bidrar med tankar från ett utomstående perspektiv. Som en själv kan ha bortsett från i mån om att genomföra andra features etc.

5.5.4 Mentalitet

Vi fann att endast en av våra respondenter var aktiv i vad som beskrivits som traditionella DevOps funktioner, vilket förvånade oss, och att problematik som skulle uppstå för de andra respondenterna eliminerades i sin helhet genom att de arbetade i molntjänster. Hos respondenten som arbetade i traditionella DevOps arbeten, med både en drift- och en utvecklingsfunktion, så var arbetsfunktionerna fortfarande inställda på att arbeta i miljöer som isolerade individerna. De var samtidigt väldigt öppna för samarbete, men det fanns tydliga fysiska barriärer som separerade personliga utrymme och kunde upplevas som motsägelsefullt. Vi kan utifrån de data vi samlat in se en förminskning eller ren avsaknad av den silomentalitet litteraturen identifierat som en av de bakomliggande faktorerna till DevOps påbörjan.

5.6 Positiva utfall

Alla respondenter upplevde en förbättrad kodkvalité och system hälsa efter implementationen av DevOps. Överlag så upplevdes en lägre teknisk skuld hos våra respondenter och därav även ett större förtroende för utvecklarnas kompetens och arbete. Leveranser utfördes oftare, med mindre incidenter och downtime än vad som tidigare varit möjligt.

5.6.1 Teknisk skuld

Både P2 och P3 upplevde en lägre grad av teknisk skuld, och påpekade att detta berodde på att utvecklarna fick agera allt mer autonomt, eller i en hierarki lös arbetsplats, efter en DevOps implementation då de fått ett större förtroende från ledning, och i de fall som en driftsfunktion finns även dom. Detta är i liknande ton med vad som förespråkas i SCRUM, där förtroende finns för individens kapacitet att producera arbete av god kvalité och det finns avstämningar som kontrollerar arbetet. I DevOps fall är dessa avstämningar på kvalité de automatiserade testerna, men då flera respondenter använde sig av SCRUM så förekom även avstämningar på av människor vid varje sprint. Eftersom DevOps inte uppfattats som ett eget verktyg hos våra respondenter, utan mer som ett komplement och fortsättning på det agila arbetet, så säkerställs arbetet på flera nivåer. Automationen höjer den absoluta kodkvalitén medan exempelvis SCRUM-sprintar hjälper till att arbetet är fortsatt rätt riktat och att tid inte läggs på att utveckla onödig funktionalitet och uppfinner hjulet igen.

6. Slutsats

DevOps har sin plats i IT/IS-världen, och med all rätt. Som det ser ut så har DevOps framgångsrikt reducerat problematik, och konceptet verkar ha mognat hos våra respondenter. En majoritet hade samma definition, det kan antyda på en viss spridning, men också att det formas efter organisationen, inte tvärtom. DevOps tydliga fokus på kvalité av kod och satsningar på automation för effektivitet går ej förlorat, även om mätningen hamnar i skymundan så är grupperna små och intima med människorna i fokus. Till följd av de mindre autonoma grupperna och agila arbetssätt verkar kultur och delning i DevOps framkomma naturligt. Tre av våra respondenter är antingen i molnet eller är på väg in i molnet, och säger att “utvecklare gör allt”. Så det är klart att problematik försvinner när en funktion inom företaget ersätts. Molnet är en del i receptet som gjort DevOps framgångsrikt hos de respondenterna, och är en krycka i de fallen. Men vi såg att vår IT-konsult som haft mest varierad DevOps erfarenhet kunnat påvisa dess framgång hos flera av sina kunder där respondenten implementerat de renodlade principerna i organisationerna. DevOps är inte en uppsättning av principer som implementeras för sig själv, för vi ser att alla respondenter framgångsrikt använder sig av både SCRUM och DevOps tillsammans för att säkerställa koden och utvecklingsarbetet. DevOps är inte “The End all and Be all”, utan det är ytterligare ett steg i riktningen mot bättre mjukvara och en hälsosammare relation mellan utveckling och människor.

DevOps is a boon!

7. Bilagor

7.1 Intervju 1

Vad är din befattning idag?

P1: Jag är utvecklare, arkitekt. Främst utvecklare, men arkitekt emellanåt.

Vad är din erfarenhet av DevOps, eller hur har du kommit i kontakt med det?

P1: Företaget har idag inga dedikerade resurser till Ops-delen och har ingen som jobbar med att sätta upp saker förutom oss då, utvecklarna gör allt. Men ett undantag är att vi har en extern firma som sätter upp servrar och så, men vi har en flytt på gång till AWS och då gör vi allt själva. DevOps är lite luddigt, som ni kanske märkt.

Hur skulle du definiera DevOps?

P1: Ett sätt att låta utvecklare ta del av drift och av det dom utvecklar, väldigt generellt. Men det innefattar att man tittar på drift och routing, vilka miljöer man ska använda.

Så ni försöker bli mer tvärfunktionella då och man har mer kompetens och har ops-delen i åtanke?

P1: Ah, det kan man säga. Det påverkar inte koden så mycket, men det har påverkat kriterierna för när koden anses vara klar för produktion. Ansvaret för driften har skiftats till oss som utvecklare.

Då kommer vi in på vår huvudfråga, och den är vad för problematik har ni stött på i DevOps?

P1: Det svåra skulle nog vara att sätta sig in i världar som man inte har stött på tidigare som utvecklare, då man bara skulle kunna kod tidigare, men nu tvingas kunna nätverk som inte ingår i en klassisk utvecklarroll. Detta inkluderar även databas implementation och så.

Hur många utvecklare är ni?

P1: Vi är nio utvecklare, uppdelade i två team så vi jobbar väldigt mycket tillsammans.

När det kommer då till er utveckling, är ni självlärda då?

P1: Vi är mycket självlärda och lär oss utefter som vi behöver det. Exempelvis hur vi använder Docker och containiserar sina applikationer så att vi kan sätta in dom i miljöer som ser likadana ut, exempelvis AWS. Docker och hela container-lagret är en klassisk del av DevOps där det ska vara lätt att implementera och testa miljön. Applikationer ska paketeras av utvecklare.

Hur är transparensen mellan er utvecklare? Märker ni av att någon sitter på mer kunskap och blir en självutnämnd expert?

P1: Tidigare så var det så, där vi tidigare jobbat rätt så isolerat och vissa personer haft specialområden. Sen så introducerade vi SCRUM och vi började arbeta och lösa uppgifterna mer som team, och vi införde ett arbetssätt där man tar nästa uppgift på listan och fick en ganska strikt kompetensspridning genom det, så vi har aktivt jobbat bort personberoende. Men "experterna" är mer insatta i vissa delar, och med SCRUM behöver de experterna hjälpa till mer i uppgifterna så att kunskaperna sprids.

Ni har väl haft en ganska långtgående transformation, har ni sett att det blivit någon omfördelning av ansvar och omstrukturering av titlar vart folk ska vända sig?

P1: Det skulle jag inte säga. Vi försöker att hela tiden dokumentera vad vi gör så att alla har tillgång till dokumentation istället för att någon exempelvis har en specifik DevOps-roll.

Det har mer handlat om att göra all information tillgänglig och på så sätt involvera alla?

P1: Det skulle man kunna säga, ja.

Så detta är en fördel som ni som en liten arbetsgrupp haft då? Eftersom ni är en mindre och intimare grupp.

P1: Precis, det hade nog inte fungerat i en större organisationer.

Det ni har liknar NoOps, där mestadels av driften automatiseras och ni som Dev hanterar resterande traditionell Ops. Har ni haft några problem att hitta verktyg och system som ni ska använda för att ta över Ops?

P1: Nej, vi har haft en liten flora av system som vi haft tidigare, och vår externa leverantör av legacysystem har använt Ansibel som ger oss ganska bra kontroll för hur servrarna ska sättas upp. Men sen när man kör det i AWS så har de extremt mycket verktyg för att man ska få till en bra miljö själv. Så att utmaningen där är hur man ska navigera AWS-världen.

Så en av utmaningarna, har varit att ta in AWS miljön och välja vad som passar er?

P1: Ah precis, det finns hur mycket som helst att välja på där. Man kan helt enkelt klicka runt på GUI:T och välja vad man vill ha, men vi har valt att köra vår infrastruktur som kod (IAC) så att vi scriptar allt och versionshanterar miljöerna på så vis. Då kan vi ha flera olika miljöer att testa mot, bland annat vår produktionsmiljö.

IAC anses vara en stor del av automationsdelen i DevOps. Där vi utgår från CAMS-ramverket så är IAC väldigt viktigt i kontexten av Automation för att underlätta arbetet.

Men då går vi in på mätningen. Har ni mätt er prestanda som utvecklare innan DevOps, och nu efter att ni börjat arbeta som DevOps? Exempelvis felfrekvens, deliveries, errorhantering.

P1: Det har vi faktiskt inte gjort. Vi har inte den typen av system som lämpar sig för det. Vi har inte riktigt den typen av system som har tydliga leverans, utan vi har en flora av små system som är ganska svårt att göra den typen av mätningar på. Känslan är som så att sen vi börjat tänka mer strukturerat har vi fått bättre kontroll, speciellt när vi tänker i termer av testbarhet och testtäckning, så är man också mer benägen av att refaktorera och ändra i kod för att man inte behöver vara lika osäker då för att förstöra något.

Så det går fortare ut i produktionsmiljön än tidigare?

P1: Det skulle jag säga. Vi har inte haft så affärskritiska system tidigare, och det har inte varit någon panik om man råkat driftsätta en bugg i produktion. Så känslan har varit att det inte varit hela världen om man råkar göra något fel. Men vi har börjat få ett mer kvalitetstänk och försöka minimera felen när de uppstår. Jag har inga direkta värden och rapporter, men känslan är att kvalitén blivit bättre.

Så ett hinder för er att kunna mäta har varit att ni haft en stor flora av system och därmed haft svårighet att hitta gemensamma mätetal, och att ni inte haft något direkt behov av det?

P1: Korrekt.

Nu när ni fört in det här DevOps-tänket, har det varit en process att få med alla medarbetare? Hur introducerades DevOps?

P1: Det har alltid varit närvarande då vi i teamet skött allt själv, men det har tidigare varit några få personer som varit bra på specifika delar, och nu så har kompetensen fördelats bättre sen vi introducerat tänket att alla kan göra allt. De flesta har varit med på banan i att följa med koden hela vägen.

Nu så kommer vi in på sista delen här, gällande delning. Sker det informationsutbyte, återkoppling, feedback i arbetsgrupperna?

P1: Delningsmässigt så har vi inbyggt i vår arbetsgång att innan vi driftar ett ärende så sker en kodgranskning så att det kan ske ett kompetensutbyte. Sen så har vi en intern wiki, confluence. Vad gäller chattsystem så har vi Microsoft Teams, det är inte mycket aktivitet där, utan vi sitter så nära varandra att vi pratar oftast sinsemellan och det tycker jag är en bra fördel.

Har ni effektiviserat något med Lean, där ni försökt kapa överflödiga arbetsprocesser som inte tillför värde?

P1: Vi gör som så att vi kör Scrum-processen där vi har ett antal produktägare som väljer vad som ska prioriteras och att vem som helst kan skapa ärenden i en stor logg. Sen så har vi utvecklare rätt så mycket makt att påverka, tycka och tänka vad vi borde göra. Så det är både och, där de har ansvaret, men vi har mycket säg i vad som prioriteras. Så inte att något skärs bort som är något vi skulle vilja ha, utan att arbetsprocessen har blivit mer strukturerad och transparent så att vi vet vad alla håller på med.

Det blir en effektivisering med Scrum då?
P1: Ja. Det är en sorts riskminimering, istället för att man håller på med något jättestort som kanske blir fel, så kör vi i tre veckors sprintar och kör ett demo för att få input om vi håller på med rätt saker. Riskminimering har skett, som blivit en typ av effektivisering så att ingen gör något onödigt för länge.

När du nämnde paketeringen av era applikationer i Docker, och IAC, så undrar vi om det sänkte behovet av kodgranskning då ni nu enklare kan testa mot en produktionsmiljö?

P1: Nej, utan kodgranskningen är dels till för att den som granskar ska få en liten kompetenshöjning och för att säkerställa att inget är konstigt med koden. Men med granskningskriterierna är det att koden skall vara testad, och där kan man säga att det hänger med på köpet. Men just Docker som så har i sig inte påverkat kvalitén.

Har paketeringen påverkat testningen och automatiseringen?

P1: Ja, det har det. Vi testar på lite olika nivåer, i basen har vi unit-tester som testar koden och sen ovanför det så har vi integrationstester som testar kopplingarna till kringliggande system, och sen ovanpå det så kör vi ett fåtal end-to-end tester som testar ett system från början till slut och det är typiskt där som Docker kommer in i att man kan använda IAC och dra upp hela miljöer, driftsätta containers och på så sätt har det förenklat det lite grann att man kan få igång en produktionsmiljö som man kan köra tester mot. Så indirekt har det påverkat tester och kvalité.

Kompletterande email svar:

Sen ni började implementera ett DevOps tänk, har du upplevt att kulturen kring hur ni samarbetar i arbetslaget gått mot en förbättring?

P1: Jag har svårt att säga att det finns en precis tidpunkt då vi införde DevOps och att man kan säga hur det var före kontra efter. Vi har i någon mening alltid varit tvungna att hantera DevOps eftersom vi är ganska få på vårt företag och aldrig haft dedikerade resurser till Ops-delen av DevOps. Med de sagt har vi en annan före/efter och det är införandet av SCRUM och mer agilt arbetssätt. Det har absolut förbättrat samarbetet, stärkt team-känslan och gett oss tydliga definitioner när ett ärende anses vara klart (testning, kodgranskning mm), och det har i sin tur gett kod som är mer robust och lättare att underhålla.

7.2 Intervju 2

Vad är din befattning och vad har du haft för kontakt med DevOps?

P2: Just nu är jag projektledare på teknisk driftsättning och det betyder att jag ser till att det vi bygger kommer ut i produktion. Det som är intressant för er är vad jag haft för uppdrag innan detta, jag var med att utveckla den DevOps lösning som företaget har i cloudet. Det kallas en pipeline (Deployment pipeline) där man bygger, testar och deployar. Så i detta arbete var vi ett gäng i USA och ett litet här i Malmö, totalt över 100 personer som var med. Sen har jag varit konsult många år både före och efter, så jag har varit med om både projekt som gått bra och mindre bra.

Vad är din tolkning av DevOps och vad inkluderar du i dess kontext?

P2: De påverkar de som jobbar med det framförallt genom att man inte längre har ett team som utvecklar och ett team som har hand om driften, utan utvecklarna tar mer eller mindre hand om allt. Det finns ju självklart fortfarande support så att utvecklarna inte plockas in för minsta lilla problem. Så om kunderna skulle skicka in en ticket eller liknande så är det ju supportpersonalen som tar hand om det. Det som är DevOps för utvecklarna är när det trillat ner en bit och man ser att det är ett fel i deras system som de själva byggt. Det kan ju vara väldigt många mikrotjänster som interagerar och om en databas är nere så är det ingen mening med att tala med utvecklarna utan då måste man se till att databasen kommer upp igen och förhoppningsvis så startar tjänsterna som utvecklarna har gjort. Så DevOps för mig är ett arbetssätt där utvecklarna har hand om och kan ha koll på driften. Det blir ju många delar, exempelvis den tekniska biten med att driftsätta, där finns det ju Continuous integration och Continuous delivery, när man skrivit sin kod och levererar den vidare så byggs det automatiskt och testas, i teorin ska utvecklarna själva kunna sköta det och ha tester redan gjorde, så när de trycker på ”ok” så ska ingen annan behöva godkänna deras arbete utan den ska kunna åka direkt in i produktion. Så långt har jag aldrig sett

någon komma, jag tror inte det är speciellt många företag som befinner sig där, att de litar på sina utvecklare tillräckligt mycket för att låta allt åka direkt ut i produktion.

När jag jobbade på cloud så har alla projekt inte bara produktion utan man har sin utvecklingsmiljö, testmiljö osv. så de finns ett antal steg. När vi utvecklade lokalt och fått det att funka så gick det senare vidare till utvecklingsmiljön där det är lite vilda västern. Dit levereras allt som man byggt och där är det inte säkert att allt funkar och att de funkar ihop. Så här upptäcker vi ifall något kanske inte funkar eller har dålig kvalité. Sen när det är klart där så går det vidare till QA, kvalitetssäkring. Så allt vi leverera gick automatiskt in i Dev (utvecklingsmiljön) och när features sen var tillräckligt bra fick vi manuellt trycka på deploy för att lägga över dem på nästa. Sista steget för att gå över till produktion så hade vi alltid möten eftersom vi var väldigt många. Då var vi tvungna att koordinera detta och då var det några gånger i veckan som vi satt stopp för att skicka vidare till QA och produktion. Då fick ingenting skickas in i produktion förens det blivit gemensamt godkänt, det var ett release möte med alla chefer. Den tekniska biten i DevOps handlar ju om att det ska vara ”zero downtime” så man hade flera instanser av samma system så då kunde man deploya och sedan ”peka om” URL:n för att slippa stänga ner systemet.

Upplevdes någon typ av teknisk skuld ”technical debt” i samband med era synkroniserade releaser?

P2: Teknisk skuld är ju jättebesvärligt. För produktchefen vill ha nya features medans utvecklarna inte vill sitta och rodda med saker som inte funkar utan de vill ju rätta till också, så det blir en trade off. I DevOps så uppfattar jag det som att utvecklarna har lite mera kontroll över det här. Man gör de som ska göras och man måste alltid refaktorisera för att se till att det fungerar. Det är teamen själva som ansvarar, vi körde lite som Spotify med produktteam och ”squads”. Då är de självständiga och får bestämma hur de ska jobba och vilka verktyg de ska ha. Så därför upplever jag det som att den tekniska skulden blir mindre i denna typ av upplägg där besluten tas på en lägre nivå och kringgår de långa beslutsleden.

Vad för problematik kring implementationen av DevOps har upplevts utifrån ett kulturellt perspektiv?

P2: Kulturen tycker jag är den svåraste biten. Vi hade t.ex. en teknisk ledare som inte tyckte om automatiserade tester speciellt mycket, så då blir det extremt svårt att få igenom de eftersom teamet måste vilja göra de för att kunna göra det. Så man måste ha folket med sig, generellt sätt så tror jag det är många som gillar agilt och att det ska vara moderna verktyg för folk vill inte sitta och koda i gamla system och rätta buggar. De vill göra nya ”hippa” grejer och på så vis är det hyfsat lätt att få folk att jobba på moderna sätt. Men just testning är det många utvecklare som inte gillar, de vill hellre koda för de tror att testning är tråkigt. Att då göra testdriven

utveckling har jag upplevt som svårt då många utvecklare inte gillar det och många tekniker är inte så extroverta och det blir då även svårt med parprogrammering pga att de inte har den läggningen.

- **Har du upplevt någon typ av ”hjältekultur”?**

P2: Jo vissa personer har haft ”jag är bäst” mentaliteten, men de betyder även att de var extremt dåliga på saker som var utanför deras specifika kompetens. Så när den komponenten kanske inte var aktuell längre så hade hen ju ingenting kvar. Som tur är så är inte alla personer så, men det finns några och det har blivit mindre i och med DevOps.

Men just nu upplever jag det som att problemet vi har idag (DevOps) istället är stress. Tillexempel om teamledaren har SCRUM of SCRUMS, man har sitt eget team med Scrum-möten och sen ska den personen även ansvara för att vara med i SCRUM of SCRUMS, arkitektmöten och det blir ”chokade”, de har svårt att hinna göra sitt jobb i sitt team och en går då miste om information som kanske borde delats.

- **Så ni upplever det som att det är mer omedvetna än medvetna missar?**

P2: Ja.

- **Har du upplevt något internt motstånd i och med införandet av DevOps?**

P2: Det har varit mest på testsidan, där det varit besvärligt. Folk har i efterhand märkt att ”shit vi har inga tester”. Vilket vi då skulle börjat med, det är jättesvårt att bygga upp bra testkedjor när man kommit långt in i utvecklingen. Det är mycket lättare att göra det från början, då har man kanske inte redan valt miljö, vilka typer av testsystem, man har byggt systemet på ett sätt som är lättare att testa. Det är sällan man blir klar vid utsatt tid utan det drar ju oftast över och om man då redan är stressad, blir försenad och ska bygga tester då hinner man ju inte alls. Det är bättre att göra rätt från början.

- **Har ni upplevt något problem med kompetensutveckling nu när driften lagts på utvecklarna?**

P2: Nej, däremot har vi sett tydliga problem och det var nog därför jag och svenskarna slutade i projektet. Man måste ha en bra organisation och kontrakt för ett sånt åtagande. Om det är så att utvecklarna ska ha ansvaret för incidenter då måste man ha organisation för det. Pagerduty är ett system för att schemalägga folk när de ska ha jour så man vet vem man ska prata med när något händer och att man får prata med rätt person. Man måste då se till att utvecklarna vet om vad som krävs och i USA är det andra regler mot vad vi har i Sverige kring arbetsregler. Där borta kan man ”beordra” folk medans här skulle det ex. krävts nya kontrakt som avsätter tid för jour och kompenserar med ledig veckodag. Så beroende på vad för typ av support som krävs så måste motsvarande kontrakt skrivas på.

Vidare till automationen, hur har det sett ut där?

P2: Vi byggde ett automatiseringssystem som jag personligen tyckte var bra, det var inte bäst men inte heller sämst. Så i ett av mina projekt så jobbade vi i SaaS, så företaget erbjuder en produkt, en mjukvara som ligger på molnet, och företags projektet anpassar den efter vad kunden vill ha. Då måste ju denna produkt vara byggd på ett sätt som klarar av DevOps. Om den är gjord på ett oflexibelt sätt och man har ett system där man inte kan automatisera, så när man levererat kod så byggs det inte, så blir det besvärligt. Så även om man vill köra DevOps så kan det finnas tekniska begränsningar i den produkt som man använder. Även om produkten i sig har stöd för att göra saker så kan man kanske inte köra ”full DevOps” och vi fick i det här fallet titta på vilka steg som vi kunde automatisera, som inte krävde manuella åtgärder. Man får i de flesta fall försöka göra en så bra DevOps som möjligt utifrån de tekniska förutsättningarna.

- **Har ni behövt göra någon typ av benchmarking vid val av era verktyg eller har ni använt företagets egna?**

P2: Vi hade ju tur när vi började med vårt Cloud (DevOps), för då fanns ju ingenting och vi kunde välja lite vad vi ville och bygga utifrån vad vi tyckte var bäst just då. Så inga problem direkt men det var ju fortfarande en urvalsprocess kring existerande verktyg som skulle väljas. I projektet jag gjort senast så var allt redan uppsatt; där är produkten, där loggar ni in för att deploya, dem pratar ni med för att få hjälp och nästan allt var specat. Och de räckte med att vi ”byggde klister” emellan och det handlade mer om att lappa upp mellan gliporna i det vi redan fått. Vad gäller verktyg så har alla utvecklare sina preferenser och får man välja så tar man oftast sitt egna IDE, Git och så. Sen finns det ju begränsningar inom företaget gällande vilka system som man kan och får använda. Ex. så går det inte att bara använda GitHub utan vi har en intern git och då kommer inte kunden åt den. I projektet just nu så ska kunden kunna komma åt allt. Vi använder då JIRA (monitoring tool) för att tracka allt, lite likt Kanban, vilket då är externt för att båda parterna ska kunna se vad som händer. Så om kunden inte bryr som vad vi gör utan bara vill ha det levererat så kan vi ju köra med våra egna interna system. Sen har det varit så att när jag kom med i projektet för DevOps så tog jag med mig kompetensen så det har inte varit så problematiskt just kring automationen. Har folk velat automatisera delar så har de kunnat vända

sig till mig så har jag berättat hur de ska göra. Det är inte svårt om man vet vad man ska göra, och det blir lättare efter varje gång man gör det. Men sen finns det ju tekniska begränsningar, t.ex. när man sitter och utvecklar och så stöter man på någonting som inte funkar och så vet man inte varför det funkar, då kan man sitta med det problemet hur länge som helst. Men det handlar väl mer om människan och individen och inte lika mycket om bristande kompetens. Så på teknisk nivå kan det vara vissa saker som tar lång tid, men att få allt i rullning är inte speciellt svårt.

- **Var arbetet med att bygga er pipeline problematisk på något sätt?**

P2: Vi byggde den under ett par år, först byggde vi en version som inte riktigt höll med skalbarheten. Så vi fick bygga om den lite allt eftersom fler variabler lades till. Själva mallen för en pipeline är ganska straight forward men att få in alla funktioner man vill ha tar lite längre tid. Men det är nog mer en kostnad i form av tid än vad det är problematiskt.

Över till tredje delen så har vi mätning och har ni upplevt några svårigheter där?

P2: Det vanliga är ju att man sticker in kod för att se användningsmönster och vart i flöden de failar. Vi hade instrumentering för att veta vad som funkar och inte funkar, vart det dök upp felmeddelande och sånt. På back-end sidan instrumentera man kod för att se hur lång tid tar anrop, vart ligger tiden någonstans. Och då kan man se vad de beror på och det finns både verktyg och system att bara plugga in.

- **Hade ni några svårigheter att identifiera vilka mätetal som var intressanta för er?**

Vi upplevde lite svårigheter med att identifiera vad för något som skulle mätas. Men det primära användningsområdet vi använder mätning till är idag för att identifiera ”bottle-necks”, så när den indikatorn hittades så har vi inte haft några större problem sen dess.

- **Mäter ni era medarbetares prestation, hur många deploys har de och vad levereras, för att få fram ert arbetssätt?**

P2: Jag tror att det är något man märker ganska snabbt i personrelationer, om man är i ett team på 5-15 personer så lär man sig ganska snabbt hur de beter sig och hur mycket de kan leverera.

Så mätning av individers prestation har vi nog inte haft något av alls, det har motverkats genom kulturen och de mindre teamen. Man lär känna hur personerna är och sen är det väldigt viktigt att man försöker hitta en gemensam beskrivning för vad begrepp som "klar" innebär. Så om någon säger att det är klart så betyder det att det här är levererat till den här miljön och testat. Annars blir det problem när man ska ha möte och alla definierar klar olika. Sen tror jag inte det ger så mycket att mäta hur duktig prestation någon har utifrån antal lösta fall och deploys. Detta beror ju helt på vilken typ av fall det är, för vissa kan ta fem minuter och vissa två veckor att bara hitta problemet.

För vi förstod det som att det var mot strateginivå som DevOps riktar sig mot, hur många deploys kommer ut i veckan, om dagen och så.

P2: Ja.

Inga större problem kring mätningen för er, bristen på behov av att mäta t.ex. Antal deploys och releasetakt är för att ni har intimare grupper med koll på varandra?

Mmm, men just kring mätning så har vi väl alltid haft problem vare sig det gäller personer eller systemen. Som med Pagerduty så har vi också arbetat fram monitoreringssystem som pingar vid avvikande beteende i systemen. Så problem stöter vi alltid på och problematiken kring detta är först att identifiera något som behöver en lösning och sedan försöka bygga lösningen på bästa möjliga sätt. Ex. om en tjänst brukar svara dåligt så beror det kanske på att koppling till en databas är svag eller inte så pålitlig. Då kanske vi måste göra någon form av cachning internt för att slippa bero på den hela tiden. Så problem finns det ju alltid, annars skulle vi inte behöva komma med några lösningar för att det inte ska förbli så sårbart.

Sista biten är då delningen och hur har ni upplevt problem där, utöver hjältekulturen?

P2: Om jag svarar allmänt så har företaget ett antal värderingar eller principer som ska levas upp till. Där sätter alla vi upp mål för året och en av de är "responsibility for others". Vi har mål som syftar till att man ska lära andra. Generellt så jobbar de som jobbar hos företaget med att man ska lära varandra, men sen vet jag inte riktigt hur det sker i praktiken. Men jag har upplevt detta som positivt och att man ofta tar andra under sina vingar. Sen beror det även på vilken form av uppdrag man har, som konsult så har du ofta som uppdrag att lägga ex antal timmar på olika projekt och då kan det bli svårt att lägga några timmar bara på att lära upp någon. Det blir som ett incitament emot upplärningen eftersom de planerats för. Det blir då viktigt att projekten planerat in för hand-over, handläggning och sånt. I allmänhet människa mot människa så funkar det bra men i projekten måste man se till att det finns tid utlagt för detta vilket varit problematiskt i vissa fall. Men de leder tillbaka till en stark organisation som identifierar detta problem**.** Jag kan även nämna på väldigt övergripande nivå att företaget är ett av de företag som

jobbar väldigt mycket med open-source. Vi köpte upp Red Hat och har jobbat med många open-source organisationer och företaget har som strategi att jobba mycket med öppna standarder vilket vi tror lönar sig i det långa loppet. De ska vara kompatibelt och de ska lätt gå att dela sinsemellan.

Sen är ju företaget ett väldigt stark företag överlag, mycket av det vi jobbar med kräver stark up-time och får liksom skita sig max 20 minuter i månaden. Detta ser ju till att motivera än att leverera och det tvingar på något sätt fram bra kvalité och tillåter inte att vi levererar något som inte lever upp till vår satta standard.

7.3 Intervju 3

Vad är din befattning och hur har du kommit i kontakt med DevOps?

P3: Jag är manager på företaget men främsta del utvecklare och solutions architect. Jag har under mina 8 år här stött på DevOps i nästan alla mina projekt i många olika konstellationer, jobbat mycket med den tekniska delen och inte så mycket med själva företaget. Utan jag har varit med och satt upp de man kallar continuous integration, continuous deployment och sånt där, åt olika kunder och på olika plattformar. Exempelvis har jag jobbat i Visual studios online som nu är Azure DevOps, Teamcity och Octopus deploy, så lite blandat.

Hur skulle du tolka DevOps, och vad anser du tillhör dess kontext?

P3: För mig personligen, är det som utvecklare att få ut testad kod, automatiserat hela vägen till produktion på en kontinuerlig basis så inte varje månad utan mer vardagligt eller veckovis.

Vad för problematik kring implementationen av DevOps har upplevts utifrån CAMS om vi börjar med kulturen?

P3: Just kring det organisatoriska och ändringar där, så har jag inte stött på det alls eftersom det är den delen av DevOps som fattas hos de flesta kunder vi har och delvis i den förståelsen som företaget har, utan alla kör på med det tekniska och missar helt organisationen i sin helhet. De kör på med sitt gamla vanliga tänk medans IT-delen trycker på från ett annat håll, så jag har aldrig behövt vara direkt involverad i det organisatoriska arbetet eftersom jag huvudsakligen suttit med det tekniska.

- **Har ni stött på någon typ av internt motstånd i samband med DevOps?**

P3: Ja när vi kommer till de tekniska delarna och folk som är emot någon form av DevOps så har vi stött på motgångar. För en utvecklare så kan de bli ganska mycket teknisk skuld om du vill ha en helt automatiserad pipeline, de är väldigt mycket jobb att automatisera, det är väldigt mycket jobb att bygga feature toggles. Du behöver organisera din kod på ett sätt som folk inte är vana vid för att kunna ha en automatiserad deployment pipeline. Man får ju ofta bygga testerna först , man måste ju bygga unit tester för koden men även system tester och massa andra tester ska ju automatiseras. Om man måste behöva klicka en massa så får man ju inte de där "sköna flödet" i sin DevOps pipeline som man vill ha. Sen med teknisk skuld som jag nämnde förut så innebär de att man måste koda mer. Feature toggles, ett begrepp som jag tror Facebook äger, innebär ju att du måste koda din kod så att du kan stänga av saker och slänga på saker, skicka ofärdig kod ut till produktion och så. Om du ska leverera saker varje dag så måste du kunna göra så, och det kräver mer jobb från utvecklarna helt enkelt.

- **Så problemet som du sätt är en typ av överbelastning för utvecklarna när ni måste ta på er ytterligare arbetsuppgifter?**

P3: Ja precis, det blir både mer jobb men arbetet kräver plötsligt mycket högre kompetens också.

- **För att möta kompetensproblemet, har ni någon form av utbildningar eller tar ni in externhjälp?**

P3: På företaget så har vi en DevOps kurs som vi själva håller i och den är tight kopplad till vår SCRUM-kurs som i princip hålls i av samma personer. Den kursen skickar vi i alla fall så många vi bara kan på. Kursen i sig är kanske inte den "starkaste" och mer av en introduktion, den är ju bara två dagar, men man kan gå vidare efteråt och ta lite certifikat och läsa på i rekommenderad litteratur.

- **Så det är lite för att motverka en tidigare tanke att DevOps bara är ett trendigt ord?**

P3: Mm, ja exakt.

- **De här med Dev och Ops, ser ni att dev tar på sig mer av driftsättningen eller har ni tidigare dedikerade ops team som nu integrerats med utvecklarna?**

P3: Ne, allt faller på utvecklarna nu, så ansvaret för driftsättningen hamnar nästan helt på utvecklarna. Om vi ser till SCRUM-principer och sådär, när vi har utvecklingsteam så ska ju

helst alla i våra utvecklingsteam kunna allt som görs i de andra utvecklingsteamen också. Operations kommer lite mer på sidan av, och de har varit något som inte funnits med alls hos vissa kunder. De har ju dock ofta ett produktionsteam om vi ser till exempelvis HM eller Electrolux. Så när vi är klara med det "fräcka" utvecklingsarbetet och gjort massa nya "fräcka" features så lämnar vi över det till ett trött produktionsteam ibland som är ett gäng gamla gubbar i serverhallar.

- **Så skulle man kunna säga att ni lämnar över det till en typ av underhållningsteam med mer av en supportroll?**

P3: Exakt, så kan man säga det. Nu som vi jobbar hos Systembolaget så är vårt produktionsteam vi själva från företaget. Så här har vi ett mycket bättre flöde i DevOps processen , men att alla devs fortfarande gör nästan allt ops.

- **Ett krav för er nu är alltså att det krävs en högre kompetens inom ops delen för alla utvecklare?**

P3: Ja men exakt.

- **I samband med att ni utvecklare har fått mer arbetsuppgifter, agerar teamen mer som autonoma grupper där fler beslut tas av er själva eller har ni en hierarkiskt beslutstagande?**

P3: Ja det är väldigt hierarkilöst så det är ofta utvecklarna tillsammans med våra manuella testare, som hos systembolaget. Som utvecklare är din pipeline automatiserad så långt som till en viss acceptansnivå eller testmiljö. Dit drar du din task så testas den och sen har testarna som är en del av teamen ett ansvar att säga "nu kommer jag trycka detta vidare till produktion" som är ett manuellt steg där det fortfarande krävs en typ av kodgranskning innan det går vidare.

- **Tidigare har det förekommit en typ av "hjältekultur" där någon sitter inne på information för att "säkra" sin plats, är det något ni upplevt?**

P3: Det blir väldigt nära SCRUM igen ur ett sånt perspektiv. Men så var det mycket förr och jag har till exempel varit expert på en del saker så när man varit borta har det blivit kaos. Men det är något vi strävar efter att få bort, så även om vi har många uppdelningar som backend och frontend så sätter vi ju aldrig någon på en sak som det inte kan men då blir det något de måste lära sig. Sen finns det ju fortfarande experter inom områdena men dem är mycket bredare nu så det kan vara en expert på en viss produkt eller så.

- **Så ni ser en jämnare fördelning av kompetensen men att de enstaka kullarna fortfarande existerar, men mer till följd av att de rent sagt bara kan mer?**

P3: Ja exakt, det enda riktiga problemet vi har här om vi pratar kompetens är att företaget har dedikerade testare som jag tycker är fel. För alla ska ju kunna testa och utveckla, i detta fall när vår testare är borta så sker liksom inga av de manuella testerna utan då är det bara automatiserade tester.

Då tänkte vi gå vidare in på Automationen och om du vill börja?

P3: Ja det största problemet är att det rent sagt är sjukt mycket jobb. Men det beror väl lite på då, att automatisera continuous integration, automatisera byggen och automatisera kod, att du ska köra unit tester, använda code metrics för att se hur din kod blir mer avancerad över tid. Sen kommer vi till release delen där vi idag har automatiserade systemtester, vi bygger en väldig massa API:er och varje API har som en healthcheck del. Så varje gång vi releasar någonting så gör vi en liten "get response" för att se att API:et fortfarande lever och funkar som den ska och nu kan de gå vidare till systemtest, acceptanstest och produktion. Däremot fattas det väldigt mycket, för den kollar mer att databasen funkar och om du har några API:er den måste kalla på så funkar de också, mer något inom infrastrukturen. Däremot om du deployar webbsidor så behöver du skriva UI tester om du ska automatisera de, och de gör inte våra utvecklare utan de testar allt manuellt. Att just ta deras manuella jobb och automatisera de efter ett speciellt ramverk så man får en bra övergripande bild blir mycket jobb. Det är också mycket jobb att få upp automatiserade lasttester. Så lasttester, UI tester och säkerhetstester görs manuellt för vi har varken råd eller tid att automatisera de.

- **Det är alltså ingen brist på kompetens att skapa dessa tester utan problemet ligger i tidsbrist och kostnad?**

P3: Ja kostnad är väl lite definitionsfråga för bygger vi testerna så blir det ju en kortsiktig kostnad men på lång sikt så behöver vi ju aldrig göra det igen. Men det handlar lite om testarna och att de gillar hellre att själva klicka runt än att sätta sig ner och koda lite tester. Det får väl göra vad de tycker är bäst men det handlar även om att arbetet är tidskrävande och sen ska det även underhållas så allt är ju relativt, man får väga för och emot.

- **Hur ser det ut vid valet av era verktyg, har ni någon urvalsprocess eller arbete kring detta?**

P3: Vi använder uteslutande Azure DevOps, förr kunde vi använda Jira, Teamcity för deploy och installera massa plug-ins för att få dem att samarbeta. Och så har du dina Git-repositories, all din dokumentation, alla dina tasks. Jira har då alla dina tasks, och du kan koppla all din kod som du

pushar till dina tasks. Så när en testare testar någonting så kan hen se exakt vilken commit som skedde i vilken miljö de ligger i. Så i Azure DevOps sätter du också upp dina pipelines, du sätter upp dina byggpipelines och dina RISC pipelines. Så Azure DevOps innehåller allt vi behöver här hos företaget. Och hur urvalsprocessen såg ut så var det inget speciellt sådär, men när Azure DevOps låg i någon typ av preview värld så var det väl lite obehagligt. Men eftersom Azure är ett Microsoft verktyg och vi ägs av Microsoft så var det ganska självklart att vi skulle ha Azure DevOps. Så överlag väljer vi alltid Microsoft verktyg i förstahand och om det nu inte skulle finnas får vi välja något annat och då blir det ju en liten urvalsprocess beroende på mängden verktyg som passar för uppgiften.

P3 visar oss Azure DevOps

Nästa område vi tänkte gå igenom är mätning och ifall ni har de och upplevt problem kring de?

P3: Vi har precis påbörjat acceptanstester och lasttester, så det här med feedback och kommunikation kring våra leveranser är sjukt viktigt. Vi kör i alla fall de som kallas mikrotjänster så man slipper ha en monolitiskt monsterapplikation med all kod, utan vi har delat upp den i kanske 20 mnindre applikationer istället. Så vi använder Azure igen, Microsofts cloudplattform, och vi använder deras grej som heter application insights. Så application insights på dina dotNET, javascript eller java applikationer innebär att du lägger in en liten kodsnutt, som säger “jag använder application insights och har den här nyckeln” så kommer den automatiskt skicka telemetric data och sen kommer de visas upp i snygga grafer och du kan skriva SQL på det och den kommer berätta för dig hur din app mår. Så om vi skulle gå in och kolla prestandan på låt säga vår produktionsuppsättning så kommer den ge oss svarstider och allt du kan tänkas behöva för att bedöma hur något mår och om något skulle behöva ses över. De är så detaljerat att vi kan se varenda liten sekvens och hur minsta lilla rad presterar. Vi har också något som kallas deathstar arkitektur i Azure DevOps. Så där kan vi se alla mikrotjänsters kopplingar och hur allt är beroende av varandra. Så application insights kör vi all typ av metorering, logging och just det här med feedback.

- **Har det krävts något typ av urval vid val av nyckeltal att mäta i och med att Application insights är så gediget?**

P3: Inte så mycket faktiskt, förut var det en ren mardröm att försöka välja ut vad som skulle gynnas oss att mäta och sånt. Men med Application insights så kan vi välja att i stort sätt mäta allt. Vi får all data presenterad på ett snyggt sätt och vi kan lätt följa vart i koden det behövs snyggas till.

- **Så ni har inte stött på någon större problematik alls i och med mätning och hur detta skulle göras, utan det blev mer en naturlig övergång i och med introduktion av Azure DevOps?**

P3: Ja exakt men sen är det ju fortfarande så att vi bygger nya saker hela tiden. Som med nya systembolaget.se, den är ju inte riktigt live än men det är ju fortfarande svårt att veta på ett övergripande sätt vad som behövs vid nya saker. Man vill ju ha en sida som säger hur saker och ting mår och det är fortfarande väldigt mycket jobb med att veta vilka nyckeltal man vill kolla på för det finns ju tusentals små saker som rapporterar data så allt måste ju kunna tas in och presenteras på ett vettigt sätt. Application Insight hjälper väl en hel del men vi måste ju ändå skapa våra egna vyer och sånt. Så det blir ju ett urval för att se vad som representerar verkligheten bäst. Sen kopplar vi ju insights till våra releaser i Azure DevOps så när du releasar någonting så får den ett ID i insights och du kan följa upp dina releaser. För att se om response time gick ner eller error-count gick upp.

Sista området är då delning och vi tänkte att du kan börja lite kring hur informationsutbytet ser ut hos er?

P3: Ja men vi använder Microsoft teams för direktkommunikation, de är chat, chatgrupper, wikis och fileshare i ett liksom. Så du har till exempel för våra team en grupp där du kan skriva och fråga saker i olika kanaler beroende på fråga. Du har wikis kopplad till den, i Azure DevOps har du också en wiki med basic info för hur du sätter upp en miljö och vanliga fel du kan stöta på. Sen har du i våra git-repon read filer också som helst ska vara väldigt utfyllda och det är ofta där på git som all information ska finnas och där vi löser mycket av problemen som stöts på.

- **Märker ni att det sker någon typ av kompetensutbyte i det dagliga arbetet, för att lite motverka hjältekulturen och sprida kompetensen?**

P3: Inom företaget har vi mycket intern kompetensutveckling, alla måste ta en viss mängd timmar varje år för att få behålla sin kompetens. Sen lär vi oss mycket när vi sitter tillsammans och kör code reviews, sen har vi även en del parprogrammering för att lära varandra och jämna ut kompetensen. Men mer än så har vi nog inte.

- **Upplevde ni något motstånd vid införandet av t.ex. Parprogrammeringen, när den sociala aspekten kom med?**

P3: Nja vi har ju främst haft seniora utvecklare som då sitter med juniora utvecklare och mest tittar på, fyra ögon är ju alltid bättre än två och sen hjälper de till lite när de körs fast och så. Men visst har vi vissa som inte lämpas för den typen av arbete och då har de blivit bortvalda från de arbetet helt enkelt. Vi har haft både seniora utvecklare som försvinner in i sin kod och tittar ut

efter en vecka men likaså juniora utvecklare som försvinner i två dagar och kommer fram när de väl suttit och stångats med ett problem för länge. Men de är i minoritet och de flesta av våra seniora utvecklare och nästan alla juniora vi tagit in de senaste åren har varit väldigt öppna för att sitta tillsammans och hjälpa varandra och så.

Har du något annat som du kommit att tänka på sådär i efterhand kring DevOps?

P3: Nja det är väl mest de att DevOps är ju inte riktigt DevOps när Ops knappt existerar för oss, mycket ny teknik som underlättar allt det tekniska arbetet och mycket kan göras av tekniken själv.

- **Skulle du säga att ni jobbar i små autonoma team med stark "koll" på varandra?**

P3: Ja exakt vi försöker ju följa SCRUM-metodologin och har team som består av 3-9, så vi är ju aldrig mer än 9 stycken men det har hänt fast då delar vi upp oss i grupper inom gruppen, blir lite olika backend grupperingar vilket kanske kan va lite konstigt egentligen. Men vi är små team och företaget följer generellt SCRUM i nästan alla sina leveranser. Sen är det ofta svårt och ibland omöjligt och ibland går det alldeles utmärkt att få över det hos kunden vi är hos. Så ibland har vid med oss SCRUM coacher, vi har minst en i varje land, och i Norden har vi bara en. Men de får i så fall åka ut till kunden och utbilda dem i hur man jobbar inom SCRUM och inom DevOps så vi från företaget kan fortsätta arbeta som vi brukar göra. Men ibland sker det inte alls som hos systembolaget nu. Här jobbar företaget efter SCRUM och DevOps till en viss nivå och när systembolaget blir involverat så blir det vattenfall.

- **Har ni upplevt något kring att de nya arbetsansvaret lätt till försämrad informationsdelning?**

P3: Jo men absolut, det sker hela tiden. Just med sprintar och SCRUM och vikten att planera sitt arbete för en viss tid framöver. De har vissa jättelätt för och vissa jättesvårt, men ju mer vi jobbar med det ju lättare blir det att se hur mycket en person klarar av. Jag tror vi alla är ganska dåliga på det i början men efter varje gång kan vi korrigera hur stor arbetsbördan ska bli och vi undviker till viss del att många blir överbelastade. Men det är svårt att gå ifrån det helt och hållet så visst upplever vi fortfarande att kommunikationen eller delningen som ni sa brister då och då. Det är ju ofta så att folk inte estimerar att föra vidare kompetens, testning och dokumentation tar sin tid och deadlinesen hamnar i kläm. Så det sker hela tiden att vi underskattar hur mycket jobb krävs.

7.4 Intervju 4

Vad är din befattning idag och hur har du kommit i kontakt med DevOps?

P4: Jag är IT konsult hos ett företag och satt som teamledare hos org1 när ni kontaktade mig. Jag ansvarade där för ett leverans och DevOps team, jag har tidigare även jobbat på org2 som utvecklare och SCRUM-master. Innan detta var jag hos org3 som något de kallade teknisk systemledare, man var tekniskt ansvarig för ett system i förvaltningen och hade koll på allt ner till servernivå. Detta var ju även innan DevOps fanns som begrepp och det handlade om att sy ihop allt detta och hur det skulle driftas.

Hur tolkar du DevOps och vad inkluderar du i dess kontext?

P4: DevOps för mig är en kultur där alla hjälps åt med sina speciella kompetenser och oavsett om man är utvecklare eller drifttekniker så samarbetar man. Man behöver inte nödvändigtvis sitta i samma team, det kan underlätta, men det beror lite på hur man vill skära organisationen. Men det viktiga är att man är öppen för samarbete och det är ett sätt att förstå varför vi vill göra ändringar i miljön hela tiden och varför vi vill ha en stabil miljö hela tiden.

- **Ni har alltså haft både drift och utvecklare på samtliga av dina tidigare arbetsplatser?**

P4: Vi har haft tydligt, här är utvecklarna och här borta har vi typiskt servertekniker, Dbaa, övervakningsspecialister och den typen av personer, till viss del även supportpersonal, första och andra linjens support vill jag även lyfta in i det här för att det är ändå de som når ut och träffar verksamheten i stor utsträckning också. DevOps tar ju inte höjd för slutanvändaren, det handlar ju om att få systemen att fungera och en IT-avdelning som kan samarbeta ordentligt.

- **Så ni har haft en ganska tydlig utvecklarsida och driftsida men med starkt tvärfunktionellt samarbete?**

P4: Absolut, och i olika drag separerade fysiskt och organisatoriskt. Hos org3 så satt jag i Norrköping men huvudkontoret var i Stockholm men även där på huvudkontoret var teamen och avdelningarna uppdelade på olika våningsplan. Hos org2 så satt vi på samma våningsplan i lite olika korridorer men med öppna dörrar. Och hos org1 så satt vi på samma våningsplan med låsta dörrar, utan ringklocka till driftsidan eller operations. Det var lite intressant för när man pratade med individerna hos operations på org1 så var det helt för att vi måste ha ett starkt samarbete. Men kollektivt så var de väldigt nöjda med att sitta och låsa in sig. De hade som en grej att kvartalets bästa händelse var när de fick bort ringklockan till deras dörr och fick sätta papper för

deras glasade yter så att ingen såg vad de gjorde där inne. Så det fanns lite utmaningar att få samarbetet att funka just hos org1.

Vad för problematik kring implementationen av DevOps har upplevts utifrån CAMS, om vi börjar med kulturen.

P4: Jo men att man tillhör organisatoriskt olika delar skulle jag vilja säga är anledningen till att "DevOps" finns. Vi jobbade ju med detta innan DevOps fanns som begrepp så fenomenet utan etiketten har ju funnits länge. Det här att man kommer från olika världar där man har olika saker man mäts på skapar en kultur där "man vill vara ifred och ha det lugnt och stabilt" kontra "vi är nyfikna, vi vill förändra, vi vill ha ny teknik" och att man då är inlåst som hos org1 ökar ju problematiken. Om man inte träffar de "andra" så får man ingen förståelse för att de inte är så farliga, vi vill ofta samma sak men vi uttrycker oss båda lite annorlunda, de tänker lite annorlunda men egentligen vill vi väl allihop.

- **Kvarstod mycket problem efter att DevOps implementerades, eftersom de var meningen att detta skulle brygga den tidigare silomentaliteten?**

P4: Vi började se en del upplösning av problemen men det var ju långt ifrån klart. Där jag haft mest mandat att jobba med DevOps hos org1 så avslutades tyvärr mitt uppdrag i förtid på grund av budgetnedskärningar helt enkelt och de hade inte råd att ha kvar större delar av DevOps teamet. Men vi hade ändå kommit ganska långt om vi bortser från kultur bitarna så hade vi kommit väldigt långt kring automatisering av byggen, deploys och skulle ta nästa steg kring tester och nästa steg för IaC. Men rent kulturmässigt hade vi en bra bit kvar, och jag hade hoppats på att vi skulle få ta steget att sätta oss ihop med ett team bildat från båda lägren och sitta ihop. Det var på tapeten men blev tyvärr aldrig av, jag tror det skulle löst en del av de kulturella problemen men nu fick vi aldrig möjligheten att göra det så jag kan bara sitta och spekulera kring det.

- **Kan en av anledningarna till att detta inte infördes tidigare vara på grund av ett bristande stöd ovanifrån inom organisationen?**

P4: Ne jag skulle nog säga att man var väldigt öppen med att börja det här arbetet med DevOps-tänk eller hur man nu ska uttrycka det. Beslutet kom nämligen ovanifrån och vi hade flera chefer med oss, det som möjligtvis stoppade oss var dels att utvecklarna hade väldig press på sig att leverera för det kom ett nytt lagkrav, så de var tvungna att leverera tills sista mars så det var inte riktigt läge att börja peta där. På driftsidan var de antingen underbemannade eller hade för ostabil miljö så det skapades merarbete hela tiden. Det bromsades nog därför snarare upp av att de som skulle jobba med DevOps inte orkade eller hade för mycket att göra. Lite kontraproduktivt i de

sammanhanget är ju att man prioriterar bort sånt som kanske skulle hjälpa till med att jobbet blev effektivare.

- **Såg du att utvecklarna och de team som fanns, fick mer ansvar och makt att ta sina egna beslut gällande sitt arbete kring t.ex. features eller funktioner?**

P4: Jag skulle nog säga att utifrån org2s perspektiv där jag var mer delaktig i utvecklingsarbetet så var vi väldigt självorganiserande och självbestämmande. Vi hade våra krav, ofta uttryckta i någon typ av user story, som haft sitt ursprung i något behov i verksamheten, en förändring, process eller stöttning i arbetsprocessen i verksamheten. Sen hur vi rent tekniskt implementerat det här var upp till teamen. Vi hade begränsningar i programspråk och servermiljöer. Vi kan t.ex. Inte säga att denna gång vill vi bygga allting i python på en windows server för det var Linux Redhat och java som gällde. Men inom ramarna för det var det väldigt fritt att välja ramverk och hur vi skulle lösa det. Frihet fanns ju där och man var även typ tredje linjens support för vi satt också och löste serverloggar och sånt. I detta fall var driftsättarna experterna som vi kunde vända oss till ifall vi själva inte kunde lösa problemet.

- **Så det växte fram ett starkare förtroende mellan utvecklarna och driftteknikerna?**

P4: Hos org2 så JA, men sen är det ju alltid så att det finns andra och andra team som har större problem. Jag ska inte säga att det inte tog ansvar men det levererade inte alltid med kvalité och de ställde ofta till de. Det hände att de leverera saker som sänkte hela servrar för att man inte testat ordentligt eller tänkt till ordentligt. Och i den änden så får man inget förtroende och man bygger bara högre murar.

- **När det levererades sämre kvalité skulle du säga att detta kan berott på bristande kompetens, eller bara mänskliga faktorn?**

P4: Det är en kombination av båda, vissa saker beror ju på att man inte kan det bakomliggande tillräckligt bra. Jag väljer att blunda för vissa risker, man blundar nog inte för ett konkret faktum att detta kommer gå rent åt helvete, men däremot blundar man nog för en del risker när man gör en riskbedömning. För man tänker att vissa saker kommer verkligen inte att hända eller risken att det händer är oerhört liten. Ibland har man inte förståelse för vad som händer på andra ställen, för andra och förståelse för hur det slår och vilka gränser man har. Om jag bygger en java kod som inte klarar att garbage collecta ordentligt så kommer den käka allt minne. Då behöver jag förståelsen för att detta kommer käka allt minne på servern och de andra system som ligger där kommer få problem på grund av mig. Intresset för just detta var lite bristande hos en del team på samtliga arbetsplatser jag varit på.

- **Har ni upplevt någon typ av hjältefenomen?**

P4: Jo det har jag upplevt på samtliga ställen och det finns nog överallt. Det kommer nog alltid finnas de som kallas hjältar, hjälteanvändare eller superusers. Och jag vill väl inte säga att det är ett problem i sig om det finns de som gärna vill göra allt och tar på sig det ansvaret. Problemet är ju när de inte släpper in andra och håller på informationen. För att få självbekräftelse, kanske skydda sitt uppdrag eller ens sätt att göra karriärer på. Det svåra där är ju att nå fram till dessa personer och få dem att förstå att de inte ska känna sig hotade bara för att fler kan de du kan. Och det är ju en jätteutmaning för det är ju inget du kan koda bort eller automatisera bort eller så. Det gäller ju att förändra en persons beteende och det är i min erfarenhet bland det svåraste man kan ge sig på och förändra. Så för att svara på din fråga har jag både upplevt det tidigare och sett det på mitt senaste uppdrag oavsett om DevOps fanns eller inte.

Hur har det sett ut med Automationen, har ni stött på några problem när ni försökt upprätta detta?

P4: Vi har ju jobbat hårt med det, och det gäller ju att ha en väldigt djup teknisk förståelse för att kunna välja rätt verktyg. Men det har snarare varit begränsningar i verktyg som gjort att det varit svårt att välja rätt snarare än att verkligen utföra arbetet. Vi kom så pass långt hos org1 att vi hade satt upp så att incheckad kod triggade enhetstester. En deploy startades till utvecklingsmiljön så fort ett bygge startats, det var alltså så att incheckning triggade ett bygge som körde enhetstester, och när det fått grönt triggade det en deploy som gick till utvecklingsmiljön. Sen hade vi automatiska tester som vi var manuellt tvungna att starta, detta berodde på legacy kod i applikationen som gjord att vi inte kunde köra det automatiskt som script inifrån team foundation server som numera heter Azure DevOps. Så vi kunde inte köra det automatiskt på grund av lite legacy kod men tanken var att det skulle triggas efter varje deploy till ovaktad miljö, så skulle vi köra de här grundläggande röktesterna som egentligen öppnade upp alla delar och läste information från alla delar av systemet.

- **Upplevde ni likt andra, motstånd kring arbetet att koda testerna och att utvecklarna ställt sig emot denna extra arbetsbörda?**

P4: Då kommer vi tillbaka till hur man är som person, de flest som är systemutvecklare, utvecklare, programmerare gillar att skriva “riktig kod” och kan möjligtvis sträcka sig till att att göra enhetstester. För det är små entiteter som bara testar små delar. Så det jag tyckt varit bäst är när man haft en prestigelös utvecklare som gör saker för att det behöver göras. Eller om man har en testledare eller testare som är tekniskt duktig och kan skriva de här testerna själv eller med lite hjälp från utvecklarna. Men ja det finns absolut, min känsla är att majoriteten av utvecklarna inte tycker det är speciellt roligt att sitta och koda tester när man kommit förbi enhetstester.

- **Du nämnde tidigare IaC, hur gick arbetet kring detta?**

P4: Det gick åt helvete, min motpart på operationssidan hos org1 valde att gå till en annan arbetsgivare så det rann ut lite i sanden. Detta även i kombination med att båda parter var

överbelastade med incidenter så man hann inte riktigt jobba proaktivt på något sätt. Det fanns väldigt goda tankar att man skulle kunna gå in på en portal och be om att få en server av en viss standard och sen skulle den beställningen trigga en lämplig konfiguration från i detta fall TFS och sen skulle scripten ligga där och starta en pipeline på något sätt.

- **Så det var huvudsakligen på grund av den tekniska skulden som arbetet inte tog fart?**

P4: Ja precis.

- **En följdfråga till förtroendet inom kulturen, höjdes den i samband med att mer blev automatiserat och att de var mottagliga till förändringarna?**

P4: I min uppfattning så är de inte att operations motsätter sig ny funktionalitet utan det är snarare förändringar i miljön som de är rädda för. Sen om det är ny funktionalitet eller gammal funktionalitet som refaktoriserat har varit ganska ointressant. Men ja absolut det har tagits väl emot och grunden för att bygga det här förtroendet har varit när vi automatiserat saker, inte bara att vi automatiserat testar utan att vi automatiserat hela deploymentprocessen gav oss jättemycket. Så ja absolut detta har byggt mycket förtroende.

Hur har arbetet kring mätning sett ut för er, har det skett någon typ av mätning för prestanda?

P4: Det blir två scenarion här, dels från org2 men också org1. På org2 som hade kommit längre än org1 när jag kom in, vilket också hade en mer mogen organisation ur detta perspektiv, hade när jag började en leverans i månaden stående, och då tog man ner hela systemen för att leverera en väldig massa system. Vi gick mer och mer mot så gott som daily deploys tillslut. För att sätta detta i perspektiv så levererade jag 22 leveranser under en 20 veckors period hos org2, detta mot deras tidigare 10 på ett år. Till detta var det ingen incident heller utan alla 22 var lyckade leveranser. Så där mätte man dels antalet leveranser och man följde även upp antalet incidenter, och en incident var allt som krävde någon form av efterarbete för att leveransen skulle gå bra. Sen gjorde man även en uppföljning på extra kostnad i verksamheten. Detta gällde då främst om systemet var nere och då hade man tagit fram tal för hur mycket varje timme kostade om systemen låg nere, då var det framförallt de centrala systemen man var intresserad av.

- **Vad motiverade mätningen av kostnaden för downtime?**

P4: Det var ett sätt att visa för chefer inom verksamheten, behovet av och vinsten av att satsa resurser på automatisering och tester. Något som kostat 30 miljoner i åtgärder kanske bara kostar 1 miljon för att förhindra. Så det var mer för att visa på det kortsiktiga och långsiktiga, att en liten investering här, väldigt snabbt betalar för sig. Jag har ju fortfarande kontakt med några på org2 och det har fortsatt på den här vägen och det har verkligen gjort det bra. Även de teamen som jag refererade till förut som kanske inte riktigt skötte sig och levererade dåligt. Är nu med och levererar med gott resultat utan incidenter, minst en gång i veckan vilket är en ganska häftig resa att göra på två år.

På org1 däremot så hade vi inte så tydlig uppföljning utan jag kan bara göra lite egna reflektioner kring hur det såg ut när jag började, så långt vi kom innan jag slutade och vad målbilden var i slutet av detta år då. Där körde man fyra leveranser på ett år, och det var såhär, tillskillnad från org2 så hade man ett system som var mer eller mindre en monolit, så när vi tog ner det systemet så stod hela org1 nästintill still. Man kunde inte komma in eller ut ur byggnaderna, rapportera misskötsamhet, man kunde inte göra någonting inom org1 om det här systemet låg nere. Därför klarade man inte av att i verksamheten ha systemet nere fyra lördagar per år. Sen gjorde vi ett antal förändringar där vi bröt upp systemet i 32 delsystem som vi kunde leverera var för sig. Inför en leverans kunde vi säga att här har vi gjort förändringar och då behöver vi bara leverera de här sju delsystemen. Om man inte gjort någon förändring i datalagret så kunde vi leverera utan nertid. Nu var det så att det nästan alltid var förändringar i datalagret i samband med det här deployerna.

Eftersom vi snabba på själva leveransprocessen också, eftersom allt automatiserades, så var det ingen som satt och kopiera filer eller startade script och sånt manuellt, utan allt följde en automatiserad process. Därför kunde vi gå från åtta timmars leveranser till cirka 35-40 minuters leveranser då vi kunde leverera saker parallellt också. Vi minskade även behovet av att ha personal på plats vilket följde av att vi jobbade mycket med kvalitetssäkring både tidigare och tydligare i processen. Så vi tog väl ner antalet personer som behövdes vid en driftsättning till en fjärdedel av vad de var tidigare. Och driftsidan behövde bara backa upp allt på servern innan vi började för resten skötte vi utvecklare. Där är det en enorm besparing men jag har inga siffror mer än att jag kan säga att vi blev betydligt effektivare i det skedet.

- **Men arbetet gick bra för er att bryta ner den monolitiska koden till fler mikrotjänster?**

P4: Ja exakt, men grejen var ju att den här monoliten fanns ju som delsystem redan från början. Men när vi talar om system som utvecklats under 16 år och över 100 utvecklare har varit inne i, så blir det rörigt och man tar genvägar och syr ihop systemen. Så det var ett stort jobb att göra den här övergången, mycket för oss att reda ut, men det gick bra.

Sista biten är delning, hur har detta upplevts?

P4: Vi kan börja i den änden att i det här fallet när det gäller att dela med sig så har org3, org1 och till viss del org2 en kulturell barriär i och med att man har väldigt känslig information. Framförallt org3 och org1 är väldigt restriktiva kring vilka som får ta del av vilken information. Den typen av kultur sprider sig även in i stödverksamheter som IT. Så man har lite de problemet att man tror inte man får dela med sig av information. Sen har man ju också strukturella problem som att slutanvändarna ska inte kontakta utvecklingsteamen utifrån de processer som fanns, utan de ska gå via någon typ av systemförvaltare som hanterar alla behov som uppstår i verksamheten. Så det gör ju att man inte har speciellt nära till en slutanvändare och kommunikationen däremellan blir en aning bristande. Men som sagt det är mycket kulturella och strukturella faktorer som spelar in här när vi talar om dessa två organisationer. org2 var bättre på delning, då satt vi dessutom mycket av verksamheten i samma hus. I vårt team hade vi även en person som jobbade som kravare och testare och hen hade 27 års erfarenhet i verksamheten innan hen började på IT-avdelningen. Hen kunde organisationen och hade ett starkt nätverk, så det var inte som att jag som utvecklare eller SCRUM-master hade kontakt med slutanvändarna men teamet fick den informationen direkt från de slutanvändare vi behövde ändå. Sen hade vi ett projekt till följd av ett lagkrav, där vi jobbade med att utveckla en lösning samtidigt som det skapades en ny organisation som skulle hantera systemet. Eftersom det skapades en tydlig användargrupp för vårt system så hade vi väldigt stark kommunikation sinsemellan. Detta kan även bero på att de började jobba i systemet innan det var klart för att förbereda register och sånt som behövde fyllas med data. Och detta berodde i sin tur på att vi hade mycket automatisering och tester så att delsystem kunde levereras och börja köras till viss del av användarna. Där hade vi alltså en väldigt stark kontakt med slutanvändarna och det är en sån där framgångsfaktor för utveckling.

- **Hur väl gick arbetet att ta vara på kompetensen inom arbetslagen?**

P4: Jag har personligen inte varit med och satt ihop något team utan man har fått ta redan befintliga team som jobbar ihop. De team jag fått jobba med har fått vara intakta för där har samarbetet funkat, och jag har haft förmånen att komma in i team som jobbat tillsammans under en längre tid och man känner varandra utan och innan. Sen vet jag ju att det funnits team som verkligen inte fungerat bra och det har behövts flyttas personer hit och dit. Så vad jag kan säga är väl att den personliga relationen är viktig för du lär dig hur andra tickar Så har man ett team där alla känner alla så blir det lättare att koordinera och utföra arbetet.

För att summera kort:

Kring de tre olika organisationerna var det byråkratin och den kollektiva mentaliteten “det är läskigt att dela information” som ledde till att man hellre var försiktig än tog chansen att nå ut till “andra sidan”?

P4: Ja precis så.

För mätning så har du inte upplevt någon problematik där i något av dina uppdrag mer än att hos org2 var det redan bra när du kom och hos org1 var det en gradvis process, ni blev aldrig riktigt färdiga med arbetet men det vart en klar förbättring med DevOps?

P4: Så skulle jag inte vilja säga, det hos org2 skulle jag nog hålla med om stämmer, det var bättre på att mäta. Hos org1 så handlade det om att problematiken låg i att bestämma vad man skulle mäta. Där var det inget direkt driv från chefer och organisationen, som faktiskt borde vara de som visat intresset för att mäta. Så där fanns det nog en problematik kring brist på stöd ovanifrån och att hitta rätt mätetal för oss. Det ena föll ju på det andra i mångt och mycket.

Gällande automationen så upplevdes ingen större problematik då ni besatt den tekniska kompetensen ni behövde och själva kunde välja och bygga era verktyg utifrån detta?

P4: Utöver tidsbrist och bristande arbetskraft så kan jag väl även bara flika in att jag upplevde ett litet motstånd mot att skriva allt för omfattande automatiska tester.

För kulturen så fanns det en tanke eller önskan kring det tvärfunktionella samarbetet men att i praktiken så föll det lite på grund av tidigare rotad mentalitet och att den geografiska uppdelningen inte underlättade något?

P4: Ja de skulle jag säga, att man hade det här önskemålet. När man väl hittade till den person som man skulle jobba ihop med. Så har jag alltid upplevt att man varit hjälpsam och vill samarbeta för att nå ett bra resultat. Så problematiken var att vända negativa personliga tankar till något positivt istället. Mycket kan sammanfattas i människan och för att knyta an till det jag sa i början med, så är DevOps kultur för mig, och det är den stora utmaningen också.

7.5 Utkast intervjufrågor:

Tolkning:

- Hur definierar du DevOps? Vad betyder begreppet för er?
- Vad fick er att välja DevOps?
- Vad har ni förväntat er få ut av det?
- Vad var era mål initialt med DevOps?

Implementation:

- Hur förberedde ni er för att implementera DevOps?
    - Specifika ramverk (CALMS), benchmarking, internanalys...
- Hur har ni gått tillväga för att implementera DevOps?
    - Vart i processen är ni idag?
        - Har det tagit längre/kortare tid än förväntat?

Resultat:

- Har ni upplevt en förbättring med DevOps?
    - Varför det/ Varför inte?
        - Kan det bero på CALMS?
- Har ni mätt förbättringen och hur har ni gjort detta?
    - Vad mäter ni?

Problematik:

- Vad har varit svårt med DevOps?
    - Culture

    - Automation
    - Lean
    - Measurements
    - Sharing
- Har ni ändrat era verktyg för DevOps?
    - Om ja, hur har den urvalsprocessen sätt ut?
    - krävdes det mycket arbete för att genomföras?

Kort om CALMS-modellen:

- Culture

Hur ser er företagskultur ut och hur står den sig gentemot (för/emot) förändring?

- Automation

Genomförs regressionstester etc. för att minimera manuellt arbete?

- Lean

Har ni försökt eliminera processer som inte medför något mervärde för kund?

- Measurement

Har produktionscyklarna sätt någon förbättring och mäts det korrekt?

- Sharing

Sker det ett kunskapsutbyte mellan de två teamen?

7.6 Utskick av Intervjumaterial

Här kommer lite av vårt intervjumaterial så att du kan förbereda dig lite inför intervjun och formulera lite svar och tankar! Tanken med detta utskick är främst att ge dig lite förståelse för hur vi tänker utforma intervjun och vilken data vi söker.

Till att börja med skulle vi vilja veta din tolkning av vad DevOps är och vad du skulle inkludera i dess kontext.

Hade ni några förväntningar på det?

Vårt arbete utgår från ett ramverk som heter CALMS; ett akronym av Culture, Automation, Lean, Measurement och Sharing. Vårt mål är att försöka koda alla problem du nämner efter dessa fem kategorier.

Företagskultur: Utbildningar, internt motstånd, omstruktureringar (team, avdelningar etc), nya roller, omfördelning av ansvar (nya ansvarstagare), godkännande, förtroende.

Automation: Tester, verktyg, script, leveranspipeline, testmiljöer.

Lean: Effektivitet (hur?), värdeskapande, motverkar redundans, långsiktigt eller kortsiktigt (technical debt).

Mätning: hur mäter ni? Nyckeltal (commits, tickets, antal “leveranser”, felfrekvens, error-hantering etc), målsättningar, processer

Delning: Informationsutbyte, återkoppling och feedback, ta vara på kompetens (arbetslagen).

Vilken problematik har du stött på i samband med implementation av DevOps?

Avslutningsvis tänkte vi oss en kort sammanfattning för att knyta ihop säcken, och validera intervjun!

8. Referenser

Adeniji, C.G., Agboola, G.M., & Motilewa, B.D. (2015). *Organizational Culture and Performance.* (International Conference on African Development Issues CU-ICADI). Hämtad från: http://eprints.covenantuniversity.edu.ng/5347/1/Paper%20101.pdf

Agarwal, Aayush & Gupta, Subhash & Choudhury, Tanupriya. (2018). *Continuous and Integrated Software Development using DevOps*. 290-293. 10.1109/ICACCE.2018.8458052.

Alvesson, M., Sveningsson, S. (2016). *Changing Organizational Culture.* London: Routledge, https://doi.org/10.4324/9781315688404

Atlassian. (2019). *What is DevOps?* Hämtad från: https://www.atlassian.com/devops

Babb, J,. Nørbjerg, J., Yates, D.J., & Waguespack, L.J. (2017). *The Empire Strikes Back: The end of Agile as we know it?*. Hämtad från: https://aisel.aisnet.org/iris2017/8/

Bass, L., Holz, R., Rimba, P., An Binh, T., Zhu, L. (2015). *Securing a Deployment Pipeline.* IEEE. Hämtad från: https://ieeexplore.ieee.org/document/7169443

Beng, Y.L., Khoong, W.L., Yip, W.T., & Fun, S.L. (2012). *Software Development Life Cycle AGILE vs Traditional Approaches.* Hämtad från: https://pdfs.semanticscholar.org/69b1/9ddc8a578f4c63d1dfe15252a465ee12fe5d.pdf

Beyer, B., Murphy, N. R., Rensin, D. K., Kawahara, K., & Thorn, S. (2018). *The Site Reliability Workbook: Practical Ways to Implement SRE*. California: O´Reilly Media, Inc.

Broberg, O. (2013). *Eight Ways Technology Is Changing Business.* Hämtad från: https://www.gomodus.com/blog/eight-ways-technology-changing-business

Burnes, B. (2014). *Managing change: A strategic approach to organisational dynamics.* Pearson Education Limited.

By, R. T. (2005). *Organisational change management: A critical review.* Journal of Change Management, Vol.5 (4), 369-380. Taylor & Francis Online. Doi: 10.1080/14697010500359250

Cohn, M. (2006). *Agile Estimating and Planning.* Upper Saddle River, NJ: Prentice Hall Professional Technical Reference.

Colavita, F. (2016). *DevOps Movement of Enterprise Agile Breakdown Silos, Create Collaboration, Increase Quality, and Application Speed.* Software Engineering for Defence Applications, Proceedings of 4th International Conference, vol 422. Springer, Cham.

Cuppettt, M.S. (2016). *DevOps, DBAs, and DBaaS - Managing Data Platforms to Support Continuous Integration.* California: Apress Media, LLC.

Dawson, P., Andriopoulos, C. (2017). *Managing Change, Creativity and Innovation.* (3.ed.) SAGE Publications Ltd.

DeGrandis, D. (2011). *DevOps: A Software Revolution in the Making?* Hämtad från: http://www.djaa.com/sites/default/files/devops.djaa_.pdf

Denison, D.R. (1997). *Corporate Culture and Organizational Effectiveness.* (2.ed.) Ann Arbor: Aviat.

DiVanna, J.A., & Rogers, J. (2005). *People - The New Asset on the Balance Sheet.* (1.ed.) Hampshire: Palgrave Macmillan.

Eficode. (2019) *Eficode quick guide DevOps.* Hämtad från: https://www.eficode.com/hubfs/documents/Eficode-English-Devops-Guide.pdf?hsLang=en

Farroha, B., & Farroha, D. (2014). *A Framework for Managing Mission Needs, Compliance, and Trust in the DevOps Environment.* Military Communications Conference (MILCOM 2014), IEEE.

Fitzgerald, B., & Stol, K.J. (2015). Continuous Software Engineering: A Roadmap and Agenda. Journal of Systems and Software. 25. 10.1016/j.jss.2015.06.063.

Gain, C. B. (2018). *Using CALMS to Assess an Organization´s DevOps.* Hämtad från: https://devops.com/using-calms-to-assess-organizations-devops/

Ghantous, G.B., & Gill, A. (2017). *DevOps: Concepts, Practices, Tools, Benefits and Challenges.* Hämtad från: https://aisel.aisnet.org/cgi/viewcontent.cgi?referer=https://www.google.se/&httpsredir=1&article=1191&context=pacis2017

Haight, J.M. (2015) *Automated Control Systems: Do they Reduce Human Error And Incidents? (*ASSE-07-05-20). Hämtad från: https://www.researchgate.net/publication/254508988_Automated_Control_Systems_Do_They_Reduce_Human_Error_And_Incidents

Highsmith, J., & Cockburn, A. (2001). *Agile Software Development: The Business of Innovation.* Computer. IEEE Computer Society, Vol 34 (9), 120-127. IEEE.

Humble, J., & Farley, D. (2010). *Continuous Delivery: Anatomy of the Deployment Pipeline.* Hämtad från: http://www.informit.com/articles/article.aspx?p=1621865&seqNum=2

Humble, J., & Molesky, J. (2011). *Why Enterprises Must Adopt DevOps to Enable Continuous Delivery.* Cutter IT Journal, 24(8), 6.

Hussaini, S. W. (2014). *Strengthening harmonization of Development (Dev) and Operations (Ops) silos in IT environment through Systems approach.* Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference, 178-183. IEEE.

Hüttermann, M. (2012). *DevOps for Developers:Integrate Development and Operations, The Agile Way.* Apress.

Jacobsen, D.I. (2002. *Vad, hur och varför? - Om metodval i företagsekonomi och andra samhällsvetenskapliga ämnen.* Studentlitteratur AB.

Kalliosaari, R.L., Mäkinen, S., Lwakatare, L.E., Tiihonen, J., & Männistö, T. (2016). *DevOps Adoption Benefits and Challenges in Practice: A Case Study.* Hämtad från: https://helda.helsinki.fi/bitstream/handle/10138/178485/RiunguKalliosaari2016DevopsAdoptionBenefits.pdf?sequence=1

Kartman, N. (2019). *How to implement DevOps with CAMS.* Squadex. Hämtad 2019-05-17 från: https://squadex.com/insights/how-to-implement-devops-with-cams/

Kim, G. (2016). *The DevOps Handbook.* (1.ed.) Portland: IT Revolution Press, LLC.

Leavy, P. (2017). *Research design: quantitative, qualitative, mixed methods, arts-based, and community-based participatory research approaches.* New York; London: Guilford Press.

Magnusson, E., & Marecek, J. (2015). *Doing Interview-based Qualitative Research: A Learner's Guide.* Cambridge: Cambridge University Press. doi:10.1017/CBO9781107449893

Malterud, K. (2014). *Kvalitativa metoder i medicinsk forskning - En introduktion.* Studentlitteratur AB.

Mezak, S. (2018). *The Origins of DevOps: What's in a Name?.* Hämtad från: https://devops.com/the-origins-of-devops-whats-in-a-name/

Mohammad, Adel & Alwadan, Tariq & Ababneh, Jafar. (2013). *Agile Software Methodologies: Strength and Weakness*. International Journal of Engineering Science and Technology. 5. Hämtad från: https://www.researchgate.net/profile/Adel_Mohammad/publication/257207655_Agile_Software_Methodologies_Strength_and_Weakness/links/02e7e524a8618f16d8000000.pdf

Mueller, E. (2019). *What Is DevOps?.* Hämtad från: https://theagileadmin.com/what-is-devops/

Poppendieck, M. (2002). *Principles of Lean Thinking.* Hämtad från: http://www.sel.unsl.edu.ar/ApuntesMaes/Anteriores/MetodologiasAgiles/LeanThinking.pdf

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit.* Boston: Addison-Wesley.

Poppendieck, M., & Poppendieck, T. (2007). *Implementing Lean Software Development: From Concept to Cash.* Boston: Addison-Wesley.

Przybyl, L. (2017). *Adopting DevOps Culture with CALMS.* Hämtad från: https://dzone.com/articles/adapting-devops-culture-with-calms

Rajpal, M. (2016). *Lessons Learned from a Failed Attempt at Distributed Agile.* Agile Processes in Software Engineering and Extreme Programming. XP 2016. Vol 251. Springer Cham.

Ravichandran, A., Taylor, K., Waterhouse, P. (2016).*DevOps for Digital Leaders: Reignite Business with a Modern DevOps-Enabled Software Factory*. DOI 10.1007/978-1-4842-1842-6

Seidman, I. (2006). *Interviewing As Qualitative Research: A Guide for Researchers in Education and the Social Sciences.* (3.ed.). New York: Teachers College Press.

Smeds, J., Nybom, K., & Porres, I. (2015). *DevOps: A Definition and Perceived Adoption Impediments.*: Agile Processes in Software Engineering and Extreme Programming. XP 2015. Vol 212. Springer, Cham.

Spaven, F. (2018). *DevOps and Error Monitoring: An Introduction to the CALMS model.* Hämtad från: https://thenewstack.io/devops-and-error-monitoring-an-introduction-to-the-calms-model/

Statista. (2018). *Extent of DevOps adoption by software developers worldwide in 2017 and 2018.* Hämtad 2019-05-18 från: https://www.statista.com/statistics/673505/worldwide-software-development-survey-devops-adoption/

Trubiani, C., Jamshidi, P., Cito, J., Shang, W., Jiang, Z.M., & Borg, M. (2019). *Performance Issues? Hey DevOps, Mind the Uncertainty.* Hämtad från: https://ieeexplore.ieee.org/document/8501933

Virmani, M. (2015). *Understanding DevOps & bridging the gap from continuous integration to continuous delivery.* Innovative Computing Technology (INTECH 2015), Fifth International Conference.

Werner, M. (2015). Gene Kim on High Performers and the Principles of DevOps - DZone DevOps. Hämtad från: https://dzone.com/articles/gene-kim-on-high-performers-and-the-principles-of

Willis, J. (2010). What DevOps means to me. Hämtad från: https://blog.chef.io/2010/07/16/what-devops-means-to-me/

Wiedemann, A. (2017). *A New Form of Collaboration in IT Teams - Exploring the DevOps Phenomenon*. PACIS 2017 Proceedings 82.

Wiedemann, A., Wiesche, M., Gewald, H., & Kremar, H. (2018). *Integrating DevOps within IT Organizations - Key Pattern of a Case Study.* Hämtad från: https://dl.gi.de/bitstream/handle/20.500.12116/18909/PVM2018_15.pdf?sequence=1&isAllowed=y