LUNDS UNIVERSITET

Ekonomihögskolan

Institutionen för informatik

Vad karaktäriserar DevOps?

En Kvalitativ Studie

Kandidatuppsats 15 hp, kurs SYSK02 i informationssystem

Författare: Gustav Friberg
Joel Joelson-Tui
Sebastian Mossberg

Handledare: Odd Steen

Examinatorer: Benjamin Weaver
Bo Andersson

Vad karaktäriserar DevOps?: En Kvalitativ Studie

Författare: Gustav Friberg, Joel Joelson-Tui, Sebastian Mossberg

Utgivare: Inst. för informatik, Ekonomihögskolan, Lund universitet

Dokumenttyp: Kandidatuppsats

Antal sidor: 165

Nyckelord: DevOps, Continuous Delivery, CAMS, Agile

Sammanfattning (Max. 200 ord):
Användandet av agila utvecklingsmetoder inom IT-organisationer har utvecklats till att inkludera driftsidan (Ops). DevOps är en rörelse för ökat samarbete mellan utvecklare (Dev) och driftpersonal (Ops) med rötter i agila utvecklingsmetoder där de fyra värdegrunderna är kultur, automation, mätning och delning. Rörelsen lockar företag i hopp om snabbare time-to-market och en bättre möjlighet till att anpassa sin organisation efter ständigt skiftande krav. Vad fenomenet innebär är dock svårdefinierat. Tidigare studier har gjorts för att definiera och karaktärisera fenomenet utifrån litteratur, men det råder brist på empirisk forskning inom området, vilket den här studien adresserar.

Studien visade att DevOps-organisationer karaktäriseras av insyn och organisatorisk transparens, där alla berörda av leveranskedjan för mjukvara ska ha ett helhetsperspektiv. Detta sker genom en rad processer, aktiviteter, värderingar och verktyg för att fostra kunskapsdelning och samarbete mellan avdelningar där liten vikt läggs på roller och bred kunskap prioriteras.

Ordlista:

Development, Dev: *Utvecklingsavdelningen, Utvecklare*: Varierande omfång, men syftar till den avdelning som står för utvecklandet av produkten. Inkluderar utvecklare, testare och QA.

Operations, Ops: *Driftpersonal, Driftavdelningen*. Hanterar den underliggande infrastrukturen som exempelvis nätverk och serverplattformar där personalen kan räknas till administratörer, såsom nätverksadministratör, lagringsadministratör, administratör för virituell serverplattform.

Release: En version ut av programvaran som görs tillgänglig för användare.

Deploy, Deployment: *Produktionssättning* av en förändring eller ny mjukvara. Förändring flyttas till en produktionsmiljö tillgängligt för externa (kund).

Versionshantering, Source Code Management (SCM): Mjukvara som tillåter återskapande av tidigare versioner av kod där ändringar kan spåras.

Commit: Att skicka in sin kod till en versionshanterare.

Automatisk byggning: Kompilation och sammanställning av programvaran efter att en kod har *commitats* till versionshantering. Dvs. så fort det sker en förändring kommer det automatiskt produceras programvaruartifakter.

Automatisk testning - Efter att kod blivit commitat och blivit byggt testas bygget automatiskt av fördefinierade test program.

Byggscript: Ett byggscript är ett eller flera script som används för att kompilera, testa och produktionssätta mjukvara. (Duvell 2009, sid. 10)

SDLC (Software Development/Delivery Lifecycle) - Leveranskedjan, omfattar alla steg i utvecklingsprocessen från krav till deployment och övervakning.

Innehåll

Figurer

Tabeller

1 Introduktion

1.1 Problemformulering

Ordet DevOps är en hopslagning av Development och Operations som myntades i samband med konferensen DevOpsDays i Ghent 2009, dedikerad till ett bättre samarbete mellan utvecklare och driftpersonal (Debois, 2009; Brunnert et al. 2015). DevOps har rötter i agila utvecklingsmetoder och flera tyckare argumenterar för att DevOps kan ses som en förlängning utav den agila rörelsen för att även inkludera Operations (Hütterman, 2012, sid. 20-26; Lwakatare et al., 2015; DeGrandis, 2011). Termen saknar dock en klar definition (Smeds et al. 2015) och det råder delad mening kring vad DevOps innebär och vad det är tänkt att omfatta (Roche 2013; Dyck et al. 2015).

Det finns ett läger som argumenterar för att själva definitionen är oviktig (Ducy, 2015) medan andra är av åsikten att en definition är av central vikt för rörelsen och dess framtid (Hendren 2016). Samtidigt argumenterar vissa att det existerar en nästintill universell uppfattning till vad DevOps innebär (Minick 2015).

Trots termens otydliga definition har rörelsen fått stor dragningskraft där man kan se att allt fler och fler förändrar sina organisationer mot DevOps (Rivera & van der Meulen 2015) i samband med att företag har insett vikten av att snabbt leverera innovativ mjukvara för att snabbt kunna möta ändringar som sker på marknaden (Dyck et al. 2015).
Continuous Delivery (CD) är en agil metod som har fått stor popularitet då det medför en snabb, automatiserad produktionssättning som möter dessa krav, där DevOps sagts vara ett koncept för organisationer att bygga en företagskultur som kan hantera de allt mer snabba och frekventa leveranser som CD ger upphov till (Wahaballa et al. 2015).

Den ökande populariteten kring rörelsen i industrin avspeglas inte i vetenskapen där bristen på vetenskaplig litteratur som beskriver fenomenet är påtaglig. Erich et al. (2014) har utfört en litteraturstudie av fältet där de ämnar att svara på vad som karaktäriserar DevOps, där de fann att litteraturen till stor del var av dålig kvalitet men att DevOps kan ses som ett konceptuellt ramverk jämförbart med agil systemutveckling som möjliggör implementationen av Continous Delivery. Författarna underströk också vikten av att DevOps inte kan ses som en lösning som ser likadan ut för varje organisation utan bör ses som en artefakt unik till organisationen.

Smeds et al. (2015) utför, likt Erich et. al (2014), en litteraturstudie för att bryta ut vad som kännetecknar och karaktäriserar DevOps. De finner i sin slutsats att DevOps är ett mångfacetterat koncept och att DevOps definition fortfarande kräver forskares

uppmärksamhet men föreslår själva en definition av DevOps som en uppsättning *engineering process capabilities* som möjliggörs genom kulturella och tekniska egenskaper i organisationen. Vidare har de även utfört en kvalitativ studie för att undersöka vilka farhågor kring förhinder en organisations anställda kan ha vid införandet av DevOps, där de bland de elva identifierade potentiella förhindren bland annat fann en otydlig definition och innebörd med DevOps var orsak till oro hos den undersökta organisationen.

Då vi, likt Erich et al. (2014) och Smeds et al. (2015), ser att tillgången till högkvalitativ litteratur är bristfällig och att det i många fall saknas en vetenskaplig undersökning av fenomenet, ses ett behov av en empirisk undersökning av vad DevOps kan innebära i en organisation. Detta för att möjliggöra en större förståelse av fenomenet, men även öka tillgängligheten för vidare forskning då den rådande litteraturen, som till största del består av industriexperters uttalanden, begränsar möjligheten och validiteten för framtida studier.

1.2 Syfte

Vi vill utforska vad DevOps innebär för utövare och deras organisationer idag och vilka egenskaper och principer som deras DevOps-organisationer innefattar. Med detta hoppas vi belysa en högst aktuellt rörelse och paradigm till hur organisationer väljer att arbeta med IT och skänka en förståelse och studerbarhet av fenomenet.

1.3 Frågeställning

Studien ämnar belysa vad som karaktäriserar en organisation som arbetar med DevOps och vilka egenskaper, arbetsmetoder och principer som återfinns i en sådan organisation, vilket formulerar forskningsfrågan:

Vad karaktäriserar DevOps-organisationer?

1.4 Avgränsningar

Med anledning av DevOps svårdefinierade natur har vi i denna studie inte för avsikt att ge en definitiv definition, utan istället presentera vad som kan observeras i organisationer som utövar DevOps.

DevOps är inte någonting som ser likadant ut för alla organisationer (Erich et al. 2014) och studien avgränsas av att den inte kommer finna alla organisatoriska karaktärsdrag som kan tänkas finnas. Studiens omfång tjänar istället syftet av att ge en inblick till ett relativt outforskat ämne. I egenskap av att vara en kvalitativ studie med ett urval om tre företag gör inte studien anspråk på en extern generaliserbarhet till en omfattande population.

2 Teori

För att kunna svara på frågan vad som karaktäriserar en DevOps-organisation behövs förståelse av var DevOps kommer ifrån, vad det innebär och vad det är tänkt att främja, samt vilka motiven är till att fler och fler organisationer väljer att introducera det i sina verksamheter.

Detta kapitel går igenom DevOps bakgrund, vilka tidigare metodiker och tekniker det stödjer sig på samt varför det är relevant idag. Teorin är en sammanställning av akademisk litteratur tillsammans med åsikter från framstående personer inom DevOps-rörelsen och Agil systemutveckling.

Först ges en bakgrund till silo-effekten som är en viktig del av den problembild DevOps är tänkt att motarbeta. Vidare beskrivs agil systemutveckling och det agila manifestet för att presentera den bakgrund och filosofi DevOps konceptet har vuxit fram ifrån och delar egenskaper med. Därefter beskrivs Continuous Delivery och hur det utgör en central del av DevOps. Avslutningsvis sker en summering av olika tolkningar och beskrivningar av DevOps från flera författare med syftet att presentera de olika synsätt som vi gör vår egen definition utifrån, samt en genomgång av det teoretiska ramverket CAMS.

2.1 Siloeffekten

Phil S. Ensor myntade 1988 uttrycket “functional silo syndrome” (funktionssilosyndom) för att förklara en ineffektiv och vanligt förekommande organisationsstruktur där avdelningarna arbetade separat från varandra i så kallade “silos” och kommunikation vanligtvis gick mellan avdelningar och ledning snarare än mellan avdelningarna. Denna struktur kritiserades då för att skapa tråkiga jobb som var lätta att kontrollera, men lämnade lite utrymme för innovation och utveckling hos medarbetarna (Ensor 1988).

Debois (2008) ser att ett konfliktområde mellan utveckling (Dev) och drift (Ops) är en vanlig företeelse. Denna splittring kan skapa silorelaterad problematik och spänningar mellan de två lägren där utvecklingsavdelningen arbetar för ständig förändring i form av uppdateringar och ny funktionalitet, medan drift arbetar för hålla produkten så stabil som möjligt för att undvika driftstörningar (Brunnert et al. 2015). Problemet i en sådan miljö har kommit att kallas “att kasta det över väggen” (Humble 2011), där utvecklare arbetar med förändringar och programvara avskiljt, för att sedan lämna över den färdiga programvaran till drift. Driften, som inte tidigare varit involverade i utvecklingen, har nu ansvaret att produktionssätta mjukvaran (Pfifer 2011).

Vidare beskriver Wettinger et al. (2014) några av de problem som kan uppstå mellan dessa yrkesgrupper, och som DevOps är tänkt motarbeta. Intressenter till en produkt ser ofta en stor rädsla för föränding efter leverans, vilket kan leda till byråkratiska och långsamma processer för att implementera nya förändringar. Vidare ses en problematik i att leveranser uppfattas som riskabla då varken development eller operations har förtroende för att mjukvaran kommer att fungera i produktionsmiljön. När problem väl uppstår är det lätt att undanbeda sig ansvaret och istället beskylla en annan gruppering. Med andra ord skyller utvecklaravdelningen på driftavdelningen och vice versa, därför att gemensamma prioriteringar saknas.

2.2 Agila utvecklingsmetoder

År 2001 samlades 17 representanter från olika lättviktsmetoder för att tillsammans skriva det agila manifestet (Beck et al. 2001). I manifestet värderas individer och interaktioner framför processer och verktyg, fungerande programvara framför omfattande dokumentation, kundsamarbete framför kontraktsförhandling och anpassning till förändring framför att följa en plan. Manifestet har beskrivits som en motreaktion mot de traditionella utvecklingsmetoderna där vattenfallsmodellen och extensiv planering

Agila utvecklingsmetoder har sedan dess fått en stor inverkan på hur mjukvaruutvecklare arbetar idag (Denning 2015; Rodríguez et al. 2015) där SCRUM och XP (extreme programming) är populära metoder som används för att kunna möta ständigt skiftande och rörliga krav och kan ses som en motreaktion till de mer traditionella utvecklingsmetoderna (Dybå & Dingsøyr, 2008). Huvudmålet var att förbättra en organisations kapacitet till förändrade krav i från kund genom att göra den mer agil. (Rodríguez et al. 2015)

Vid sidan om SCRUM står också kanban som ett viktigt arbetssätt bland agila utvecklingsmetoder. Detta är ett sätt att prioritera arbetsuppgifter med visuellt stöd som tillåter hög synlighet. Lappar med uppgifter sätts upp på en vägg och ska gå horisontell riktning gå igenom ett antal faser steg för steg. Begränsningar sätts på hur många som uppgifter som får befinna sig i en och samma fas samtidigt för att säkerställa att uppgifter faktiskt utförs hela vägen. Verktyget möjliggör prioritering av de uppgifter som anses nödvändiga för att ro projektet i hamn och för varje steg finns regler som eliminerar svinn, exempelvis ska inte större specifikationer framställas än vad som kan kodas och testare ska inte utföra fler tester än vad som kan produktionssättas. Kanban har visat sig vara ett effektivare arbetssätt vid jämförelse med SCRUM i mindre företag (mindre än 50 anställda), medan de är ungefär lika effektiva i större (Lei et al. 2015).

Metoderna främjar bland annat heterogena arbetsgrupper vars medlemmar ska komplettera varandra i form av kunskap och olika perspektiv. Dessa ska bli tilldelad den auktoritet som krävs för att kunna genomföra sitt arbete (Dybå & Dingsøyr 2008; Lee & Xia 2010). Vi ser numera företag undergå agila transformationsprocesser för att anpassa för agila metoder (Gandomani et al. 2015) där utvecklare, testare och QA arbetar tillsammans i kortare iterationer med målet att ha mer frekventa kodintegrationer (Fitzgerald & Stol 2015) i kontrast mot traditionella utvecklingsmetoder som föredrar extensiv planering (Hütterman 2012, sid. 7; Rodríguez et al. 2015).

2.3 Continuous Integration & Continuous Delivery

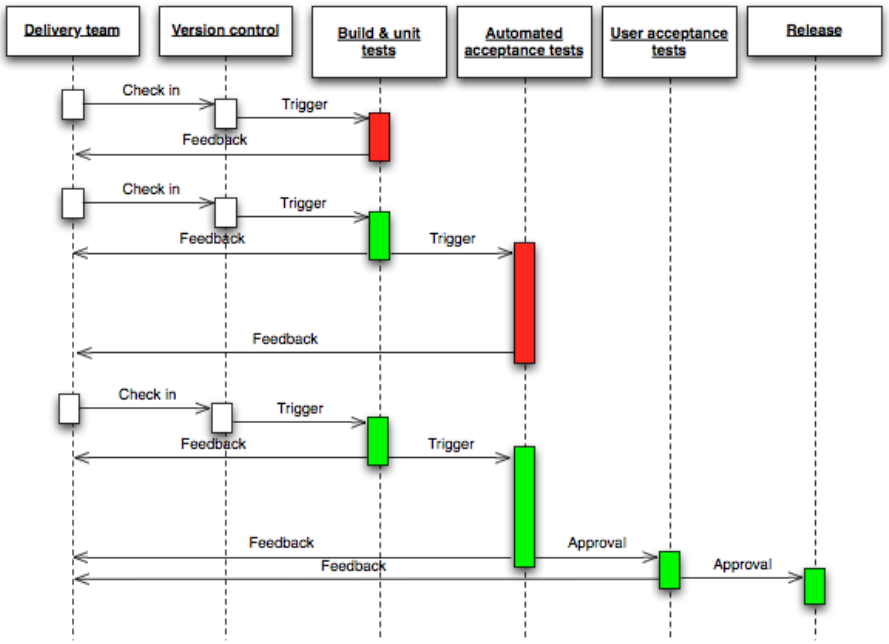
Övergången till agila utvecklingsmetoder har gjort Continuous Integration (CI) praktiker populära (Huo et al. 2004; Rodríguez et al. 2015), och har sitt ursprung i XP (Beck 1999 citerat i Ståhl & Bosch, 2014), där utvecklare uppmanas att integrera, bygga och testa sin kod ofta för att hantera fel medan de fortfarande är små och hanterbara (Brandtner et al. 2014; Rodríguez et al. 2015) och ses ofta som en nyckelkomponent inom agil utveckling (Stolberg 2009). Detta har gett upphov till ökad reliabilitet och frekvens av *releaser* och ökad produktivitet för utvecklare (Ståhl & Bosch 2013).

Duvall (2007, sid. 4-5) beskriver ett typiskt CI-scenario som att en utvecklare i en projektgrupp *commitar* sin kod till en versionshanteringsserver. Efter att kodversionen har uppdateras exekveras en automatisk byggnad utav koden. Med en byggnad menar Duvall en sammanställning av kompilation, testning och verifiering av att programvaran fungerar i sin helhet, som ofta sker med hjälp av ett byggscript och därav också automatiskt. Vidare genereras feedback till utvecklaren och andra projektmedlemmar.

CI associeras med de verktyg för till exempel versionshantering och mjukvara som sköter den automatiska byggningen och testningen, men enligt Fowler (2006) kan CI användas utan några verktyg alls, istället är det uppmaningen att integrera ofta och tidigt snarare än senare, då integrationen av programkod har visat sig vara en besvärlig och ofta tidskrävande process. Vidare är det sagt att CI inte har ett fastslaget tillvägagångsätt, utan praktiken kan variera i hur man realiserar det, vilket framgår i Ståhl & Bosch's artikel (2014).

Framstegen inom CI har möjliggjort vad som kommit att kallas Continuous Delivery(CD) (Humble & Farley 2011, sid. 4; Olsson et al. 2012), först myntat av Jez Humble och David Farley som inspirerades av det agila manifestets principer (Humble 2010). Continuous Delivery tar den agila principen om att leverera tidigt och leverera ofta (Olsson et al. 2012) och CI konceptet ett steg ytterligare genom att den kod som integrerats, byggts och passerat de automatiska testerna, ska kunna sättas i produktion. (Claps et al. 2015).

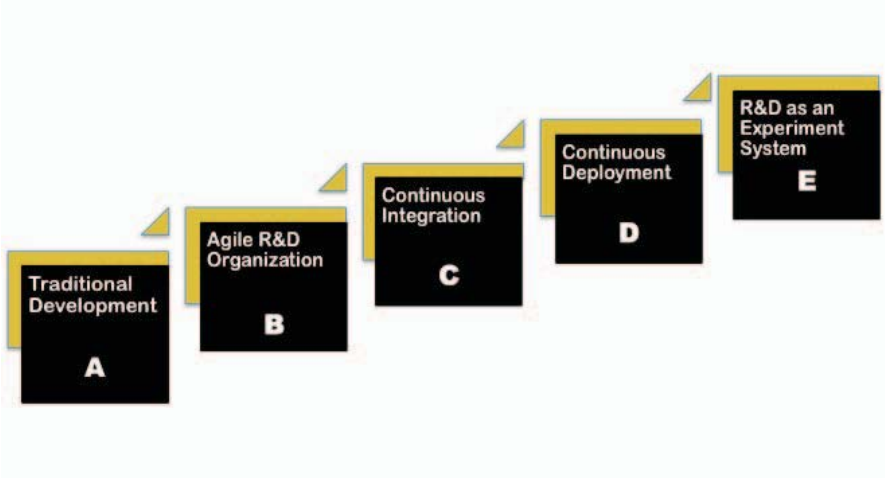
På så sätt minskas arbetsbördan för vad som tidigare varit en omfattande process, att gå från utveckling till produktion (Rodríguez et al. 2015). Humble & Farley (2011, sid.6) motiverar användningen av automation istället för manuella produktionssättningar med att fel alltid kommer uppstå med extensiva steg-för-steg manualer. Med automation blir processen repeterbar och därför pålitlig, samtidigt som beroendet av experter vid produktionssättningstillfället minskar. Automationen utförs likt CI med hjälp av verktyg och skript. Flödet från integrerad kod till produktionsmiljö, dvs. vad som kallas *delivery pipeline*, beskrivs i figur 2.1.



Figur 2.1. Deployment pipeline (Humble 2010)

Det görs en avskiljning mellan Continuous Delivery och Continuous Deployment, där Continuous Deployment är principen att förändringen direkt hamnar i produktionsmiljö, medan Continous Delivery gör det möjligt att sätta förändringen i produktion, ofta till den grad av ett enkelt knapptryck (Humble 2010). Termerna innebär således stora likheter, där Continuous Delivery automatiserar *delivery pipeline* fram till det slutgiltiga steget, att leverera förändringen ut i produktionsmiljö, medan Continuous Deployment syftar till en fullt automatiserad process.

Resan en organisation tar från att introducera ett agilt arbetssätt till Continuous *Deployment* beskrivs i Olsson et al. (2012) som skapat en konceptuell modell över hur en typisk mognadsprocess för en organisation ser ut (fig. 2.2). I ett första steg introduceras agila arbetsmetoder i verksamheten där nästa målsättning är att uppnå Continuous Integration. Först när organisationen anses mogen och använder sig av CI blir steget till CD möjligt.



Figur 2.2 Konceptuell modell av mognadsprocess för Continuous Deployment (Olsson et al., 2012)

2.4 DevOps

Med de ökade kraven på snabb leverans av mjukvara väljer organisationer att implementera Continuous Delivery som till stor del automatiserar processen och kortar ledtiden från idé till produktion. För att kunna realisera CD i sin organisation skriver Humble och Farley (2011, sid. 28) att fler individer och avdelningar bör vara involverade i leveranskedjan och kunna se hur projektet fortlöper, vilket de påpekar är en av grundprinciperna i DevOps-rörelsen. Vidare beskriver Humble och Molesky (2011) att den högsta mognaden av Continous Delivery innebär vetskapen att produktionssättningar kan ske riskfritt. De tillfällen som tidigare varit en omfattande process blir till en icke-händelse då det sker såpass ofta, men för att uppnå den mognadsnivån krävs det att organisationen anammar DevOps.

> *Achieving these benefits within enterprises requires the discipline of devops: a culture of collaboration between all team members; measurement of process, value, cost, and technical metrics; sharing of knowledge and tools; and regular retrospectives as an input to a process of continuous improvement. (Humble och Molesky 2011, sid. 11-12)*

Wettinger et al. (2014) beskriver DevOps som en uppstående paradigm vars mål är att motverka den traditionella distansen mellan utvecklare och drift, då det orsakar längre ledtider innan ny programvara når produktion samtidigt som organisationer idag ser allt större krav från kunder att leverera fortare och oftare. Konceptet att DevOps har som mål att öka

samarbetet mellan utvecklare och drift för att uppnå en bättre leveranskedja är väletablerat (Wettinger et al. 2014; Virmani 2015), vilket även kan ses i Humble och Molesky‘s (2011) beskrivning:

> *DevOps is about aligning the incentives of everybody involved in delivering software, with a particular emphasis on developers, testers, and operations personnel. A fundamental assumption of DevOps is that achieving both frequent, reliable deployments and a stable production environment is not a zero-sum game. DevOps is an approach to fixing the first three problems listed above through culture, automation, measurement, and sharing. (Humble & Molesky 2011, sid. 8)*

DevOps är på så sätt en möjliggörare för Continuous Delivery, praktiken att kunna produktionssätta förändringar kontinuerligt och riskfritt, genom en kombination av kultur, automation, mätning och delning. Dyck et al. (2015) utför i sin artikel en kritisk analys av flera existerande definitioner, för att klargöra skillnaden mellan *Release Engineering* och DevOps, där de ger definitionen:

> *DevOps is an organizational approach that stresses empathy and cross-functional collaboration within and between teams – especially development and IT operations – in software development organizations, in order to operate resilient systems and accelerate delivery of changes. (Dyck et al. 2015, sid. 3)*

Författarna ger utöver definitionen en ytterligare förklaring att DevOps är ett organisatoriskt tillvägagångsätt som betonar vikten av bra kommunikation och empati mellan grupperingar. De medger att definitionen kan sakna validitet då en universell definition som tillgodoser alla perspektiv är svår att hitta och att definitionen gav upphov till ytterligare diskussion.

Hütterman (2011, sid. 4) fångar problematiken med att en definition är svårnådd och erbjuder en mer övergripande förklaring att DevOps kan ses som en blandning av välkända avancerade tekniker och nya innovativa tillvägagångssätt för problem inom produktionssättning av mjukvara och drift.

Smeds et al. (2015) definierar i sin litteraturstudie DevOps som en uppsättning organisatoriska *kapabiliteter*. Till kapabiliteterna hör bl.a. CI och CD tillsammans med en kontinuerlig övervakning och feedback. Dessa kapabiliteter möjliggörs genom kulturella och tekniska *möjliggörare* där vi kan se bl.a. delade mål, obesvärad kommunikation och automerad produktionssättning.

Även ifall en slutgiltig definition verkar till synes omöjlig, står det klart att DevOps har ett fokus på att genom organisatoriska *principer*, i form av vad som kallas kultur, automation, mätning och delning, kunna anpassa organisationer för att bättre nå målet om frekventa och riskfria produktionssättningar. Vi har i studien valt att använda oss av följande synsätt av DevOps, som utgörs av en summering av de nämnda definitionerna och beskrivningarna:

DevOps är en strategi för mjukvaruutveckling och produktionssättning av mjukvara där målet är en effektivare leveransprocess. Detta görs genom kultur, automation, mätning och delning.

2.5 Teoretiskt Ramverk: CAMS

Damon Edwards och John Willis, pionjärer inom DevOps, myntade under DevOpsDays konferensen i Mountainview 2010 akronymen CAMS för att definiera de grundläggande värderingarna inom DevOps (Willis 2010). Dessa var kultur (*culture*), automation, mätning(*measurement*) och delning (*sharing*) (DevOpsDictionary 2010). Dessa värderingar har fått ett stort genomslag och ligger till grund för DevOps definition i välciterad litteratur inom området (Humble & Molesky 2011; Hütterman 2012, sid. 4; Bang et al. 2013; Erich et al. 2014).

Värdegrunderna är, trots sitt stora genomslag, i vårt tycke många gånger svårförstådda och kommer nedan summeras från olika författares beskrivningar. Värdegrunderna är i sitt ursprung ofta kort formulerade och av generaliserande karaktär.

2.5.1 Kultur

Den första värdegrunden innefattar en kultur av samarbete mellan grupperingar (Bang et al. 2013) där Humble och Molesky (2011) beskriver vikten av att drift är inkluderade i både utvecklingen och övergången till produktion där de ger de praktiska råden att personer från drift regelbundet ska delta i planeringsmöten och projektgrupper, samtidigt som utvecklare ska rotera i driftgrupper och representanter från utvecklingssidan regelbundet ska hålla möten med driftpersonal. Avslutningsvis nämner de att utvecklare ska hålla jour och kunna bistå driftgruppen vid felsökning av incidenter i produktion.

Swartout (2012, sid. 72-73) nämner inte CAMS explicit men redogör för en kultur av öppen, ärlig och modig dialog och pekar på att öppenhet och ärlighet är kritiskt för implementation av Continuous Delivery och DevOps där alla involverade i leveranskedjan måste ha ett forum.

“Everyone involved in the product delivery process, from developers and testers through change and release controllers to Product Owners and Senior Managers, must have a forum, which they can use to share their thoughts. Not only that, they need to be actively encouraged to do so, which can be quite a challenge.” (Swartout 2012, sid 72-23)

Med modig dialog menar Swartout att de involverade i leveranskedjan måste få tillåtelse att uttrycka sina åsikter utan risk för reprimander, oberoende om vart i organisationsstrukturen personen befinner sig.

Hütterman (2012, sid. 4, 8) nämner CAMS som, likt Willis (2010), gör i sin beskrivning av värdegrunden en referens till agila manifestet med “People over processes and tools. Software is made by and for people” och skriver att det måste infinnas en kultur av tillit och gemenskap inom organisationen för ett framgångsrikt DevOps-initiativ.

Feitelson et al. (2013) såg i sin studie om utveckling och produktionssättning på Facebook att en kultur av att utvecklare som själva tar ett personligt ansvar för sin kod, och en kultur av gemensamt ansvar för produkten, var kritiskt för DevOps-organisationen. Ett större kvalitetsansvar för utvecklaren ses även i kartläggningen Rodriguez et al. (2015) utfört av Continuous Deployment där de uppmanar utvecklare att ansvara för kodens kvalitet, testernas kvalitet och den driften av den kod som är skapad (sid. 15).

Av beskrivningarna bedömer vi att värdegrunden kultur i en DevOps-kontext omfattar hur kommunikationen och rollers involvering sker i organisationen, hur ansvar tas och fördelas samt vilka värderingar som infinnes.

2.5.2 Automation

Fitzgerald & Stol (2015) beskriver automation inom DevOps som en fullständig automation av byggning, produktionssättning och testning av mjukvaran. Med hjälp av detta kan man uppnå kortare ledtider, och därmed snabbare leverans och feedback från slutanvändare. Enligt Humble & Molesky (2011) bör det användas en *deployment pipeline* för att uppnå detta, som tidigare beskrivet under *Continuous Delivery*.

Hütterman (2012, sid. 111) ser ökad feedback som det viktigaste inom automation, men nämner även färre risker och hög repeterbarhet som starka argument till att använda sig av det. Delning av versionshanteringssystem för kod (SCM) mellan utvecklare och drift ses som ett verktyg som kan förbättra samarbetet mellan avdelningarna (sid. 121), och han förespråkar ärendehanteringssystem såsom JIRA för att uppgifter skall kunna spåras mellan avdelningarna (sid. 128).

2.5.3 Mätning/Utvärdering

Enligt Humble & Molesky (2011) kan mätningar göras på två nivåer: på hög nivå, som omsättning eller uppfyllnad av affärsmål, eller på låg nivå med väl utvalda indikatorer på prestanda. Författarna varnar för att man skall vara försiktig med att använda tester på låg nivå då det kan påverka hur utvecklarna arbetar. Som exempel nämner de att enhetstester för kod kan modifieras för att koden skall gå igenom testet om detta är något utvecklaren utvärderas utefter (Humble & Molesky 2011, sid. 8).

Swartout (2012, sid. 105-108) talar om vikten av att ständigt mäta och analysera framstegen inom ett DevOps-arbete. Han uppmanar organisationer att mäta sina “best practices” och medger att detta kan låta som ett märkligt koncept, men nämner test för kodkvalitet, kodkomplexitet och enhetstest som befintliga verktyg för att utföra detta. Andra mätdata som kan visa på effektiviteten av ett DevOps-arbete kan vara frekvens av *commits* av kod. Fitzgerald & Stol (2015) understryker vikten av att kunna mäta sin leveranskapacitet och att nya målsättningar för detta endast kan sättas genom att först utföra mätningar.

2.5.4 Delning

Värdegrunden Delning har en stark anknytning till organisatorisk kultur vilket Hütterman (2012, sid. 8) pekar på när han beskriver en kultur avdelning:

> *Everything starts with how people perceive one another. That is, does the company have an “us vs. them” culture or a “we” culture? Thus, DevOps centers on the concept of “sharing”: sharing ideas, issues, processes, tools, and goals.*

Humble och Molesky (2011) understryker också vikten av en gemenskap där de ser att utvecklare och driftpersonal gemensamt bör fira framgångsrika produktionssättningar. Swartout (2012, sid. 81) håller med om detta och tillägger att belöning av en gemensam framgång kan inge en känsla av samarbete och DevOps-anda istället för att belöna enskilda prestationer.

Humble och Molesky (2011) betonar också vikten av delning av kunskap och verktyg. Att dela kunskap kan enligt författarna exempelvis vara att förvarna respektive avdelning inför en stor driftsättning.

Hütterman (2012, sid. 27) pekar på att ifall en tjänst eller applikation slutar att fungera är det meningen att så många som möjligt ska kunna felsöka problemet och att det endast kan ske ifall det finns ett bra kunskapsutbyte. De verktyg som delas kan innefatta verktyg som hanterar miljöer och infrastruktur inom organisationen, något som de ser som stor del av vad DevOps karaktäriseras utav (Humble & Molesky 2011, sid. 9).

3 Metod

I detta kapitel presenteras motiveringar till forskningsmetoden, som bestått av en kvalitativ ansats där semi-strukturerade intervjuer utförts. Intervjuerna har sedan kodats enligt ett kodningsschema inspirerat av Allan's (2003) kodningsschema.

3.1 Kvalitativ ansats

Jacobsen betonar vikten av att välja en undersökningsuppläggning, och nämner två typer av uppläggningar: deskriptiva och kausala (Jacobsen, 2002, sid. 56). Studiens problemställning är deskriptiv då den vill ta reda på vad som karaktäriserar en DevOps-organisation och ge en detaljrik beskrivning av fenomenet.

Vidare bör forskargruppen ta ett beslut ifall studien skall ske med en extensiv uppläggning, där få variabler undersöks men många undersökningsenheter, eller intensivt; få undersökningsenheter men många variabler. Den extensiva undersökningen kommer med fördelen att uppläggningen kan ge studien en större generaliserbarhet sett till populationen och att med en stor bredd kan undersökningen ge en mer exakt beskrivning av fenomenet. Men med nackdelen att ett bredare upplägg begränsar studiens möjlighet till detaljrikedom (Jacobsen, 2002, sid. 99-101). Alternativet, en intensiv uppläggning, erbjuder däremot forskargruppen en möjlighet till större detaljrikedom och nyanseringar av det studerade fenomenet, där Jacobsen beskriver utgångspunkten för en intensiv uppläggning är att finna individuella variationer och skillnader i uppfattningen av ett fenomen men samtidigt få fram likheterna. Vidare skriver han att gå på djupet är ett försök att skaffa sig totalförståelse och att ge bästa möjliga beskrivning av fenomenet (Jacobsen, 2002, sid. 94).

I vårt val av utformningen såg vi till studiens problemställning och syfte. Uppläggningen var deskriptiv och ämnade att ge en fenomenet en så rik beskrivning som möjligt för att skänka en större förståelse i ett område med en bristande tillgång till litteratur. Samtidigt kunde vi se att fenomenet saknade en gemensam uppfattning där det länge kallats efter en tydligare definition. Behovet av en studie med hög generaliserbarhet fanns, samtidigt som behovet av en mer detaljrik beskrivning.

Studien utfördes med intensiv ansats då vi såg tillgången till undersökningsenheter som begränsande och skulle på så vis påverka studiens generaliserbarhet, dvs. att resultatet skulle bli svårt att överföra till en större population, men att det samtidigt skulle det ge oss tillgång till en djupare förståelse och en större grad av detaljrikedom, vilket vi ansåg var av stor betydelse. Mer specifikt utformades studien mot vad Jacobsen beskriver som en *Små-N* studie, där forskarna väljer ut ett fåtal enheter från olika kontexter. Styrkan i detta förfarande

är att det ger en rikare bild av ett undersökt fenomen eftersom beskrivningar ofta skiljer sig åt beroende på miljö. De olika kontexterna i den här studiens fall var att det utfördes intervjuer från tre olika organisationer med varierande DevOps initiativ. Denna intensiva utformning menar Jacobsen lämpar sig bäst när forskaren vill ha en rik beskrivning av ett visst fenomen. (Jacobsen, 2002, sid. 99)

Efter valet av uppläggning bör forskaren ta ställning till huruvida en kvalitativ eller kvantitativ ansats ska tas och vilken typ av metod som ska användas. En kvantitativ metod ger data som kan mätas med siffror, ofta införskaffade med hjälp av frågeformulär där frågor med tydliga svarsalternativ. Detta förutsätter att studien har en klar utgångspunkt och att den tydligt kan avgränsas med en god kunskap om det undersökta ämnet. (Jacobsen, 2002, sid. 143-146). Kvalitativa metoder ämnar istället utforska ett ämne utifrån en mer öppen ansats, där forskargruppen intresserar sig av processen och människors tolkning av den (Maxwell, 1996, sid. 19).

Den här studien har utförts med en kvalitativ ansats, motiverat av att en kvalitativ metod lämpas sig bäst när forskaren har en liten kunskap om det studerade fenomenet och vill veta vad fenomenet innehåller (Jacobsen 2002, sid. 150). En kvalitativ metod erbjuder större flexibilitet än den kvantitativa och möjliggör gradvis anpassning av metoden för datainsamling allt eftersom studien fortlöper (Maxwell 1996, sid. 17; Jacobsen 2002, sid. 38-39). Flexibiliteten skulle visa sig var värdefull för studien då det gav möjlighet att anpassa intervjuguiden efter att nya fynd upptäckts i förgående intervjuer och gav möjligheten att vidare undersöka upptäckten.

Miles och Huberman (1994, sid.17) talar om vilken grad av struktur som är lämpligt att arbeta med i en kvalitativ studie. Här höjer författarna ett varningens finger för oerfarna forskare som, liksom vi, har begränsat med tid. En helt induktiv ansats var således inte realistiskt för studiens omfång, samtidigt rekommenderar de inte heller en starkt strukturerad utformning utan förespråkar en mellanväg. På så vis höll undersökningen en öppen utformning som erbjöd detaljrik data, guidad av ett mindre omfattande teoretiskt ramverk. Samtidigt förhöll sig undersökningen fortfarande inom ramen för vad som kunde förväntas omfatta DevOps.

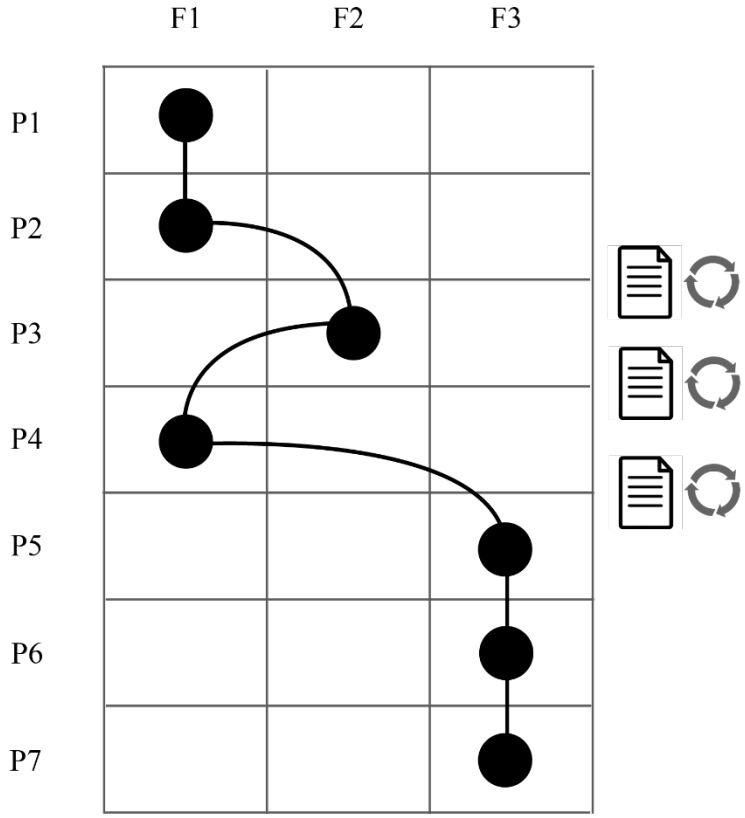
3.2 Öppna intervjuer

Studien utfördes med öppna intervjuer som instrument för empiri då personens berättelser av miljön var av intresse (Jacobsen 2002, sid. 160). Datainsamlingen utfördes iterativt med dataanalysen i en kvalitativ empirisk sekvens (Steen 2016) där vi i enlighet med Maxwells rekommendationer påbörjade dataanalysen av data direkt efter insamlingen utförts. Dataanalysen påverkade således efterföljande intervjuernas strukturerade form gradvis (Maxwell 1996, sid. 77).

De första intervjuerna utfördes till viss grad semi-strukturerat, styrt av en enklare intervjuguide utformad efter det teoretiska ramverket CAMS. Efter dataanalysen växte

kategorier fram ur datan och påverkade på det sättet vårt instrument, där intervjuguiden utvecklades utifrån vad tidigare uppgiftslämnare sagt. Vi har inledningsvis följt Charmaz (2006, sid. 25) råd att utgå ifrån några få breda och öppna frågor för att sedan ha möjlighet att gå in på högre detaljnivå där det ter sig lämpligt. I enlighet med *intensive interviewing* hade frågorna ett spann från löst guidade och utforskande frågor till halvstrukturerade och specifika frågor.

I figur 3.1 presenteras hur intervjuerna fortlöpte med tiden. Första intervjuerna med intervjupersonerna P1 och P2 tog vid på Företag 1. Efter intervjutillfällena analyserades datan och intervjuguiden reviderades till nästa intervjutillfälle. Dokumentsymbolerna till höger om tabellen hänvisar till tillfällena revidering av intervjuguiden tog vid.



Figur 3.1 Intervjuflödet med revidering av intervjuguide

Intervjuerna utfördes på uppgiftsgivarens arbetsplats i avskilda rum och spelades in efter uppgiftsgivarens godkännande. Gemensamt för intervjuerna var att temat för samtalet var hur deras syn på DevOps i sina respektive organisationer såg ut där vi lät deltagarna fritt berätta om deras arbete.

3.3 Dataanalys

Efter intervjuerna slutförts förbereddes datan genom transkription. Transkriptionerna annoterades med minnesanteckningar tillsammans med markering av textstycken av intresse. För att hantera den stora datamängden som de öppna intervjuerna genererade valde vi att använda oss av ett kodningsschema där nyckelpoänger skapas ur datan. Schemat utvecklades med inspiration av Allan's (2003) kodningschema.

Varje nyckelpoäng märktes med ett **P**, siffran därefter denominerar intervjun ifråga och det sista talet visar vilken nyckelpoäng det motsvarar i intervjun. Till exempel refererar **P1-15** till den femtonde nyckelpoängen från den första intervjun, hämtat från den markerade texten i transkriptet. Se figur 4.

> *[...] när vi anställer så brukar vi sällan leta efter nån viss nyckelkompetens. Våra annonser för utvecklare är ganska generiska och breda.. Har ni jobbat med … och så en lista på 300 olika grejer som inte är förenliga med varandra egentligen och sen så kommer det in folk som kanske kan en tredjedel av de här grejerna. Jamen kul! Kom in på intervju.*

P1-15	Ser bred kunskap framför nyckelkompetens.

Figur 3.2 Exempel på konceptgenerering utifrån transskript

I enlighet med Maxwell (1997, sid. 78-79) kategoriserades våra nyckelpoänger både i kategorier baserade på litteraturen och sådana som induktivt grundades i insamlad data. Den ursprungliga litteraturbaserade indelningen utgick från CAMS. Nyckelpoänger från transkriptet sammanställdes i ett separat ark och blev efter bedömning tilldelade en värdegrund i en första kategorisering.

Kultur	P1-15	Ser bred kunskap framför nyckelkompetens.

Figur 3.3 Exempel på tilldelning av CAMS värdegrund till nyckelpoäng

Värdegrundskategoriseringen gav sedan upphov till en ytterligare kategorisering efter nyckelpoängernas samspel och utsagor, vad Jacobsen (2009, sid. 237) kallar hänföring av enheter. Vi har valt att kalla denna typ av kategorisering för konceptgenerering, då nyckelpoängerna tillsammans antyder på ett gemensamt koncept. (Se tabell 1)

Tabell 3.1 Exempel på konceptgenerering utifrån data.

Dev överlämnar inte ansvaret av produkten, utan tar ett helhetsansvar för produktens hela livslängd	
P1-55	Utvecklare ansvarar för sin produkt konstant
P3-18	Ops ansvarar för plattformen, Dev för produkten som kör på plattformen.
P4-26	Att lämna över ansvaret för mjukvaran till driftavdelningen ses som något i många fall sämre.

Allteftersom våra intervjuer analyserats och koncept genererats ur datan, framgick det gemensamma kategorier som ansågs viktiga. Kategorierna namngavs efter breda begrepp för att kunna rymma koncepten men samtidigt ge en gemensam nämnare. Kategorierna fungerade som en hänvisning till instrumentet, dvs. intervjuer och intervjuguide. Därav kunde vi i den fortsatta analysen av transkriptioner använda ett mer fokuserat angrepssätt där dessa viktiga kategorier prioriterades framför att kategorisera den totala datamängden (Jacobsen 2002, sid. 230-231).

Tabell 3.2 Exempel på kategorisering av koncept "Ansvarsfördelning och helhetsansvar"

Kategori	**Ansvarsfördelning och helhetsansvar**
Koncept	Dev överlämnar inte ansvaret av produkten, utan tar ett helhetsansvar för produktens hela
Koncept	Ops ansvarar för den underliggande plattformen, men inte produkten som sådan.
Koncept	DevOps-organisationer karaktäriseras utav att lägga mindre vikt till roller och titlar.

3.4 Urval

Med forskningsfrågan vill studien besvara vad som karaktäriserar en DevOps-organisation, vilket ställer krav på urvalet i studien. Enligt Weiss (1994, sid. 17) kan en respondent ses antingen som en informant med expertkunskap inom ett visst område, eller en representant av en studiepopulation. Då vi utfört en kvalitativ studie och intresserade oss för respondenternas upplevelser och åsikter utifrån deras roll som DevOps-utövare snarare än deras expertis inom ämnet som sådant, såg vi respondenten som en representant av DevOps-rollen.

En DevOps-utövare definierades som någon som arbetar i, eller direkt i anslutning till en arbetsgrupp som går under namnet DevOps. Respondenter till studien införskaffades genom förfrågningar över e-post till företag som sade sig använda sig utav DevOps i deras arbete och företag som aktivt annonserade efter personal med kunskap inom DevOps.

För att säkerställa att respondenten kunde tala med erfarenhet inom DevOps-arbete, utan att utesluta respondenter med erfarenheter av tidiga skeden av ett DevOps-arbete färskt i minnet, valde vi att enbart inkludera personer som arbetat med DevOps i minst tre månader. Urvalet ställde dock inga krav på vilken roll personen i fråga har eller har haft tidigare i DevOps-arbetet, då det hade begränsat bredden och variationen och därmed begränsat datan. Det förekom stora skillnader i hur respondenterna såg på DevOps, men ingen bedömdes ha en så avvikande syn på DevOps att en exkludering ansågs nödvändig.

3.5 Undersökningskvalitet

3.5.1 Validitet

Inom samhällsvetenskapen gäller generellt att validitet uppnås genom *intersubjektivitet*, dvs. att flera människor har samma uppfattning om ett begrepp. Intersubjektivitet anses vara det närmsta vi kommer sanningen inom detta fält (Jacobsen, sid. 256). Validitet delas upp i två delar där intern validitet rör resultatens giltighet medan extern rör överförbarhet (generaliserbarhet).

Det är inte avgörande för en kvalitativ studie som denna att resultatet är externt generaliserbart till större populationer. Däremot är det viktigt att studien är *internt* generaliserbar, dvs. att inte ge studien ett alltför smalt fokus inom tillämpningsområdet för studieobjektet (Maxwell, 1996, sid. 201). Ett exempel i vårt fall skulle kunna vara att ge fokus till enbart en roll inom ett av de undersökta företagen. För att motverka detta har vi avsiktligt sökt intervjupersoner med varierade roller, som utvecklare, driftansvariga, konsulter, projektledare och avdelningschefer.

> *“Indeed, the value of a quantitative study may depend on its lack of external generalizabiliity, in the sense of being representative of a larger population; it may provide an account of a setting or population that is illuminating as an extreme case or ideal type”*
> *(Maxwell, 1996, sid. 201)*

En del av de nyckelpoänger som tagits fram platsar i flera kategorier beroende på vilket perspektiv som tas och således kan kategoriseringen ses som godtycklig, vilket kan hota studiens validitet. Koncepten som genererats har dock starkt stöd i studiens resultat och speglar DevOps-arbetet i de undersökta organisationerna.

3.5.2 Reliabilitet

Reliabiliteten i en studie avser hur pass tillförlitliga dess resultat är och hur tillvägagångssättet kan ha påverkat vilka resultat som uppnåtts. Nedan följer tre viktiga aspekter som kan påverka en studies reliabilitet, som presenterats av Jacobsen (2009).

Intervjuareffekten är en term som beskriver effekten en intervjuare har på den som blir intervjuad. (Jacobsen, 2009, sid. 270) Som intervjuare har vi strävat efter att hålla oss neutrala med avseende på klädsel, kroppsspråk, språkbruk, talesätt etc. Det finns dock risk för att intervjusituationen kan ha upplevts intimiderande som resultat av att vid varje intervjutillfälle varit tre personer som hållit intervjun. Även detaljer som att dörren till mötesrummet i vissa fall stått öppen kan ha influerat svaren då respondenten möjligen kan ha oroat sig för förbipasserande.

Vidare talar Jacobsen (2009, sid. 271-273) om kontexteffekten som handlar om den kontext intervjuer utförs i. Vi har i vår studie enbart utfört planerade intervjuer på de intervjuades respektive arbetsplatser. Detta för att i så liten grad som möjligt påverka de utvunna svaren genom att optimera den intervjuades bekvämlighet.

Slutligen har vi valt att spela in samtliga intervjuer för senare transkribering. På detta sätt har också reliabiliteten stärkts då vi via inspelning och transkription undviker slarv vid nedteckning av data under intervjuns gång (Jacobsen 2009, sid. 273-274).

3.6 Etik

Tre etiska grundkrav diskuteras av Jacobsen (2009, sid. 482-489): informerat samtycke, krav på privatliv och krav på att bli korrekt återgiven. Informerat samtycke innebär att deltagande i undersökningen ska vara frivilligt och att deltagaren ska vara fullt medveten om, samt förstå, eventuella risker eller vinster som ett deltagande innebär. Frivilligheten rör också sådant som påtryckningar från medarbetare vilket kan vara subtilt och svårt att uppfatta. Vår bedömning är att inga av dessa aspekter brutits mot, fastän vi inte kan vara helt säkra på om eventuella påtryckningar manifesterats utan vår vetskap.

Krav på privatliv gäller information som kan vara identifierande, känslig och/eller privat. För att förhindra identifierbarhet har vi valt att inte ange information som ålder och kön. Inte heller har sådan information som kan anses vara integritetskänslig uppkommit i våra intervjuer.

Slutligen finns ett krav på korrekt presentation av data vilket innebär fullständig och korrekt återgivelse av intervjuresultat. Med detta menas också att citat ej ska tas ur sitt sammanhang då det i sådana fall kan förvränga den ursprungliga meningen i uttalandet. Genom inspelning och transkribering samt presentation av fullständiga intervjuer till respondenterna har vi försäkrat oss om att presentationen av data är riktig.

4 Empiri och Analys

I detta kapitel presenteras inledningsvis de organisationer och de intervjudeltagarna som ingick i studien. Vidare presenteras intervjuernas resultat i förhållande till CAMS-ramverket. Slutligen sker en genomgång av de kategorier och koncept som tagits fram ur analysarbetet.

4.1 Intervjudeltagare

I studien har sju intervjuer utförts med intervjudeltagare från tre organisationer.

4.1.1 Företag 1 (F1)

Ett växande IT-bolag inom telekombranschen med fokus på kommunikationsverktyg för företag. Bolaget driver flera produkter där deras flaggprodukt är en digital telefonväxel. Huvudkontor i Malmö med ett öppet kontorslandskap med grupperingar efter support, drift och utvecklare.

Intervjuperson 1 (P1):

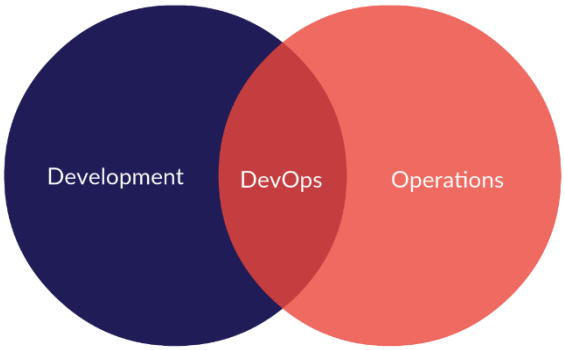
P1 kom in i bolaget i ett tidigt skede och att företaget nu har sett en stor expansion där de nu sitter 46 anställda på utvecklingsavdelningen. P1 nämner att när han först kom dit var han förvånad över upplägget och gav en skämtsam liknelse med kaosteori, men att P1 utvärderat och försökt hitta ett förslag till en förbättring, men inte funnit någonting.

Vår uppfattning är att P1 ser upplägget som fördelaktigt, att det innefattar en öppen miljö där alla har möjligheten att väga in i form av idéer och vad som bör prioriteras. F1 ser hellre att deras personal har en bred kompetens framför specialister och betonar vikten av att deras anställda ska visa intresse för tekniker och produkten.

Intervjuperson 2 (P2):

P2 arbetar som mjukvaruutvecklare på en avdelning som fångar upp sådant arbete som produktteamen skjuter framför sig. Detta kan vara allt ifrån stora ombyggnationer av arkitektur till uppgifter som trillar ned på deras bord från andra avdelningar.

P2 upplever DevOps som en miljö där mycket av arbetet sker ad-hoc genom en knackning på axeln eller annan form av spontan kommunikation på arbetsplatsen, och där deployments kan göras av utvecklarna med ett par knapptryck.



Figur 4.1 DevOps överlappande

"DevOps är att att man suddar ut det mellan de som försöker se till att grejerna är igång och de utvecklare som skall stå till svars varför det inte är det. Om man nu har en cirkel med alla utvecklare här och en cirkel med alla dom som håller på med operations där, att dom överlappar varanna litegrann. Den överlappningen där, det är DevOps för mig"

- P2, Rad. 163

Intervjuperson 4 (P4):

P4 är idag själv organisationens driftavdelning, även om det arbetas för att förändra detta. P4 drivs av att automatisera bort delar av sitt arbete i verktyg som sedan tillhandahålles till utvecklarna.

"Jag strävar ju egentligen efter att typ automatisera bort mig själv."

- P4, Rad. 126

P4 ansvarar för att allt det tekniska, såsom servrar och nätverk, är igång och fungerar. Han hjälper även till med att sätta upp testmiljöer till utvecklarna, och tillhandahåller virtuella maskiner för att köra utvecklarnas kod. P4 använder sig mycket av interna chattverktyg, och tycker att kommunikationen inom organisationen är både det bästa och det mest utmanande med DevOps.

"Trots att vi kommunicerar jättemycket så tror jag att man skulle kunna göra det ännu mycket mer."

- P4, Rad. 204

P4 ser mycket individuellt ansvar inom organisationen och få direktiv ovanifrån. Detta tros ge högre kvalitet på produkterna som utvecklas, även fast det ibland kan vara problematiskt att prioritera sina arbetsuppgifter.

4.1.2 Företag 2 (F2)

Intervjuperson 3 (P3):

P3 har en konsultroll på Företag 2 och arbetar med att implementera Continuous Delivery i andra organisationer. P3's senaste kunder har varit stora bolag med större siloliknande avdelningar, och den största problematiken har enligt P3 varit att få de anställda att lita på varandra över avdelningsgränserna. Vid intervjutillfället arbetade P3 på ett större svenskt bolag med över 100 000 anställda världen över, med huvudkontor i Malmö. P3's arbete har främst legat på utvecklingssidan i organisationen och fokus har legat på att främja snabb feedback mellan utvecklare och drift genom CD. P3's roll skiljer sig avsevärt från övriga respondenter då han är inhyrd hjälp för att förverkliga CD och på så vis kommer i kontakt med DevOps

P3 anser att bra DevOps är något som är svårdefinierat då det saknas en tydlig definition av fenomenet.

> *"DevOps är som fint väder, vissa tycker det är fint väder när det regnar, vissa tycker det är fint väder när det är molnigt. Men det är ju fortfarande väder." - P3, Rad. 126*

P3 såg att det fanns stora svårigheter i att främja kunskapsdelning inom organisationer, men beskrev det som viktigt.

4.1.3 Företag 3 (F3)

Företag 3 är ett svenskt bolag som tillhandahåller business intelligence produkter. Produkterna har traditionellt varit klientinstallationer men erbjuder sedan ett tag tillbaka molnlösningar. P7 berättar att det pågår en organisationsförändring för att mer och mer arbeta enligt DevOps där man rör sig mot en tjänstifiering av molnlösningens komponenter.

Den delen av organisationen som hanterar molnlösningar karaktäriseras av snabba leveranser och korta ledtider och ägandeskap av kod hela vägen ut i drift, medans den delen av organisationen som arbetar med bolagets mer traditionella arkitektur med klient- och serverapplikationer som distribueras med installationspaket ser längre ledtider och mindre frekventa releaser.

Intervjuperson 5 (P5):

P5 är chef för två team som arbetar med verktygen i CI- och CD-kedjan och har arbetat med det i ungefär ett halvår. Han ansvarar för prioritet av uppgifter som skall utföras av sina underordnade, och ser beräkning av business-värdet för nya features som viktigt.

Mycket av arbetet går ut på att strukturera arbetsuppgifter i deras ärendehanteringssystem JIRA, som han uppger dokumentera en stor del av deras arbete.

Intervjuperson 6 (P6):

P6 arbetar likt P5 på avdelningen Release Readiness som stöttar resten av organisationen med hjälp av en CD-pipeline. Han arbetar som arkitekt för ett team som tillhandahåller diverse system som stödjer releaseprocessen med hjälp av tester, automatiska byggen och konfiguration.

DevOps uppgavs finnas på organisationen redan för åtta år sedan då P6 började arbeta där, även om initiativen inte gick under det namnet på den tiden. P6 ser DevOps som en stödjande aspekt till releaseprocessen och ser vikten av att se utvecklarnas behov som något av det viktigaste i hans arbete.

Vidare ser P6 kommunikation och feedback som något som ett viktigt förbättringsarbete inom organisationen.

Intervjuperson 7 (P7):

P7 håller titeln Sr. Solution Architect och började på F3 för två år sedan. P7 nämnde att han hade en lång erfarenhet av produktionsleverans.

P7 beskrev att organisationen inte kunde ses som DevOps fullt ut, utan istället fanns DevOps i visa delar utav organisation, samt att organisationen genomgår en förändringsfas där man arbetar för att tjänstifiera applikationer och skapa team utifrån detta. P7 såg detta som en fördel då tjänstifierade team kan ta ett helhetsansvar från början till slut.

> *"[…] DevOps är ju ett sätt att göra det till allas problem hur det funkar i produktion. Det är det som är intressant. Det spelar ingen roll hur snabbt du kan få fram det till Ops ifall de inte kan deploya det" - P7, Rad. 134*

4.2 CAMS

De koder som framtogs vid analysen har i ett första steg kategoriserats efter CAMS-ramverket (se Appendix L). Vid kategoriseringen framkom att många koder kunde placeras i flera kategorier och överlappade varandra. I de koder som hade multipla kategorier var *Kultur* den mest frekvent återkommande, då exempelvis uppmaningar till ökad kunskapsspridning kunde bedömas som både en kulturellaspekt och en aspekt knuten till delning.

Vi har i dessa fall valt att kategorisera koderna enbart i den kategori där de är mest passande, vilket i det flesta fall har inneburit de kategorier som *inte* är kultur. De koder som har tvetydiga kategorier har färgkodats för att matcha de sekundära kategorier som de skulle kunna tillhöra.

Tabell 4.1 Översikt av CAMS-nyckelpoänger

Kategori	Antal Nyckelpoänger
Kultur	200
Automation	43
Mätning	8
Delning	81
Totalt:	**332**

4.2.1 Kultur

Kultur innefattar de koder som kan relateras till organisationsstruktur, kommunikation, värderingar och arbetsmetodiker samt hur man ser på ansvar och prioriteringar.
De vanligast förekommande koderna kunde relateras till kommunikation mellan olika avdelningar, ansvarsroller, olika former av feedback samt vikten av att skapa en kultur där kunskap delas.

Tabell 4.2 Urval av nyckelpoänger associerade med Kultur

Nyckelpoänger kategoriserade i Kultur (urval):	
P1-30	Utvecklare tar ett helhetsansvar för produkten.
P2-29	Kommunikation viktigt mellan support och dev.
P3-37	Geografiska avstånd har påverkar kommunikation och tillit.
P4-2	Team är funktionsorienterade runt applikationer.

4.2.2 Automation

I Automation har de koder som berör automatiserad byggning, driftsättning och testning placerats. De vanligast förekommande koderna innefattade driftsättning, CD och verktyg.

Tabell 4.3 Urval av nyckelpoänger associerade med Automation

Koder kategoriserade i Automation(urval):	
P2-13b	Utvecklare blir motiverade till att frekvent släppa sin kod.
P1-36	Miljön tillåter snabba rollbacks.
P3-3	CD behövs för att man ska vara bra på DevOps.
P4-93	Operations arbete automatiseras bort när det är möjligt.
P6-2	DevOps involverar miljöer för att stödja byggen

4.2.3 Mätning

Mätning innefattar de koder som handlar om mätning av arbetet. Väldigt få respondenter såg mätning som någonting centralt i DevOps, och många såg mätning som svårt.

Tabell 4.4 Urval av nyckelpoänger associerade med Mätning

Koder kategoriserade i Mätning(urval):	
P1-18	I och med öppen och agil struktur blir det svårmätt och man får förlita sig på magkänsla och
P3-50	Svårt att veta vad som utgör “bra” DevOps.
P4-84	DevOps är svårt att mäta.
P5-32	Behovet av att mäta sjunker när anställda tycker det är roligt att arbeta

4.2.4 Delning

I delning har de koder som berör delning av exempelvis kunskap och verktyg placerats. De vanligaste koderna innefattade kunskapsdelning genom verktyg, kommunikation, code-reviews och ärendehanteringssystem.

Tabell 4.5 Urval av nyckelpoänger associerade med Delning

Koder kategoriserade i Delning(urval):	
P1-52	Bred kunskap och helhetsperspektiv ska ge upphov till bättre beslut
P4-100	Kunskapsspridning kan ske genom automation.
P3-17	Ops ansvar att förse Dev med verktygen och kunskapen för att kunna släppa själv.
P2-13a	Code reviews används för att minska antalet fel och dela kunskap mellan utvecklare.
P7-6	Kunskapsspridning får folk mer intresserade av hela leveranskedjan.

5 Karaktäriserande Egenskaper

CAMS-ramverket användes i den inledande fasen av studien för att sedan ersättas av ett nytt ramverk som växte fram ur emprin i den empiriska sekvensen. I detta kapitel presenteras de koncept av organisatoriska egenskaper som karaktäriserar de DevOps-organisationer som studerats, vilka sammanställts i bredare kategorier.

*Initialt presenteras en översikt av kategorierna och de underliggande koncepten. Därefter presenteras koncepten var för sig, tillsammans med ett urval av de tillhörande nyckelpoängerna. En mer detaljerad kodning och kategorisering av intervjutranskripten återfinns i appendix **B** (P1), **C** (P2), **E** (P3), **G** (P4), **I** (P5), **J** (P6), **K** (P7) och en sammanställning av koncepten med nyckelpoänger i appendix **M**.*

Tabell 5.1 Översikt av genererade koncept

Kategori	Koncept
Insyn (gemensam)	DevOps-organisationer karakteriseras utav en organisatorisk transparens, där varje medarbetare ska ha en insyn till andra delar av organisationen utanför sitt eget område och ha ett helhetsperspektiv av organisationens värdekedja.
	DevOps karakteriseras av en organisationsstruktur och ledning som stödjer insyn.
Arbetsfördelning och helhetsansvar	DevOps-organisationer karakteriseras utav att lägga mindre vikt till roller och titlar.
	Inom DevOps-organisationer finns fortfarande olika ansvarsområden.
	Dev överlämnar inte ansvaret av produkten, utan tar ett helhetsansvar för produktens hela livslängd.
	Ops ansvarar för den underliggande plattformen, men inte produkten som sådan.
Kunskapsdelning	DevOps möjliggörs av kunskapsdelning.
Gruppöverskridande samarbete	DevOps organisationer karakteriseras av samarbete och kommunikation mellan gruppgränserna.
Verktyg	DevOps karakteriseras av användandet av verktyg för att understödja aktiviteter
Feedback	DevOps organisationer karakteriseras av snabb feedback där organisationen så tidigt som möjligt ska få uppmärksamhet kring problem.
Agilitet	DevOps organisationer karakteriseras av korsfunktionella teams framför stora avdelningar.
	DevOps organisationer karakteriseras av att snabbt agera utifrån ändrade förutsättningar.
	DevOps organisationer använder agila metoder.
	DevOps organisationer karakteriseras av att snabbt och ofta kunna produktionssätta

		förändringar.

5.1 Arbetsfördelning och helhetsansvar

F1 ser roller av mindre vikt (P1-12, P1-37, P2-32, P4-1a, P4-1b). En anställds titel ska inte begränsa vilka uppgifter den kan utföra, utan istället uppmanas en anställd att utföra arbetsuppgifter ifall den anser ha förmågan och intresset (P1-3). En approach som P1 pekade på som en kvarleva från företagets start då de var få anställda. F3, som är ett större bolag kunde detta ej explicit observeras, men P6 pekade på att i dagens läge har den gemene utvecklaren kunskap om det mesta och att de på så sätt kan bistå varandra i arbetet (P6-32).

Tabell 5.2 Urval av koncept: Mindre vikt till roller och titlar

DevOps-organisationer karaktäriseras utav att lägga mindre vikt till roller och titlar (urval):	
P1-37	Finns inga tydligt definierade roller.
P2-32	DevOps är att sudda ut gränserna mellan Utveckling och Drift.
P4-89	Otydliga ansvarsroller har fler fördelar än nackdelar.
P7-27	Produktionen ska vara allas problem.

Till exempel kan produktionssättningen av mjukvara, som traditionellt faller under driftavdelningens ansvar, utföras av utvecklaren också (P2-24). P3 beskriver ansvarsroller som att Dev ansvarar för produkten i produktion, medan Operations ansvarar för den övergripande plattformen produkterna körs på samt infrastrukturen till det (P3-18).

Då de två olika beskrivningarna till en början verkar motsägelsefulla passar dock P3's beskrivning med vad som beskrivs i F1 (P1-32, P2-26) med att utvecklarna har jouren för produkten medan Operations agerar som second-line och att utvecklare får det övergripande ansvaret för produkten även i produktion (P1-30, P1-55, P4-25). Man skiljer också på vilka problem som faller under utvecklarna att åtgärda i produktionsmiljön (P2-24a), där Operations fortfarande ansvarar för och åtgärdar problem med den underliggande plattformen och infrastrukturen (P2-26, P4-44, P6-23).

Tabell 5.3 Urval av koncept: Dev har helhetsansvar

Dev överlämnar inte ansvaret av produkten, utan tar ett helhetsansvar för produktens hela livslängd (urval)	
P1-30	Utvecklare tar ett helhetsansvar för produkten.
P1-32	Utvecklaren har jouren för att produkten fungerar, medan Ops fungerar som secondline
P4-26	Att lämna över ansvaret för mjukvaran till driftavdelningen ses som något som i många fall är sämre.
P3-18	Ops ansvarar för plattformen, Dev för produkten som kör på plattformen.
P2-28	Utvecklare ansvarar för sin kod hela vägen till och med drift.
P6-24	Applikationsägare ansvarar för applikationslagret.
P7-16	Strävar mot utvecklarteams helhetsansvar.

Att utvecklare tar ett helhetsansvar återfanns också i F3's cloud-tjänster där utvecklare tillsammans med driftpersonal ansvarade för produktionsmiljön med monitorering (P6-27). Detta skedde dock inte i leveranskedjorna i F3's klient- och serverapplikationer, då dessa erbjuds till kunden som installationspaket i mer traditionell modell. Detta påverkar cykeltiden av produktleveranser, där en release manager ansvarar för godkännande av släppet och kundsupport ansvarar för distribution (P5-1, P5-2). Således är möjlighet till helhetsansvaret kopplat leveranskedjan och den applikationsarkitektur som utvecklarna arbetar med. Det övergripande målet med DevOps ses hos F3 som att "produktionen ska vara allas problem" (P7-27).
De tidigare nämnda otydliga ansvarsrollerna ses i vissa fall som problematiska för F1 (P4-88). Exempelvis finns risk att problem eskalerar för långt då det ej råder övertygelse om vem som ansvarar för vad och en viktig uppgift lämnas ogjord(P1-38). Det blir även svårare att se vilken kompetens som saknas inom företaget (P1-13). Enligt P4s beskrivning väger dock fördelarna över med detta sätt att hantera ansvarsroller gentemot nackdelarna (P4-89).

Tabell 5.4 Urval av koncept: Ops ansvarar för underliggande plattform

Ops ansvarar för den underliggande plattformen, men inte produkten som sådan (urval)	
P6-23	Operations ansvarar inte för applikationslagret.
P2-26	Driftavdelningen har ansvar för problem kopplade till hårdvara och nätverk.
P3-18	Ops ansvarar för plattformen, Dev för produkten som kör på plattformen.
P2-25	Jour-ansvar för drift läggs på (vissa) utvecklare.
P1-32	Utvecklaren har jouren för att produkten fungerar, medan Ops fungerar som secondline.
P4-44	Operations är inte involverade i mindre deployments som inte påverkar nätverk.

På så sätt existerar det fortfarande separata ansvarsområden men uppgiftsfördelningen skiljer sig från den traditionella fördelningen. Utvecklare tillåts utföra uppgifter som tidigare hört till driftens område, som produktionssättning, monitorering av produkten i produktionsmiljö samt felsökning. Detta möjliggörs genom att utvecklare tillhandahålls verktyg där de kan commita sin kod, låta tester köras och på ett enkelt och snabbt sätt kan leverera den testade koden till produktion på egen hand (P2-10, P2-11, P4-44). Här är anknytning till CD stark där utvecklarna, med hjälp av verktyg försedda av drift (P3-17, P4-94), kan låta sina förändringar nå produktionsmiljö på egen hand utan Operations direkta inblandning. När en organisation använder sig av vad som kan kallas CD försvinner den tydliga separationen som tidigare fanns, då uppgiften att sätta kod i produktion faller på utvecklarna att utföra men i samarbete med driftavdelningen.

P1's beskrivning om att grupperingar sker efter en viss produkt (P1-39, P1-40) och att team byggs upp utifrån de resurser som behövs (P1-27) tyder på en agil approach. P4 berättar att team bär ansvar för specifika funktioner eller applikationer istället för att vara uppdelade efter mer traditionella roller (P4-2, P4-3). Denna struktur återfanns även i den organisationsförändring P7 berättar om, där organisationen gradvis arbetar mot en tjänstefiering av sin cloud-produkt (P7-14).

Tabell 5.5 Urval av koncept: Otydliga ansvarsroller innebär utmaningar

Otydliga ansvarsroller innebär utmaningar (urval):	
P1-13	Svårt att se saknad kompetens i och med Otydliga roller.
P1-8	Delat ansvar kan innebära missar.

Detta arbetssätt kräver ett stort personligt ansvar (P4-118) vilket i sin tur för med sig en risk att anställda lägger mycket tid på uppgifter som saknar kundnytta (P4-66, P4-72). Prioritering av arbetsuppgifter är såväl viktigt som svårt (P4-108) och det kan kräva mycket tid bara att upprätta genomtänkta prioriteringar (P4-65), vilket ofta görs på prioriteringsmöten (P4-69). Vidare prioriteringssvårigheter märks i den kommunikationsmässigt öppna miljön där nya arbetsuppgifter ofta uppkommer spontant vilket stör de ordinarie arbetsuppgifterna (P2-7, P1-47). Balansgången mellan långsiktiga och kortsiktiga mål uppges av P2 också vara en utmaning för F1.

De tidigare nämnda otydliga ansvarsrollerna kan ses som problematiska för F1(P4-88). Exempelvis finns risk att problem eskalerar då det ej råder övertygelse om vem som ansvarar för vad och en viktig uppgift lämnas ogjord (P1-38). Det blir även svårare att se vilken kompetens som saknas inom företaget (P1-13). Enligt P4s beskrivning väger dock fördelarna över nackdelarna när det gäller detta sätt att hantera ansvarsroller (P4-89).

5.2 Kunskapsdelning

Det är viktigt för F1 och F3 att kunskap sprids mellan de olika rollerna (P1-7, P1-42, P1-44, P2-12, P3-22, P6-13, P7-20). Detta kan ske genom utökad feedback mellan avdelningarna (P1-56, P3-7), feedback genom CD-verktyg (P3-8) och feedback genom code reviews (P1-33, P2-13a). Det kan även ske automatiskt när man möjliggör kontakt mellan vissa typer av personligheter och kompetenser (P1-43).

> *Det skall liksom vara så få människor som möjligt som skall vara "den viktiga personen". När den personen har semester eller har gått för dagen ska inte det hindra arbetet. -P2, Rad. 29.*

Tabell 5.6 Urval av koncept: DevOps möjliggörs av kunskapsspridning

DevOps möjliggörs av kunskapsdelning (urval):	
P2-12	Kompetens sprids och binds ej i nyckelroller.
P1-3	Många ska kunna göra mycket.
P3-7	Utvecklare får möjlighet att ta ansvar med hjälp av snabb feedback.
P6-18	Champions medför kunskapsdelning.
P7-7	Kunskapsspridning skapar ett helhetsperspektiv
P7-9	Organisationens kontroll underlättas genom kunskapsspridning.

Inom F1 lades stor vikt vid att kompetens inom organisationen skulle vara så pass utbredd att andra anställda skulle kunna ta över arbetsuppgifterna vid frånvaro, och att ingen nyckelkompetens skulle gå förlorad om någon lämnade företaget (P1-48, P2-12, P2-31). P3 såg kunskapsdelning som något som var svårt att uppnå, men nämnde stöd från ledningen som en viktig faktor (P3-22). Att DevOps river murar och ge de anställda en större insikt i organisationen sågs även som en bidragande faktor till ökad kunskapsspridning (P3-41).

Hos F1 ser de till bredden av kunskap och personlighet i deras nyanställningar, där man hellre såg intresse, social kompetens och samarbetsvilja framför nyckelkunskaper (P2-19) då man vill öka kunskapsdelningen inom företaget. P3 såg fördelen i att ha samarbetsvilliga personer (P3-32b) men poängterade att det inte var ett krav för DevOps. F3 hanterar kunskapsspridning inom organisationen genom att introducera nya verktyg till så kallade "champions", personer med ett stort tekniskt intresse inom en avdelning/team, innan de gick ut till resten av organisationen (P6-18). Champions används för att skapa intresse, förmedla kunskap och möjliggöra feedback för nya verktyg innan de når resten av organisationen.

5.3 Gruppöverskridande samarbete

Samarbete mellan avdelningar är av stor vikt och värderas inom DevOps-organisationen (P1-42, P4-75). Det är genom den ökade kommunikationen och samarbetet mellan avdelningarna som en organisation uppnår gemensamma mål. När avdelningarna delar gemensamma mål underlättas prioritering av arbete (P3-46).

> *"Har man bra DevOps så har man bra samarbete. [...] med DevOps har de gemensamma mål, business-mål, och då vet de hur de ska prioritera."*
> *-P3, Rad. 158*

Tabell 5.7 Urval av koncept: Gruppöverskridande samarbete och kommunikation

DevOps organisationer karakteriseras av samarbete och kommunikation mellan	
P1-42	Gruppöverskridande kunskapsdelning och hjälpsamhet
P2-1	Det existerar olika ansvarsområden
P3-17	Ops ansvar att förse Dev med verktygen och kunskapen för att kunna släppa själv.
P6-11	Face-to-face kommunikation mellan avdelningar viktigt.
P7-4	Ops förser utvecklare med verktyg för CD.

En av de vanligaste orsakerna till att en lösning fallerar är otillräcklig insyn i Operations verksamhet från Development (P2-16, P3-25), vilket kan resultera i dålig tillit mellan avdelningarna (P3-16). För att förbättra detta set F1 det som viktigt att tidigt involvera driftavdelningen inför en release (P3-30) och att ha en öppen dialog mellan avdelningarna (P2-16).

> *"Vi försöker hålla varandra underrättade om det är grejer. [...] Så att supporten får reda på att något är på väg att gå ner eller att det är ett planerat driftstopp." -P2, Rad. 31*

I F1 ses ett nära samarbete mellan Development och Support (P1-22, P2-5, P2-29), vilket ger utvecklarna möjlighet att snabbt anpassa sig till kundens önskemål. Utvecklarna har också möjliget att hjälpa Support genom att vara proaktiva i deras arbete. Exempelvis kan de förvarna inför en större release och meddela vad som kan påverkas samt hur eventuella resulterande problem kan lösas från kundens sida (P2-38). Att Development och Operations delar utrymme kring kaffeautomaten ses också som något som kan ge insikt och förståelse för varandras arbete genom spontan kommunikation (P2-36).

Kommunikation är kopplat till kunskapsdelning då kommunikation mellan avdelningar och läger uppmanas så att kunskapsdelningen kan öka. Kommunikationen avser även en tidig involvering av exempelvis Operations i planeringsfasen (P3-30, P4-38) så att de tidigt kan ge synpunkter (P1-31) och får en tidig vetskap om vad som kommer att behövas från drift, till exempel vid utvecklingen av en ny produkt eller vid en ändring (P2-38). Avdelningarna ser

nyttan i varandras input (P4-36, P4-37) och utvecklarna, som har helhetsansvaret för produkten, kommer närmare slutkunden genom dialog med Operations och Support (P4-12). I F1's fall sker detta genom veckomöten med installatörer och support, som har den primära kontakten med kunder, och utvecklarna (P4-13). Vidare har supporten i F1 en möjlighet till direktkontakt med utvecklare ifall akuta ärenden skulle inträffa (P4-30), vilket är någonting som uppmuntras. Både F1 och F3 kommunicerar aktivt via en chat som är gemensam för hela organisationen och sammankopplar avdelningarna (P2-16, P2-36a, P7-23a).

Då kommunikation och samarbete är av central vikt i organisationerna (P3-43), utgör de också den största utmaningen. (P4-114). En direkt utmaning till uppmaningen att öka samarbetet ligger till den fysiska och geografiska placeringen av avdelningar (P3-37). En större organisation kan ha sina avdelningar utspridda och går på så sätt miste om viktig direkt och spontan kommunikation (P2-36). P2 såg dock fortfarande att organisationsstorleken inte behöver utesluta DevOps-arbete (P2-17).

Den fysiska distansen kan motarbetas med hjälp av kommunikationsverktyg (P4-14, P4-15, P14-16), men med det ökade samarbetet och den ökade kommunikationen ser P4 en utmaning att filtrera all kommunikation. Den spontana och rika kommunikationen tillsammans med breda roller kan även uppfattas som distraherande i arbetet, då alla har kunskap till att göra det mesta (P2-7).

5.4 Verktyg

Verktyg som en karaktäriserande kategori har ett brett omfång och stödjer de andra egenskaperna som för kommunikationen inom ett team eller mellan avdelningar. Även då direkt face-to-face kommunikation ses som en fördel (P6-11) är fysisk närhet inte alltid en möjlighet, där man använder sig av, ibland organisationstäckande, verktyg för kommunikationen. Inom team använder man ärendehanteringssystem och agila verktyg som JIRA (Kanban) för att fördela och prioritering (P4-53, P6-28).

Tabell 5.8 Urval av koncept: Verktyg

Verktyg (urval):	
P1-36	Miljön tillåter snabba rollbacks.
P2-14	CI/CD sker genom SCM & byggverktyg som Git och Jenkins.
P2-11	Det är enkelt att deploya sin kod.
P4-53	Ett ärendehanteringssystem används för att spåra alla aktiviteter.
P3-26	Avdelningar ökar tillit genom att själva sätta upp verktyg och failsafes till sina inputs. Svårt!
P6-2	DevOps involverar miljöer för att stödja byggen.
P7-22	Verktyg underlättar frekvent kommunikation mellan avdelningar.

Verktygen inom DevOps tillhör till stor del Continuous Delivery pipelinen (P1-35, P2-14, P3-2, P3-3, P6-7). Dessa verktyg förenklar och effektiviserar deployment av kod (P2-10, P2-11), samt minskar risken för fel när kod deployas (P1-36, P3-43, P3-26, P3-42). CD kan förverkligas genom source control managers (t.ex. Git) och byggverktyg (t.ex. Jenkins) (P2-14).

Det är av stor vikt att snabbt kunna återställa de ändringar man har gjort om något visar sig vara fel (P1-36, P3-42b). Med hjälp av tydligare processer och verktyg kan kod deployas mer smärtfritt, vilket leder till bättre relationer mellan Development och Operations (P3-23). P3 beskrev hur driftavdelningen sätter upp säkerhetsanordningar som garanterar säkra releases (P3-26) och hur drift tillhandahåller en testmiljö som är så likvärdig produktionsmiljön som möjligt för utvecklare att utföra tester på (P3-23).

5.5 Feedback

Alla uppgiftslämnare har sett feedback som en viktig komponent, vilket har en stark anknytning till de fynd vi funnit kring organisationernas agilitet då CD möjliggör snabb feedback till utvecklare (P2-5, P2-10, P3-8) genom att automatisera stora delar av produktionskedjan och det går därmed snabbt och enkelt att deploya kod (P2-10, P2-11, P2-13b, P2-15). Men effektiviteten här beror inte enbart på verktyg utan kräver ett välfungerande samarbete mellan Dev och Ops för att komma ut i produktion snabbt (P3-11, P3-12). Snabb feedback uppnås dessutom genom att tidigt involvera både Dev och Ops i gemensamma aktiviteter (P1-31, P3-31, P2-38). Den snabba feedbacken är i sin tur möjliggörare till ansvarstagande för utvecklarna (P1-30, P3-18) då de får veta direkt om deras förändring ger fel eller fungerar bra (P3-7, P3-10). Förut kunde denna feedback ta upp till flera månader, och i dessa fall har det varit svårt att minnas vad som gjorts samt snabbt åtgärda problemet (P3-29). Dessutom undviks via DevOps situationer där utvecklare arbetar vidare på felaktig kodbas p.g.a. lång återkopplingstid (P3-30). Istället sker mindre förändringar med kortare mellanrum vilket minskar dylika risker (P3-42a, P3-42b) och därmed även risken för osämja (P3-42).

Tabell 5.9 Urval av koncept: Snabb feedback

DevOps-organisationer karaktäriseras av snabb feedback där organisationen så tidigt som möjligt ska få uppmärksamhet kring problem(urval):	
P3-10	Utvecklare ska veta direkt ifall deras förändring inneburit framgång eller fel.
P4-73	Kommunikation med support hjälper utvecklare att förstå kundbehov.
P3-8	CD möjliggör snabb feedback.
P2-5	Support kan direkt vidarebefordra önskemål från kund till utvecklare.
P7-18	DevOps team har tillgång till statistik från produktion.

En annan typ av feedback är hur väl olika initiativ eller förändringar fungerar inom organisationen. Det finns inga etablerade mätetal för detta inom F1 och vissa intervjuade rapporterar att de till stor del går på magkänsla (P1-18, P3-49). F3 har ingen stark process för mätning av DevOps-arbetet(P6-33), men DevOps-team har tillgång till statistik från produktion(P7-18). Mätning och utvärdering anses vara en viktig del för att optimera produktionskedjan(P7-5).

Feedback från kund (P1-22) sker inte direkt till utvecklare utan däremellan finns ofta en säljare eller liknande som buffert (P1-20) för att förebygga att bli överväldigad av för mycket feedback (P1-21). Code reviews är ett kontrollverktyg som används för att se till så att koden håller god kvalitet innan den deployas (P1-33, P2-13a, P5-12, P4-56, P4-56, P4-60, P4-104-107, P5-13). Vidare anses chatt som innefattar alla avdelningar inom hela organisationen som en värdefull kanal för feedback (P2-36a, P7-23).

5.6 Agilitet

Gemensamt för alla företagen som ingick i studien är användandet av agila metoder, verktyg och agila värderingar. Vi har valt att kategorisera Continuous Delivery, att kunna produktionssätta förändringar frekvent, som en agil värdering och praktik då det stödjer korta leveranscykler och konstant addering av kundvärde. Enligt P4 karaktäriseras DevOps på F1 av korsfunktionella team (P4-2a, P4-71) som saknar den traditionella, stora uppdelningen i avdelningar efter arbetsområde (P4-3). Istället är team organiserade med ansvar för särskilda funktioner eller applikationer (P4-2, P4-4). Detta liknas med den tjänsteorientering som F3 arbetar mot där varje team ska ta ett helhetsansvar för en modul i cloud-tjänsten (P7-14).

Tabell 5.10 Urval av koncept: Korsfunktionella team framför stora avdelningar

DevOps organisationer karakteriseras av korsfunktionella teams framför stora avdelningar	
P4-2	Team är funktionsorienterade runt applikationer
P4-4	Team är funktions & applikationsorienterade
P7-14	Tjänstifierar för att minska komplexiteten,Microservices starkt anknutet.

Snabba ändringar och snabb reaktionsförmåga prioriteras framför utförlig planering (P1-14, P1-46). Som förutsättning för snabba förändringar ges anställda fritt mandat att själva göra de ändringar som bedöms nödvändiga (P1-6) samt själv bestämma vilka uppgifter som bör prioriteras (P4-64). Prioritering av arbetsuppgifter identifieras som både viktigt och svårt (P4-108).

Tabell 5.11 Urval av koncept: Snabbt kunna agera utifrån förändringar

DevOps-organisationer karaktäriseras av att snabbt agera utifrån ändrade förutsättningar: (urval)	
P1-46	Prioriterar snabba ändringar framför extensiv planering och fokuserar på att reagera snabbt.
P1-14	Problematik löses ad-hoc framför planering.
P1-6	Fritt mandat för förändring ifall personen tror det är nödvändigt.
P4-66	Kundnytta i fokus vid prioriteringar.
P4-108	Prioritering av arbetsuppgifter är viktigt och svårt.
P2-7	Många spontana arbetsuppgifter stör de ordinarie arbetsuppgifterna
P2-9	Balans mellan kortsiktiga och långsiktiga mål.
P7-5	Mätning och utvärdering en viktig del för att optimera leveranskedjan.

Distansen mellan avdelningar är av betydelse då man ser en stor fördel till snabb kommunikation direkt mellan kollegor (P2-35, P3-37, P4-30). Att snabbt kunna springa över med ett ärende face-to-face i brådskande situationer sågs som en stor fördel för P2, som tidigare upplevt att liknande ärenden hanterades genom långsamma formulärsdrivna processer (P2-35). P3 berättar att storleken på organisationen kan vara avgörande för hur en enskild person kan uppfatta sin roll och på så sätt helheten.

> *"Ju mindre ändringen är som du släpper, desto mindre är chansen att det blir fel. Om man går från att släppa en gång i månaden till fem gånger om dagen så innebär varje ändring mycket mindre. Sannolikheten att det blir fel är ju för det första mycket mindre, och är det så att det blir fel då har du en sådan snabb feedback-loop att du kan laga den snabbt. Antingen kan du laga den eller så kan du ta bort den utan att skada något." -P3. Rad 139.*

För att förebygga att felaktig kod levereras går det att använda sig av code reviews, där versionshanteringsverktyg används för att granska varandras kod innan det sätts i produktion, vilket även bidrar till kunskapsspridning mellan utvecklare (P1-33, P2-13a, P5-12).

Tabell 5.12 Urval av koncept: Snabbt kunna produktionssätta förändringar (CD)

DevOps-organisationer karaktäriseras av att snabbt och ofta kunna produktionssätta förändringar: (urval)	
P2-13b	Utvecklare blir motiverade till att frekvent släppa sin kod.
P3-43	Minskad risk i ändringar innebär en minskad risk för osämja.
P3-42a	Mer frekventa och mindre ändringar för minskad risk.
P4-21	Ärendehanteringssystem med kanban används för att strukturera egna uppgifter.
P4-22	Ärendehanteringssystem med kanban används för att dokumentera arbetsuppgifter.
P5-5	Cloudlösningen produktionssätts kontinuerligt.
P6-7	Jenkins används för byggmiljöer.

P2 uppger att DevOps gör att utvecklare blir motiverade att frekvent släppa sin kod (P2-13b). F2 uppmanar också till frekventare produktionssättningar då frekventare och mindre ändringar minskade risker i samband med släpp (P3-42), och dessa minskade risker leder i sin tur till minskad risk för osämja (P3-43). Synsättet återfinns också i F3 där man arbetar mot en organisationsförändring för att anpassa sin applikationsarkitektur i mindre tjänstifierade applikationer för att kunna produktionssätta oftare (P5-5, P7-14).

Applikationsarkitekturen har en inverkan ifall Continuous Delivery är uppnåbart. Arkitekturen måste stödja frekventa produktionssättningar, där P5, P6 och P7 förklarade att deras klientinstallationspaket inte såg lika frekventa förändringar då denna typen kräver en annan typ av distribution, samt att klientmjukvaran innebär flera miljöer och därav ökar antalet miljöer som måste testas. F3 erbjuder både molntjänster och konventionella installationspaket, där det var molntjänsterna som kunde tjänstifieras och brytas ut och stödjas av mindre utvecklingsteam för att tillåta frekventare produktionssättningar (P5-5, P7-14).

Tabell 5.13 Urval av koncept: Applikationsarkitektur som stödjer CD

DevOps karakteriseras av en applikationsarkitektur som stödjer snabba leveranser (CD)	
P5-5	Cloudlösningen produktionssätts kontinuerligt.
P7-25	Testbarheten av produkten avgör hur väl en pipeline kommer att fungera.
P7-24	SaaS produkter lämpar sig bättre för delivery pipeline.
P7-14	Tjänstifierar för att minska komplexiteten, Microservices starkt anknutet.
P5-11	Skillnader produktionsflöde mellan cloud och desktop beror till stor del på geografiskt

5.7 Insyn

Gemensamt för alla uppgiftsgivare är att insyn i organisationen är en återkommande egenskap, där samtliga ser en fördel med att individen bör kunna se utanför sitt specifika område/avdelning. Detta ses som den återkommande utmaningen för organisationerna och någonting man säger sig arbeta för att skapa mer av. Exempelvis ser vi att P3 beskriver det som att en utvecklare kan skriva sin kod och "kasta den över väggen" och inte veta vad nästa steg är.

> *"[...] ju större ett företag är desto svårare är det för en enskild person i företaget att förstå sin roll i helheten. Man gör sin grej och så levererar man och sen så ... man vet knappt vad nästa steg är för något. I ett mindre företag, helst, ifall man kör DevOps då vet man ju att det här är kraven som kommer in, här utvecklar man och det kommer se ut så här på slutet. "*
> *-P3, Rad 136.*

Hos F3, som är ett större bolag än F1, arbetar de med vad de kallar *champions.* Dessa personer är nominerade av sin avdelning för att skapa insikt i DevOps-organisationen och skapa en kunskapsdelning inom organisationens avdelningar (Hütterman, 2012, sid. 72).

"Champions är något vi använt, det tycker jag är jättebra. Det innebär att olika delar av organisationen får utpeka någon som skall vara lite mer kunnig. Och då stödjer vi dem jättemycket i början och pratar jättemycket med dem i grundfasen. Det hjälper på olika sätt: För det första får vi organisationen intresserad av förändringen, för då bidrar dom hela tiden. Och då ger dom kompetens, både om förändringen men också om det vi faktiskt implementerar samtidigt som vi implementerar det. Så, feedback till oss, en kompetens till dem, en förståelse för dem, en förståelse till dem" - P6, Rad 40

Som vi kan se av P6's beskrivning är kunskapsspridning starkt anknutet till insynen i organisationen, och överlappar med vad man se som samarbete mellan gruppgränserna.

Tabell 5.14 Urval av koncept: DevOps organisationer karaktäriseras av transparens/insyn

DevOps organisationer karakteriseras utav en organisatorisk transparens, där varje medarbetare ska ha en insyn till andra delar av organisationen utanför sitt eget område och ha ett helhetsperspektiv av organisationens värdekedja: (urval)	
P1-48	Undvika individberoende genom att alla har en helhetsbild
P3-41	DevOps för med sig en större insikt i organisationen för individen
P4-95	Insyn i andra roller ökar förståelse.
P4-109	DevOps metod för att få bredare synsätt på sitt arbete.
P6-6	Viktigt att se behov hos andra avdelningar.
P6-8	Viktigt att se business value i allt som utvecklas.
P6-21	Champions ger insyn till utvecklarna åt de som tillhandahåller verktygen.
P7-7	Kunskapsspridning skapar ett helhetsperspektiv

6 Diskussion

Vi har i den här studien använt oss av definitionen att DevOps är en strategi för mjukvaruutveckling och produktionssättning av mjukvara där målet är en effektivare leveransprocess. Detta görs genom kultur, automation, mätning och delning. Detta kapitel börjar därför med en utvärdering av ramverket CAMS ställt mot den data vi erhållit i undersökningen. Därefter följer en diskussion kring hur DevOps bättre skulle kunna beskrivas utifrån det analysarbete vi utfört ställt mot tidigare forskning.

6.1 CAMS

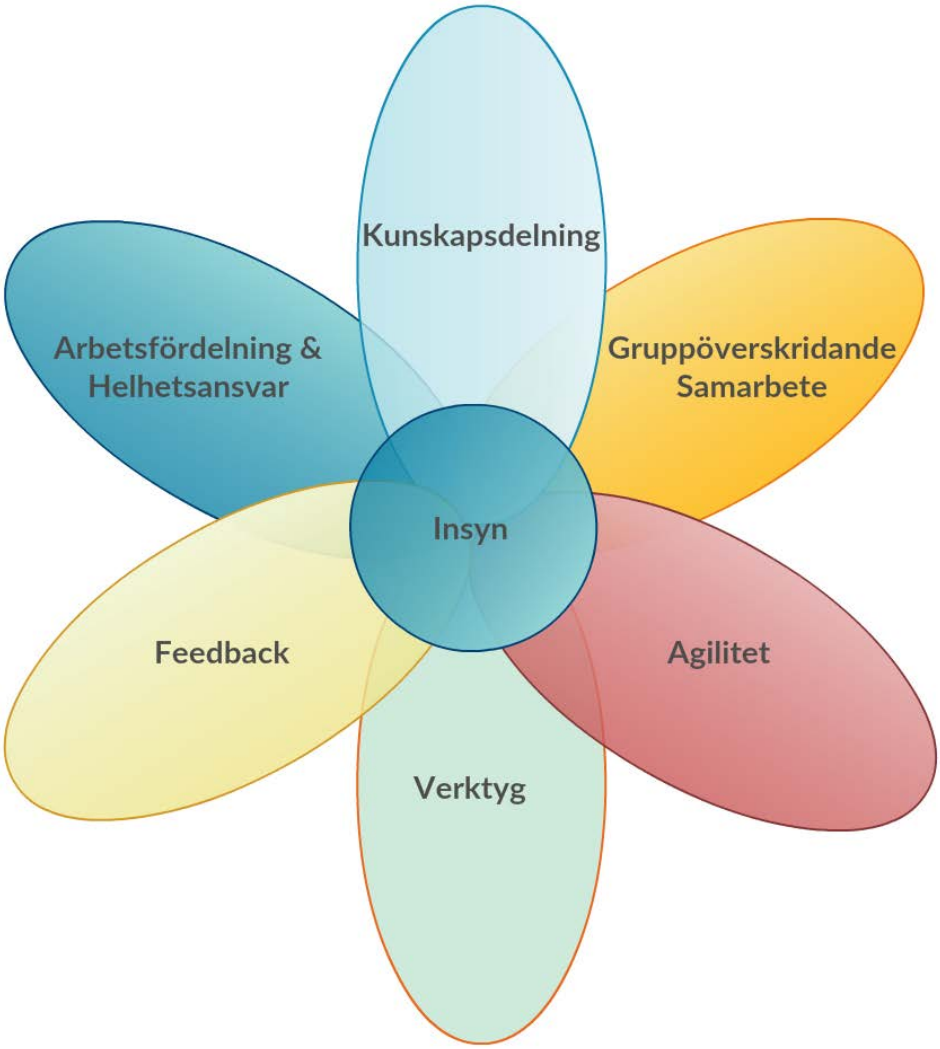
Vi har i studien använt oss av CAMS-värdegrunden som teoretiskt ramverk för finna vad som karaktäriserar en DevOps-organisation. Även ifall dessa har väglett vår undersökning till värdefulla fynd, bedömer vi att det inte främjat en värdefull kategorisering av fynden. De nyckelpoänger vi fann i vår analys var svåra att placera till en särskild värdegrund som ett resultat av dess vaga beskrivning i litteraturen, vilket i många fall resulterade i överlappningar som beskrivits i kapitel 4.2.

Lwakatare et al.(2015) redovisar som utmanare till CAMS ett konceptuellt ramverk baserat på deras litteraturgenomgång av 22 studier rörande DevOps. I deras ramverk läggs ett större fokus vid monitorering där delning tagits bort och ersatts av övervakning och kultur har bytts ut mot samarbete. Övervakning innebär, enligt studien, bland annat att analys sker av prestandarelaterad data i anknytning till användardata från kund. Detta står i kontrast mot de fynd vi gjort i denna studie som talar för att monitorering och mätning inte har en lika viktig plats för alla de studerade DevOps-organisationerna.

Likt Lwakatare et al. (2015) fann vi en avsaknad av en av värdegrunderna, då F1 och F2 pekade på att mätning inte var någonting som kännetecknade respektive organisations arbete utan snarare sågs som en utmaning. I F3 påpekades vikten av mätning (P7-5) men fokus låg på kundvärde i utvecklingsarbetet snarare än på effektiviteten i leveransprocessen och kunde inte kopplas till den mätning som påträffats i litteraturgenomgången av CAMS. Med anledning av den tydliga avsaknaden av värdegrunden *mätning* och den mångtydiga värdegrunden *kultur* såg vi inte att ramverket tjänade till att svara på frågeställningen. Ett nytt ramverk skapades därför utifrån de kategorier som togs fram i analysen.

6.2 Insyn

Från de koncept och kategorier som funnits i datan ses en gemensam nämnare, det vill säga en kärnkategori. Kärnkategorin representerar kategoriernas gemensamma bidrag, och vi har i denna studie funnit *insyn* som den gemensamma nämnaren. De underliggande kategorierna arbetar gemensamt för en större insyn och transparens i organisationen, och många av kategorierna kräver att en insyn finns för att det skall kunna existera. Kategorierna kan således både möjliggöra insyn och vara beroende av insyn.



Figur 6.1 Konceptuell modell: Insyn

Insyn syftar till den översikt en person uppmanas att ha utav organisationen och hur organisationen bör vara transparent för att kunna erbjuda en större insyn. En anställd i ett företag ska ha en bra uppfattning av sin egen position och roll i leveranskedjan och samtidigt ha en god förståelse för vad andra i organisationen, tidigare och senare i SDLC utför. Genom att ha en ökad förståelse för de moment som innefattar SDLC, desto bättre beslut och arbete kan individen utföra i sitt moment (P1-52).

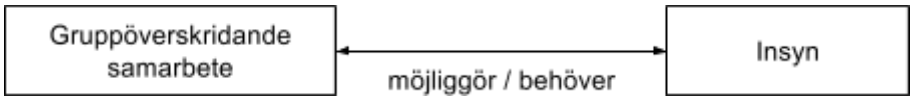
De aktiviteter är således de teoretiska kopplingar som har visat sig ha relationer sinsemellan, vilka kommer att presenteras i det här kapitlet. Detta kommer även att ställas mot tidigare forskning av ämnet DevOps för att se var det finns stöd och var det skiljer sig.

Vi har i datan identifierat arbetsfördelning & helhetsansvar, kunskapsdelning, gruppöverskridande samarbete, feedback, verktyg och agilitet som kategorier; variabler som tillsammans påverkar insynen och helhetsperspektivet. Helhetsperspektivet har varit ett återkommande tema i alla intervjuer, där deltagarna pratat om vikten av att varje person har en bred insikt och kunskap, samt en förståelse för vad andra utför i leveranskedjan (P1-48, P3-41, P4-109).

6.2.1 Gruppöverskridande Samarbete

Att minska avståndet mellan utvecklare och drift är en uttalad grundpelare i DevOps, vilket även återfinns i våra resultat. Det är genom att uppmuntra en kommunikation och samarbetsvilja mellan lägren som en effektivare leveransprocess kan uppnås, då leveranskedjan består utav moment som berör båda parter.

Hüttermans syn på DevOps målsättning att öka kommunikation mellan drift och utvecklare (2012, sid. 8), och Swartouts syn på DevOps som ett sätt att arbeta där drift och utvecklare arbetar i harmoni mot samma mål utan organisatoriska barriärer (2012, förord) har stöd i studien, där respondenterna såg samarbetet som en av de viktigaste aspekterna (P2-38, P3-45, P4-75 m.fl.).



Figur 6.2 Gruppöverskridande samarbete möjliggör och behöver Insyn

Ett gruppöverskridande samarbete är ett praktexempel på att insyn är en karaktäriserade egenskap, då man i ett samarbete får en förståelse för varandras situation och kunskap sprids.

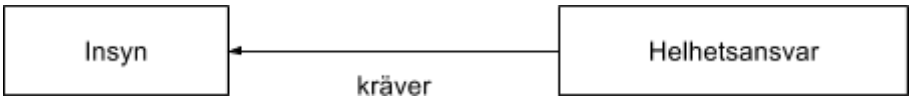
6.2.2 Arbetsfördelning och helhetsansvar

Humble och Molesky (2011) manar för att utvecklare ska vara tillgängliga när incidenter inträffar och kunna bistå Operations med felsökning, men i F1's fall är det utvecklarna som ansvarar för koden i produktion (P1-30) och Operations är inte inblandade mer än ifall det skulle vara ett problem med den underliggande plattformen (P1-32, P4-44). P3 ser det också som utvecklarens ansvar i produktion (P3-18) och att det är driftens uppgift att erbjuda utvecklaren att ta ansvar genom rätt verktyg men framförallt tillgången till kunskap.

Operations ansvarar ju för att systemen är uppe medan utvecklare ansvarar för att produkten funkar som den ska. Och det betyder ju att för att development ska veta hur man ska deploya en sak på en server så måste de ju få "det berättat för sig" -P3, Rad. 30

Detta motsätter sig vad Humble och Molesky (2011) säger om att utvecklare ska bistå operations, men får samtidigt stöd av den bild Feitelson et.al (2013) har av att utvecklaren äger koden i produktion. Ifall en tjänst eller applikation slutar att fungera pekar Hütterman (2012, sid. 27) på att så många som möjligt ska kunna felsöka problemet, vilket ses i vissa delar av F3 där ansvaret delas mellan driften och utvecklarna genom att utvecklarna ständigt övervakar produktionen (P7-19). Klart står dock att en involvering utav utvecklare i produktion ingår i båda fallen, där insyn mellan avdelningarna är centralt.

Utvecklarens tillgång till insyn i produktionen och driftens tidiga involvering möjliggör det helhetsansvar av produkten och den möjlighet att bistå driften som Humble och Molesky (2011) förespråkar, samt möjliggör att alla ska kunna bistå vid en incident likt Hütterman's beskrivning av rollfördelningen (2011, sid. 27).



Figur 6.3 Helhetsansvar kräver insikt

6.2.3 Kunskapsdelning

Vidare talar man hos F1 om otydliga roller och att målet är att alla bör kunna utföra flera arbetsuppgifter. För att minska roller och titlars betydelse behövs det att flera personer i viss mån ska kunna utföra samma arbetsuppgift. Detta kräver att det sker en kunskapsdelning sinsemellan och att individer delar sin expertis och erfarenheter med andra (Swartout 2012, sid. 54). Genom att motivera kunskapsdelning inom organisationen och uppmuntra en god kommunikation kan insynen för individen öka. Ett exempel är att man i F3 arbetade med champions i olika team och avdelning för att sprida kunskapen om nya verktyg och för att skapa engagemang i gruppen mot vad som pågår i resten av organisationen och leveranskedjan.

Figur 6.4 Kunskapsdelning möjliggör insyn

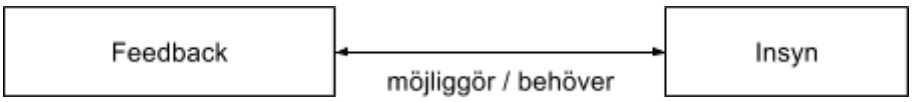
Vidare ser vi att strukturen och storleken på organisationen kan ha en stor roll i hur insynen påverkas, där fysisk närhet underlättar en snabb kommunikation och på så sätt även kunskapsspridningen. Här ligger också en stor del av problematiken, då organisationer kan ha en tidigare struktur där utveckling och drift sitter separerat, vissa fall rent av på olika

lokationer. Att arbeta i geografisk anslutning till andra avdelningar bidrar till kommunikation och tillit (P3-38), något som även stödjs av Swartout (2012, sid. 97) som hävdar att “en avsaknad av fysisk närvaro alltid är en barriär”.

6.2.4 Feedback

Feedback kan vara ett medel för att skapa insyn för individen, där till exempel en utvecklare direkt får feedback från *delivery pipeline* ifall dennes kod inte har passerat de uppsatta spärrarna. Genom att få en detaljerad feedback på vad som inte passerade ökar utvecklarens insyn till vad som krävs av produkten för att få passera vidare i kedjan och således en förståelse för hela leveranskedjan. Detta stöds av Hütterman (2012) som ser just feedback som en av de viktigaste bidragen i en automatiserad leverans (sid.111).

Till skillnad från feedbacken i *measurement* i CAMS-perspektivet, där större delen av feedbacken kommer från den automatiserade leveranskedjan, har studien visat ett fokus på feedback som kommer från andra avdelningar via ökat samarbete och kommunikation (P1-22, P2-5, P3-28, P4-73). Feedback från de automatiserade leveranserna är dock inget som helt glömts bort (P3-10, P3-28), men studien fann ett överväldigande fokus på just feedback via kommunikation.



Figur 6.5 Feedback möjliggör insyn samtidigt som det behövs

Feedback mellan avdelningar nämns även av Fitzgerald & Stol (2015), som ser att utvecklarna ofta arbetar på produkter med låg kundnytta om det inte finns bra feedback från Support-avdelningen (sid.5). Feedback kan även ske genom agila verktyg, såsom JIRA, där avdelningarna får insyn i varandras arbete och kan se var andra avdelningar har tagit vid av arbetet och dess fortskridande (Hütterman 2012, sid. 128).

6.2.5 Agilitet

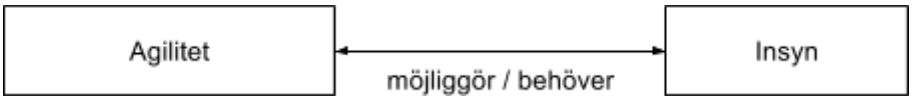
Gemensamt för alla organisationer är att de uttalat använder sig av Continuous Delivery, som är en agil praktik där snabba och frekventa produktionssättningar uppmanas. CD får en framstående roll och starkt karaktäriserande för DevOps-organisationen. Kategorin verktyg handlar till stor del om att förse organisationen med kapabiliteten att förverkliga CD.

För att få en större insyn i arbetet kan man med fördel använda sig ett uppsjö agila metoder, exempelvis parvis programmering och code reviews (Swartout 2012, sid.54). Code reviews används i stor utsträckning hos F1, där varje ändring skall gås igenom av åtminstone en kollega (P4-60). Code reviews används även till viss grad inom F3 (P5-12).

I resultaten framgick att kundnytta skall ligga till fokus vid prioriteringar kring vilket arbete som skall utföras (P4-66, P4-72) vilket ligger i linje med var som förespråkas i det agila manifestet: *"Vår högsta prioritet är att tillfredställa kunden genom tidig och kontinuerlig leverans av värdefull programvara"* (Beck et al. 2001). Prioriteringen av kundnytta kräver en insyn i vad som genererar kundvärde, vilket P4 menade ibland är en utmaning.

> *Det är rätt mycket tankeverksamhet som går åt till att väga "hur viktigt är det här? vad ger det för kundnytta?" eller sitter vi och gör grejer som jag inte borde göra, typ? I hela organisationen är det ansvaret inte särskilt toppstyrt utan det ligger på personnivå i princip i hela organisationen. Ganska mycket i alla fall. Det du gör, är det relevant? Borde du göra det? Är det kundnyttigt?*
>
> *-P4, Rad. 126*

Vidare används kanban för att strukturera arbete och ge insyn mellan avdelningar (P4-14, P4-23 m.fl.), något som har stöd i Swartouts uppmaning om att implementera agila arbetsmetoder, som till exempel kanban, i arbetet (2012, sid. 15).



Figur 6.6 Agilitet möjliggör insyn och behöver insyn

Det agila manifestet förespråkar även face-to-face kommunikation och ser det som det mest effektiva sättet att förmedla information både till och inom utvecklingsteamet (Beck et al. 2001). Face-to-face kommunikation sågs även som något viktigt hos F1, där brådskande ärenden snabbt kunde kommuniceras mellan avdelningarna (P2-35).

6.2.6 Verktyg

Hütterman (2012, sid. 8) har gjort referensen till det agila manifestet att DevOps ser personer framför verktyg. I det agila manifestet förklarar man att då man värderar båda så värderar man det förstnämnda före det sistnämnda (Beck et al, 2001). Vi kan se att verktyg har ett stort värde i DevOps-organisationen då det understödjer de flesta av de identiferade aktiviteterna och värderingarna som främjar leveranskedjan.

I F1 och F3 används en chatt över alla avdelningar där releaser kan diskuteras mellan avdelningar (P2-16, P6-12, P7-23a), och kommunikation förenklas (P2-36a). Swartout (2012, sid. 75) uppmanar organisationer att arbeta enligt just gruppchattar eller onlineforum för att stärka kontakten mellan avdelningarna. Denna arbetsmetodik skapar ett öppet informationsflöde och kan ge en större insyn mellan avdelningarna som sammansluter organisationerna då alla har tillgång till samma information. I F1 och F3 har den som önskar tillgång till andra avdelningars dokumentation och kanban-boards vilket skapar en insyn för den enskilde att den kan se alla aktiviteter i hela organisationen.

Användandet av automatiserade bygglösningar har varit centralt för DevOps-arbetet hos samtliga organisationer (P1-35, P2-14, P3-8, P4-39, P6-1). Hütterman(2012) nämner SCM och byggverktyg som medel för detta (sid. 63), vilket var starkt förankrat i de undersökta organisationerna (P2-14, P4-54, P4-57, P6-2). Linjen mellan vad som är kunskapsdelning och vad som hör till kommunikation blir svår att dra då kunskapsspridning ofta sker i form av kommunikation, men det kan argumenteras att kunskapsdelning även kan ske i form av ett förenklande skript för ex. produktionssättningen och testningen av den commitade koden. Verktyget, eller den automatiska testningen som satts upp av produktägare kommer att generera ett fel tämligen direkt där utvecklaren blir notifierad om vad som gått fel. På så sätt blir utvecklaren medveten om de krav som satts upp för den koden och får en insikt i vad som krävs av.

Figur 6.7 Verktyg möjliggör insyn

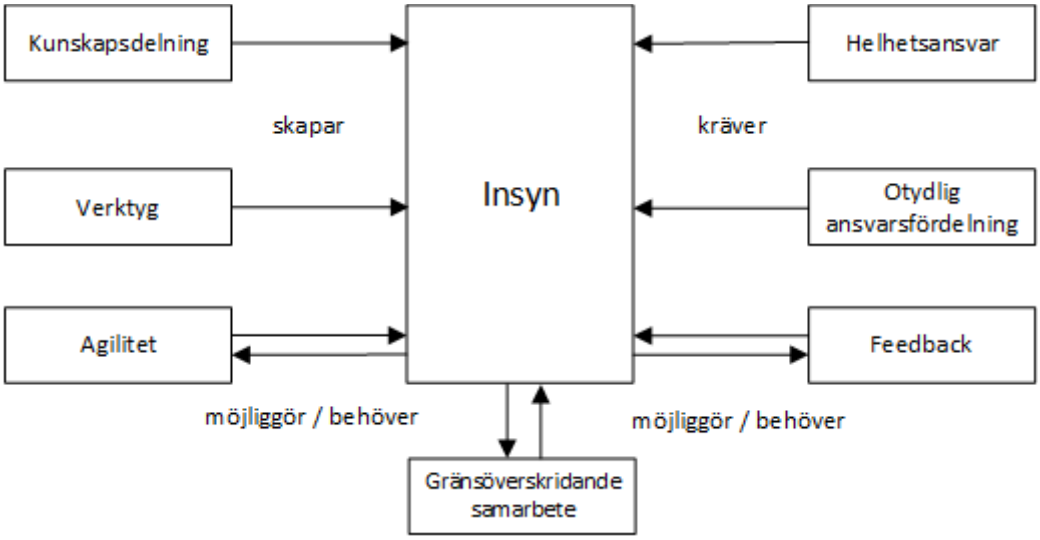
6.3 Sammanfattning

Koncepten som tagits fram ur datan har många gemensamma beröringspunkter, och det sker samspel kategorierna emellan. Exempelvis kan kunskapsspridning ske genom gruppöverskridande samarbeten och feedback, och agila arbetsmetoder stödjs av verktyg.

Dessa kategorier och koncept bidrar tillsammans med att skapa en organisatorisk insyn för att skapa en effektivare leveranskedja och ses som karaktäristiska för DevOps-organisationerna i studien.

> *"Ja bra fråga, det handlar väl mer om en organisation där så många som möjligt egentligen har bra insyn och kunskaper om hela flödet utifrån krav till produktion och tillbaka. "*
>
> *-P7, Rad 13, på frågan vad DevOps är för honom.*

Olsson et al. (2014) fann i sin studie om vilka barriärer en organisation måste överkomma för att uppnå sitt mål om Continuous Delivery att transparensen ofta var bristfällig och att avsaknaden av överblick stod i vägen för den effektiva leveranskedjan. För att överkomma detta såg de att fler enheter från organisationen behövde inkluderas och dela utvecklingsavdelningens mål att släppa oftare och mindre. Då DevOps är ett medel för att förverkliga CD (Humble & Molesky 2011) och avsaknad av inblick kan stå i vägen för en effektiv leveranskedja (Olsson et al. 2014) ser vi ett teoretiskt stöd bakom insyn som en central del av DevOps.



Figur 6.8 Relationen till insyn

Vi öppnar upp för att fler komponenter, organisatoriska egenskaper och arbetsmetoder går att applicera på modellen beskriven i figur 7 och 13. Det som skapar en organisatorisk transparens och insyn för individen i leveranskedjan kan ses som typiskt DevOps, där insynen är en avgörande faktor. Insynsmodellen går därmed även i viss mån att applicera på tidigare föreslagna ramverk som t.ex. Smeds et al (2015), med insyn som både stöd och resultat av vad de valt att kalla *cultural enablers*, där de kulturella möjliggörarna kräver att organisationens olika avdelningar har en helhetsförståelse för leveranskedjan.

Capabilities	Continuous planning Collaborative and continuous development Continuous integration and testing Continuous release and deployment Continuous infrastructure monitoring and optimization Continuous user behavior monitoring and feedback Service failure recovery without delay
Cultural Enablers	Shared goals, definition of success, incentives Shared ways of working, responsibility, collective ownership Shared values, respect and trust Constant, effortless communication Continuous experimentation and learning

Figur 6.9 Capabilities & Cultural Enablers av Smeds et al. (2015, sid. 171)

7 Slutsats

7.1 Sammanfattning

Studien ämnade svara på forskningsfrågan: *"Vad karaktäriserar DevOps-organisationer?"*. För att svara på frågan utfördes en empirisk undersökning där intervjuer hölls med anställda inom DevOps-organisationer om hur DevOps såg ut på deras arbetsplats.

Det teoretiska ramverket CAMS visade sig ha ett svagt stöd för fynden i undersökningen då ett mycket litet fokus låg vid *mätning*, samtidigt som *kultur* sågs som ett alltför otydligt och omfattande begrepp. En ny modell för organisatoriska egenskaper som karaktäriserar DevOps skapades därför utifrån de nyckelpoänger och koncept som togs fram i analysen.

Av modellen framgår att DevOps-organisationer karaktäriseras av en stor insyn och transparens. Insynen ses som förmågan att se hela organisationens värdekedja och ha ett helhetsperspektiv av livscykeln, där alla delar det gemensamma målet om att leverera mjukvara snabbt och frekvent. Detta främjas genom ett nära samarbete över gruppgränser där det sker en aktiv kommunikation och kunskapsdelning i hela organisationen som understödjs av verktyg. I organisationerna såg man det som viktigt att viktig kunskap inte knöts till enskilda personer, vilket motarbetades genom automation och ständig kunskapsspridning.

Gränser mellan roller visade sig vara otydliga då de anställda förväntades utföra arbetsuppgifter som traditionellt ligger utanför den formella rollen. Exempelvis förlängdes utvecklarnas ansvar för sin kod hela vägen ut i produktion. Feedback visade sig i första hand komma ifrån andra avdelningar snarare än som ett resultat av mätning, och agila arbetsmetoder som t.ex. code reviews och kanban var ett stort stöd när det gällde såväl struktur som feedback och kunskapsdelning.

7.2 Förslag till framtida forskning

Då studiens omfång enbart omfattade tre organisationer vill vi öppna upp för ytterligare forskning på området för att stärka teorin och den framtagna modellens överförbarhet.

Appendix A

Intervjuguide för intervju P1 & P2

Culture	☐ Hur sker Kommunikation? ☐ Hur ser man på ansvar? ☐ Hur ser man på samarbete mellan grupperna? ☐ Involvering? ☐ Vad värderas?
Automation	☐ Hur automatiserat är arbetet? ☐ Continuous Delivery?
Measurement	☐ Hur ser man på och hanterar feedback? ☐ Hur utvärderar man arbetet?
Sharing	☐ Hur ser man på kunskapsdelning? ☐ Delning av verktyg?

Appendix B

Intervjutranskript och nyckelpoängskodning av intervju med **P1**.

1.	**I och med att ni arbetat med Cloud från början, är det det som fått er in på DevOps.**
2.	-- Ja, det gör ju att man behöver ha en struktur som är mer ett IT bolag än ett Telefonibolag. Telia och sådär har ju stora rum med massa sladdar och sådant i, det har vi ju aldrig haft. Utan vi har ju bara haft folks datorer och nån serverhall någonstans, så hela inställningen har ju varit IT vänlig från början och då tror jag DevOps är ett naturligt steg.
3.	*Key Point:* P1-1 DevOps naturligt för utvecklande IT-bolag
4.	**Har ni alltid kört med DevOps?**
5.	Det beror ju på hur man definierar DevOps. Det kan vara ganska normalt när man är ett litet startup bolag med få anställda, så vill man ju gärna att alla ska kunna göra allting. Det ska inte vara *"Jag måste be Knut göra om det här, för att han är den enda som kan det och då måste jag vänta på att han först hjälper Arne här borta.* " Utan man vill ju kunna agera snabbt och bara kunna fixa den där servern eller patcha den här applikationen. Det började ju som det att alla skulle kunna göra allting och sen så visst är där ju alltid folk som är bättre på vissa saker, men man får sprida den kunskapen. Sen efterhand som det har växt från 1 -3 utvecklare så har det ju hela tiden.. Man har hela tiden funderat på hur länge funkar den approachen, med liksom den nya startupen, ett litet gäng som programmerar tillsammans.. Hur länge fungerar den, när behöver man införa en större process, dela upp i avdelningar och sånt, och vi har inte riktigt nått det taket än, man tänker att det bör hända vid 8-10 pers, men nu är vi 46 tror jag, och det fungerar med en ganska liknande approach att alla ska kunna göra allting.. Samtidigt så har ju serverbitarna kräver inte så mycket ändring, det är mycket som man sätter upp det, nu har vi virtualiserade miljö också, så man lägger ju bara in nya virtuella hostar vi behov och det är ju ganska enkelt att göra men större ändringar i infrastrukturen är fortfarande lite trögare och det är nog inte alla som det kan det det är nog väldigt få, en eller två stycken som skulle våga gå in i serverhallen och dra ut ett rack och flytta ut det liksom.
6.	☐ Definitionen av DevOps kan variera ☐ DevOps mer anpassat för mindre minde bolag ☐ Önskan om att flera ska kunna göra mycket ☐ Är att agera snabbt länkat till att alla ska kunna göra allt?

	☐ Ska sprida kunskap ☐ En oro att upplägget inte kommer hålla i längden ☐ Approachen verkar förändras med ökning i personal styrka ☐ Behovet av förändring beror till hur man har satt upp server bitarna ☐ Vad innebär större ändringar i infrastrukturen? Vad innebär trögare? Innebär det mindre *tillit*? Ser vi en koppling mellan tillit och storleken på förändringen? ☐ Bara vissa **vågar** göra en större förändring i infrastrukturen. Beror hur det är uppsatt Key Points: ☐ P1-2 Definitionen kan variera ☐ P1-3 Många ska kunna göra mycket ☐ P1-4 Kunskap ska spridas ☐ P1-5 DevOps faller naturligt för mindre bolag
7.	**Är det så att, ifall den personen som vågar så har den personen tillåtelse att göra det?**
8.	Ja, mandatet är liksom lite fritt, man får göra det man anser viktigt eller anser behövas. Sen är det ju ganska vanligt att folk vill gärna få saker godkända av andra liksom så att de inte bara: “Amen jag tycker vi skulle slänga hela den här servern men det tyckte inte de andra 30 så att.. “ Då kanske det inte är bra. Man får ju ändå kolla av, känna av .. men det är ingen en stor omröstning som sker i något råd eller något sånt,
9.	*Fritt mandat för ändringar, ifall den personen ser det som viktigt* *Betonar vikten av att andra ser en poäng i det.* Key points: ☐ P1-6 Fritt mandat för förändring ifall personen tror det är nödvändigt ☐ P1-7 Inget större beslutsorgan
10.	**Och det känner ni har fungerat bra?**
11.	Ja! Det är ju en vanlig klyscha det här med delat ansvar är inget ansvar och saker faller mellan stolarna och visst det kommer det alltid att göra men änsålänge så har det plockats upp ganska mycket bollar och vid behov. Så det brukar fungera.
12.	Key Points: ☐ P1-8 Delat ansvar kan innebära missar
13.	**Om det är någon som inte känner sig bekväm med de här serverbitarna, går han då över och spontant frågar ifall någon kan fixa det då?**
14.	Ja, jag tror det första steget för de flesta är att så här: “kan inte du fixa den här grejen? “. Då har det ofta blivit att, ja jo det kan man göra, så de kan fokusera på andra grejer längre upp i stacken. Men jag hade

	ju tyckt det var roligt ifall alla kunde göra allting, det är ju en utopi någonstans liksom, det går ju inte, så man försöker ha ett överlapp där. Vi sitter just nu i en situation där en nyckelperson har slutat. Då försöker man täcka in det spannet han täckte in, nu hade han överlappande ansvar in in i vad andra också hade koll på, men där är också en glugg som bara han hade koll på. Så där märker vi ju.. man får ju smaka på, även ifall man tänker DevOps, så missar man ju de här puckarna ibland.. Och då får man lida för det!
15.	Key Points: ☐ P1-9 Alla borde kunna göra allting ☐ P1-10 Man kan tänka DevOps ☐ P1-11 Vill undvika nyckelpersoner
16.	**Är det svårare att se vilka gluggar man kan bli av med när det här med ansvarsområden faller bort eller?**
17.	Ja, vi har ju inte tydligt definierade roller som sådana, så man vet inte riktigt vad vi saknar och vi har inte någon som ansvarar för det här .. Så då vet vi inte att vi har ett behov av någonting nä. Samtidigt så om vi inte märker att behovet inte finns så kanske det inte riktigt finns liksom. Vi brukar ofta har den approachen att vi , äh man får lösa problem efterhand som de kommer istället för att sitta och fundera fram alla problem som skulle kunna uppstå, och lösa dem på förhand. Avvägt självklart mot säkerhetsrisker och ja, vad är värt att investera tid i och sådana grejer, så någon form av horisont försöker man ju alltid ha uppåt.. Men organisatoriskt är vi väldigt självbalanserade på något sätt.
18.	Key points: - P1-12 Inga tydliga roller - P1-13 Svårt att se saknad kompetens iom. Otydliga roller - P1-14 Problematik löses ad-hoc framför planering
19.	**Är det så att ni har sett att man försökte täppa till en kunskap, en glugg, med en ny person, men att den personen har hamnat någon annanstans sedan, eller att man blir anställd för en sak men att man plockar åt sig lite olika?**
20.	Vi har ju en konstigt approach där också faktiskt, när vi anställer så brukar vi sällan leta efter nån viss nyckelkompetens. Våra annonser för utvecklare är ganska generiska och breda.. Har ni jobbat med … och så en lista på 300 olika grejer som inte är förenliga med varandra egentligen och sen så kommer det in folk som kanske kan en tredjedel av de här grejerna. Jamen kul! Kom in på intervju! Så försöker vi istället hitta folk som har en bra personlighet och en bra inställning liksom, är intresserad av teknik och vill lära sig mer och vill använda teknik för att lösa problem, så tror vi istället att om man bara tar ett gött.. Goa människor som är smarta och sätter dem på samma

	plats och låter dem jobba tillsammans så kommer det komma bra grejer utav det.. Istället för att ousch.. Vi har ingen som kan MySQL Server 2.93, det måste vi anlita 3 pers som kan sitta med. Då är det istället, ta in folk som kan någonting annat kanske och så slänger vi det på den. Försöker lösa det på det sättet istället.
21.	Key points: ☐ P1-15 Ser bred kunskap framför nyckelkompetens ☐ P1-16 Ser personlighet och intresse framför nyckelkompetens
22.	**Och det fungerar bra för er?**
23.	Ja! Det tycker jag! Det funkar framför allt förvånansvärt bra egentligen. Jag kom in när vi var 10 pers och det var lite.. "Osh, vi måste nog har lite struktur på det här stället", men sen så har vi ändå fortsatt och nu har vi fyrdubblat det. Det funkar ändå.
24.	Key points: P1-17 Osäkerhet över konceptet till en början
25.	**Fräckt!**
26.	Ja man blir lite shockad nästan!
27.	**Är det någonting som inte fungerar så bra med den approachen?**
28.	Det är en svår fråga det! Vad funkar inte bra.. Alltså det är ju alltid.. Eftersom vi har en såpass agil eller öppen approach till hela utvecklingsmetodiken och DevOps tänket så är det ju lite svårt att utvärdera. Så vi vet ju inte riktigt hur bra vi är, eller om vi gör en ändring så märker vi inte av.. vi har inga mättal på liksom säg att: "Oh det gick bättre och vi har kunnat skala ned det här teamet från 10 till en person". Utan det är mycket gut-feeling man får gå på liksom, hur känns det, vad är känslan hos snittutvecklaren här nu, känns det bra när han går hem från jobbet eller är han orolig för att det ska krascha på natten och att han ska bli uppringd och någon skriker på honom? Så man försöker fånga upp det, och det är ju social engineering som kommer in där. Hur fångar man upp det då och hur märker man att gut-feelingen i snitt är dålig eller är gut-feeling bra hos dem som inte har kundperspektiv med den är dålig hos dem som har kundkontakten liksom. Det är ytterliggare ett problem som man behöver lösa.
29.	Key points: P1-18 ch agil struktur blir det svårmätt och man får förlita sig på magkänsla och social engineering P1-19 DevOps ses som ett tänk
30.	**Hur ser det ut hos utvecklare, hur nära kommer de kund?**

31.	Det är ju ganska isolerat egentligen. Skulle säga att utvecklarna har sällan direkt kontakt med våra slutkunder. Det kan ju ha och göra lite med vilken branch vi är inne i. Vi siktar ju enbart på företagstelefoni och företagskommunikation, och på dem större företagen så är det ju någon IT-chef som brukar köpa in grejerna och då vill dem ha en professional account manager som de har sin kontakt med här. Så i dem fallen så är det ju ofta en buffert där på en säljare eller någonting som vi kallar adviser uppe på teknisksupportavdelningen eller adviseravdelningen som har kontakten med och känner kunden och har en personlig anknytning till där. Men sen har ju dem direkt kontakt med utvecklarna. Vi försöker hålla nere den processen också, om vi nu inte kan ha direkt kontakt med utvecklarna till kund, vilket kan vara väldigt fördelaktigt i vissa typer av grejer, men ibland kan det bli lite overwhelming också. Speciellt om man har väldigt många kunder, så är det jobbigt för utveckларn och bolla det i huvudet. Så då är det bra om man sållar det via lite trattar, men håller ändå feedback loopen så kort som möjligt.
32.	Key Points: P1-20 Närhet till kundkontakt kan vara beroende av vilken branch man tillhör P1-21 - För mycket feedback kan vara överväldigande
33.	**Är det så att er vidareutveckling är feedback-styrt? Är det genom customer-feedback ni ser vad ni ska utveckla härnest?**
34.	Ja det är ju. Det är mycket mycket buggfixar som kommer in via customerfeedback loopen. Det är inte jättemycket, nu har jag inte hundra procent koll på det här, men jag tolkar det som att det inte görs jättemycket innovation som kommer från kunderna. Än en gång så är det kanske kopplat till branschen. Det är inte så många som tänker att vi skulle kunna ha kortare telefonnummer.” Kan vi få det? Nä det går inte “. Då försöker man istället innovera på ja, kolla av andra brancher. Vi har utvecklat vår egen chat ganska nydligen för att kunna knyta in den in i paketeringen av helhetskommunikationen för företag och att man försöker identifiera trenderna ute i världen; “att okej, fler och fler kommunikationer nuförtiden sker över chat istället för att man sitter och ringer.” Ja då kanske vi skulle ha det färdig paketerat i produkten vi säljer och så utvecklar vi det och försöker fånga upp det på det sättet. Vi har kallat utvecklingsavdelningen för produkt och utvecklingsavdelningen, där det har legat på utvecklarna att också innovera och komma på nya produkter. Sen så har man alltid individer som brinner väldigt mycket för produktutveckling och vissa som är mer som hellre vill sitta och knacka kod. Så det kommer ju lite olika mycket input från olika människor. Men jag tolkar det som att mycket av innovationen sker internt, snarare än kundstyrt.
35.	Key points: P1-22 Utvecklingen är beroende av kundfeedback

	P1-23 Trender kan styra utvecklingen P1-24 Utvecklare står för stor del av innovationen
36.	**När det uppkommer en idé som inhouse vill göra, hur ser då processen ut från idé till att det är satt i drift?**
37.	Ja, det beror ju på scopet av projektet såklart. Vi har ju buggfixar som är lite mer; bara gör det! Sen större, ja , vi har ju.. Försöker komma på något bra exempel som jag kan relatera till. Vi har ju nu t.ex. En stor satsning på sign-up flödet för just PRODUKT, eller gratisvarianten utav det, att streamlina hela den processen så att den är smidig. Då har det ju varit en som har, egentligen den som brinner mest för det, som tar projektledarrollen lite, som får agera main stakeholder på något sätt och har final say. Sen så bygger man ju upp teamet lite utifrån de andra resurserna på vad som behövs. Nu har vi ju folk som jobbar mycket med UX som vi tagit in för ett halvår sedan, innan var det bara utvecklare som satt och gjorde HTML sidor, men nu har vi dem som faktiskt tänker på hela upplevelsen. De är ju kanske inte så DevOps(ia) egentligen, men de är ju med i det här projektet då det bara är kundupplevelsen som är viktig, så de blir inblandade på nån viss procent, kanske 100% i det här fallet då den är så viktig. Sen så försöker man identifiera vilka andra produkter som blir påverkade av att sign-up flödet ska vara smidigt, det ska ju vara snyggt på webben och i iPhone appen och i Android appen och i någonting annat kanske. Så då får man bygga upp ett team där som ska försöka få igenom det. Nu är det han som leder just det projektet han som är chef för avdelningen också, så han får de resurserna.
38.	Key points: P1-25 Större förändringar kräver mer planering P1-26 Personligt Intresse kan avgöra ledarrollen P1-27 Team byggs upp efter de resurser som krävs P1-28 Team kan innehålla personal utanför Development och Operations
39.	**Är det så att från det här signup-flödet, är Ops med då från början?**
40.	Ja (osäker), definiera Ops?
41.	Key points: P1-29 Ordet ops är inte självklart
42.	**Det är svårdefinierat för mig, men jag tänker just hur det ska hostas, hur det ska driftas.. Är de med från början eller kommer de in senare?**
43.	Just i det projektet så har de inte varit inblandade och det är väl då för att de inte riktigt behöver vara det. Signup-flödet projektet är snarare en omstrukturering av, eller är egentligen bara en ny presentation av existerande funktionaliteteer, så den går inte så långt ner och vänder i serverna som man kanske hade tänkt i en hel sådan DevOps kjedja, så

	där kan jag nog säga att de som jobbar mer med Ops grejerna har inte varit med, för att det inte har behövts. Har vi något annat exempel som är lite mer genomgående (tänker) Vi bygger ju stödsystem internt hela tiden för att centralisera viss logik så att det inte är utspritt på massa olika applikationer. Vi har ju t.ex. När vi slår upp vem som äger vissa nummer eller kontaktinformation till vissa telefonnummer så går vi ju ibland mot ENIRO, ibland mot någon annan tredjeparts lösning. Det har varit varje applikation som vi har internt har gjort det individuellt. Nu har vi ett projekt där vi ska centralisera det så att vi bygger en produkt internt som vi frågar om ett nummer, så skickar den själv vidare. Där har vi ju direkt en ny applikation som ska upp som vi vill köra på en egen host. Så där har vi ju ett behov som har identifierats någonstans ifrån, vet inte om det har varit styrt av ekonomiavdelningen som har varit inblandade, då det är en viss kostnad varje gång man slår upp ett nummer. Så ekonomiavdelningen har varit med och tryckt på, de som har haft applikationen som tidigare har gjort uppslag hit och dit har tyckt att det varit en kul teknisk approach. Jag som jobbar med allmän systemarkitektur mestadels tycker det här är ett vettigare lösning att centralisera det där. Så det har ju varit rätt brett redan där och direkt blir ju Ops inblandade där för att vi ska ha en ny host. Så då frågar vi Johan om en ny host, det har vi väll plats till? Då säger han argh! men sen säger han jo det har vi.. För där är ju alltid en buffert är förhoppningen. Så då sätts den upp och då försöker man diskutera hur ska denna användas, behöver den mycket CPU eller är det RAM, ska den ha jättestor hårddisk? Så försöker man hitta den lösningen. Ofta är det utvecklaren som får speca själv och då tvingar man utvecklaren själv att tänka på hur den här applikationen ska fortleva under en lång tid istället för; jag har byggt den här JAR filen, kan du hosta den någonstans? Som man har hört skräckhistorier om från andra ställen. Så läggs så mycket ansvar på utvecklaren som möjligt sen så till att det finns hårdvara att köra den på som matchar behovet. Är det sen någonting som går fel; hosten krashar oavbrutet. Ja det är bara den applikationen som kör på den här hosten, så det är förmodligen applikationens fel. Då får utvecklaren fortsätta äga det och hålla koll på det. Så Ops delen av det hela är egentligen bara att se till att förutsättningarna finns där, för att utvecklaren ska kunna köra sin applikation.
44.	Key Points: P1-30 Utvecklare tar ett helhetsansvar för produkten. P1-31 Det sker en dialog mellan Dev och Ops vilka miljöer som sätts upp
45.	**Okej! Ops har inte den insikten i appen i sig..?**
46.	Ops går inte in och patchar den; liksom att här är en bugg eller den här drar för mycket minne, utan Ops kanske har ett utökat övervakningsansvar och kolla att hosten mår bra; eller när den mår dåligt, då pekar den på utvecklaren; du borde fixa det här. I verkigheten så är det en större jourverksamhet som har koll på det

	mesta och hur sakerna ligger och mår och i den är det bara utvecklare med egentligen. Så vi övervakar ju ops delarna också. Sen så har vi då Ops killen som secondline, ifall det verkligen skiter sig; hosten har gått ned helt och vi måste åka till serverhallen, då är det han som har nycklarna.
47.	Key Points: P1-32 Utvecklaren har jouren för att produkten fungerar, medan Ops fungerar som secondline
48.	**Tänkte på det du sa innan, där vid buggfixar, eller någonting som snabbt ska utföras. Hur automatiserat är en sådan process? Från det att man löser buggen till det att det ska testas och ..**
49.	Continous Delivery där då?
50.	**Ja**
51.	Ja det är semi-automatiserat eller hur man ska säga. Jag har funderat lite på; att ja okej, nu har jag commitat min kod och pushat det och nu ska det bara rullas ut. Vi har ju inte den approachen än och jag är inte helt säker på att vi vill gå mot den heller. I dagsläget är det att man skriver sin ändring, man lägger upp den för review så att man får lite andra ögon på det. Är det en enklare buggfix, säg en felstavning, så skickar man upp den direkt. Medan rör det sig om någon logik så är det alltid bra ifall någon annan tittar på det. Om inte iallfall bara för att sprida kunskapen. Sedan mergas den in till master; vi använder git, och sedan så har vi ett manuellt steg för att uppgradera men det är såpass automatiserat så det är några klick på ett webb gui. Vi har det lastbalanserat mellan ett antal noder, så man deaktiverar en nod, kollar att okej nu är där ingen trafik kvar på den, då uppgraderar vi den. Sen så väntar man tills den är klar.
52.	Key points: P1-33 Kodreview för större ändringar P1-34 Kodreview för att sprida kunskap P1-35 Continuous Delivery används
53.	**Innebär det att man kan köra ut en ändring på en nod utan att de andra påverkas?**
54.	Precis! Så man kan också testa den här då; “oj jag hade inte löst det, vi väntar, rollback på den sålänge”. Så det är ganska nice att kunna göra det. Är man sen då nöjd så aktiverar man den och deaktiverar den gamla och uppgraderar den också. Så det är en ganska trevlig approach! Hade man haft fullt automatiserat så hade det bara skjutits ut och förmodligen byggt någon staging miljö; automatiskt ut till den och så kollar man att det verkar fungera bra, men då ska man ändå in och klicka någonstans för att godkänna det. Hur mycket tid ska vi lägga på att få till den processen när vi typ har det

	redan?
55.	Key points: P1-36 Miljön tillåter snabba rollbacks
56.	**Skulle man kunna fråga egentligen vad ser du som de största utmaningarna när det är såpass spritt med roller och det inte finns någon definition? Blir det konflikter ibland som man annars kanske inte hade sett?**
57.	Ja (osäker), det kan ju alltid bli en konflikt, eller man kan säga att problem skulle kunna eskalera för långt innan man upptäcker dem, om man inte har någon som sitter och tittar på att ex. Nätverksinfrastrukturen är okej, så kanske man inte stöter på problem förän allting slutar fungera samtidigt. För att nätverket är fullt av trafik; shit vi kan inte skicka fler ettor och nollor på den här sladden, vi skulle haft två sladdar ju såklart, eller 4 eller 12. Så det är ju en risk att man inte tänker på sådana grejer om man inte har någon som ansvarar för det. Samtidigt vill vi inte ha någon som räknas sladdar hela dagarna, det är inget utmanande jobb och det är ett problem som är lösbart. Det är också, att inte ha tydligt definierade roller, vi har ju lite grupper indelade. Vissa som sysslar med iPhone appen t.ex. Och vissa som sysslar med telefoni-plattformen. Det kan ju alltid bli konflikter då mellan dem ; Jag tycker att ni ska lösa det här problemet för att det ligger liksom inte riktigt bara hos telefonin och samtidigt så är det iPhone-appen använder ju telefonin. I problem som inte har någon tydlig ägare, det kan bli bråk om vem som ska göra det och de som behöver göra det till sist blir lite upprörda över det att; det här är inte vår grej egentligen. Så sådana grejer kan bli problem.
58.	Keypoints: P1-37 Otydliga roller kan innebära att problem eskalerar för långt P1-38 Finns inga tydligt definierade roller P1-39 Grupper kan utgöras utifrån en spec. Applikation P1-40 Applikationsgrupperingen kan ibland ge upphov till att problem inte går att placera
59.	**Hur tycker ni ert kunskapsutbyte fungerar?**
60.	Ja, det är ju just det att man försöker samla personligheter snarare än detaljkunskaper gör ju att man får en bra gruppdynamik överlag. Folk som trivs med varandra och snackar gott. Då kommer det liksom ut bra idéer och man försöker hjälpa varandra över gruppgränserna. Så det funkar ju, vi fortsätter!
61.	Key points: P1-41 Samlar viss typ av personligheter för att öka kunskapsdelning P1-42 Gruppöverskridande kunskapsdelning och hjälpsamhet
62.	**Måste man inte vara väldigt duktig över stora ansvarsområden som**

	utvecklare?
63.	Jo, det blir ju att olika utvecklare tar olika stor plats i den här stacken. Det blir ju lite att det har med självbalanseringen att göra. Man har de som är drivna och de som är vana vid att ta plats, de bubblar ju upp till någon form av ansvarsposter och då blir det lite såhär; du har första tjing att titta på databasen. Sen är det ju i den personens egenintresse och hur hela organisationens synpunkt väldigt viktigt att sprida den kunskapen, så att inte sitta så; jag sitter hela dagen och avbryter dumma queries till databasen. Då är det dens jobb att gå bort till utvecklarna som jobbar och ställer de här frågorna mot databasen och säga att de inte kan göra såhär utan bör göra såhär istället. Så alla har ju uppdraget att egentligen bygga bort sig själva.
64.	Key points: P1-43 Självbalans av personligheter och kunskaper P1-44 Kunskapsdelning en viktig del
65.	**Det låter som att ni tycker att det fungerar bra?**
66.	När jag kom hit så var jag skeptisk först, tänkte det här kan inte fungera det är ju helt galet, någon kaosmetodik! Men sen har jag ju försökt utvärdera den och hitta på bättre varianter och jag har inte kommit på någonting annat. Så, det fungerar tillräckligt bra för att vi inte ska behöva dra i någon handbroms. Snarare så att de vi anställer som kommer från andra större företag drar en lättnads suck; vad skönt utan den här corporate grejen, “fixa den här buggen? Då behöver du göra en request om det”. Det var någon som jobbat på bank tidigare som är extremmotsatsen. Det planeras i halvår på att göra ändringar som bara tar fem minuter att fixa, medan vi då försöker att planera i fem minuter för ändringar som kommer ske över ett halvår. Sedan styr vi ju om ofta. Om vi bestämmer oss idag att vi ska satsa på det här och sedan börjar vi.. (paus) kommer halvägs och upptäcker att marknaden har svängt! Släng det här då och hoppa på det här andra spåret istället. Genom att vara lite kaosartig ger ju en frihet i att anpassa sig bättre efter omständigheterna. Sen är det ju nackdelar också att man har en tendens som ny utvecklare eller unga utvecklare, att svara på den som skriker högst för tillfället. Så om VD’n springer in och säger: Ah vi måste ha den här featuren för att kunna sälja till den här kunden. Då börjar den på den featuren. Sen kommer någon från supportavdelningen och säger att nu ringer kunder in om den här grejer.. Då blir det det istället liksom. Så det är där man försöker tratta in grejjer med att ha advisers som fångar upp det som verkligen behövs och försöker hålla någon gemensam prioritering. Det är lite sådana utmaningar.
67.	Key points

	P1-45 Initial skeptism, men visat sig fungera bra för P1's organisation P1-46 Prioriterar snabba ändringar framför extensiv planering och fokuserar på att reagera snabbt P1-47 P1-46 Kan orsaka svårigheter i hur prioritering ska ske
68.	**Är det någonting man fastställer på något sätt? Att prioriteten ligger från den *advisern*?**
69.	Ja vi försöker ju sätta individer, det är ju de personer som gillar att ta ansvar, de bubblar upp någonstans. Säg den som har ansvar för iPhone appen, då är det den som har the final say någonstans i prioriteringsordningen; "Vi jobbar på det här, sedan det här och sen det här!" På ett sätt blir det individberoende i och med att det är han som sätter prioriteringen, men samtidigt så är alla involverade i vad som ska göras framöver. Så ifall han skulle bli påkörd av tåget så kan man alltid ta in någon annan som gillar att prioritera för alla har ändå helhetsbilden, det är förhoppningen.
70.	- Prioriteteringen kan bli individberoende? - Undviks genom att alla har samma helhetsbild? P1-48 Undvika individberoende genom att alla har en helhetsbild
71.	**Hur lång erfarenhet har du inom..**
72.	Inom DevOps?
73.	**Dels inom DevOps, utveckling och här på Bolaget?**
74.	Här har jag jobbat i typ 3 år och på den tiden har vi gått från 10 till 46 personer så det är ju en fruktansvärd expansion. Största delen utav jobbet är att hitta skrivbord till alla nya. Innan dess har jag jobbat 2-3 år på olika webbutvecklingsbyråer där DevOps tänket har funnits där också. Kanske inte uttalat "Vi ska satsa på DevOps" utan snarare kommit som en naturlig grej eller en nödvändighet då vi inte varit så många, så då måste vi ha koll på allting. Vi har inte 10 pers som bara sitter och ser till att dessa 2 servrarna fungerar, utan det är bättre att utvecklarna har koll på det också.
75.	- DevOps passar bättre för mindre bolag? - DevOps är inte någonting man bestämmer sig för? - DevOps är någonting som sker undermedvetet? - Undviker att ha för strama arbetsuppgifter? - Bättre att alla har bred uppfattning P1-49 Effektiverare resursanvändning genom breda roller och kunskap P1-50 DevOps kan komma naturligt eller som en nödvändighet vid få anställda
76.	**Kallades det DevOps när du kom hit?**

77.	Nä då hette det ju utvecklingsavdelningen, eller ja development, sedan döptes det om till Product & Development, men det DevOps tänket har ju alltid funnits där ändån. Det är nog ingen som aktivt har tänkt “Nu ska vi uppfylla metodiken DevOps” utan det har ju varit en naturlig del att utvecklarna ska kunna så mycket som möjligt, fullstackiga för att kunna ta klokare beslut som “ifall jag gör den här grejen här längst ute i applikationen, vad får det för påverkningar på den applikationen bakom som jag ligger och pratar med, hur påverkas den dator som vi använder.. Kommer det då bara blåsa minnet eller spränga CPU:erna på den faktiska eller virtuella hosten som kör det här och hur påverkas då den fysiska hosten som hostar den virtuella och de andra syskon virtuella, kommer de påverkas också?”
78.	☐ DevOps är ett tänk? ☐ DevOps är inget aktivt mål att uppnå? ☐ Ifall det finns en bred kunskap kommer man ta bättre beslut? ☐ Förståelse för hårdvara? P1-51 DevOps kan vara ett tankesätt P1-52 Bred kunskap och helhetsperspektiv ska ge upphov till bättre beslut då
79.	**Så det här helhets ..**
80.	Ja det är fullstack tänket som man försöker satsa på. Sen så är det ju ingen som har koll på hela, utan det är att man försöker få"in folk som har ett intresse någonstans, i någon av nivåerna. Sedan vill man sprida så att de kan lite uppåt och lite nedåt, så att vi som grupp täcker in hela grejen. Man blir cross-functional som team. Sen har man ju vissa individer som blir specialister som har en lite smalare bredd, sen har man de som har bredd och översiktskollen på hela men kanske inte detaljkunskapen.
81.	P1-53 Försöker nå ut till personligheter som har intresse för att ha bred kunskap
82.	**En avslutande fråga angående termen DevOps, vad innebär den för dig?**
83.	
84.	Den svåra million-dollar question.. Nämen det innebär ju att man inte separerar. Att man som utvecklare tar ett helhetsansvar, nu pratar jag ju ur utecklarens perspektiv eftersom jag är det själv. Att man ansvarar för den koden man skriver, inte bara under den tiden man skriver den utan även under hela dess livslängd och om den skapar problem för andra så är det ditt problem. Då får du själv ta och lösa det tillsammans med andra och hitta lösningar, istället för att bara knacka ihop någonting och slänga över staketet och sedan får en driftorganisation ta hand om den och sen blir det dem som alla klagar på för att “den här grejen fungerar inte”. Det är helhetsansvaret! Den som

	ska ha feedbacken får den.
85.	P1-54 Separationen mellan roller ska vara så liten som möjligt P1-55 Utvecklare ansvarar för sin produkt konstant P1-56 Feedback når rätt person
86.	**Någonting du vill tillägga?**
87.	Nä jag känner mig helt uttömd på svar!
88.	**Stort tack!**

Appendix C

Intervjutranskript och nyckelpoängskodning av intervju med **P2**.

1.	**Hade du kunnat berätta lite om rollen som du har här?**
2.	Jag började förra sommaren och som alla utvecklare här så börjar man på ett visst system, på affärssystemet, så jag har varit där tills för några veckor sedan, en månad sedan ungefär så nu jobbar jag för X som ni träffade precis i en grupp som har förkortningen EET, jag vet inte om han gick in på det.. Står för Engineering Effectiveness team, går helt enkelt ut på att vi gör alla saker som teamen inte brukar hinna med kan man säga, och skjuter på framtiden. På något sätt försöker vi bygga det bra eller ta dom stora ombyggnadena av arkitekturen, och sådana saker. Det tenderar då bli så att det är jag som kan mycket om mycket så att säga såatteh det trillar in mycket saker utifrån som faller ner på vårat bord.
3.	-Hinner inte med allt, är detta typiskt för DevOps? -Personer som kan mycket om mycket,
4.	**Så det är dig folk går till när dom behöver hjälp?**
5.	Lite så, sen är det klart att det beror på vad det är.. Vi har en tre-fyra system som är rätt stora och distinkta, och är det ett typiskt X eller Y problem så finns det ju folk som jobbar med det heltid och då hamnar det där.. Men är det saker som trillar in på databaser, servrar och sådant hamnar det hos oss förstås.
6.	*P2 håller med och påpekar att finns olika grupper som ser typiska problem dagligen. De som inte hamnar i deras zon går till P2 och dens grupp.* *- Vissa löser typiska problem, vissa löser andra* P2-1 Det existerar olika ansvarsområden
7.	**Skulle du säga att det är något typiskt DevOps att ha ett sådant team?**
8.	Njeh. Njae det vet jag inte. Nej det tycker jag nog inte. Det är nog det första jobbet av dom här fem (som intervjupers. tidigare haft, egen anm.) fem som har ett sådant team för det.. Typisk sådan sak. Här är det skönt för här sitter beställarna inhouse.. Det är rätt mycket prioriteringar som vi kan göra sjävla på något sätt.. Vi tillåts tycka till vad som är en vettig prioritet. Till skillnad från ställen där man lyder under väldigt starka budget-krav och det är någon annan stans som tycker att “kan vi inte göra den snabba grejen nu och göra den långsiktiga lösningen senare..” Jag vet inte

	hur många gånger jag dumt nog tackat ja till sådana saker. Om man sitter i en DevOps kultur kanske dom tycker att “det här är fel och dom måste fixa det” med flåsande över axlar och det tas mycket genvägar. Så det är nog en väldigt bra sak att ha om man har en tydlig DevOps struktur (avdelningen dvs, egen anm.)
9.	*P2 kan inte direkt säga ifall det är typiskt DevOps med att ha ett catch-everything team. Vi reflekterar att frågan kanke upplevs som främmande och svår besvarad. P2 har inte sett ett sådant team tidigare.* *Fortsätter med att förklara närheten till beställare är en bra sak, då prioriteringar kan själv göras. Vidare berättar han om en öppen miljö där personer tillåts utala sig om vilka prioriteringar som de anser viktigast, tillskillnad från andra ställen (osäkerhet ifall P2 menar andra DevOps arbetsplatser eller icke DevOps arbetsplatser).* *P2 nämner DevOps kultur och att man kan befinna sig i en sådan. Antyder sedan att en sådan kultur kan variera i sin struktur, där P2 anser att en tydlig struktur kan vara fördelaktig.* - DevOps kultur kan bli stressfullt iom. lösa strukturer? - DevOps kan ha varierande strukturer? - Svårt att se vad som går innan för ramen av DevOps? - Ifall beställare är nära betyder det egen prioritering? - Öppen miljö där personer tillåts tycka till om prioriteringar är DevOps? - Antydan om att arbetsplatsen är unik P1-2 DevOps kan ha olika strukturer P1-3 Svårt att se vad som ingår i DevOps P1-4 Öppen miljö, alla får tycka till om prioriteringar
10.	**Skulle du säga att det är en tydlig DevOps struktur här?**
11.	Ja, det tycker jag, för att det är så himla.. Ni kanske såg att alla ovanför läktaren här är support av olika slag, både telefon och mail. Dom gör vad kunden begär och så fort det tricklar ner något dom inte kan lösa så har vi en hel avdelning med utvecklare nedanför. Så ja, det tycker jag .
12.	*Även då P2 kan svara direkt att det är en tydlig DevOps kultur, så blir det senare svårt att sätta ord på. Fortsätter med att förklara närheten mellan supporten och utvecklarna och låter det uttrycka den tydliga strukturen.* *P2-5 Support kan direkt vidarebefordra önskemål från kund till utvecklare*
13.	**Har du jobbat i en sådan struktur tidigare innan du kom hit?**
14.	Jaa, det har jag gjort. När jag satt och gjorde en app till en kund, en mobil

	hemsida och en app, då fanns det lite strukturer i när det kom supportärenden. Deras kunder ringde in till dom, och sedan kom ärendet till vår operations-avdelning.
15.	*P2 har varit i liknande strukturer tidigare då han arbetade med mobila applikationer och hemsidor. Vidare antar vi att P2 menar att det fanns struktur i hur ärenden transerverade till rätt avdelning. Här misstänker vi att vi har talat om olika strukturer*
16.	**Har det alltid varit så på de tidigare 5 jobben du haft?**
17.	Nae, inte alla… Vissa ställen har varit mer ad hoc, och vissa andra ställen har det hållits mer personliga. Speciellt om det är större organisationer så är det tätare skott. Ofta blir det lite längre ledtider från att man rapporterar problemet till att det hittar ända ner till utvecklare. Jag har jobbat på en bank senast där det av förståerliga skäl är en ganska lång bit däremellan.. Mellan support/operations och utvecklare.
18.	*P2 antyder att ledtiden mellan utvecklare och driftavdelning kan påverkas av organisationens storlek och i vilken bransch den befinner sig i.*
19.	**Är ni snabba här tycker du?**
20.	Ja, det är vi. Vi måste vara det, vår business är att sälja telefoni till många många företag och få vara lösningar till det. Om det skulle börja strula någonting så skulle det göra det för väldigt mycket folk samtidigt. Vi har några hundra tusen abonnenter så, om telefonin ligger nere kan det börja kosta pengar.. Både att folk inte kan ringa eller att de blir sura och kan lämna oss. Så det måste vi vara (snabba, egen anm.)
21.	*P2 anser att sin organisation är snabb och att de måste vara det för att inte löpa risken av ekonomiska förluster.* *P2-6 Arbetet är effektivt.*
22.	**Från att gå från den här tighta strukturen till den lösa.. Jag antar att du ser positiva saker, men ser du någonting svårt i det?**
23.	Det beror ju på vad man är för utvecklare tror jag också. Är man en sådan som gillar att problemlösa och grejer så tror jag att man kan gilla en sådan här miljö. Är man en sådan som gillar lugn och ro och att få jobba med en sak.. Och när det kommer in en rapport om en bugg så får man göra det i lugn och ro.. Om man föredrar det och inte hela tiden får någon som kommer och petar en på axeln fyra gånger om dagen.. Det har funderats många gånger här borta på.. Men ska man ha en person som Operations-sidan går igenom för att kontakta ett team eller ska dom gå till allihopa.. Och det är inte alldeles glasklart heller. Jag kan gilla att det är väldigt transparent här, men samtidigt så.. Många gånger har man lite arbetsro man vill ha och så.. Känner att.. Den här veckan försvann förbi och jag

	hann inte alls göra den egentliga uppgiften som jag jobbar på, så är det lätt att glömma av att man kanske fixat 14 problem istället på den tiden.
24.	*P2 ger sken av att en DevOps miljö passar en viss sorts person; en som gillar att problemlösa, medan de som prioriterar att få jobba med en sak i taget och lugn och ro kan se mer negativt på det.* *P2's företag har funderat över frågan hur de ska hantera hur ärenden delas ut och att det inte är en självklar fråga.* *Transparens är en bra sak enligt P2 men att det lätt kan bli att den inte hinner med det som var tänkt från början.* *Vi märker att frågan utvecklats till ett ev. känsligt område där P2 tänker över hur han ska formulera. Vi ser en potentiell risk att omgivningen påverkar.* *P2-7 Många spontana arbetsuppgifter som stör de ordinarie arbetsuppgifterna*
25.	**Hur ser organisationen på det du har gjort då om du fixat 14 grejer.. Finns det något krav på att bevisa det i tidsrapporter t.ex.?**
26.	Nae, det har vi inte, och det är väldigt skönt. På banken där jag jobbade tidigare skulle varenda minut man lagt på något räknas, för det fanns i budget att vi skulle kunna debitera det. Det sköna här är att vi är utvecklarna och vi gör det vi bedömer är viktigast. Är det beställt tre saker är det någon som säger vilken av de beställda sakerna som är viktigast. Om vi känner att det här systemet ligger nere så får vi absolut ta och lösa problemen med någon sorts balans i att om man gör det långsiktiga jobbet så man slipper göra 14 saker i veckan och lösa… är ju alltid den balansen man får ta men det är ju snarare balansen mellan nya features och att förbättra systemet.. Sen kommer det alltid saker utifrån.. Minns inte ens vad frågan var haha.
27.	*De behöver inte tidsrapportera vad de gjort under en dag i P2's organisation, vilket han uppskattar. På sitt tidigare arbete med stramare strukturer var de tvugna att rapportera allt.* *Vidare så nämner P2 igen att det är upp till utvecklarna att prioritera vad osm är viktigast. Nämner även att det är en ständig balans mellan långsiktiga och kortsiktiga mål.* *Avslutar med att påpeka att det alltid finns utomstående faktorer som kan påverka.* ☐ Utvecklare bestämmer agendan? ☐ Mindre krav på rapportering inom DevOps? P2-8: Inga krav på tidsrapportering P2-9: Balans mellan kortsiktiga och långsiktiga mål.

28.	**Hmmm. I ert arbete.. Jag förstår från X att ni använder continuous delivery (hänvisning till intervju med P1) till viss del.. Hur tycker du att det fungerar?**
29.	Framförallt är det väldigt bekvämt att ha det väl på plats, att det är så få tryck som möjligt. På det största affärssystemet så ingår det att om man jobbat här en vecka så ska man ha sett på när någon gör en release, och att man kan göra det själv. Det skall liksom vara så få människor som möjligt som skall vara "den viktiga personen".. När den personen har semester eller har gått för dagen ska inte det hindra arbetet. Och det gör ju att.. Klart det är ju en väldigt smidig sak.. Framför allt om operations kommer och säger att "vi kan inte logga in här" "åh herregud vi har gjort något dumt här" och så fixar man det. Om man nu skall leta efter en nackdel så är det väl att när man väl inser att "titta vad lätt det är att göra det här" så.. Nu säger jag inte att det är så här, men det är alltid en risk att det utnjyttjas.. Jag skulle ändå säga att det överväger ju definitivt att ha en väloljad continuous delivery mekanism.. För det är så.. Det blir.. Om man är på ett annat ställe och det är lite nervöst över att man släppt något och det kommer en bugg så blir man lite "gun-shy" och skjuter upp nästa release så att den blir jättestor och det kommer massa nya features… Så kommer det ju vara något.. Då är det bättre att ta en liten bit, och om man ställer till något, vilket man förhoppningsvis inte gör, så är det liksom lika snabbt att slå tillbaka. Och det är ju.. Hellre liksom vara vågad på det viset, sen får man så klart se till att det .. Tar testerna skitlång tid att köra och man försöker få bra continuous deliveries.. Om det tar timmar och köra det och man lyckas köra 3-4 releaser per dag så får man inse att testerna inte komma hinna vara körda innan det kommer ut.. Nu har vi lite andra mekanismer, vi kör obligatorisk code review på alla ändringar, så att två andra utvecklingar skall ha ögnat över alla ändringar.. Ingenting är ju fool-proof men…
30.	*P2 uppger att det är en snabb och enkel process för deployment.* *P2 uppger även att de undviker att ha personer sittande på nyckelkompetens för processerna.* *Vidare ser P2 att en väl fungerade CD mekanism gör utvecklarna mer självsäkra i att våga släppa egna releases.* *Code reviews uppges kunna minska antalet fel i koden som släpps.* *P2-10 Det går snabbt att deploya sin kod* *P2-11 Det är enkelt att deploya sin kod* *P2-12 Kompetens sprids och binds ej i nyckelroller* *P2-13a Code reviews används för att minska antalet fel och dela kunskap mellan utvecklare.* *P2-13b Utvecklare blir motiverade till att frekvent släppa sin kod*
31.	**[12:40]Hur ser er organisation på att det blir fel? Är detta något ni uppmuntrar eller?**

32.	Ja man uppmuntrar ju inte fel haha.. Ja men… Jag skulle säga att det finns lite olika bilder av det och det går väl lite upp och ner.. Har det varit saker, och det behöver ju inte vara att det trillar ut precis i samband med realease utan kan vara något som smyger sig fram och dyker upp, så.. När jag pekar upp dit (operations, egen anm.) så är det lite för att de är våra närmsta kontaktpersoner från operations.. Om det blir att det är, framför allt stora driftstörningar, så har vi möten efteråt, “Reason For Outage”, RFO, och debriefar “vad kom det här ifrån?” och “vad gör vi för att det inte skall ske igen?”. Ja. Det kan ju vara kritiskt om man precis plockat in en ny kund som är värd jättemycket och så första dagen, eller nånting händer första veckan. “Oj, nu kan vi inte ringa här..” Ja då är det klart att man jobbar i uppförsbacke med den kunden.. Kanske flera år efter det bara för att man fått en dålig start. Så vi vill ju att… Det var någon gång, det har säkert ni hört någon gång också.. Ska man jobba skitlänge och ha en perfekt produkt eller släppa något medans det fortfarande är relevant för marknaden? Det är någon slags balans man måste hitta.
33.	**Det här balansen, blir den tydlig eller är det svårt att se den?**
34.	Jaa, den kan nog vara lite klurig kanske. Vi försöker ju hålla ribban, vilka tester är tillräckligt små för att vi kan sätta på unit test nivå och få dom att gå igenom, eller försöka ribba och vara lite liksom hård men rättvis på reviews att , “det funkar men är inte snyggt och kommer inte hålla i längden.”eller vad det nu är. Den här avdelningen som vi nu har X och jag.. Vi försöker ändå att inte låta de här surdegarna stå och jäsa någonstans utan fixa det direkt. “Don’t clean broken windows.” Sådana saker skulle jag säga är den farliga biten. Om man har det lite stökigt och så så är det lite lätt att man: “den här grejen borde ha varit omskriven för många år sedan, men nu är den som den är och pillar man i ett litet hörn på det och löser sitt case. Så har man sabbat nån helt annan grej för den var ihoptrasslad upp och ner.. Hopplöst att veta.. Då är det svårt att veta. Det man tycker är en harmlös liten ändring kan bli en skitstor förändring någon annanstans. Men det är absolut inte en tydlig balans. Det är faran i att ha det så men att vi ändå.. Nu gör vi ändå ett rätt tydligt initiativ för att sakerna inte skall vara så i längden. De kommer fixas. Långt svar haha.
35.	**Om du har utvecklat någonting och fått det reviewat av dina två code reviewers, vad gör du då sen för att få det produktionssatt, vilka pratar du med då eller..?**
36.	Ja, alltså om vi tar vårt det här affärssystemet för jag tror att det är det tydligaste caset. I GitHub gör man själva mergen.. Och det gör att vårat byggsystem som använder Jenkins.. Ja det är Jenkins det heter nu Hudson hette det förr.. Vi har ett byggsystem som heter Jenkins.. Det kommer bygga sjäva systemet och trigga byggen av alla andra grejer, så gör man något som är långt ner kommer det att bli världens grej. Det leder sedan till att prylen är redo att släppas för release om testerna går igenom.. Sedan så lyfter man det lite olika beroende på hur farlig sak det

	är. Om det bara är en pytteliten sak pratar man bara kanske inom teamet när man bestämt sig så säger man till alla utvecklare att nu är det på väg att komma något här. Är det någonting som är lite större så går det till DevOps-chatten.. För vi har en chatt för alla som är involverade i DevOps. Sedan så sju-åtta klick senare så är systemet uppe i drift.
37.	*P2 uppger att de använder GitHub för Source Control Management och Jenkins för att automatisera sina byggen i deras CI/CD.* *Releases kan göras direkt av utvecklarna utan att gå via operations om de inte är omfattande.* *Omfattande releases diskuteras först i en DevOps chatt.* *Kod kan släppas snabbt med ett antal knapptryck.* *P2-14 CI/CD sker genom SCM & byggverktyg som Git och Jenkins.* *P2-15 Små releases görs enkelt inom Dev-teamet.* *P2-16 Större releases diskuteras med Operations genom DevOps chatt.*
38.	**Så du kan egentligen deploya allting helt själv om du vill?**
39.	Ja, det är ingenting som krävs utomstående påverkan. Sedan finns det ju någon sorts riktlinje på när det är lämpligt att göra en release. Kanske inte för sent på eftermiddagen, är det fredag så kanske man funderar på att lägga det lite tidigare.. Framför allt i ljuset av “hur sent kommer jag tvinga en person att vara kvar på en tid som dom inte vill vara kvar”. Finns det risker över helgen? Närmar det sig fakturering? Så är det typiskt en sådan sak.. Sista i månaden kanske man inte lägger upp något för att det är riskabelt.
40.	**Jag tänkte på dom här testerna, iom att det är CD.. De här testerna man behöver skriva för det. Är det svårt att skriva ett test som man kan lita på såpass mycket att det går ändra fram dit?**
41.	Jae Ja eller.. Det är lite lurigt och det beror på vad man.. Ett integrationstest på webb-nivå, så att man tänker att man går inte här på affärssystemet att man kan klicka på den och då leder det till det och sådär.. Det är rätt omständigt och rätt pyssligt. Om man istället fokuserar på den logikintensiva delen alltså, nere och rotar i databasen eller så så har dom gjort det ganska smidigt där. Så länge folk bara tänker på att göra det. Det beror på vad man har fått med sig från tidiga arbetsplatser eller skolan. Vi har relativt mycket folk som kommer direkt från skolan här. Då kanske man tänker såhär: “En uppgift den skall lösas och efter det kan man göra såhär och såhär och såhär..” Det är ju skitbra.. Men vad är det som säger att den här grejen kommer fungera imorgon? Ingenting för där är inget… Men det ingår lite i review-processen. “Det vore trevligt med ett test här.” Och så ett smiley-face. Inget agg i den alls utan det skall vara väldigt god ton och mycket encouragement!

42.	**Ni var 45 personer här idag eller?**
43.	På utveckling ja.
44.	**Hur tror du att det skulle påverka att ha väldigt många fler och ha DevOps på det sättet när mansstyrkan är större?**
45.	Nja alltså. Just DevOps delen av det tror jag inte behöver vara vansinnigt mycket svårare i och med att det ändå är en relativt utvald grupp… Låt oss säga att idag i den här DevOps chatten är det 15-20 från utvecklingssidan och jag tror inte att den mängden behöver förändras om man exempelvis dubblar antalet utvecklare så tror jag inte att det är dubbelt så mågna representanter från utveckling. Så jag tror man har nog andra utmaningar som är större om man ser till att växa dubbelt. Sen är det klart att det alltid där mycket bitar när man växer, när det är fler personer kan det bli fler merge-konflikter, svårare att hålla koll på vad alla gör och det man tidigare trodde var en rätt så harmlös grej kan ställa till med ännu mer saker, framför allt om man gör saker exakt samma samtidigt. Man kan inte hålla koll på att när man mergar tillbaka.. Även om man inte hållit på med samma kod så är det något som påverkar varandra.
46.	*P2 ser inte problem relaterade till DevOps vid växande antal utvecklare. Mjukvaruutvecklingen i sig kan drabbas vid växande antal utvecklare då det blir fler konflikter i SCM.* *P2-17 DevOps fungerar väl trots växande utvecklingsteam.*
47.	**Är det sådant ni stöter på idag redan?**
48.	Jo, det gör man alltid i viss mån och det.. Jag tror.. Nu har jag ju inte varit här i ens ett år men det hörs lite på snacket på dom som varit här .. har man varit här ett par år är man vetaran på det här stället.. Man hör ibland att dom pratar om att dom vet precis vem som har skrivit vilken grej “det gjorde han och han den där gången”. Absolut inte koll på sådana nivåer idag. Förr hade dom ett måndagsmöte där alla personer på utvecklingssidan berättade vad dom gjorde, man gick runt men det kan man inte göra idag utan då får man ha ett teams representant som summerar vad teamet gör. Målet är att gå från 40 till 80 här på utvecklingssidan i år, så det är ju en stor förändring/utökning.. Jag tror inte dom var mycket mer än 20 pers i början på förra året så det är en snabb ökning.
49.	*P2 uppger att det är svårt att veta vad kollegorna sysslar med när det är många arbetskamrater.*
50.	**Jag förstår det, så ni satsar på nu 80 under året?**

51.	Ja, det är målet. Det är svårt att hitta bra folk för oss, så vi får se hur det går.
52.	**Jag tänkte höra där i och med att man sade att man gärna vill ha en viss sorts folk att passa in i DevOps. Är det extra svårt att hitta dom personerna eller är det som att leta kompetens?**
53.	Det är nog definitivt en extra bit alltså. Klassiskt för mig inom mjukvara är att vi som samlad klump kanske inte är den mest hypersociala av folk… Så då vill man alltid hitta folk som funkar i grupp. Vissa ställen kan man ha en person som istället bara är riktigt vansinnigt duktig men går inte att jobba med riktigt. Ger man den personen en egen uppgift så funkar det kanske skitbra men att få det samarbetet att fungera i en DevOps-struktur.. Det blir liksom ännu mera man behöver göra ett avkall.. Man märker också nu när vi håller på att expendera internationellt att då blir det liksom en DevOps-struktur där.. Nu har vi ju den danska supporten sittandes här men nu är det frågan hur länge det blir.. Vi har ju Norge också just nu och sen tre länder till vi pratar om. Frågan är ju då när man börjar distribuera en sån här struktur också.. Kan bli lite lurigt.
54.	*P2 uppger att det är svårare att hitta bra personal till ett DevOps team.* *Det ställs enligt P2 högre krav på social kompetens hos utvecklare.* *Att distribuera en DevOps struktur över teams internationellt kan ses som problematiskt.* *P2-18 Det är svårt att hitta bra personal till ett DevOps team.* *P2-19 DevOps ställer höga krav på social kompetens hos utvecklare.* *P2-20 Att distribuera DevOps till internationella teams kan ses problematiskt.*
55.	**Spännande**.
56.	Ja absolut.
57.	**[25 ish]Har ni fler kontor än det här nu?**
58.	Inte utvecklingskontor. Vi har en utvecklare som sitter i Stockholm. Det är en av grundarna så för att han skulle få stanna kvar i bolaget så fick vi gå med på att han skulle sitta där. *fniss* Annars har vi massa säljkontor, men det kan vara två personer som hyr ett skrivbord någonstans. Sen har vi lite större i Köpenhamn, Olso och Stockholm. Jag tror inte riktigt att vi siktar på att få fler utvecklingskontor utan nya länder integrerar vi härifrån. Som det ser ut nu. Malmö är ju jämfört med Stockholm en lite lättare stad att hitta folk och hitta bostad till folk. I Stockholm är det svårt att locka in någon utifrån för det är inte alltid den personen hittar någonstans att bo. Det är lite lurigt med en sådan situation.

59.	*P2 uppger att säljarna ej är lika geografiskt bundna som resten av organisationen.*
60.	**Vill du utveckla det med att ni har supporten till Danmark här… Är det några svårigheter med det vad du tror?**
61.	Alltså.. I och med att support som sådant väldigt sällan har en face-to-face relation med själva kunden… För oss är det ju en fördel att vi inte också behöver kommunicera med chatt och telefon med en person som inte talar samma språk. Det kan vara mycket begrepp och så. Dels att det finns svenska fackbegrepp och så finns det danska sådana.. Och det danska språket i största allmänhet och så. På det sättet. Om dom har kontakt med den danska säljkåren som inte är här utan på 2-3 orter i danmark så kanske dom tappar operations vs utåt. Vad lovar deras säljare eller dom som sitter och gör saker.. Och har den faktiska kontakten gentemot kunden. Det kan vara det luriga för dom tror jag. För oss funkar det ganska bra. Det är skönt att ha dom här uppe. Då kan man öva på sin danska i lugn och ro, haha.
62.	*Problematik med språkbarriärer mellan kund och utvecklare elimineras med hjälp av supporten, enligt P2.*
63.	**Ett förtydligande: Operations för dig, är det dom som sitter och pratar med kund eller?**
64.	Alltså, bland annat. Nu har ju vi en operations-del som sitter mer integrerat med oss som kör drift-biten. Den är nästan.. Det faller nästan under min grupps ansvar också. Om man tänker den ops-biten, om man jobbar på affärssytemet med servrar och drift och liknande så var det nästan en person som satt dikt-an utvecklingsavdelningen som man pratade med, och det blev mer tekniker till tekniker. Den andra operations-biten är ju dom som både pratar med kunden men kanske om vi har massa system som talar mot telia för mobiltelefonin och så där plötsligt går ner.. Eller folk som har hand om simkort och liknande att dom har sina system som dom pratar med och så. Inte BARA telefon-supporten utan en blandning grejer. Vad är er bild av operations?
65.	*Operations sitter integrerat med development hos P2.* *P2 uppger att utvecklingsavdelningen utför delar av driften.* *Hos P2 har varje team oftast en person som sköter en stor del av kontakten med drift.* *P2-21 Operations och Dev jobbar på samma kontor.* *P2-22 Utvecklingsavdelningen utför delar av driftsättningen av mjukvara.* *P2-23 Det finns ofta en person i varje team som sköter den huvudsakliga kontakten med driften.*
66.	**Den är inte helt tydlig ännu. Det är väl mer åt det tekniska drift-hållet. Som ser till att koden körs på maskinerna och blir patchad. Databasadministratörer osv.**

67.	På det sättet är ju vi kanske lite från att ha varit litet men även att vi fortfarande är ganska små.. Har vi ju det liksom inhouse utvecklingsavdelningen i mångt och mycket. Så att dom operations-bitar som vi inte vet själva är knasigt, för det har… vi har en liten jour-organisation som prenumerar på alla sms från systemen och blir väckta om nätterna och sådana saker.. Om det är slut på disk, vi kunde inte nå den tjänsten eller så. Det är ju en del bland oss. Så finns det då två personer. Vi försöker återanställa så att det blir två personer.. Som har själva servrarna.. Dom har mer slut på disk problem och nätverksinfrastruktur. Sånt gör inte utvecklarna.
68.	*P2 uppger att bolagets storlek lett till att en stor del av driften sköts inhouse av utvecklingsavdelningen.* *Vidare hävdar P2 att jour-ansvar för drift är fördelat hos personer på utvecklingsavdelningen.* *Endast en till två personer har enligt P2 huvudansvar för driften.* *Problem kopplade till hårdvara och nätverk sköts av driftansvariga enligt P2.* *P2-24 Utvecklarna sköter själva många av de uppgifter som typiskt utförs av drift, såsom produktionssättning.* *P2-24a skiljer på driftproblem. Utvecklare löser en viss sorts problem, medan Operations tar mer övergripande infrastruktur problem.* *P2-25 Jour-ansvar för drift läggs på (vissa) utvecklare.* *P2-26 Driftavdelningen har ansvar för problem kopplade till hårdvara och nätverk.*
69.	**Ni jobbar extremt nära?**
70.	Ja det gör vi. Dom delar liksom space med oss. Och det är vi som håller koll på om vi slår i max-tak på antal rader i en databas. Det har aldrig skett förut men det kunde tydligen hända. Det har aldrig hänt mig innan. Då är det liksom vi som sitter och klurar hur sjutton löser vi det.
71.	*Drift och utveckling arbetar jämte varandra enligt P2's utsago.* *Utvecklarna ansvarar själva för att åtgärda defekt kod om de själv har skrivit den.* P2-27 Utveckling och drift arbetar jämte varandra. P2-28 Utvecklare ansvarar för sin kod hela vägen tillochmed drift.
72.	**Hur ser utbytet ut annars, jag tänker kunskapsutbyte.. Mellan Dev och Operations Operations?**
73.	När du säger operations operations menar du er eller min definition av operations?
74.	*P2 finner det problematiskt att definiera vad operations är för något.*

75.	**Din definition.**
76.	Ja, alltså. Vi försöker hålla varandra underrättade om det är grejer. Det kan vara dom som håller kontakten med externa systemleverantörer.. Att dom får reda på att något är på väg att gå ner eller planerat driftstopp, var beredda för nu ska det och det ske då och då.. Det försöker vi göra, det är dom här debriefingarna, RFO, som vi har när vi haft en driftstörning. Annars är vi nog lite dåliga på att… vi har pratat om att försöka få upp förståelsen för varandras situation och det är fortfarande lite sådär.. Nu på torsdag ska vi ha ett möte för hela utvecklingsavdelningen och den diskussionen har startat först idag att “förresten det kommer inte vara en enda utvecklare på kontoret på torsdag och så..” så blev det en diskussion med det att man ska vara beredd att .. liknande situation när vi åker på kick-off hela gänget. Vi var i miami i 5 dagar i vintras.. Det är också sådär att om alla utvecklare sitter på samma plan och inte har tillgång till wifi under åtta timmar så..hur löser man det då? Det gäller att tänka på. Nu var jag den enda utvecklaren som inte kunde åka så jag fick lära mig ha jour istället. Men det är ju mer någon sorts planeringsutbyte än ett generellt kunskapsutbyte. Vi har märkt nu när den ena på tekniska operations, driften, sade upp sig, att det plötsligt.. Från att det var känsligt till väldigt känsligt nu när vi bara har en person.. Blir han sjuk eller vabbar eller nånting.. Hur löser vi det? Det kan hända saker plötsligt. Det har grävts av fiberkablar mellan oss och danska serverhallen. Och sådant där. Det ställer ju till det sån skit, för oss. Det kan ju vara rena grekiskan för många där.
77.	*Kommunikation mellan support och utvecklare är viktigt enligt P2. Support håller kontakten med externa systemleverantörer och kan meddela utvecklarna om planerade driftstopp.* *Det kan vara svårt för utvecklare att sätta sig in i supportens situation, och vice versa enligt P2. Det kan de bli bättre på.* *P2 uppger även att det kan vara farligt att ha specifik kompetens knuten till ett fåtal personer.* *P-29 Kommunikation viktigt mellan support och dev.* *P-30 Det kan vara svårt för dev och support att få förståelse för varandras situationer.* *P-31 Viktigt att inte knyta kompetens till specifika personer*
78.	**Om vi på något sätt skulle återgå till definitionen, hur man definierar DevOps.. Vad innebär termen för dig?**
79.	Ja, alltså, för mig och framför allt här så sammanfattas den nog nästan i den här.. Vad ska man säga.. Gränssnittet, istället för att ha en väldigt stark gräns att man suddar ut det mellan dom som försöker se till att grejerna är igång och dom utvecklare som skall stå till svars varför det inte är det … att om man nu har en cirkel med alla utvecklare här och en cirkel med alla dom som håller på med operations där, att dom överlappar

	varann litegrann. Den överlappningen där, det är DevOps för mig.
80.	*DevOps för P2 är att gränserna mellan Utvecklare och Drift suddas ut. Detta kan visualiseras som ett Venn-diagram där den gemensamma mängden mellan Utvecklare och Drift är DevOps.* *P-32 DevOps är att sudda ut gränserna mellan Utveckling och Drift.*
81.	**Kanon. Får rita den o ha med i vår rapport nu.**
82.	Ja precis, ett venn-diagram sådär. Det är ju inte så att man tillhör operations helt plötsligt liksom. Men man tillhör trots allt DevOps, det är någon slags subset av…
83.	**Tror du att DevOps kan betyda något annorlunda för någon annan?**
84.	Jag kan tänka mig att DevOps på många ställen blir en formell roll eller representant som man ringer, och den kanske har någon sorts formellt flöde att får man en buggrapport så går den vidare.. Då blir det lite längre ledtider helt enkel. Här är det lite, framförallt eftersom vi sitter så smidigt och tillgängligt… om det är riktigt KRISKRIS behöver dom knappt skriva något på chatten utan kan springa bort o säga att “nu fan är det bråttom här!” Det kan vara massa grejer sådär, kanske något jtäteviktigt inspelat samtal som skulle kunna råka försvinna, “hur sjutton får vi tag i det här innan det är borta permanent?”. På andra ställen känns det mer som att det är “om det är ett sådant problem vem hör vi av oss till?”. Konstiga strikta vägar.. Alternativt att det är väldigt odefinierat, väldigt löst. Det kommer någon… Någon sitter på IT-driften nånstans och “jobbar inte du i det teamet?” så kommer en helt lösryckt fråga över en kaffe nästan i lunchrummet. Så det kan vara superdefinierat och superflummigt och här är det… Jag vet inte, det är den där chatten som förändrar allting sådär. Det är bra att ha något, iom att vi är ett såpass litet företag och så där.. På vissa ställer är det kanske så att.. På ett ställe jobbade vi i 8 olika projekt så det fanns ju liksom 8 olika DevOps-grupper, en för varje projekt. Det var egentligen bara en person i varje projekt som IT-avdelningen kände till/talat med tidigare. Det blev liksom någon sorts lösryckt DevOps-struktur.
85.	*P2 har på andra ställen sett DevOps som en formell roll eller representant som fungerar som kontaktperson. Detta kan ge längre ledtider. Detta ser P2 som en väldigt lösryckt DevOps-struktur.* *Hos P2 sitter support, dev och ops nära varandra.* *Vid brådskande ärenden kan kommunikation ske omedelbart och face-to-face.* *Spontan kommunikation över en kaffe kan ge nya insikter.* *P2-33 DevOps är ingen formell roll eller representant.* *P2-34 Support Dev och Ops sitter nära varandra rent fysiskt.* *P2-35 Vid brådskande ärenden finns möjlighet att direkt kommunicera*

	face-to-face *P2-36 Spontan kommunikation över en kaffe kan ge nya insikter.* *P2-36a Chat-Verktyg underlättar kommunikationen*
86.	**Känner du att man delar samma uppfattning om vad DevOps skall innebär här?**
87.	Njae, jag tror det är lite sädär vagt. Jag tror att för många utvecklare så gör man sitt system här.. Och så har man olika uppfattningar om hur allvarligt det är att något är nere. Att operations-delen kanske tycker det är väldigt viktigt att vi ställer och diskuterar kring pingisbordet varför och hur ska vi se till att det inte sker igen. En del kanske tycker att det är.. Nu hände någonting men nu är det ju igång igen, att man kanske inte är helt.. Menar.. Det kan vara samma sak inom utvecklingsdelen, att vissa utvecklare tar det på större allvar att något är lite fel, att man känner en större urgency. Man märker förstås att det är vissa personer som känner sig mer delaktiga i DevOps - delen och är mer snabba på att svara på delen där, eller tänker på att förvarna vissa saker, att “nu kan det bli problem här, eller får dom något problem så säg till dom att starta om telefonen så kommer det att funka igen.” Så att folk vet om det INNAN folk ringer in o slipper svara att de får återkomma efter att ha pratat med tekniker.. Så kan man säga att vi har gjort det här, och kanske tillochmed skicka ut ett mail till kunden. Hur pass proaktiv man är, det är nog något som skiljer sig rätt markant.
88.	*P2 tror att en homogen bild av vad DevOps är inom organisationen saknas. Men ger inte sken av att det är ett större problem* *P2 ser att utvecklarnas uppfattning om hur allvarligt ett driftstopp kan vara skiftar.* *Det finns i organisationen olika grader av proaktivitet vid releaser.* *P2-37 Det saknas en homogen uppfattning av vad DevOps är.* *P2-38 Det är viktigt att förebygga problem genom att förvarna varandra (Dev <-> Ops)*
89.	**Har du något du vill tillägga?**
90.	Det kan jag inte påstå. Jag har halsbränna så mycket som jag pratat.
91.	**Sorry**
92.	Ingen fara

93.	**Stort tack!**

Appendix D

Intervjuguide inför intervju med **P3**.

Culture	☐ Hur sker Kommunikation? ☐ Hur ser man på ansvar och roller? ☐ Hur ser man på samarbete mellan grupperna? ☐ Hur ser man på Involvering mellan grupperna? ☐ Vad värderas? ☐ Hur tydligt utformade är riktlinjerna för DevOps? ☐ Ser du att DevOps lämpar sig bättre för en viss typ av personlighet? ☐ Hur tycker du att organisationensstorlek påverkar? ☐ Behövs bred kunskap hos individen? ☐ Hur ser tilliten ut mellan avdelningar? ☐ Hur prioriterar man?
Automation	☐ Hur automatiserat är arbetet?
Measurement	☐ Hur ser man på och hanterar feedback? ☐ Hur utvärderar man arbetet?
Sharing	☐ Hur ser man på kunskapsdelning? ☐ Hur sker delning av verktyg? ☐ Hur prioriteras kunskapsspridning ?

Appendix E

Intervjutranskript och nyckelpoängskodning av intervju med **P3**.

1	Som jag tror att det kom fram i mailet också är att jag har inte jobbat så mycket direkt med devops, däremot så har vi jobbat nära och försökt införa Continuous Delivery och sånt. Där är DevOps rätt knutet, i alla fall om man ska vara bra på det.
2	*P3 nämner att han inte arbetar direkt med DevOps utan jobbar nära DevOps genom att införa CD. CD som är nära knutet (assoicerat) ifall man ska vara bra på det?* Key Points: P3-1 DevOps är någonting man kan vara bra eller dålig på P3-2 CD är anknutet till DevOPs, P3-3 CD behövs för att man ska vara *bra* på DevOps
3	**När ni jobbat nära, är det att ni kommer in i några som uttalat ”Vi tänker köra devops”?**
4	Ja, just nu sitter jag ju på (*företag)* och där har de ju gått en DevOps-kurs och försökt liksom.. hur ska man jobba när man gör devops och såna grejer. Då sitter jag på utvecklingssidan och försöker få igång continuous delivery så att … man tar emot det på ett bra sätt
5	Key points: P3-4 DevOps är någonting som lärs ut P3-4a DevOps är någonting man gör P3-5 DevOps / CD kan tas emot på ett bra eller dåligt sätt
6	**Okej, och då är det att du sätter upp verktygslådan åt dem då eller?**
7	Jaa, på IKEA så coachar jag mer de som sätter upp miljöerna och sånt, det är det jag gör. Mer teori och liksom vad ska man tänka på? Vad är viktigt när man .. Ska man [?? 01:15]
8	P3-6 DevOps omfattande nog att behöva coaching för
9	**Ja, vad ger du till dem för tips då egentligen, eller vägledning med det här (verkligen, vad består det utav? 01:21)**
10	Egentligen, allting handlar ju om att ge utvecklare möjlighet att ta ansvar Och det gör man genom att ha snabb feedback tillbaka, och det är det som är Continuous Delivery. Men sen ute i produktion så måste ju de ha ett bra samarbete med Operations.
11	P3-7 Utvecklare får möjlighet att ta ansvar med hjälp av snabb feedback

	P3-8 CD innefattas av snabb feedback P3-9 Produktion kräver ett bra samarbete med Operations
12	**Förlåt, vilka är det som ger feedback till utvecklarna?**
13	Allt, alla, hela vägen utåt. Så att testningen måste vara helst automatisk och alla miljöerna måste sättas upp.. automatiskt igen (?1:56), så att man får snabb feedback på hur det gick för min commit. Och även när det kommer till produktion så ska ju den komma snabbt ut i produktion, och det gör den genom att de samarbetar bra här, Ops i DevOps.
14	*Feedback kan komma från flera ställen. P3 verkar hänvisa till att alla miljöer, vilket vi i den här kontexten innebär de verktyg som P3 coachar bolaget att sätta upp.* P3-10 Utvecklare ska veta direkt ifall deras förändring inneburit framgång eller fel P3-11 Feedback till utvecklaren beror till viss del på samarbetet mellan Ops
15	**Så samarbetet mellan dem ska ju vara viktigt, för det här med att sätta upp miljön, faller det på operations egentligen då? Är det därför samarbetet behövs?**
16	Samarbetet behövs så att när Dev gör sina .. sätter upp sina miljöer för test, då ska det helst göras på samma sätt som i produktion. Då ska de gärna veta hur gör man, så samarbetet måste ju finnas där. Annars testar de på ett sätt och så gör Op på ett annat sätt. DevOps för mig är ju mer kulturen,jag vet inte vad det är för er?
17	*P3 berättar för oss att man vill ha sina testmiljöer så lik produktion som möjligt, och att det krävs ett samarbete för att kunna möjliggöra detta. P3 berättar snabbt efteråt att han ser DevOps för honom är mer om kulturen. Vi är osäkra ifall han menar att DevOps handlar mer om kulturen än att ha en lik testmiljö eller ifall en liknande testmiljö är typiskt kultur. [Frågar för uppklarning]* *Vidare frågar P3 om vår uppfattning om vad DevOps är.* P3-12 Testmiljöer ska vara så lik produktion som det går. För att det ska ske krävs ett samarbete med Operations P3-13 DevOps handlar om kultur P3-14 DevOps kan uppfattas olika
18	**Mycket så, mycket kulturen mer än just kanske vilka verktyg du använder är väl vår sammanfattade uppfattning utav det. Att det är viktigare med samarbetet än det.**
19	Lär ni er om DevOps på skolan eller är det någonting som ni har hittat på själva?
20	Teoretisk avfärgning *En uppfattning av att DevOps kan vara ett fenomen som börjar bli aktuellt och undrar ifall skolan visat intresse.*

21	**Det har vi hittat på själva. Och ämnet är väl diffust, det verkar finnas mycket..**
22	Ja alltså det är skitsvårt att veta om man, om ett företag har bra DevOps. Det är som att försöka speca upp "Vad är en bra utvecklare?". Man kan ha punkter men man kan fortfarande lyckas med allting men ändå dålig.
23	*P3 väljer igen att säga att man kan vara bra eller dålig på DevOps och jämför det med en abstrakt fråga.* P3-15 DevOps är svårmätt då det är för abstrakt
24	**Är det någonting du tar med dig egentligen när du lär ut att nu ska vi fostra kultur? Eller hur ser ni, hur ser råden ut när du vill att de ska samarbeta?**
25	Oftast brister det i trust. De litar inte på varandra. Innan så har det varit mycket att utvecklare kastar över en vägg till testarna som kastar det över en vägg till Operations. I och med att scrum och sånt så har ju den här väggen mellan test och utvecklare som blivit "kort nu eller lägre" ???4:14 så man ser i alla fall igenom och se över?? Men den väggen är ju kvar mellan operations. Det är den väggen som .. om man berättar för dem att i förlängningen "går det bra?"4:25 då är det lättare att förstå liksom att ta med Ops i scrum-teamen på något sätt. Då är det lättare att förstå liksom vad man ska göra.
26	*P3 nämner att det ofta är tilliten till varandra som är hindret för en bättre kultur. Berättar hur Utvecklare tidigare lämnat över till testare, som testar och sedan lämnar över till Operations. P3 förmedlar till de utvecklare han träffar att om de "river" den väggen, ger mer insikt i andra delar, kommer det att gå bättre.* P3-17 Tilliten är ofta det som brister som ett resultat av dålig insyn i andra avdelningar
27	**Hur ser ni på det här med ägandeskapet? När samarbetet är igång, hur ser ni på kollektivt ägandeskap?**
28	Av vad, samarbetet?
29	**Ja, eller egentligen produkten tänker jag. Vem är det som ansvarar för någonting när det väl är i produktion?**
30	Det är olika saker. Operations ansvarar ju för att systemen är uppe medan utvecklare ansvarar för att produkten funkar som den ska. Och det betyder ju att för att development ska veta hur man ska deploya en sak på en server så måste de ju få "det berättat för sig?"5:38 av Ops eftersom det är Ops som har det ansvaret. /* Och allt det här är ju en mans åsikter, så jag är otroligt nyfiken .. snack om att dela med oss av resultatet */
31	*P3 pekar på att det finns olika ansvarsområden, Operations ansvarar för att miljön produkten befinner sig i, medan utvecklarna har ansvaret för själva produkten. För att Dev ska kunna göra förändringar så behöver de veta hur de ska gå tillväga, vilket är upp till Ops att förklara.* P3-17 Ops ansvar att förse Dev med verktygen och kunskapen för att kunna släppa själv. P3-18 Ops ansvarar för plattformen, Dev för produkten som kör på plattformen

32	**NoOps?**
33	Ja, NoOps.
34	**Det är inget du har stött på eller?**
35	Jo, det har jag. Bara i teorin, inte i livsform, någon som vågat. Men det funkar ju kanske på mindre företag. Behöver inte ha någon operations alltid. Det räcker ju .. det är lite som när man tänker så här att scrum utökar sig till Operations också, så kommer alla medlemmar vara testare, utvecklare och operations. Och det är ju på något sätt noops. Det funkar ju på vissa ställen tror jag, och då är det ju bra.
36	P3-19 Med scrum så får alla alla roller
37	**Det här med mindre bolag, vad ser du för stor skillnad mellan ett stort bolag som X och de här mindre. Alltså noops verkar ju vara en att det inte behövs operations men när ökar behovet utav operations?**
38	Det vet jag inte..
39	**Nej, svår fråga kanske. Men passar DevOps mer eller mindre bra beroende på storleken av företaget?**
40	Jag tror att..i mindre bolag så är det nog ett slöseri med resurser att ha en specifik operations. Det är ju inte så komplicerat tycker vi då, vi utvecklare. Men det är det ju. Alltså det är ju inte så lätt.?7:56 Men alltså det beror ju på, kan man få in dem i teamen, kan man ändå fortsätta 7:58 ??. Det är samma sak med test, för det är ändå en förlängning. Test, om man sätter ut testresurser i de här teamen, hur kan de då samarbeta och ändå liksom testa så det är samma kvalitet ?8:12. Samma sak är det i Ops, om man lägger ut den kunskapen i teamen, hur ska de kunna bibehålla den kvalitén? På alla servrar och allting. Så att de inte helt plötsligt har olika portar de har öppna och saker?8:28 . Så på något sätt måste de kunna samarbeta mellan sig, men ändå vara en del av sitt. Men det var inte alls ett svar på din fråga.
41	*P3 verkar antyda att Operations uppgifter är inte lika komplicerade när omfattningen är mindre, men menar att det kan variera och inte en spikad faktor. Frågan var svår att få svar på och vi förstår inte delar av svaret vi fick. Vi tycker dock att vi ser en antydan om att ifall kunskapen decentraliseras i team, hur ska kvalitén då bibehållas. Ser även en hänvisning till förra frågans svar om agila team och att Ops numera ingår i det.* P3-21 Att Ops ingår i team ses som en förlägning av redan agila team (SCRUM) P3-22 Decentralisering kan ses som ett orosmoment då kärngruppen(kunskapsområdet, homogena arbetsroller) inte existerar i samma omfattning.
42	**Tack ändå, tack ändå.** **Vi är intresserade av de här teamen och på något sätt, min uppfattning är att teamen ska vara så kompletta som möjligt. Och att det kanske utgör .. kunskapsdelning däremellan ska vara komplett. Hur fostrar man en sådan sak, att kunskapsdelning ska ske?**

43	Ja det är ... kommer du på det så är du rik. Det är ju svårt redan nu när man försöker dela upp det på test .. nu är jag tillbaka på det här test och .. det här är ju bara en förlängning, typ. Det är ju många utvecklare som har lärt sig mer om test. Det är många testare som lärt sig mer om utveckling. Så de kan ju stötta varandra. Det är ju egentligen bara en tredje sak, att man tar in en operations som kan lära de andra och så kan han lära sig annat. Så det där är ju jättesvårt, det gäller att det kommer uppifrån på något sätt att så här ska vi göra. Men det här moderna sättet att tänka på .. skitsvårt på (stort? 9:53) företag, speciellt med chefer som tycker att men så här har man alltid gjort och så här ska vi alltid göra så. ...(DevOps grej något?? 10:02)
44	*Att fostra kunskapsdelning är en stor utmaning enl. P3 och menar på att motivet bör komma ifrån ledning. Detta kan dock i P3 erfarenhet vara problematiskt, speciellt i samband inom större organisationer med tydligare ledning* P3-23 Kunskapsdelning måste uppmuntras av ledning vilket kan vara problematiskt
45	**Men du försöker alltså, jag förstått det rätt, att få Dev och Ops att jobba närmare tillsammans.**
46	Ja
47	**Hur gör du det?**
48	Som det är nu så försöker jag ju få instruktioner av DevOps eller Ops så som vi i Dev kan använda för att deploya våra testservrar på samma sätt som de gör i Production. Och vi vill inte ha en checklista, vi vill ha ett verktyg som - ett recept som gör det åt oss.
49	*P3 vill få Ops att instruera Dev att använda sig av verktyg som liknar det Operations gör. Vår förståelse är att Ops ska visa Dev hur de kan göra för att deploya till en miljö som är likt produktionen (som är Ops ansvar).* P3-24 Testmiljön ska vara så likt produktion som möjligt och det är Ops som håller i kunskapen.
50	**Så genom att sätta upp denna miljön så kommer det av sig själv lite eller att när man sätter upp verktyg egentligen för att kunna deploya till testmiljöer, det kommer av sig själv då?**
51	Ja, alltså om de har ett visst sätt att göra det på så gör vi på samma sätt så att istället för att använda alla servers (som är ändå??11:05) maskiner så använder vi servicevirtualisering. För att liksom täcka upp liksom, vi kan ju inte öppna upp allt liksom på samma sätt.
52	*Vi förstår det som att P3 talar från en utvecklarperspektiv. Utvecklare ska efterlikna Operations processer.* P3-25 Utvecklarna ska efterlinka Operations processer
53	**Men rent fysiskt och socialt, är det fortfarande lika stor separation som innan då?**
54	Just nu är det det, men vi försöker få med dem mer och mer. För det är just den

	där tilliten som saknas där också. För det är ju, jag läste att förra året så .. Nu är det närmare hörsägen men 80% av de leveranser som gjordes via, till Operations fallerade. Eller de, de som fallerade i produktion var på grund av instruktionerna för hur man ska sätta upp den här, ifrån utvecklingssidan. För de förstod inte att men den porten måste ju vara stängd eller där har man ju brandvägg liksom eller den servicen är inte där(??11:49) Så att det här att de inte visste hur det såg ut på riktigt ställer till det rätt mycket.
55	*P3 nämner tilliten som ett problem. Leveranser till produktion falerar pågrund av för bristfälliga instruktioner från Operations, att utvecklare inte i sin utveckling har en tillräcklig uppfattning om hur det ser ut på riktigt (produktion).* P3-26 Otillräcklig insyn i produktion orsak till fallerande lanseringar
56	**Tilliten, jag tänker bara om det finns någonting du ser som en magic bullet till hur man direkt ökar då utan de här ..**
57	Ja en magic bullet, ha det jag haft det hade det kostat. Nej det finns inget.
58	**När jag hör tillit så tänker jag "jag litar på dig" men att istället så är det att man sätter upp något failsafe verktyg så att tilliten blir automatisk då, är det det som menas?**
59	Ja, genom att man gör en massa dubbel-extra-kontroller bara för att .. och så tar det lite längre tid alltihopa. För att man vet ju inte vad de har testat, man tror inte att de har ordentligt. Operations gör så mot utvecklingen för att ofta så går det sönder så de måste ha en egen test, produktionsserver för att göra testdeployments innan de gör det på riktigt för att de vågar inte gå på riktigt direkt ifall man måste rollbacka och det är jobbigt. Samma sak med test, de litar ju inte på att utvecklarna har testat det ordentligt så de kör alla testerna fast de här har gjorts rätt mycket. Så därför försöker jag också ta bort rätt mycket att det ni har testat här behöver inte testas någonstans mer. Det ska vara transparent och tydligt liksom. Och det är svårt.
60	*Tillit kan skapas genom att Operations själva sätter upp en produktionsmiljö till Test att använda. Vidare så gör Test det till Dev som är deras input. Avdelningar ökar tilliten till inputen genom att själva bygga verktyg och failsafes. P3's bedömning är att det är svårt att genomföra* P3-26 Avdelningar ökar tillit genom att själva sätta upp verktyg och failsafes till sina inputs.
61	**Jag tänker hur approachar de er som konsulter då. Är det att nu har vi ett DevOps-initiativ här så kan ni komma och hjälpa oss eller?**
62	Just DevOps har ju inte vi skyltat med så mycket utan det är ju Continuous Delivery, så de har ju på utvecklingssidan har ju sagt att vi vill ju kunna leverera på ett bättre sätt. Och så har de frågat då om ni kollar om ni kan få till Continuous Delivery hos oss. Och då har jag kommit dit och kollat hur de jobbar och så fick de följt en modell så att ja ni ligger så här (?14:13).. Och en av sakerna är ju att det ska finnas DevOps och så har man pratat med dem också, för att det är ju hela kedjan. Och utvecklarna, Development förstår ju inte oftast att Continuous Delivery

	är ju faktiskt så att produktion skulle kunna ta det och slänga ut det om de vill. De vill ju bara till nästa steg, över muren va.
63	*DevOps bör finnas där när de ska implementera CD. DevOps omfattar hela kjedjan. CD till för utvecklare ska kunna lansera till produktion.* P3-27 DevOps är involverat i hela utvecklingsarbetet
64	**Blir de paffa då när du kommer med den här insikten då på något sätt, ja men om ni ska ha Continuous Delivery så ska tänka på det här (?**14:47 **) Hur är reaktionen på en sån sak då?**
65	Nej alltså egentligen inte, egentligen blir de inte det utan de blir bara glada över att det finns en väg fram.
66	**När du säger att de vill leverera bättre mjukvara, på vilket sätt menar du då? Hur ska de göra det bättre?**
67	Leverera bättre mjukvara. Sa jag det? **Kunna leverera bättre bara**
68	De gör ju sina sprintar, de gör ju sina stora tjok av kod, skickar det till test som testar jättelänge, skickar till produktion som får testa och deploya. Och sen säger de det är fel, då kommer det tillbaks en rapport på att man för två månader sedan så blev det ett fel där. Då har de ingen aning om vad de gjort för något. Så man vill ju liksom att det ska ta någon, nästan en dag max liksom. Helst hela tiden men vissa tester tar så lång tid så att..
69	*Ju längre feedback väntar, desto svårare att åtgärda.* P3-28 Snabbare feedback lättare att åtgärda problemet
70	**Det är så här lasttester då eller?**
71	Till exempel. Man vill ju ändå att det de får en felrapport på ska vara aktuellt i minnet så de kan laga det utan att context switcha för mycket. Och då .. jag försöker återkoppla till vad jag sa men .. ja. Om det ligger två månader gammal kod på deras main line på något sätt, så kommer alla placera sin nyutveckling på den koden. Så att om man då säger ja men det var helt fel, du tänkte fel här. Då kommer alla andra som har levererat och håller på och kodar också liksom "jaha vi gjorde fel, ja då måste vi ändra alltihopa".
72	*Ju fortare ett fel blir tillrättar, desto mindre löper risken att bygga på felaktig kod* P3-30A Undvika context-switcha för mycket P3-30B Undvika att bygga vidare på kod som fått feedback för sent genom snabbare feedback
73	**Är det det man menar med snabbare feedback så att säga?**
74	Snabbare feedback är att den som introducerar ett fel ska snabbt få veta att det var fel, laga det snabbt liksom. Så att det stör så lite som möjligt resten va. Och samma sak när man deployar till DevOps, om det är DevOps vi snackar om. Om i deployment, jag behöver ändra i deploymentscriptet liksom för att jag behöver

	en till service eller en till port uppe. Då ska man göra på ett konkret sätt tillsammans med DevOps så att när det väl kommer dit, att det inte smäller då liksom. Utan att de varit med från början och öppnat upp den porten kanske eller något sånt.
75	*Feedback från Operations genom att inkludera dem i ett tidigt stadium* P3-31 Tidig involvering främjar snabb feedback
76	**Krävs det en viss person för att vara med i ett sånt här DevOps-samarbete på något sätt. Ställer det några extra krav på personen, bör den vara på ett visst sätt för att kunna ha med den här kunskapsdelningen på något sätt?**
77	Samarbetsvillig? Nej jag vet inte. Det är väl alltid bra liksom. Nej men det beror ju lite på vad man menar med "Nu har vi DevOps", var är gränsen, var ligger liksom.. Man gör en massa saker sen helt plötsligt kan man säga "nu har jag DevOps". Jag vet inte vart den ligger liksom. Det är det som är problemet.
78	*P3 ser samarbetsvilja som fördelaktigt för att ha "bra" DevOps. P3 har svårt att se var man drar gränsen för vad som räknas som att ha DevOps eller inte.* P3-32a Svårt att avgöra när man *har* DevOps P3-32b Samarbetsvilja hos personal en fördel.
79	**Det låter som att .. skulle man kunna säga att DevOps är någonting man uppnår? Eller är det någonting man gör?**
80	Jag tror att de flesta företag nästan .. alltså man kan ha, nej man har DevOps. Antingen så har man det bra eller dåligt. Men de flesta har faktiskt DevOps, fast de flesta har nog jättedålig DevOps. Det är väl så man kan se det. Så det är väl en slags .. alltså om all mjukvaruföretag har utvecklare, men vissa utvecklar dåligt och vissa utvecklar bra. DevOps är mer en ...?18:46 en kultur, något som finns. När man har bra DevOps, det vet jag inte men alla har det på något sätt. På någon slags nivå. Det är otroligt svårt att definiera DevOps. Men det är samarbetet med kulturen, att samarbeta.
81	*P3 vill beskriva DevOps som någonting man har och att det finns nivåer till hur pass bra man har det. P3 utrycker det som svårt att sätta ord på (kultur, något som finns).*
82	**Vad gör folk oftast för fel i DevOps, som du ser?**
83	Det är det där att inte vilja samarbeta. Eller att de som sitter på operations inte har kontroll på hur de deployar till produktionsservrar till exempel. För att har de inte typ det scriptat eller versionshanterat eller har det i någon slags repository, då är det otroligt svårt att kommunicera det här bakåt. Så att de kan titta på det och sätta upp tester på samma sätt. Så att..
84	*P3 ser att det finns ovilja att samarbeta.* *Operations processer påverkar hur effektivt ett samarbete kan vara. Har inte Operations processer som tillåter ex. Scriptning så blir samarbetet svårare.* P3-33 Finns en ovilja att samarbeta P3-34 Operations nivå(mogenhet) av kontroll och rutiner påverkar samarbete

85	**För att man bygger upp kedjan på deras practice då?**
86	Ja, det är ju de som styr det mest på server produktionen, så man vill ju vara i lärande (?20:02) med dem. Men kan man inte kommunicera alltså visa att så här gör vi, då är det svårt. Så att någon slags orkestreringsverktyg, om ni har lite koll på några.. vet inte om ni använt chef eler puppet eller ansible och sånt. Någonting sådant liksom, för att kunna visa att så där gör vi, nu kan ni göra likadant. Och då kan även utvecklare kanske commita kodförändringar till deras typ.. som förslag då på förändringar. Och då kan jag även godkänna det, och då är det klart liksom. Då kan vi .. då har vi den porten liksom.
87	*Fortsatt tema från tidigare svar; att desto tydligare Operations kan visa hur produktionsättning ska gå till desto mer kan samarbete främjas. P3 nämner orkestreringsverktyg.* P3-35 Tydligare processer och användning av verktygunderlättar samarbete
88	**Har du stött på att de Opsarna som egentligen sätter upp detta, på ett sätt, de ger ju ifrån sig något som är från deras domän. Finns det något motstånd till sånt? Att de delar med sig utav sin puppetkunskap eller liknande så att "nu kan ni också göra det"?**
89	Jag tror att om man har kommit dit att man använder ett sådant verktyg så är man ganska villig att dela på det. Så ligger det ju liksom i någon slags versionshanteringsverktyg, och då är det ju oftast .. lägger man in det på ett sådant sätt så är det oftast klart liksom. Då är det bara att ta alla stegen så är man där liksom. Men att få dem att använda det från sina personliga script liksom och göra om dem till något sånt, det steget är mycket svårare. Så kommer jag till ett företag där de har något sådant verktyg på plats, då känner jag att då är det ganska lugnt. Då kan man bara haka sig på det där och använda det, då går det mycket fortare. Men att övertyga dem att de behöver det här, då är det mycket svårare liksom.
90	*P3 ser en svårighet att motivera till att börja använda orkestreringsverktyg som ett större problem än att få folk att dela med sig av kunskap av verktyget. En tröskel som ska nås över.* P3-36 Organisation ser inte alltid nyttan i verktygen
91	**Vad, just i den övertygelsen, kan du se någon sådan här "detta är de mest motsträviga emot"?**
92	Förändring
93	**Förändring, det är bara att man inte kommer och rör till det liksom**
94	Ja alltså de flesta tycker ju inte om förändring. Ingen tycker ju om förändring.
95	**Och du jobbar med förändring?**
96	Ja
97	**Det här med kunskapsdelning och så, använder ni några verktyg för**

	kommunikation till exempel chat eller.. som spänner över Dev och Ops?
98	Inte vad jag vet, jag tror inte det gör det. Det är ju jättebra att ha det. Vi har det på softhouse, slack heter det, och då har vi liksom olika grupper, olika topics liksom. Så att vi kan kommunicera med varandra med den topicen på övriga kontor i landet också. Och det är ju jättebra, men just på IKEA nej det har de inte.
99	*P3 ser dedikerade kommunikationsverktyg som någonting positivt för samarbetet*
100	**Finns det någon roll som kanske ansvarar för samarbetet mellan Dev och Ops, alltså som en kontaktperson liksom.**
101	Inte än, men det försöker vi ju få till. Inte kanske en roll men att nånstans ska ansvaret finnas. Vi försöker egentligen .. just nu sitter vi och pratar med dem för att se hur villiga är de att samarbeta, och det är ju verkligen positivt. De kör puppet just nu, faktiskt.
102	*P3 ser nyttan i att ha en ansvarsroll för DevOps initiativet.* P3-37 Ser nytta i att ha en ledarroll i initiativet
103	**Du har mest sett att man sitter separat liksom, det har aldrig blivit att rollerna har suddats ut helt?**
104	Mellan Dev och Ops?
105	**Ja**
106	Nej, de sitter ju olika hus tror jag också. Det stora datacentret är i Älmhult och utvecklarna är i Helsingborg och det finns någonting i Malmö också, det är väldigt stora avstånd.
107	**Ser du någon negativ impact utav det, skulle man tjäna på att sitta närmre?**
108	Definitivt, sitter du nära någon som du kan se och prata med så är ju .. kommunikationen blir ju mycket högre, plus att tilliten ökar ju. Ser du en person så .. det var han som skickade till mig
109	*P3 ser en ökad kommunikation och tillit av att geografisk/fysiskt vara närmare varandra* P3-38 Geografiska avstånd har påverkan på kommunikation och tillit
110	**Mm, ett ansikte liksom. Och det här med fikarummet och det också**
111	Ja. Det är ju svårt, de är ju många. De är ju hundratals liksom, svårt att sitta bredvid alla. Så därför blir det såna DevOps (?24:58) för att man är så många.
112	*Storleken på samarbetet kan påverka möjligheterna för den fysiska närheten.* P3-39a Organisationens storlek påverkar möjligheter för fysisk närhet
113	**Du sa att alla tyckte det var så jobbigt med de här förändringarna. Är det någonting specifikt de brukar tycka är riktigt mycket pain med de här förändringarna?**

114	Nä inget specifikt så. Menar du operations eller de här andra?
115	**Vilket som egentligen**
116	De flesta som tycker det är dåligt med förändring är ju de som inte är .. som inte jobbar med det direkt, alltså själva operationspersonerna är testpersonerna är utvecklarna. De förstår ju oftast att de här förändringarna ..??25:41, det är ju deras chefer och deras projektledare och sånt som. Kör man continuous delivery, då har man inte den här projektplanen. Och då blir det så svårt, vart är vi nu? Vilken tollgate ska vi passera nu?
117	*Ofullständiga meningar, men vi uppfattade P3 som att det är ledning och projektledare inte kan relatera och har en negativare inställning* P3-39b Traditionell projektledning ställer sig motstridigt till förändringen
118	**Slå omkull det klassiska på något sätt.**
119	Precis
120	**Och då är det alltså ledning som egentligen är mest rädda för det eller?**
121	Ja alltså de är nog för att de inte förstår, det är lite så här svårt att tro att det här är bra.
122	*Att DevOps/CD är svårförklarat kan påverka lednings inställning och motivation.* P3-40 Oförståelse påverkar inställning och motivation till DevOps/CD
123	**Tror du utvecklarna och de kanske har hört om DevOps men ledningen kanske inte har? Finns det en stor spridning på uppfattningen utav det?**
124	En kollega till mig var till en DevOps-konferens i Oslo i .. någon månad sedan, vet inte när det var. Men då var det ju massvis med konferens(?26:36)-personer där. Men grejen var att det var jättemånga personer som kom fram till dem och sa att "Det här DevOps, är det något nytt?" liksom. Man har ingen aning om att det fanns. Man trodde att det var något helt annorlunda, helt nytt liksom. Så jag vet inte varför de inte vet något, de tror ju att det är att nu ska vi göra det enklare för development så vi kallar det för DevOps. Men egentligen är det ju tvärtom liksom. De ska ju, DevOps är ju för Ops skull egentligen för att de ska få .. eller för bådas skull så att de får bättre saker till sig.
125	*Vi uthämtar att DevOps upplevs som ett nytt fenomen och uppfattningen skiftar kring vad det innebär.*
126	**Det var nytt för många då?**
127	Min kollega han hade ju varit där med Chef och han sa att han hade ju hört talas om någonting som dålig kunskap om DevOps just i så här Sverige, Norge. Så att .. så det var lite dåligt betyg.
128	**Visste han vad DevOps var när han kom hem?**
129	Ja måste vi .. nu får ni se till så att vi sprider kunskapen

130	**Ja vi jobbar på det. Men det här med milstolpar .. när man har mer frekventa releases liksom, kan man sätta upp .. ser du att man sätter upp andra mål då? Än tidigare.**
131	Andra mål, vad man releasar menar du?
132	**Jag menar att förut kanske man hade att släppa version ett och så kanske man hade det som en milstolpe i projektplanen. Har man några andra sätt att mäta det .. alltså progress.**
133	Ja alltså förr var det mycket så att man kör vattenfall, att det här ska ingå och det här ska släppas vid den här tidpunkten. Då utvecklar man, sen testar man sen släpper man. Nu i och med det allt det här agila och allting, då har du ju en lista, en prioriterad lista på saker som ska släppas. Och då har du oftast en tid att där ska du släppa det, sen släpper du dem en i taget tills du kommer till den punkten. Och så får man se allt det här hann vi släppa, liksom. Det finns det här tid, vad den ska innehålla och kostnad, så man får välja vilken ska man .. alla kan inte vara statiska. Och kör man agilt så är det ju innehållet som är flexibelt. Det har inte liksom med DevOps att göra men..
134	*P3 gör en avskiljning på att agil utveckling och släppa ofta inte har med DevOps att göra. Vi är osäkra på budskapet, ifall P3 menar att DevOps faktiskt inte har med det som nämns, eller ifall det hör till agilitet och att det finns en viss gräns var agilt tar slut och DevOps börjar.*
135	**En annan syn på mål på något sätt att .. känns det som att alla jobbar mot samma sak, är staketet helt rivet på det sättet nu?**
136	Nej, det .. ja, ja alltså det håller ju på. Alltså pratar vi om IKEA så har det precis börjat liksom tärningen är kastad högst upp. Men ju större ett företag är desto svårare är det för en enskild person i företaget att förstå sin roll i helheten. Man gör sin grej och så levererar man och sen så .. man vet knappt vad nästa steg är för något. I ett mindre företag, helst, ifall man kör DevOps då vet man ju att det här är kraven som kommer in, här utvecklar man och det kommer se ut så här på slutet. Så det är ju också en stor fördel med DevOps att man har helheten klar för sig. Att man vet varför man gör saker. Och det har ju med mål att göra, men i större företag på ett gammalt sätt så har man murar runt varje person nästan. Jag gör mitt liksom och sen .. ta hand om sig själv.
137	*Storlek på organisationen påverkar individens möjlighet till insyn; större organisationer är det svårare se sin plats i kedjan. P3 nämner att DevOps breddar en persons insikt och helhetssyn. P3 nämner att det är ledning högre upp i organisationen som initierat omorganisationen.* P3-41 DevOps för med sig en större insikt i organisationen för individen P3-41a
138	**Jag tänkte på om man får dem att släppa mer frekventa releases hela tiden, utvecklarna, hur får man de att våga göra sådana ändringar som kan vara farliga hela tiden liksom? Är det inte lätt att det blir fel om det måste releasas mycket?**
139	Alltså ju mindre ändringen är som du släpper, desto mindre är chansen att det blir

	fel. Och om man byter eller går från att släppa en gång i månaden till fem gånger om dagen så innebär varje ändring mycket mycket mycket mindre. Så att sannolikheten att det blir fel är ju för det första mycket mindre, och är det så att det blir fel då har du en sådan snabb feedback-loop att du kan laga den snabbt. Antingen kan du laga den eller så kan du ta bort den utan att skada något.
140	*Release frekvens och storlek på ändring i direkt anknytning med risk. Fler releaser minimerar risken. Mindre ändringar minimerar risken* P3-42a Frekventare och mindre ändringar för minskad risk P3-42b Rollbacks minskar risk vid deployments
141	**Så man får mindre skit för att göra fel?**
142	Precis, mycket mindre skit. Men är det så att du jobbat i två månader, sen testas det, då får du ju skit för att du stoppar utvecklingen jättemycket. Men nu är det så här att släng in en commit liksom och så får du svar att "Oj, det var fel", då ändrar jag väl ite där då.
143	*Med minskad risk för releaser minskar också osämja.* P3-43 Minskad risk i ändringar innebär en minskad risk för osämja
144	**Hur gör man med incitament för arbete? Är det någon skillnad när man jobbar med DevOps?**
145	Incitament för att jobba över huvudtaget eller?
146	**Nej jag tänker på olika avdelningar brukar ha lite olika mål, till exempel Dev kanske vill göra många och snabba releases medan Ops vill ha allting stabilt. Hur får man dem att samarbeta?**
147	Ja, ska samarbetet funka så måste man gå ett steg upp och kolla vad är det för business-mål vi har? Så man får liksom kolla vad har business för mål? Är det så att ska vi liksom ge kunderna ny content varje dag eller är det att de kunder vi har, att de har stabil grund att stå på? Till exempel, det är ju business som måste bestämma det. Kan ni scrum?
148	**Ja**
149	Där har du liksom en slags product owner som liksom bestämmer att det här är prioriteten på alla .. det ni ska utveckla. På samma sätt här, om man har DevOps utvecklinssätt att jobba på så måste man gå till business så blir det att ..?33:19 , att det här är prioriteringen på alting som ni ska göra. Tänk på det här. Vad ni än gör får ni aldrig sänka en server säger de kanske. Det är jätteviktigt. Eller vad ni än gör så måste säkerheten vara bäst eller vad ni än gör så måste ni sälja så mycket som möjligt. Så det är liksom business-målen som går över avdelningsmålen. Har ni läst the phoenix project? Den rekommenderas. Den är en novell fast inte på riktigt. Den är ju skriven utav någon som försöker förklara DevOps, fast de har skrivit en berättelse istället. Den är den enda bok ni behöver läsa. Just det här med hur de ska samarbeta är en stor grej.
150	*P3 ser att Dev och Ops bör gemensamt se till vilka business mål som finns.* P3-44 Dev och Ops tillsammans ser till business mål.

151	**Är det någonting du skulle vilja säga överlag om DevOps och utmaningar utöver tilliten och utöver någonting som**
156	Det vet jag inte om jag sagt eller inte men det som tar stor plats nu är ju verktyg. Man behöver inte verktyg för att ha DevOps men det underlättar ju jättemycket. Verktyg för att orkestrera, verktyg för att automatisera testning, verktyg för att .. man kan ju börja där liksom men det är ju inte nödvändigt, men det är ju bra. Det är ju en bra hjälp liksom. Annars så ja .. har jag uttömt ..
157	**Hur når feedback från kunderna .. påverkas det i DevOps? Feedback från kunder når utvecklare och operations**
158	Ja, lite gör det ju det faktiskt. För jag menar i bra .. eller det är egentligen bra Ops att i DevOps .. okej, man måste ju hela tiden monitorera, kunden kan ju komma med feedback på två sätt. Antingen så går de direkt via .. antingen så mäter man hur de beter sig på servrarna, eller så kommer de via business och säger att det där såg ju .. Men all mätning och all statistik och trender och sådant görs ju på Ops-sidan, och har man bra DevOps så har man bra samarbete. Då kan man lätt påverka utvecklingen och säga att "nu är någonting ett problem där" liksom. Och eftersom de har DevOps då har de gemensamma mål, business-mål, då vet de liksom hur de ska prioritera det. Så ja, det blir ju enklare att kvalitetssäkra feedback från kunder om man har DevOps. Annars finns det ju chans att det blir så att Ops säger att "nu är kunderna missnöjda för det där" och så säger [utvecklarna] "nej vi har inte tid, vi håller på med en ny feature". Det funkar ju inte, då.
159	*P3 ger oss en indikator på att bra DevOps innehåller en bra kommunikation mellan Dev och Ops. I detta sammanhanget att Ops ser kundfeedback först och genom dialog visar Dev. Gemensamma mål innebär att rätt saker blir prioriterade* P3-45 Bra DevOps innebär kommunikation och gemensamma mål. P3-46 Gemensamma mål anknyter prioritering
160	**Prioritering, löper det inte risk att det är svårt att prioritera tänker jag?**
161	Det är också svårt, alltid svårt. Men just nu business måste vara med och säga vad är våra topp tre punkter, vad vårt företag står för liksom.
162	P3-47 Verksamheten bidrar till gemensam prioritering
163	**När du är ute, vem är det som ger prio oftast?**
164	Nu har jag inte varit på ett ställe som har bra DevOps, alltså bra och bra. Så det är oftast är det ju liksom att inom development så får de direktiv från business och så översätter de det till sina saker hos de personer (utpekade för att prioritera?? 37:57). Men samma sak gäller där och där också så att. Och det är olika personer som tolkar vad business har sagt, det blir för många steg liksom. Så det är svårt. Men med en bra DevOps så är det tydligare för de har bättre samarbete.
165	**Skulle du kunna peka på någon att de har bra DevOps? Är det bolag som .. Netflix har man ju hört mycket om, har de bra DevOps eller man behöver en större insikt än vad de berättar för att kunna avgöra det?**

166	Jag tror de har bra DevOps. Men jag vet tyvärr inte några stora .. tomt tyvärr
167	**Det verkar på något sätt, för mig har jag svårt att se någon klar definition utav det och det verkar väldigt svårt. Något slags fenomen samtidigt som att det är en kultur och rörelse och på något sätt känns det som att det är en best practice någonstans också. Har du funderat på det perfekta ordet för vad DevOps är?**
168	Det .. jag kan inte hitta något .. Jag menar DevOps är ju, det är svårt.. det är svårt att säga.. Och sen sitter man i det också, då tror man att ja men vi har DevOps men vi har mycket problem här också, så vi har inte bra DevOps. Men tittar man utifrån kanske man tycker att ni har ju jättebra DevOps, bara någon sak som saknas liksom. Eller, ni har ju allt som behövs. Så DevOps är som fint väder, vissa tycker det är fint väder när det regnar, vissa tycker det är fint väder när det är molnigt. Men det är ju fortfarande väder.
169	*P3 förklarar att DevOps behöver inte vara "bra" för att man har komponenter på plats. Vår förståelse är att P3 är av uppfattningen att DevOps är någonting subjektivt.* P3-48 DevOps är någonting subjektivt
170	**Etablerade bolag som funnits sedan tidigare och som nu ska skifta, har de en större uppförsbacke eller?**
171	Ja det är verkligen så. Det är gamla rävar som tycker att man ska göra på precis samma sätt som sist. Jag vet att det var någon professor som Chalmers som sa att bästa sättet att inför Continuous Delivery och DevOps är att sparka ...?42:19 chefer. Så kan man införa vad man vill.
172	*P3 ser att bolag som existerat och arbetar mer enl. Traditionella metoder kan vara motstridiga.* P3-49 Bolag med traditionella metoder ser mer motstridighet
173	**Är det väldigt kostsamt för ett större företag att övergå till DevOps, sätta upp verktyg och sätta upp konsulter etc.**
174	Nej jag tror inte det är så dyrt att göra det.
175	**Hur mäter man hur väl övergången har fungerat?**
176	Ja hur mäter man om det är bra DevOps? Jag vet inte. Men man får ju ta några mätpunkter som hur snabbt feedback kommer tillaka, hur snabbt kunders klagomål har fixats, hur mycket är (är de med tillsammans?? 43:11), hur mycket fel blir det från att en utvecklare levererar till att en operatör, Ops-personen ska deploya liksom. Blir det något fel när .. man får kolla, beroende på vad som har funkat dåligt innan så att man behåller de mätvärdena och mäta igen sen.
177	*P3 ger oss vad som kan indikera P3's syn på vad som utgör bra DevOps, men antyder att det är svårmätt.* P3-49 Svårt att veta vad som utgör "bra" DevOps
178	**Det var nog det för oss, tack igen!**

Appendix F

Intervjuguide inför intervju med **P4**.

Intervjuguide C (P4)

Bakgrund	☐ Vad är din roll här? ☐ Skulle du säga att du är Operations på företaget? ☐ Vad skulle du säga är Operations, vem hör dit och vad är uppgifterna? ☐ Har du jobbat med DevOps förut? ☐ Om ja: Skiljer sig DevOps mellan din dåvarande och nuvarande arbetsplats? ☐ Om nej: Vad skiljer sig från dina tidigare arbetsplatser och den DevOps-organisation du befinner dig i nu? ☐ Vad är DevOps för dig? ☐ Tror du att det är den gemensamma bilden för alla i er DevOps-organisation? ☐ Om inte: *Ser du detta som problematiskt?*
Insyn	**Kommunikation/Dialog** ☐ Hur ser Ops kommunikation ut till andra avdelningar? ☐ Hur bör det se ut enligt dig? ☐ Vad ser du som problematiskt? **Kunskapsspridning** ☐ I er dialog, sker det någon form av kunskapsspridning där? ☐ Har du ett exempel? ☐ Kan du mer om Utveckling idag än du kunde tidigare? ☐ Har det hjälpt dig? ☐ Hur ser ni på proaktivitet och förebyggande åtgärder? ☐ Vad delar du med dig av? ☐ Vad ser du som problematiskt? **Organisationsstorlek/struktur/branch** ☐ Ser du att DevOps är begränsat till en viss sorts organisation? ☐ Påverkar storleken? ☐ Beslutsorgan? ☐ Nya och etablerade företag? ☐ Branch? ☐ Vad ser du som problematiskt? **Personligheter** ☐ Ser du att person behöver ha en viss personlighet för att kunna arbeta med DevOps? ☐ Om ja: Är det enkelt att få tag i sådana personligheter?

	☐ Ifall det finns personligheter som inte är X, hur påverkar det arbetet? Hur kommer man runt det?
Agilitet	**Prioritering** ☐ Hur prioriterar du ditt arbete? Dvs. vilka uppgifter prioriteras? ☐ Kommer prioriteringar utifrån eller inifrån? ☐ Hur tycker du att sådan prioritering fungerar? **Planering** ☐ Hur involverad är du i planering av produkter? ☐ Vad är din åsikt om det? **Ansvar** ☐ Hur skulle du beskriva dina ansvarsområden? ☐ Vad är typiskt för Ops? I en DevOps miljö? ☐ Vem bär ansvaret för testning och QA? **Roller** ☐ Hur ser ni på nyckelkompetenser/specialiseringar
Tillit	☐ När alla ska kunna lite av varje, känner du ett förtroende för att kvalitet uppehålls? ☐ Hur garanterar ni kvalité?
Verktyg/CD	☐ Vilka DevOps verktyg använder ni? ☐ Går det att ha DevOps utan dessa verktyg?
Feedback	☐ Hur ser feedback ut vid deployment? ☐ Hur ser feedback ut från support? ☐ Hur ser feedback ut från kund? ☐ Hur mäter ni hur väl DevOps fungerar för er? ☐ har DevOps-initiativet inneburit nya mätetal?

Appendix G

Intervjutranskript och nyckelpoängskodning av intervju med **P4.**

1.	**Hur ser din roll ut här, vad gör du på företaget?**
2.	Min roll här.. nu ser det ut .. nu har jag en roll ungefär som en sysadmin skulle jag säga. Alltså jag ansvarar för i princip all hårdvara som vi har i produktion, vi har liksom delat på .. vi har ju en massa intern IT som kör kontoret här och alla andra kontor och sajter och det har inte jag någonting med att göra. Men däremot så kör vi ju alla våra applikationer, alltså telefoni och webbsidor och så här, kör vi ju på egen hårdvara. Så de grejerna ansvarar jag för, och för nätverk kan man säga. Alltså så att de .. vi har ju prylar i Lund framför allt, och sen en datahall i Stockholm, lite i Oslo och lite i Malmö. Sen däremellan har vi ju nätverk och sen sitter vi ihop med en massa tredje parts leverantörer och vi peerar med en massa motsvarande aktörer som oss, och sen köper vi internet av några aktörer också. Så de grejerna ansvarar jag för som ungefär sysadmin skulle jag säga.
3.	*P4 beskriver sitt ansvarsområde som all hårdvara och alla nätverk som används i produktion.* *P4 beskriver sin roll som "ungefär sysadmin"*
4.	**Skulle du säga att det hör till Ops, Operations?**
5.	Jaa det skulle jag nog säga. Tidigare så var vi organiserade rätt så .. fanns det en operationsavdelning som var vi som höll på med teknik. Men framför allt sen support.. alltså det är ju rätt så många, ja operations sitter ju där uppe. Eller det som tidigare var operations, nu håller vi på att försöka liksom gå över till en tydligare DevOps-organisation. Och då ska .. men tidigare så på något sätt vår, min roll har varit liksom Operations snarare än Dev.
6.	*P4's organisation försöker bli en tydligare DevOps-organisation. P4 antyder på att en tydligare DevOps organisation innebär en mindre avgränsning* *P4-1 Organisationer kan vara olika grader av DevOps* *P4-1a DevOps innebär en mindre definitiv skiljning på Dev och Ops*
7.	**Okej, men support på något sätt hamnar under Operations?**
8.	Det handlar hos oss hamnar det under operations också.

9.	*Hos P4 benämns även Supporten som Operations.*
10.	**Är ni på väg att bli någon slags tydligare.. alltså blir ni mer som Dev eller börjar ni bli något eget utanför Dev nu?**
11.	Det var väl så ungefär att vi eller jag, just nu är det bara jag men vi var två stycken tidigare och ska bli det snart igen, att vi hade mer liksom interaktion med utvecklarna än med övriga operations som då är support och leverans och så här. Så därför så blev det naturligt att vi liksom flyttade ner så nu ingår vi snarare under liksom Dev-paraplyet. Men jag skulle säga att min roll är ju fortfarande liksom en klassisk Operations-roll. Men det är mer liksom internt organisatoriskt att min chef sitter på .. är en utvecklare nu istället för .. ungefär så
12.	*P4 uppger att Operations har mer kontakt med Dev än någon annan avdelning, och att Operations rent organisatoriskt ligger under Dev.* *P4-1b Ses organisatoriskt som en och samma avdelning*
13.	**Okej, och det är det man menar med när ni försöker gå mot mer DevOps, att..**
14.	Nja, egentligen inte .. ja kanske delvis, men tanken är också att vi ska skapa team som eh ..snarare liksom applikationsfokuserade eller funktionsfokuserade så att vi har ett team för säg en av våra webbsidor. Då är där några utvecklare som utvecklar den produkten eller tjänsten, där är kanske någon support-tekniker som supportar den och där är kanske någon, ungefär som jag, som liksom driftar den mer eller så här. Så att vi ska liksom, det ska vara mer så här cross-funktionella team än snarare en stor utvecklingsavdelning och en stor operationsavdelning.
15.	*P4 uppger att teamen är byggda utifrån funktion snarare än arbetsuppgift. Enligt P4 är teamen korsfunktionella snarare än strukturerade som stora avdelningar.* *P4-2 Team är funktionsorienterade* *P4-2a Team ska vara korsfunktionella* *P4-3 Organisationen saknar stora avdelningar efter arbetsuppgift*
16.	**Så man bygger team efter applikationen? så att..**
17.	Snarare, precis så, snarare team efter liksom applikation eller funktion än vilken tjänst eller roll vi har på företaget. Är tanken men vi är lite uppe i den omorganisationen så för mig är det också lite luddigt var vi kommer landa och så. Men det är väl ungefär målet att vi ska åt det hållet.
18.	*P4 uppger att teamen är byggda efter funktion eller applikation snarare än roll/tjänst på företaget.* *Enligt P4 är det otydligt hur rollerna kommer att se ut i framtiden.*

	P4-4 Team är funktions & applikationsorienterade *P4-5 Rollernas fördelning i framtiden är oviss*
19.	**Den här luddigheten, ser du det som något negativt att man kanske inte riktigt kan beskriva det?**
20.	Tänker du kring DevOps i allmänhet?
21.	**Ja, eller mer tänker jag nog på den här omorganisationen. Känns den tydlig för alla?**
22.	Jag tror att det kan finnas en del luddighet, sen tror jag kanske inte att det går ut över.. jag tror att alla kan liksom göra sitt jobb ändå. Men.. ja, det kan nog finnas nersidor med att det är lite luddigt och att vi.. vi testar ju oss fram lite kan man väl säga, det är ju inte.. det finns liksom inget supertydligt "exakt här ska vi vara om ett år och exakt så här ska det se ut". Utan det är ju snarare att vi testar oss fram lite. Och det är ju klart att det finns nersidor i det liksom att kanske ansvarsbitar är lite otydliga, "vem har ansvar för det här?" och så.
23.	*Det finns en otydlighet kring var rollerna är på väg i organisationen. Denna otydlighet kan ha nedsidor.* *Otydligheten kan mynna ut i otydliga ansvarsroller.* *P4-6 Otydlighet kring rollernas framtid.* *P4-7 Otydlighet kring rollernas framtid kan vara problematisk.* *P4-8 Ansvarsroller kan bli otydliga vilket kan vara problematiskt.*
24.	**Var kommer motivet ifrån till omorganisationen?**
25.	Det handlar väl, framför allt skulle jag tro att det handlar om att liksom skapa någon slags synergieffekter mellan, utvecklarna kommer i och med en sån här organisation hamna närmre liksom våra slutkunder, för att de har närmre kontakt med de som levererar tjänsten eller de som supportar den och så här. Och det tror jag är en stor fördel liksom att det blir mindre täta skott och mer liksom kommunikation.
26.	*Att utvecklarna kommer närmare kunderna ger synergieffekter för organisationen. Detta ger mer kommunikation och mindre täta skott mellan avdelningar.* *P4-9 Utvecklarna kommer närmare kunderna.* *P4-10 Närmande när kommunikationen mellan avdelningar ökar.* *P4-11 Siloeffekter motarbetas med kommunikation.*
27.	**De som levererar, menar du utvecklarna då?**
28.	Nej, utan då menar jag snarare .. alltså vi har ju en leveransavdelning som liksom leveranssätter våra kunder ungefär. De lite större kunderna gör ju inte

	allting själva utan de behöver lite hjälp för att få igång sina prylar. Och då, det steget kommer att vara närmre liksom, utvecklarna kommer sitta närmre kunden egentligen.
29.	P4 uppger att installatörerna (support) av deras mjukvara kommer närmare utvecklarna. P4-12 Installatörer, som har kundkontakten, kommer närmare utvecklare och därav närmar utvecklare kunden.
30.	**Hur kan en sådan kommunikation se ut?**
31.	Det skulle väl se ut ganska mycket så här tror jag att man sitter i möte liksom, kanske en gång i veckan i teamen och pratar om "vad efterfrågas", "vad bör vi göra" från alla håll liksom. Mycket så. Och sen använder vi ju Jira här, superaktivt liksom. Hela nya DevOps-organisationen är liksom med och lägger tickets och alla kan liksom följa hur progress på buggar eller nya funktioner eller vad det..
32.	*Möten mellan support och övriga avdelningar veckovis redovisar vilket behov som finns på marknaden.* *Alla organisationen får lätt en översikt över vad som händer med hjälp av ärendehanteringssystemet Jira (Kanban).* *P4-13 Veckovis möten mellan support och utvecklare hjälper utvecklare att förstå kundernas behov.* *P4-14 Ärendehanteringssystem (Kanban) ger alla insikt i vad som håller på att utvecklas.* *P4-15 Ärendehanteringssystem (Kanban) ger alla insikt i vilka buggar som finns.*
33.	**Okej! I den Jira, skulle du säga att det sker någon kunskapsspridning där?**
34.	Ja, absolut! Det är ju absolut vår största kunskapsbank för i princip allting vi gör för företaget som har med liksom utveckling eller så här att göra eller efterfrågan på funktioner och så finns ju där och dokumenteras ju där superväl. Så att det är ju vår absolut största kunskapsbank.
35.	*P4 uppger att deras ärendehanteringssystem är deras främsta resurs för kunskapsdelning.* *P4-16 Ärendehanteringssystem står för största delen av all kunskapsspridning på företaget.*
36.	**Har du alltid varit Operations, i tidigare yrkesroller och så?**
37.	Näe, det skulle jag inte säga. Jag är utvecklare egentligen och jag är anställd

	här också som utvecklare faktiskt fast på Operations för att liksom utveckla automatiserade utrullningar av applikationer till exempel, och övervakningar och så här. Så att jag kommer ju snarare ifrån utvecklingshållet. Vilket gäller rätt stora delar av företaget alltså traditionellt så har F1 varit super-utvecklingsdrivet, alltså att det .. liksom från .. de som grundade det var typ utvecklare liksom och väldigt många kommer från det hållet. Så nej, tidigare har jag inte jobbat i någon Operations-avdelning eller så här.
38.	*Operations utvecklar automatiserade utrullningar av applikationer.* *Operations utvecklar också övervakning av applikationer.* *Företaget är i stor grad utvecklingsdrivet.* *P4-17 Operations tillhandahåller automatiserad utrullning av mjukvara.* *P4-18 Operations övervakar det applikationer som körs.* *P4-19 Utvecklingsdrivet företag.*
39.	**Hur blev du introducerad till Operations-biten, var det en slags naturlig utveckling för att du kunde det sedan tidigare eller?**
40.	Ja, lite så och sen .. för tidigare var jag lite mellan Operations och Utveckling i det att jag jobbade som utvecklare på Operations. Det handlar ganska mycket om att det fanns luckor liksom på den sidan som vi behövde täppa igen. Däremot så har vi en 40 pers nu som är svinbra på att utveckla. Men det fanns inte så mycket folk på liksom tekniksidan, så att det är lite som att jag blev knuffad ditåt för att jag kunde och ville göra de grejerna, skulle jag säga.
41.	*Roller inom organisationen kan bytas om luckor behöver fyllas och det finns intresse från någon i en annan roll.* P4-20 Utvecklare bytte ansvarsuppgift för att det fanns ett behov
42.	**Hur plockar man upp en sån boll? Jag tänker att ditt intresse där hjälpte dig att ta reda på saker men var det så att det fanns en Operations-bit här sedan tidigare som lärde dig eller?**
43.	Ja, det skulle jag säga att det gjorde. Men, jag vet inte svårt att svara på.
44.	**I Jira, där kunskapsspridningen sker, vad ser du oftast att du delar med dig av där?**
45.	Ja, vi använder Jira eller jag använder Jira för min egen skull också liksom. Det är helt publikt, alla kan se allting jag gör i Jira. Men även smågrejer som jag egentligen bara inte ska blanda in någon annan i, även de skapar jag ofta tickets på. Alltså "gör det här, på grund av det" ungefär. Och sen försöker jag ha med liksom progress i "vad händer här". Både för min egen del för att ha det liksom dokumenterat till ungefär nästa gång jag ska göra det, eller om .. nästa gång så är det kanske inte jag som ska göra det utan någon annan som ska göra det, och då finns det också dokumenterat liksom. Så det är inte bara som kommunikation utan jag tänker att det funkar mycket som dokumentation också för.. liksom procedur, alltså både detaljbitar ungefär så här "jag körde det här kommandot och det funkade bra" liksom, eller mycket mer

	övergripande.
46.	*P4 använder ett ärendehanteringssystem med kanbanboard, både för att själv hålla koll på vad han själv skall göra och för låta andra i organisationen se vad han håller på med/har hållit på med.* *Användandet av Jira underlättar dokumentation av såväl processer som småfixar.* *P4-21 Ärendehanteringssystem med kanban används för att strukturera egna uppgifter.* *P4-22 Ärendehanteringssystem med kanban används för att dokumentera arbetsuppgifter.* *P4-23 Ärendehanteringssystem med kanban bidrar till informationsdelning.*
47.	**Ja, okej. Och sen så kan vem som helst gå in och läsa där?**
48.	Och så kan vem som helst gå in och läsa det.
49.	*Allmän tillgång till all dokumentation och ärenden.* P4-23a Alla har tillgång till dokumentation och tidigare ärenden
50.	**Brukar du referera till de sakerna du skrivit om någon har problem som de behöver lösa?**
51.	Ja, det händer ju absolut. Och framför allt så brukar .. ja, både att jag refererar till det men också att jag går tillbaks och letar själv om jag har liksom inte.. jag kommer ihåg att jag höll på med något förra året men jag vet inte vet inte riktigt vad och hur jag gjorde det. Och förhoppningsvis så har jag då dokumenterat det hyffsat. Så Jira; superbra. Skulle jag säga.
52.	**Tidigare när du sa att Telavox var skapat utav utvecklare och på något sätt varit utvecklardrivet så, är det något som krävs för att DevOps ska användas, eller passar det bara en viss sorts bolag?**
53.	Nej, det tror jag inte skulle krävas. Jag tror att många, nu kan jag inte riktigt sätta mig in i alla .. den här typen av bolag men jag tror att många skulle tjäna på att mer liksom .. mindre vattentäta skott och mer att .. mer att Operations och Dev jobbar tätare ihop. För oss funkar det ju skitbra i alla fall. Och jag tror att många skulle tjäna på mer av den biten.
54.	*P4 uppger att ett bättre samarbete mellan Utvecklare och Drift ses som en stor fördel.* *P4-24 Organisationer tjänar på att ha mindre avskiljningar*
55.	**Vad menar du med vattentäta skott mellan avdelningar?**

56.	Jag tror att på andra ställen så är det ofta så att utveckling tar fram en produkt eller någonting. Och sen är de klara med det och då skyfflar de vidare det till en liksom drift eller Operations-avdelning som sen ska drifta deras grejer, och att däremellan är det liksom inte så mycket kommunikation. Men hos oss så är det ju .. det är fortfarande utvecklarna som driftar sina applikationer och det är liksom de som får ta skiten om applikationerna inte funkar ungefär. Och jag tror på .. jag vet inte hur det ser ut överallt annars såklart men jag tror att på många, framför allt ännu större bolag, att det finns .. i alla fall klassiskt sedan tidigare, att utvecklarna tar fram en produkt och sen är de klara med det och då ska någon annan drifta det. Och det tror jag är sämre, i många fall i alla fall.
57.	*P4 uppger att utvecklarna fortfarande har ansvar för den kod som finns i produktion, olikt hur det kan se ut på andra ställen.* *P4 ser problematik i att överlämna ansvaret för en produkt till driften när den är färdig.* *P4-25 Utvecklare har ansvar för kod i drift.* *P4-26 Att lämna över ansvaret för mjukvaran till driftavdelningen ses som något som i många fall är sämre.*
58.	**Tror du att storleken på bolaget skulle ha någon påverkan på ifall DevOps fungerar eller inte?**
59.	Nej, egentligen inte. Förmodligen så ju större företaget är, destor större blir ju omställningen om man ska försöka göra den här omställningen. Men annars så tror jag inte, nej det tror jag inte. Det blir kanske mer naturligt om man är ett företag som är tre personer för att då vet alla allt liksom, än om man är ett företag på tusen personer. Så det är klart, i den mån att ..
60.	*P4 ser ingen problematik med att köra DevOps inom större organisationer.* *P4 tror att övergången till DevOps kan bli svårare om det är en större organisation.* *DevOps ses som naturligt i små organisationer.* *P4-27 DevOps fungerar i både små och stora organisationer.* *P4-28 Att övergå till DevOps kan vara svårare för en stor organisation.* *P4-29 DevOps ses som naturligt i väldigt små organisationer.*
61.	**Okej. Du sa att utvecklarna får skiten? smällen på något sätt.**
62.	Mm
63.	**Vad är det Ops ansvarar för då, ifall man säger att Dev har..**
64.	Man kan väl säga att vi ansvarar för att hårdvaran funkar och nätverk funkar, ansvarar upp till operativsystem-nivå i princip. Så att de .. alla servrar liksom ska funka upp till den nivån. Dessutom så ansvarar ju liksom supporten för .. vad ska man säga, det är ju de som tar emot felanmälningar eller så här. Och

	ofta hamnar ju de grejerna inte om att det kanske är ett fel på produkten eller en bugg utan bara att det var.. kunden inte visste hur den skulle ställa in den här funktionen eller så. Men ibland handlar det ju också om att bara relaya buggar till utveckling.
65.	*P4 uppger att operations ansvarar för att hårdvara och nätverk, ända upp till operativsystem-nivå.* *P4 uppger att Operations ofta skickar vidare buggar till utvecklarna.*
66.	**Okej, den relay-biten, hur ser den ut?**
67.	Det beror lite på hur.. är det superakut så är det liksom ringa eller gå och prata med den eller de som har ansvar för den produkten. Men annars använder vi Jira till de grejerna också, liksom skapa tickets för att "den här funktionen funkar inte som den ska", eller "den borde fungera så här" eller "när jag gör det här så händer det här". Och sen så plingar det till i mailen hos de som liksom ansvarar för den produkten.
68.	*P4 uppger att kommunikation kan ske face-to-face om det är akut. Andra akuta ärenden kan kommuniceras via telefon.* *Ärenden som inte är akuta går via organisationens ärendehanteringssystem.* *P4-30 Akut kommunikation sker face-to-face eller via telefon.* *P4-31 Icke-akuta ärenden förmedlas via ärendehanteringssystem.*
69.	**Ja, okej. Men ni ansvarade upp till?**
70.	Upp till operativsystem kan man säga.
71.	*P4 ansvarar för hårdvara och mjukvara upp till operativsystemsnivå.*
72.	**Är det så underliggande paket och sånt som behövs för applikationen också som räknas till det?**
73.	Nej, det gör det egentligen inte utan vi installerar liksom .. utvecklarna beställer ungefär av mig.. ofta en VM med lite specar liksom "jag vill ha så här mycket disk och så här mycket minne" liksom och så. Och sen levererar jag då typ standardmaskiner liksom med standardpaket. Och vi kör ju Debian typ rakt över. Men bara då med liksom, ja vi har någon så här standard att detta ska finnas installerat på alla maskiner. Och sen är det utvecklarna som liksom installerar sin applikation och paket som den kräver, som går utanför standardgrejer.
74.	*Operations tillhandahåller virtuella maskiner till utvecklarna.* *En uniform standard på miljöer underlättar.* *P4-32 Standardisade miljöer och paket för Dev att beställa*

75.	**Så de specar och sen så sätter du upp den?**
76.	Ungefär så.
77.	*Utvecklarna specar vad de behöver och Operations sätter upp den miljöerna.*
78.	**Men är det så att du kan ha synpunkter på det de har sagt vid något tillfälle eller du tar specen och sätter upp det bara?**
79.	Det är ofta diskussioner kring det. Det är ju lite såhär en brottningsmatch hela tiden. Allt kostar ju liksom pengar. Det kostar mer att ha en terabyte disk än att ha en fem, tio. Så vi krigar ju alltid lite om.. Behöver du verkligen det här liksom? I varje fall är det nästan diskussioner skulle jag säga.. Vad tror vi att det behövs för att köra det här?
80.	*P4 uppger att det ofta är en konflikt mellan hur mycket prestanda Ops är villiga att avvara och hur mycket prestenda Dev säger sig behöva. Det är i Ops intresse att spara så mycket pengar som möjligt medans Devs behov fortfarande tillgodogörs.* *P4-33 Det är ofta diskussioner över hur mycket prestanda Ops skall upplåta till Dev för olika projekt.*
81.	**Så ni har diskussioner och sen så ….?**
82.	Ja, det skulle jag säga.
83.	**Okej. Är det den huvudsakliga kommunikationen ni har mellan varandra: när det är request på hårdvara.. och buggar..?**
84.	Det förs ju diskussioner hela tiden kring hur vi kan effektivisera saker och göra det bättre, går det att göra på något annat vis? Då är det ganska bra med input. Jag tror att min input är ganska värdefull för utvecklarna, och tvärtom, deras input är supervärdefull för mig. Innan varje ny grej vi gör, varje nytt projekt, så förs det en diskussion kring "hur ska dom här grejerna köra?" och liknande.
85.	*P4 uppger att diskussioner hjälper organisationen att hela tiden effektivisera processer och hitta nya metoder.* *P4 ser sin input som värdefull för utvecklarna, och ser utvecklarnas input som värdefull för P4.* *P4-34 Diskussioner mellan avdelningar hjälper organisationen att effektivisera saker.* *P4-35 Diskussioner mellan avdelningarna hjälper organisationen att hitta nya metoder.* *P4-36 Operations bidrar med viktig input till Dev*

	P4-37 Dev bidrar med viktig input till Operations.
86.	**Är du med från början då eller?**
87.	Det skulle jag säga att jag är, typ alltid.
88.	*Operations involveras i ett tidigt skede av projekten.* *P4-38 Operations involveras i ett tidigt skede av projekten.*
89.	**Vi förstod från när vi pratade med Jonas att ni kör med Continuous Delivery...**
90.	Ja, det skulle jag säga.
91.	*P4 uppger att organisationen har Continuous Delivery.* *P4-39 Organisationen har Continuous Delivery*
92.	**Kan du berätta lite kring det, hur det funkar hos er?**
93.	Jag vet inte riktigt hur ni definierar continuous delivery.. men jag antar att ni menar att vi liksom... Hur tänker du att ni definierar det?
94.	*P4 frågar oss om vår definition av CD. CD som någotning abstrakt som kan uppfattas och definieras olika.* *P4-40 Continuous Delivery svårdefinierat*
95.	**Vi har väl kommit fram till att det är lite som DevOps: väldigt svårdefinierat. Vår tolkning just nu, utan att gå för djupt ner, är att utvecklarna kan, på egen hand, låta någonting nå produktion.**
96.	Mm, så jobbar vi absolut. De flesta grejerna vi kör skulle jag säga att vi uppdaterar nästan dagligen, eller i alla fall veckovis. Med små buggfixar eller små tillägg och så. Och det är ju ingen stor process eller så, utan det är i princip varje utvecklare som commitar en bit kod och sedan rullas det ut.
97.	*Vi svarar på frågan om vad CD är för oss för att låta konversationen fortlöpa. Stort inflytande av P4's svar.* *P4 uppger att applikationer oftast uppdateras dagligen.* *P4 beskriver processen att uppdatera en applikation som väldigt simpel.* *Enligt P4 kan varje utvecklare själv sätta sin kod i produktion.* *P4-41 Applikationer uppdateras dagligen* *P4-42 Det är enkelt att uppdatera en applikation.* *P4-43 Alla utvecklare kan själva sätta sin kod i produktion.*
98.	**Och då är det på något vis, i och med att utvecklarna ansvarar för**

	produkten så känner inte du att man behöver sätta upp gates från att skydda dom från att riva systemet i och med att dom själva ansvarar för att det dom gör fungerar?
99.	Ja, ungefär så. Är det större grejer som involverar.. nätverksgrejer och sådär.. Då förs det ju diskussioner kring hur vi skall göra det där. Alla utvecklare kör i princip en testmiljö lokalt också. Innan dom rullar ut grejer skall det ju vara ordentligt testat. Men det är ju upp till varje utvecklare i princip, eller åtminstone upp till varje grupp utvecklare. Sen reviewar dom varandras kod och testar varandra kod och så. Så exakt den processen kan jag inte.
100	*P4 berättar att Operations endast är involverade i stora deployments som påverkar nätverk med mera. Då sker diskussioner kring hur det ska gå till.* *P4 uppger att testmiljöer finns lokalt för alla utvecklare.* *Kod reviewas alltid av kollegor i samma team.* *P4-44 Operations är inte involverade i mindre deployments som inte påverkar nätverk.* *P4-45 Testmiljöer finns lokalt för alla utvecklare.* *P4-46 Kod reviewas av kollegor inom teamet.*
101	**Så det finns inga specifika testgrupper eller testroller?**
102	Nej, det finns inte. Vi har inga testare utan alla utvecklare testar.
103	*P4 Uppger att alla utvecklare testar och att ingen specifik testroll finns.* *P4-47 Alla utvecklare testar själva sin kod.* *P4-48 Det existerar ingen test-roll på företaget*
104	**Testmiljöerna, är det någonting du sätter upp?**
105	Det är lite olika. Delvis är det det, testmaskiner som kör remote på våra sajter. Men vi har också.. Alla har lokala testmiljöer också i princip. Dom kan köra dom applikationerna dom utvecklar lokalt på sin maskin. På det sättet gör dom det.
106	*P4 uppger att det existerar testmiljöer remote på deras sites, men att de flesta utvecklare själva har lokala miljöer för sin applikation.* *P4-49 Vissa testmiljöer körs remote.* *P4-50 Utvecklare har lokala testmiljöer för sin applikation.*
107	**Skiljer sig.... ?**
108	Ja det kan det faktiskt göra. Vissa prylar måste testas i en miljö som är mer lik den skarpa än vad man kan producera på sin lokala maskin.

109	*P4 berättar att vissa tester ej går att köras lokalt utan måste köras remote.* *P4-51 Vissa tester måste köras remote.*
110	**Men det varierar?**
111	Ja, precis, det varierar. Men de allra flesta tester körs lokalt hos utvecklaren.
112	*De flesta tester körs lokalt hos utvecklaren.* *P4-52 Tester körs oftast lokalt.*
113	**Verktyg.. Vad använder ni för verktygslåda för detta, från test till kommunikation och...**
114	I huvudsak är det två.. Vi använder Jira för att tracka allting och så använder vi Git och GitHub för att liksom.. Där all kod checkas in och som används för liksom review och där vi har möjlighet att rulla tillbaka saker om det går fel och så. Så Jira och GitHub skulle jag säga är de två främsta. De två stora. Dessutom bygger vi alla grejer på Jenkins. Där kör vi massa tester vid byggerna, dom är jag inte superinsatt i exakt hur det funkar. Där kör vi massa automatiserade tester liksom vid alla byggen.
115	*P4 uppger att de använder ett ärendehanteringssystem för att spåra allt som sker. De använder också SCM-system för att hantera olika versioner, rollbacks och för att reviewa varandras kod.* *För att automatiska bygga och deploya kod används Jenkins.* *P4-53 Ett ärendehanteringssystem används för att spåra alla aktiviteter.* *P4-54 SCM används för versionshantering.* *P4-55 SCM används för rollbacks* *P4-56 SCM används för code reviews* *P4-57 Jenkins används för att automatiskt bygga och deployakod*
116	**När det går från test till produktion, hur ser den processen ut?**
117	Som jag sade så händer det i princip varje dag för varje applikation, som jag tror. Oftast så väljer vi att uppgradera våra grejer kanske inte när det är som mest utan snarare senare på eftermiddagen, nu om någon timme kanske så kommer nog en massa uppdateringar att köra. Och då har utvecklarna testat koden först då lokalt, och sedan skickat den till GitHub för review, så har andra utvecklare kollat att det ser okej ut och eventuellt testat också. Sedan committas den och så bygger vi nya applikationer och rullar ut, ofta då lite senare på eftermiddagen.
118	*P4 uppger att uppdateringar oftast körs på eftermiddagen då systemen inte är så belastade.* *Koden har då testats lokalt och blivit reviewad i organisationens SCM-*

	system. *P4-58 Uppdateringar i applikationer sker oftast då systemen är lågt belastade.* *P4-59 Kod testas lokalt innan den deployas.* *P4-60 Kod reviewas av kollegor innan deployment.*
119	**Den där utrullningen... Är det att ni klistrar in det nya ... eller sker det automatiskt också?**
120	Njaa, det sker inte automatiskt, men de flesta ... ??? har flera hostar att köra på, eller flera servrar. Så då funkar det oftast så att vi har två vilande servrar och två aktiva servrar. Då kör vi upp den nya applikationen på de två vilande, och sedan när den kör så klippar vi över så att kunder hamnar på den applikationen istället. Ungefär så. Sen är det lite olika beroende på vad vi uppgraderar. Vissa grejer kanske vi måste uppdatera söndag natt för att det är kundpåverkande, men i princip alla tjänster vi har skall kunna uppdateras utan någon kundpåverkan alls. Dom kan vi uppgradera när som helst för att vi har flera körbara instanser.
121	*P4 uppger att organisationen har flera körbara instanser av sina applikationer aktiva och flera vilande instanser. Vid uppdateringar uppdateras de vilande instanserna och trafiken kopplas sedan över från de aktiva.* *P4-61 Flera körbara instanser underlättar uppdateringar.* *P4-62 Flera körbara instanser innebär låg kundpåverkan vid uppdateringar.*
122	**Du sade att du lade upp dina uppgifter i Jira för vad du skall göra och när och så.. Är det du själv som bestämmer vilka arbetsuppgifter som skall ingå där eller trillar det ner mycket direktiv?**
123	Det är liksom förfrågningar från utveckling och förfrågningar från marknad kanske.. att nu är det en kund som vill ansluta till oss på det här viset.. Eller förfrågningar från supporten att dessa grejer verkar inte funka, är det problem med någon nätverks grej där eller? Kanske 40% eget initiativ och sedan spritt från övriga delar av organisationer, kanske 60% skulle jag säga.
124	*P4 uppger att det är möjligt att själv prioritera när han skall utföra sina arbetsuppgifter. En stor del av arbetsuppgifterna kommer som direktiv från andra avdelningar.* *P4-63 Direktiv kommer oftast från andra avdelningar och inte uppifrån.* *P4-64 Det är möjligt att själva prioritera sina arbetsuppgifter.*
125	**Är det svårt att hålla en balans där och hålla alla nöjda samtidigt?**

126	Ja, lite så är det ju. Det är rätt mycket tankeverksamhet som går åt till att väga "hur viktigt är det här? vad ger det för kundnytta?" eller sitter vi och gör grejer som jag inte borde göra, typ? I hela organisationen är det ansvaret inte särskilt toppstyrt utan det ligger på personnivå i princip i hela organisationen. Ganska mycket i alla fall. Det du gör, är det relevant? Borde du göra det? Är det kundnyttigt?
127	*P4 uppger att mycket tid går åt till att prioritera arbetsuppgifter. Kundnytta ligger i fokus vid prioritering.* *P4 uppger att väldigt lite är toppstyrt inom organisationen utan att ansvaret ofta ligger på personnivå.* *P4-65 Tidskrävande att prioritera sina arbetsuppgifter* *P4-66 Kundnytta i fokus vid prioriteringar.* *P4-67 Platt organsation, ej toppstyrt.*
128	**När det ligger på en sådan person-nivå, hur viktig är personligheten då?**
129	Det är nog ganska viktigt. Sen tror jag det är viktigt men... Nu är min organisation superliten (Ops egen anm.), det är bara jag egentligen, men annars är det viktigt med liksom.. Alla jobbar ju i team, det gör jag också men de andra i mitt team gör inte samma sak som jag men annars tror jag det är superviktigt med möten där man sätter upp prioritet och sådär. Det skulle kunna bli spretigt om precis alla gjorde vad dom kände för. Men det är ändå lite ditåt vi vill sträva tror jag. Vi har tre ledord: "Enkelt, roligt och relevant", så allt vi gör skall vara enkelt, roligt och relevant. Är det inte det ska vi helst inte göra det. Nu tappade jag bort mig lite...
130	*P4 uppger att personligheten är ganska viktig i en DevOps organisation.* *P4 uppger även att det är väldigt viktigt med möten där man sätter upp prioritet för vad som skall utföras.* *P4-68 Personlighet ganska viktigt inom DevOps.* *P4-69 Det är viktigt med möten för att prioritera vad som bör göras.*
131	**Det var just vikten av personligheten...**
132	Ja, precis, precis. Det är klart att ledorden är ju också ganska luddiga.. Det kan man ju tolka lite hur man vill. Så jag tror det är viktigt med kommunikation.. I en DevOps organisation där det är lite olika kompetenser i samma grupper kanske.. Då blir det ännu viktigare kanske, så att inte utvecklarna sitter och lägger två månader på någon cool pryl som ändå ingen använder. Det har dom som sitter i supporten mycket bättre koll på. "Det är fan ingen som frågar efter det här, varför sitter du och gör det? Gå och gör något vettigt istället!"
133	*P4 uppger att kommunikation är viktigt inom DevOps. Att ha spridda kompetenser i varje team ses som viktigt.*

	Med möjlighet att själv prioritera sina arbetsuppgifter finns det en risk att utvecklare lägger tid på något med låg kundnytta. *Kommunikation med supporten hjälper utvecklarna att förstå kundens behov.* *P4-70 Kommunikation viktigt inom DevOps* *P4-71 Spridda kompetenser inom team viktigt.* *P4-72 Med eget ansvar finns risk att utvecklare jobbar på projekt med låg kundnytta.* *P4-73 Kommunikation med support hjälper utvecklare att förstå kundbehov.*
134	**Händer det?**
135	Kanske inte till den extremen, men delvis händer det nog. Jag tror att den kommunikationen är superviktig, och svår att prioritera! Vad är det egentligen som är viktigt? Är det här viktigt om ett år?
136	*P4 uppger att det är svårt att se framtida kundbehov och att detta kan vara ett problem vid prioriteringar.* *Kommunikation mellan Support och Dev är viktig men svår att prioritera.* *P4-74 Svårt att se framtida kundbehov kan ge upphov till problem vid prioritering.* *P4-75 Kommunikation mellan Support och Dev är viktig.* *P4-76 Kommunikation mellan Support och Dev är svår att prioritera.*
137	**Är det svårt att sia om det eller?**
138	Det är svårt att sia om det ja, och det finns olika.. Säljarna har kanske vissa önskemål, att "vi skulle kunna sälja mer om vi gjorde såhär" och supporten är inte så intresserade av att sälja massa nytt, utan dom vill kanske snarare att grejerna ska funka så att folk inte är tokiga och ringer in och klagar på allting. Så det är klart att det finns olika önskemål från olika håll!
139	*P4 uppger att säljare och support styrs av olika intressen. Säljare vill kunna sälja bättre produkter och support är mer intresserade av stabila och fungerande produkter.* *P4-77 Säljavdelningen vill ha nya och bättre produkter.* *P4-78 Supportavdelningen vill ha stabila och fungerande produkter.* *P4-79 Support- och säljavdelningen har olika mål.*
140	**Men man vill att det skall finnas någon slags liknande uppfattning om**

	saker och ting..?
141	Ja, det skall ju finnas någon slags balansgång.. och kan man liksom komma överrens i grupp om att det här är det man borde göra, och att det är relevant, så tror jag att det är det bästa.
142	*P4 uppger att beslut bäst tas genom överenskommelse i grupp.* *P4-80 Beslut tas med fördel genom överenskommelse i grupp.*
143	**Sade du att det är någon särskilt personlighetstyp som krävs för DevOps, eller?**
144	Nää, det vet jag inte om jag sade.. Jag tror inte det. Eller det gjorde jag kanske. I sådana fall vet jag inte riktigt vad jag menade.. Men det är nog.. Det krävs ju liksom att man kan kommunicera på olika nivåer med personer som kanske inte är lika insatta i det man håller på med. Det skulle jag säga är nästan en av dem viktigaste förmågorna att ha i sitt jobb.
145	*P4 uppger att det är viktigt att man kan kommunicera på olika nivåer med personer som inte är lika insatta i ens kompetens och arbetsområden. Detta anges vara en viktig förmåga i jobbet.* *P4-81 Det är viktigt att kunna anpassa sin kommunikation till mottagaren.*
146	**Har du jobbat i en DevOps organisation tidigare?**
147	Nej, egentligen inte. Nej det har jag inte. Jag har varit här ganska länge, och innan dess så har jag typ jobbat som frilans och så, så jag har inte jobbat i en större organisation tidigare. Så det har jag inte!
148	**Hur vet ni att DevOps fungerar bra för er? Har ni något mätverktyg?**
149	Nä, det tror jag inte att vi har. Även innan "DevOps" kom upp så tror jag att företaget hade fattat att den kommunikationen och kunskapsspridningen.. Att det är vettigt. Genom trial and error ungefär. Men inte vad jag vet. Sedan finns det ju folk som jobbar mer med organisation lite allt... Men jag tror inte heller att dom har någon data på att det funkar bra liksom. Det är nog snarare att det känns bra och folk tycker att det är kul. De anställda är nöjda.
150	*P4 uppger att DevOps föll sig naturligt för organisationen som sedan tidigare haft ett starkt fokus på öppen kommunikation och kunskapsspridning.* *P4 uppger att det är svårt att mäta hur väl DevOps fungerar i organisationen men att alla verkar nöjda.* *P4-82 DevOps naturligt för organisationer med öppen kommunikation.* *P4-83 DevOps naturligt för organisationer med stor kunskapsspridning.*

	P4-84 DevOps är svårt att mäta. *P4-85 DevOps upplevs som uppskattat bland de anställda i organisationen.*
151	**Tror du att alla här har ungefär samma bild av DevOps?**
152	Nae, det är jag inte säker på. Jag tror att det kan variera och jag tror att jag kanske har funderat på det lite mer än gemene man, men jag har ju inte funderat färdigt på vad det innebär, tror jag inte. Och jag tror att det är väldigt många som inte har funderat alls särskilt mycket på det. Så det finns nog ganska olika bilder av det. Av vad DevOps är?
153	*P4 uppger att det inte finns någon homogen bild av vad DevOps är i organisationen.* *P4-86 Otydligt vad DevOps innebär.* *P4-87 Olika bilder av vad DevOps innebär existerar.*
154	**Det kan inte finnas någon problematik i att...**
155	Jo, det kan nog finnas problematik i att det blir luddigt. Att ansvarsbitar blir lite luddigare, "vem tar ansvar för det här?" det tror jag. Jag tror att det kan finnas en sådan problematik. Jag tror att fördelarna väger över ganska tungt. Men visst, absolut, om alla hade en supertydlig roll, att vi ska göra just det här och just det här, då blir ju ansvarsbiten tydligare... Också lättare om högsta chefen berättar precis vad alla ska göra.. Det är också enklare, men jag tror att det skulle bli sämre. Vi skulle liksom producera mycket sämre grejer.
156	*P4 uppger att ansvarssroller blir luddigare med DevOps och att detta kan vara problematiskt. Fördelarna med fritt ansvar väger över tungt.* *P4 uppger att direktiv oftast inte kommer uppifrån, vilket gillas.* *P4 tror att en platt organisation ger bättre produkter.* *P4-88 Otydliga ansvarsroller kan vara problematiskt.* *P4-89 Otydliga ansvarsroller har fler fördelar än nackdelar.* *P4-90 Direktiv kommer inte uppifrån vilket uppskattas.* *P4-91 Platt organisation ger bättre produkter.*
157	**Ni ser en högre kvalitet att jobba med DevOps?**
158	Ja, det tror jag. Absolut.
159	*P4 uppger att DevOps ger en högre kvalitet på arbetet.* *P4-92 DevOps ger högre kvalitet på arbetet.*

160	**Du sade att det var svårt att se vad DevOps som sig gjorde för er rent specifikt.. Men kan du se på något sätt om du har gjort ett bra jobb eller inte?**
161	Alltså vid någon slags överlämning av...
162	**Om du går hem och känner att "det här gjorde jag bra...".**
163	Ofta handlar det i sådana fall om att.. Jag strävar ju egentligen att typ automatisera bort mig själv. Det är ungefär mitt mål, att jag inte ska behöva ha något att göra. För väldigt mycket av dem grejerna jag gör går att automatisera ännu mer, och jag kanske skulle kunna lägga över ännu mer grejer på utveckling eller på andra om det är tillräckligt enkelt att göra och att det är automatiserat. Så lyckas jag med sådana grejer och egentligen lyfta bort en arbetsbörda, då är jag ganska nöjd! Sen ska det inte bara vara att jag skyfflar över det på någon annan utan det skall vara för att den är förenklad eller automatiserad. Det skulle jag nog säga, att då är jag mest nöjd.
164	P4 uppger att han automatiserar bort sin grad i den grad det är möjligt. P4 skapar automatiserade verktyg och ger dessa till Dev för att Operations ej skall behöva utföra onödigt arbete. P4-93 Operations arbete automatiseras bort när det är möjligt. P4-94 Operations tillhandahåller automatiserade verktyg till Development.
165	[43:49] **Du är utvecklare från början..**
166	Ja
167	**Skulle du säga att du har nytta utav det i din roll nu?**
168	Ja det skulle jag säga, absolut! Jag utvecklar ju ganska mycket. Det är nog många som har ungefär min roll men som sitter i liksom Windows Server och typ klickar på grejerna, men jag, i princip allt jag gör scriptar jag och jag sitter sällan i GUI'n med knappar och så. Så då har jag väldig stor nytta av att kunna .. amen ut och skriva, ofta python grejer. Men ibland större java-grejer.
169	*P4 uppger att utvecklarkunskaper är nyttiga i Ops-rollen.*
170	**Många andra, de är utanför F1 då?**
171	Ja precis! Så tror jag att det ser ut på många andra ställen. Jag tycker att jag har en väldig nytta av att jag kan utveckla. Jag har också lättare att se utvecklarnas.. Både vad det finns för potential av vad den här grejen skulle

	kunna göra. Eller vad som är väldigt svårt att åstadkomma på applikationsnivå. Det tror jag också är en fördel att kunna se det.
172	*P4 uppger att det är lättare att förstå utvecklarnas behov när man förstår hur de arbetar.* *P4-95 Insyn i andra roller ökar förståelse.*
173	**Angående kunskapsspridning, är det din uppgift att lära ut uppgifter som vanligtvis hade hört till operations, till medarbetare?**
174	Ja, det vill jag säga! De försöker vi nog sträva efter mer och mer. Vi vill ju liksom sprida i princip all kunskap och liksom jobba bort… Vi har ju upptäckt att det är jättedåligt med spoffar. Alltså folk som sitter på kompetens som bara den kan.
175	*P4 uppger att organisationen vill sprida kunskap och sprida nyckelkompetens.* *P4-96 Kunskapsspridning är viktigt.* *P4-97 Nyckelkompetens skall jobbas bort.*
176	**Nyckelpersoner?**
177	Ja! Nyckelpersoner! Historiskt.. Det är väl naturligt när man är kanske lite mindre. I företaget finns bara en som kan just det här. Men i och med att vi växer så är det ju superviktigt att det inte bara är jag som kan göra det här, eller liknande. Så jag ser det som jätteviktigt för mig att sprida den kunskapen som jag har. Annars som jag sa, antingen då automatisera bort dem grejerna som jag sitter och gör och dokumentera det, eller göra det tillräckligt enkelt så att någon annan kan sätta sig in i och kunna göra det också.
178	*P4 uppger att nyckelkompetens kan vara svårt att undvika i små organisationer. Han uppger att kunskappsirdning kan ske genom kommunikation, automation och dokumentation.* *P4-98 Nyckelkompetens svårt att undvika i små organisationer.* *P4-99 Kunskapsspridning kan ske genom kommunikation.* *P4-100 Kunskapsspridning kan ske genom automation.* *P4-101 Kunskapsspridning kan ske genom dokumentation.*
179	**Nu är du ju utvecklare från början men, känner du att det är vice versa? Att utvecklare lär dig saker också?**
180	Ja! Verkligen! Absolut! Vi har massor av nytta av varandra! Det är jättemycket synergieffekter. Så det skulle jag verkligen säga!
181	*P4 uppger att avdelningarna får synergieffekter av att kommunicera med varandra, och att detta ger upphov till lärande.*

	P4-102 Avdelningarna får synergieffekter genom att kommunicera med varandra. *P4-103 Kommunikation mellan avdelningar skapar lärande.*
182	**Ifall alla ska kunna lite av varje, känner du förtroende för att kvalité upphålls?**
183	Jag tror.. Jag tror inte att det är något problem! Men att det kräver att man diskuterar grejer mer, att folk tittar på varandras jobb. "Vad gjorde du här" och liksom reviewar varandras jobb. Men jag tror inte att det ska betyda, eller behöver betyda, att det blir sämre kvalité. Snarare tvärtom, att det sprids kunskap så att fler kan. Då blir det också fler synpunkter, fler kritiska ögon. Så jag tror snarare tvärtom.
184	*P4 uppger att utvecklarna reviewar varandras kod, och att detta sprider kunskap och ökar kvalitet.* *P4-104 Code reviews ökar kvalitet.* *P4-105 Code reviews ökar kunskapsspridning.*
185	**Kan det bli så att, ifall alla har en insikt och alla reviewar, kan det leda till att arbetet inte går frammåt för att många har någonting att säga till om?**
186	Mmm! Den risken finns. Att det blir längre ledtider liksom. OM jag är superexpert på den här grejen och det är bara jag som kan det, då går det ju kanske lite snabbare för mig att göra det och jag kan bara skita i vad alla andra tycker för jag vet ändå bäst. Än att tre andra ska ha synpunkter också. Så visst det kan nog ta längre tid i vissa fall.
187	*P4 uppger att code reviews kan vara tidskrävande.* *P4-106 Code reviews kan vara tidskrävande.*
188	**Det beror på alltså?**
189	Ja, det tror jag beror på. Men jag tror ju att kvalitén blir bättre.
190	*P4 uppger att code reviews ger högra kvalitet på mjukvara.* *P4-107 Code reviews ger högre kvalitet.*
191	**I en sådanhär miljö, är det mer att man har flera bollar i luften?**
192	Nää jag vet inte om det skiljer sig egentligen. Sen har man ju.. Allihopa skulle jag säga, det nämnde jag nog tidigare, att prioritering kan vara ett problem och är något som man hela tiden måste tänka på. Gör jag, springer jag på rätt boll.. Eller jonglerar jag med helt fel grejer. Men jag är

	inte säker på att det är fler bollar i luften. Jag tror inte det.
193	*P4 upplever att det kan vara problematiskt att ständigt priortera vilka arbetsuppgifter som skall utföras.* *P4-108 Prioritering av arbetsuppgifter viktigt och svårt.*
194	**Vad skulle du säga DevOps för dig?**
195	Det är lite klurigt (skratt) Amen jag tror.. Att det är liksom en metod och .. för att få fler.. liksom ett bredare synsätt på vad vi gör. Alltså mer input från olika håll. Liksom en bredare vy istället för att titta på sin egna grejer. Ja, ungefär så skulle jag nog säga att jag ser på det.
196	*P4 beskriver DevOps som en metod för att få ett bredare synsätt på vad man gör. Han tror att DevOps ger mer input från olika håll på det man håller på med.* *P4-109 DevOps metod för att få bredare synsätt på sitt arbete.* *P4-110 DevOps ger mer input från olika håll på sitt arbete.*
197	**Skulle du säga att det går att ha DevOps även utan de här verktygen?**
198	Som.. njaa (osäker). Jag tror det är viktigt med olika typer av liksom kommunikations verktyg om man inte bara är tre personer som ses, som sitter brevid varandra, så tror jag det är superviktigt med kommunikationsvägar. Vi har ju.. De verktygen som jag sa tidigare.. Det jag glömde var att vi har en chatt, vi chattar ju vet inte hur många chattmeddelanden vi skickar om dagen (antydan på stor mängd, egn. anmärkning), men många tusen liksom i olika grupper.
199	*P4 uppger att det är viktigt med verktyg för kommunikation, och att de på företaget använder sig av en chatt där alla kan vara med.* *P4-111 Verktyg för kommunikation viktigt.* *P4-112 Gemensam chatt främjar kommunikation.*
200	**Vad använder ni då?**
201	Då använder vi vår egen. Den är ju en av dem funktionerna som vi utvecklar (F1 produkt, egen anmärkning), så vi använder det. Alla ringer och chattar och så i våra egna verktyg. (54:32)(Visar telefonappen) (Visar meddelanden över hela skärmen)

	Nu är det någon som käkar glass! (Skratt) Men… ja! Våra egna verktyg använder vi.
202	*P4 uppger att de själva använder ett egenutvecklat chattverktyg.* *P4-113 Organisationen använder sig av egenutvecklad chatt.*
203	**Vad är det svåraste med att jobba med DevOps?**
204	Kanske att det är kommunikationen också! Att trots att vi kommunicerar jättemycket så tror jag att man skulle kunna göra det ännu mycket mer. Ja det skulle jag tro!
205	*P4 ser kommunikationen som den största utmaningen när man arbetar med DevOps.* *P4-114 kommunikation den största utmaningen inom DevOps.*
206	**Är du aktiv där (på chatten)?**
207	Ja det skulle jag säga att jag är! Delvis! Men visst, det är på ganska olika nivå. Det är ju många.. Jag är inte så aktiv och skriver om glass! Ibland har vi fått ha såhär, “Nu måste ni lugna er med det där”, för det blir lite för mycket skitsnack. Men det är också.. Jag tror ändå att fördelarna väger över, det är ganska trevligt om det finns glass så är det rätt nice att det är någon som säger till att det finns.
208	*P4 uppger att det är lätt att det blir mycket “skitsnack” med öppna kommunikationskanaler.* *P4-115 Öppna kommunikationskanaler kan ge upphov till överflödig kommunikation.*
209	**Tänker på det här med context-switcha. Jag vet ju själv när jag sitter hemma och man har en chat uppe så måste man kolla den hela tiden..**
210	Jo jag håller med! Ibland är det ett problem för mig med. Men chatten kanske inte är så. Kanske att det snarare är ett problem.. Och det kommer ifrån att vi är liksom en.. Kanske delvis en DevOps, för att vi är en DevOps organisation, att det är rätt mycket spring. Folk är borta och frågar varandra grejer hela tiden. Jag får ju mindre gjort ibland för att någon sitter bredvid och snackar om någonting som inte jag har att bry mig om egentligen, men jag kan inte låta bli att lyssna. Och vi har ju pratar mycket om såhär öppet kontorslandskap som vi sitter i.. Att det finns rätt mycket,

	det finns supermycket fördelar men det finns också nackdelar.. Och där är chatten liksom likadan. Det är skitsvårt att låta bli att kolla, för att det kommer upp en studsande gubbe där nere, eller..
211	*P4 upplever att det är mycket spring och prat i arbetsmiljön och att det har både för och nackdelar. Kan ibland vara distraherande.* P4-116 Mycket spring och prat i öppna landskap P4-117 Kommunikation kan vara distraherande
212	**Är det därför du tog hörnet då eller?**
213	(skratt) Ja precis! Jag har ju bästa platsen ! (skratt) Tidigare satt jag uppe på Operationsavdelningen (supporten, egen anm.), mitt i den och det var ju värdelöst. För det första att alla som sitter där uppe pratar ju i telefon hela tiden och jag pratar ju ganska sällan i telefon. Sen jättemycket spring! Men de håller ju på att riva väggar och sånt nu så jag ska flytta alldeles strax igen. Så jag blir av med det jag anser är bästa platsen på kontoret. (skratt) Tyvärr!
214	**Ser du några generella utmaningar med det här arbetssättet?**
215	Ja det finns nog flera stycken. Jag tror att det kräver lite större personligt ansvar. Det tror jag. Men det kanske snarare handlar om att vi inte är så hierarkiska än att det är en DevOps miljö. Det är ganska få detaljbeslut som kommer uppifrån som går ner i organisationen. Men nu när jag tänker efter så vet jag egentligen har med DevOps att göra. Vad frågade du, ifall jag ser några utmaningar?
216	P4 uppger att DevOps kräver större personligt ansvar, mycket pga. deras platta organisation/hierarki. P4-118 Kräver stort personligt ansvar P4-119 Platt organisation P4-120 Direktiv kommer sällan uppifrån
217	**Ja**
218	Det kanske också är problem att, men det tror jag är en positiv utmaning, att det kräver att folk breddar sig lite, inte bara gör det som de kanske snöat in på, utan titta bredare. Det kan nog finnas motstånd åt det, att man tycker det är trist eller jobbigt för att jag kan de här grejerna svinbra, så ska jag försöka göra massa annat också. Det är klart att det kan vara en utmaning, men jag tror att i det stora att det är en positiv utmaning.

219	*P4 uppger att DevOps kräver bredare kunskaper och individer som är mottagliga för dessa kravdessa krav. Anställda som känner sig duktiga och säkra på en sak måste gå ur sin comfort zone och lära sig fler saker. P4 ser detta som en utmaning, men en positiv sådan.* P4-121 DevOps kräver breddning av kunskap P4-122 Anställda ägnar sig aldrig åt bara en specialiserad uppgift
220	**Stort tack du!**

Appendix H

Intervjuguide inför intervjuer med **F3**.

Intervjuguide D (P5, P6 & P7)

Bakgrund	☐ Hur länge har du arbetat på F3? ☐ Hur länge har ni arbetat med DevOps? ☐ Är ni en typisk DevOps Organisation?
Roller	☐ Hur skulle du beskriva din roll här? ☐ Är du Dev eller Ops eller lite av varje? ☐ Hur tydliga ansvarsroller skulle du säga att ni har?
Ansvar	☐ Hur skulle du beskriva dina och andras ansvarsområden? ☐ Vad är typiskt för Ops respektive Dev i en DevOps-miljö? ☐ Vem bär ansvaret för testning och QA? ☐ Vem bär det största ansvaret för produkten/koden i produktionsmiljö?
Insyn	**Delivery Pipeline** ☐ Har ni en Delivery Pipeline, Continuous Delivery? ☐ Hur är din inverkan på/påverkar du den? som (Ops) eller (Dev)? (Dev) har du tillåtelse att commita och sätta kod i produktion? (Ops) Hur väl skulle du säga att du *vet* vad de andra i kjedjan gör? ☐ Vad är din uppfattning av vad personer involverade i er Delivery Pipeline gör som inte du utför? **Gruppöverskridande Samarbete** **(kommunikation)(kunskapsdelning)** ☐ Hur ser den övergripande kommunikationen ut till andra avdelningar? ☐ Hur bör det se ut enligt dig? ☐ Vad ser du som problematiskt? ☐ Skulle du säga att det sker ett samarbete mellan Dev och Ops? ☐ Utför ni projekt ihop? Kommer de till dig om råd eller ber dig visa dem hur saker ska göras? **Kunskapsspridning/Feedback** ☐ I er dialog, sker det någon form av kunskapsspridning/feedback mellan avdelningarna? ☐ Har du ett exempel? ☐ Kan du mer om Utveckling/Operations idag än du kunde tidigare?

	☐ Har det hjälpt dig? ☐ Vad delar du med dig av? ☐ Vid en deployment, hur ser kommunikationen ut då? ☐ Vem ska notifieras, godkänna, vara med innan knappen trycks? ☐ Hur ser ni på proaktivitet och förebyggande åtgärder? ☐ Får ni på (Dev) (Ops) feedback från kund? ☐ Hur får ni tillgång till kundfeedback? (Om oviktigt) Vilken feedback ser ni som mest viktig? ☐ Mäter/utvärderar ni ert arbete/DevOps? **Organisationsstorlek/struktur/branch** ☐ Ser du att DevOps är begränsat till en viss sorts organisation? ☐ Påverkar fysiskt avstånd mellan Dev och Ops? ☐ Hur tas beslut? ☐ Hur ser du på DevOps i nya respektive etablerade företag? ☐ Kan branchtillhörighet påverka hur DevOps ser ut? ☐ Vad ser du som problematiskt? **Personligheter** ☐ Ser du att person det krävs en viss personlighet för att arbeta med DevOps/i en DevOps organisation?
Agilt	**Prioritering** ☐ Hur prioriterar du ditt / ni ert arbete? Dvs. vilka uppgifter prioriteras? ☐ Kommer prioriteringar utifrån eller inifrån? ☐ Hur tycker du att sådan prioritering fungerar? **Planering** ☐ (Om OPS) Hur involverad är du i planering av produkter? ☐ (Om Dev) Hur involverade är Ops i planering av släpp? ☐ *Vad är din åsikt om det?*
Tillit	☐ När alla ska kunna lite av varje, känner du ett förtroende för att kvalitet uppehålls? ☐ Hur garanterar ni kvalité?
Verktyg / CD	☐ Vilka DevOps-verktyg använder ni? ☐ Går det att ha DevOps utan dessa verktyg?

Appendix I

Intervjutranskript och nyckelpoängskodning av intervju med **P5**.

1.	Sen så skulle ni träffa P6 och P7 va? De sitter i samma organisation, eller i min organisation, som arkitekter så att det är de som är liksom hjärnan eller hjärnorna bakom den här Continuous Delivery-kedjan som vi sätter upp.
2.	**Men ni jobbar i samma team?**
3.	Vi jobbar i samma team, jag är chef för ett team som heter Trunk och Release Readiness, det är två olika team. Där trunk, det är ett väldigt konstigt namn, men det är väl liksom stammen, main branch. Trunk är det teamet som är ansvariga för att integrera nya verktyg i Continuous Integration- och Continuous Delivery-kedjan. Så det är väl de som från början plockar upp ett verktyg som .. för att lagra source code eller för att lagra artefakter eller för att bygga .. de plockar upp de verktygen och integrerar dem i maskineriet. Medans Release Readiness är mer ett team som supportar utvecklingsorganisationen med hur man använder de här verktygen och monitorerar så att byggena går bra. Och sen har vi även i mitt team något som heter release managers som är de som .. man kan säga "gate keepar" eller vaktar release branchen innan man ska släppa ut den till kund så att det inte kommer in något strunt där. Och sen så ta den hela vägen till vi levererar den ut från R&D ut till de som sen skickar det till kund. Så min organisation, vi skickar ingenting till kund utan vi lägger upp en färdig artefakt och säger till en kundorganisation: "här har ni den, ladda ner den".
4.	*P5 uppger att en roll inom organisationen har till uppgift att se till att en release ser bra ut och inte innehåller något “strunt” innan den ska släppas till kund* P5-1 Release Managers ansvariga för godkännande av release ut mot kund.
5.	**Och de plockar ner den själva eller?**
6.	Ja, precis och så ser kundsupport-organisationen till att distribuera ut det till slutanvändaren. Och det är en jättestor organisation där jag har liten eller ingen insikt alls hur det går till när produkten väl går ut till kund. På F3 är vi ungefär 2500 anställda, 500 inom R&D, och sen i alla fall 1500 in sälj och säljsupport och kundsupport och sånt där.
7.	*Från att P5 lämnar över till kundorganisationen saknar P5 insyn i hur distributionen går till. Kundorganisationens stora storlek betonas.* P5-2 Kundsupport bär ansvaret att distribuera en release till slutanvändaren. P5- Insyn saknas från distributionsansvarig till kundsupport.
8.	**R&D är det Dev då?**
9.	Det är Development, ja precis. Så Research and Development. Men lejonparten utav R&D är ju rent utvecklingsarbete.

10.	**Så ni plockar upp den första biten utav kedjan så att säga, och sen så är det några som tar vidare den då eller?**
11.	Ja, alltså vi ser ju till att produkten bygger till en färdig produkt som är färdigtestad och sen så lagrar vi den på ett artefakt-repository där sen andra kan använda den.
12.	*P5 uppger att avdelningen denne tillhör ansvarar för att produkten är färdig och färdigtestad* P5- Release Readiness ansvarar för att produkten är färdig och färdigtestad.
13.	**Skiljer det sig från kanske någon annan sån här, exempelvis någon som har en webbtjänst, utan det är snarare att den publiceras någonstans så att någon laddar hem den så att säga?**
14.	Vi har alla möjliga varianter höll jag på att säga. Vi har ju en ren cloudlösning, som en SaaS-offering som heter QlikSense Cloud. Det är en .. den snurrar på amazon web services som en SaaS offering. Och då deployas den produkten kontinuerligt liksom. Den ska ju egentligen deployas, på varje commit så ska du uppgradera den, våran SaaS offering. Riktigt så automatiserat är det inte just nu utan..
15.	*P5 uppger att QlikSense Cloud produktionssätts kontinuerligt. Ambitionen är att varje commit ska uppgradera produkten, men i dagsläget har den graden av automation ej ännu realiserats.* P5- Cloudlösningen produktionssätts kontinuerligt. P5- Cloudlösningens produktionssättning är ej fullständigt automatiserad.
16.	**Men det är tanken?**
17.	Ja, det är tanken och den uppdateras ju, alltså flera gånger i veckan. Så det är en offering, det är en SaaS-offering. Och sen så finns det även en enterprise och en desktop. Desktop är ju den som man kör på liksom en standalone, laptop eller desktop. Och sen så har vi enterprise, enterprise är det som de flesta stora företag installerar i sin, i sitt privata nät eller private cloud. Och båda de två, både desktop och enterprise, de paketeras som liksom i exekverbara filer. Såna man sitter och trycker next next next på, den typen av installationer, medan cloud-produkten eller den här SaaS offering, där byter man ju ut enskilda komponenter liksom, där .. det finns ju bara en cloudlösning, det är den som alla använder i hela världen, sen så skalar den ju ut. Men det finns bara, det finns inte multipla sådana utan det finns ju en. Och där har alla komponenter samma version. Om du då committar kod, du bygger, sen paketeras det och sen så byter man ut den komponenten i molnet.
18.	*P5 uppger att flödet skiljer sig mellan cloudlösningen och desktoplösningen. Cloudlösningen byter enbart ut komponenter och kräver inte samma distribution som desktoplösningen.* P5- Releaser i cloudlösningen sker genom utbyte av komponenter. P5- Cloudlösningen kräver inte samma distribution som desktoplösningen.
19.	**Skiljer man på flödena mellan SaaS-tjänsten och det kompilerade som folk laddar hem?**

20.	Ja, vi gör det faktiskt i vårat företag. Men det är nog mest utav geografiska skäl. Att de som jobbar med cloud inte sitter här. Och sen så finns det kanske lite andra historiska skäl. Men det är ju, deployment-mekanismerna är ju lite olika. Det ena är ju en installable, en paketerad fil liksom, och det andra är docker-containrar som hot swappas. Så att de flödena skiljer sig ganska mycket.
21.	*P5 uppger att cloud har annorlunda produktionsflöde än desktop, mycket på grund av att utvecklarteamen sitter geografiskt separerade.* P5- Cloudtjänsten och desktoptjänsten har olika flöden för produktionssättning. P5- Cloudtjänsten och desktoptjänsten har geografiskt separerade teams.
22.	**Det var av geografiska skäl som man ..**
23.	Ja, alltså att det är så hos oss, att de här flödena är så parallella, det är nog mycket av geografiska skäl. Att vi råkar sitta på olika ställen. Annars så tror jag man skulle kunna ha väldigt lika flöden, men vi har ju vissa beröringspunkter mellan enterprise desktop-flödet och cloud-flödet och det är ju att vi använder ju samma artefakter, alltså det är ju samma kod som paketeras på olika sätt. Så att de flödena går ju ihop. Men sen går de också isär när de ska paketeras inför installation.
24.	**När man säger geografisk skäl, är det att man behöver vara nära för att ha en sån kedja, att man måste kunna ha tillgång till varandra?**
25.	Nej, det tror jag inte. I vårat fall så, att vi inte har harmoniserat de här flödena helt och hållet det beror nog på att tekniken för flödet är olika och det gänget hos oss som jobbar med cloud-deployment har hamnat i Kanada. Och sen har vi desktop och enterprise deployment som sitter här i Lund. Väldigt företagsspecifikt.
26.	P5 uppger att flödet för produktionssättning mellan produkterna skulle kunna vara mer lik, men inte fullständigt. Skillnaden beror till stor del på de geografisk separerade utvecklingsteamen. P5- Skillnader produktionsflöde mellan cloud och desktop beror till stor del på geografiskt separerade utvecklingsteam.
27.	**I ditt team, hur delar man upp ansvar? Vad som ska utföras och vem som får committa och.. finns där någon slags hierarki, att ni hade release managers för att släppa ut det, men finns det någonting som ..**
28.	Nej, utan det är. Man gör ju en pull request på sina commits och sen får ju någon plocka upp det liksom. Det får ju granskas och tas in.
29.	P5 uppger att kod granskas innan den släpps in i master branch. P5- Kodgranskning sker innan release.
30.	**Det granskas utav alla eller?**
31.	Utav någon skulle jag vilja säga. Vi har inte riktigt samma, vi använder inte Gerrit till exempel, så att vi har hårda regler på att det ska vara två som sätter plus ett liksom för att det ska få lov att komma in utan det ska granskas innan det går in. Man kan tänka

	sig att vi skulle kunna styra upp det bättre men så är det inte idag. Jag tänkte, du kanske funderade på hur vi .. gäller det bara liksom committen för att få in sin kod eller gäller det även hur vi bestämmer vem som gör vad i dagligt arbete?
32.	P5 uppger att företaget saknar strikta regler för granskning av commits, men att en commit ska granskas av någon innan det går vidare i processen. P5- Kod granskas innan en commit går vidare.
33.	**Ja, det är också intressant.** [hur de bestämmer vem som gör vad i dagligt arbete, egen anm.]
34.	Ja, för vi jobbar med en projektstyrningsmodell som heter Kanban som ni säkert känner till. Då har man ju.. man har sin .. nya taskar som kommer in, kommer in överallt ifrån, kommer från utifrån organisationen, de internt, de kommer från mig, de kommer från min chef. Så lägger vi in dem nya liksom, sen kommer de in i backlogen och då har vi bestämt oss för att göra någonting med dem, sen går de in i in progress och sen kommer de till done. Och det viktigaste för oss är att ha en väldigt bra prioriterad backlog, så vi försöker grooma den här backloggen heter det då när man omprioriterar backloggen en gång i veckan eller varannan vecka. Sen jobbar vi i en sånhär Kanban board, vi använder ett verktyg som heter Jira som är extremt vanligt.
35.	*P5 uppger att F3 använder sig av Kanban för prioritering och visualisering av arbetsuppgifter och arbetsflöde. Omvärdering av prioriteringar sker varje eller varannan vecka* P5- Kanban används för prioritering av arbetsuppgifter P5- Kanban används för visualisering av arbetsuppgifter. P5- Viktigt att omvärdera prioriteringar veckovis.
36.	**Hur prioriterar ni vad som ska göras? Kommer det ovanifrån eller via möten eller..**
37.	Hos oss är det jag och arkitekten, så det är antingen jag och Jan eller jag och Christoffer som sitter och gör en prioritet. Sen så .. min chef har ju, han kommer med input till mig vad han vill att vi ska göra och i vilken ordning men då tar jag med det till det mötet, så att det är jag som prioriterar.
38.	*Prioriteringar sker i första hand top-down där ledare och chefer anger vilka uppgifter som ska prioriteras.* P5- Prioriteringar av arbetsuppgifter sker i första hand top-down.
39.	**Finns det något utrymme för utvecklare själva att göra några slags prioriteringar?**
40.	Ja, absolut det tycker jag nog att det finns men då får man ju göra sin röst hörd. Oftast är det ju så att de utvecklarna som sitter hos mig som lägger sina egna backlog-taskar i backloggen liksom som, bara för att vi ska få ordning på .. få koll på vad vi gör, kunna göra en plan och kunna prioritera det mot allting annat vad vi har att göra. Och sen så gör de sin röst hörd och säger att "jag tycker detta är viktigt", då prioriterar vi det.

41.	*P5 uppger att det finns möjlighet för utvecklarna själva att påverka prioriteringen av arbetsuppgifter, men då framför allt genom att göra sin röst hörd till en överordnad om vad utvecklaren i fråga anser vara viktigt att prioritera.* P5- Utvecklare kan själv påverka prioritering av sina arbetsuppgifter.
42.	**Hur är det med de här Kanban, är det bara ni inom avdelningen som ser det eller brukar andra avdelningar gå in och kolla vad ni håller på med?**
43.	Vi har ett antal olika projekt liksom, vi har ett eget projekt men alla projekten är öppna. Så vem som helst som har access till Jira inom R&D kan också gå in och se hur vi jobbar. Och det händer ju att jag, när de undrar "varför händer det ingenting med våran ticket? Vi har sagt att vi behöver hjälp med detta", ja så skickar jag tillbaka "ja, detta är vår backlog och så här ser det ut och .. " ja så försöker jag förklara. Så det finns inga hemligheter. Sen så, jag sitter ju inte och tittar i andra Kanban-board än mina egna liksom, det känner inte jag att jag har tid och ork till. Men det finns inga hemligheter, så vill man så får man.
44.	*P5 uppger att Kanban-projekt är öppna och tillåter insyn från alla anställda inom R&D. Att gå in och titta på andras Kanban är ovanligt, men det är i vissa fall användbart att kunna hänvisa till relevant Kanban-projekt i diskussioner.* P5- Kanban-projekt är öppna för alla utvecklare. P5- Möjligheten att se andras Kanban-projekt utnyttjas ej frekvent. P5- Möjligheten att se andras Kanban-projekt upplevs som värdefullt.
45.	**Nu var SaaS i Kanada ..**
46.	Ja, alltså de som utvecklar SaaS-lösningen sitter i Kanada. Men samtidigt .. det de gör, det är ju framför allt att de wrappar en enterprise-produkt. De wrappar ju och paketerar komponenter som kommer överallt ifrån liksom. De tillhandahåller ett ramverk som de här komponenterna ska liksom in i. Så att produkten, även den som finns som en SaaS-lösning, den är ju till 90% gjord här i Lund även om själva cloud-teamet sitter i Ottawa i Kanada.
47.	**När denna är uppe och säg att något går ner, vem är det som plockar upp den bollen?**
48.	I SaaS-lösningen?
49.	**Ja**
50.	Vi har ett cloud operations team som är utanför R&D som är de som skall monitorera och supporta. Så att de har ju second line support i cloud operations och sen så har man third line support i R&D. Det kan ju ändå vara så att det är vi som rättar felen i produkten, men det är inte vi som monitorerar den.
51.	*P5 uppger att cloud-lösningen har ett cloud operations team som är separat från R&D. Monitorering samt first och second line support sker här, medan third line support utförs av R&D.* P5- Cloud-lösningen har ett operations team som är separat från Utveckling

52.	**Hur ser en sådan kommunikation ut? Hur får de den kunskapen att kunna lösa produktproblem?**
53.	Tänker du hur cloud-teamet kan lösa produktproblem om de är i de komponenterna de har använt?
54.	**Ja**
55.	Då kommer de felen att rättas utav komponentleverantör. Så att om det är en komponent som kommer ifrån .. vi har ju kontor i Italien och på Island och här liksom. Är det fel i en komponent som kommer ifrån italien, då kommer italien att rätta det i sin komponent. Och då använder vi Jira för att dokumentera vad som behövs rättas, rättningarna och vi routar även problemen vidare via Jira.
56.	*P5 uppger att när fel upptäcks av operations så delegeras uppgiften att lösa felet till komponentleverantören. Jira används som stöd för dokumentation och delegering i problemlösningsprocessen.* P5- driftproblem delegeras till komponentleverantör P5- Jira används för att dokumentera lösning av problem i koden P5- Jira används för att delegera uppgiften att lösa problem i koden
57.	**Man använder Jira för att dokumentera också?**
58.	Vi använder Jira för att dokumentera kravställningen på produkten, alltså vi skapar liksom produktscenario, det som adderar värde. Och sen bryter vi ner det till .. så småningom blir det subtaskar som går ut på en individuell medarbetare. Så att det går från kravställningen från produktledning som i ett produktscenario, så bryts det ner och i slutändan så är det taskar som går ut på medarbetare men man kan se kopplingen liksom. Jag adderar värde .. vi kommer sälja den här featuren X och jag måste göra detta, detta och detta och hon eller han gör detta, detta och detta. Och tillsammans kommer vi sen kunna leverera produktscenario X som vi kan ta betalt för.
59.	*P5 uppger att JIRA används för dokumentation och att kundvärde är i fokus vid utvecklingen.* P5-Jira används för dokumentation P5-Kundvärde är i fokus vid utvecklingen.
60.	**Så den som gör subtasken där längst ner, den vet också vad de andra gör?**
61.	Det kan man enkelt se. Om de är intresserade utav det det vet jag inte, men man kan se vilka andra saker som måste göras för att man ska kunna realisera produktscenario-värdet som man sen kan ta betalt för.
62.	P5- Jira möjliggör insyn
63.	**Så ni har alltid ett värde för slutprodukten som är satt innan projektet är börjat som ni jobbar för att ..?**
64.	Ja, det är väldigt viktigt att vi har koll på vad det är som genererar värde för slutanvändaren innan vi börjar utveckla. Så det ska finnas ett tydligt värde, man ska

	addera värde.
65.	*P5 uppger att värde är i fokus innan utveckling startar.* P5-Kundvärde i fokus vid priorteringar i utvecklingsarbetet.
66.	**Hur utvärderar ni ifall ni har skapat värde?**
67.	Det kan faktiskt inte jag svara på, jag har ingen speciell insikt i hur det går till
68.	P5- Insikt saknas i hur värdeskapande utvärderas
69.	**Utvärdering av teamet, hur sker det i en chefsposition?**
70.	Jag kan svara på hur jag tittar på det. Man kan mäta vad man har för throughput i sin Kanban-board, liksom se hur mycket taskar man lyckas exekvera. Vi gör ju ett estimat på de här taskarna, hur mycket vi tror att de ska ta. Och sen så kan vi mäta hur många taskar vi får igenom. Men jag funderar inte så mycket på att mäta performance i mitt team faktiskt. Utan jag tänker mig att jag har med vuxna människor att göra som känner sig engagerade och intresserade så att då kommer de att .. om de kan se värdet utav det de gör så kommer det att ordna sig av sig själv.
71.	*P5 uppger att inget fokus ligger på att utvärdera hans team.* P5-Ingen inre utvärdering av team.
72.	**Det här med intresse, är det en viktig del för er?**
73.	På vilket sätt tänker du ..?
74.	**Tänkte på det du sa innan om att i och med att de visar intresse och ansvar så försvinner lite behovet av att mäta.**
75.	Ja, alltså det är ju .. ja, absolut, det är klart att det är jätteviktigt att de brinner för vad de gör. Då får man ju ut mycket mycket mer. Är det roligt att jobba så får man ju ut mycket mycket mer. Man kan ju inte gå med piskan och säga att nu ska vi leverera utan det ska vara roligt att jobba, det är då man får ut mycket.
76.	P5- Behovet av att mäta sjunker när anställda tycker det är roligt att arbeta
77.	**Det är något ni värderar?**
78.	Ja, F3 som företag är ju väldigt måna om att F3 ska vara en attraktiv arbetsplats. Man jobbar ju supermycket med det och skryter mycket med att man blivit vald till liksom .. jag vet inte vilken plats vi var på i år men vi har alltid haft en toppnotering där på best place to work, så att vi har varit liksom femte bäste företaget att jobba på, åttonde bästa och så här. Det verkar lite som att vi går neråt i och för sig men det är ändå, då mäts man alla företag i Sverige. Inte mjukvaru enbart utan liksom som är motsvarande stora. Så att F3 jobbar väldigt mycket på att det ska vara roligt att gå till jobbet.
79.	**Kanske lite mer bakgrundsfråga, hur länge har ni haft ett DevOps-initiativ här?**
80.	Det kan jag inte riktigt svara på, för att det är före min tid. Jag kom hit här i höstas och

	då var det ganska väletablerat med DevOps.
81.	**Har du arbetat med DevOps tidigare?**
82.	Nej det har jag inte gjort utan jag har varit chef inom annan .. kvalitetssäkring och så där.
83.	**Hur ser du på kvalitetssäkring inom DevOps? Hur skiljer det sig, om det gör det alls?**
84.	Alltså, det som är .. jag kan ju inte svara för all DevOps, men det vi gör här, alltså jag är ju mer ansvarig för att tillhandahålla ett Continuous Integration-, Continuous Delivery-maskineri. Och då är ju DevOps mer att jag ska tillhandahålla möjligheten att köra testning i en tidig fas, alltså jag ska tillhandahålla möjligheten att köra komponenttester och integrationstester och kanske första systemtest och så vidare. Så att jag levererar inte testfall, jag ser bara till att maskineriet funkar liksom. Om teamet själv har satt upp att "de här testfallen ska passera och man ska få lov att gå in med sin komponent", ja då ser vi till att de testfallen körs. Men innehållet i den sviten, det bryr inte jag mig om. Utan det är upp till teamet själva att sätta kvalitetskrav på sig själva. Tids nog kommer vi få betala priset ändå, om de släpper en komponent hela vägen in i en slutprodukt liksom då blir det bara jobbigare för dem. Det bör finnas ett incitament hos teamen att säkra kvalitet i tid och vi tillhandahåller ramverket för att göra det.
85.	*P5 uppger att hans jobb är att ansvar för tillhandahållningen av CI och CD. I det ingår testning.* P5- Organisationen jobbar mot CI P5- Organisationen jobbar mot CD
86.	**Kommunikationen med operations och om ni har några slags verktyg eller metoder att hålla den kommunikationen uppe, eller ge insyn eller ..**
87.	Ja Jira är ju ett verktyg, men vi jobbar väldigt mycket med ett verktyg som heter Slack för att kommunicera. Så att vi har ju kanaler där vi publicerar information dagligen liksom. Är det några problem med byggkedjan så publicerar vi information om det. Gör vi några förändringar så publicerar vi information om det. Plus att vi jobbar med, faktiskt fortfarande med mail, man skickar ut liksom "vi ska göra en uppgradering av detta verktyget i helgen så att de här API:erna kommer att förändra sig" då publicerar vi det via mail också om det är stora saker. Annars är det mycket Slack, och sen har vi ett verktyg som är integrerat med Jira som heter Confluence där vi bloggar och dokumenterar mycket av de förändringar vi gör. Förändringar i arbetssätt och förväntningar på development-team och sen publicerar vi det i Confluence.
88.	*P5 uppger att Slack, ett chattverktyg, används för kommunikation. Slack används för att informera andra avdelningar.* *Dokumentation sker i Confluence via JIRA.* P5-Chatverktyg används för kommunikation. P5-JIRA-verktyg används för dokumentation.
89.	**Dagligen?**

90.	Slack är många gånger, alltså det är på minutbasis, det är ett enormt flöde. Men vi försöker också blogga på confluence om de förändringar vi gör.
91.	**Är det så att det blir väldigt mycket kommunikation, är det bara inom teamet eller är det hela bolaget eller?**
92.	Vi kommunicerar ut till hela bolaget eftersom vi tillhandahåller ju .. vi är ju litegrann hjärtat i organisationen liksom som tillhandahåller liksom integrationsmöjligheter, vi är ju integrationspunkten i företaget så att, det är ju klart, vi kommunicerar ju med alla. Och det kan ju vara så att vi ser att den ena komponenten har ett beroende av den, men de själva sitter itne så nära varandra. Så att vi är liksom en spindel i nätet, så vi kommunicerar med en stor del av företaget.
93.	*P5 uppger att att Release Readiness är "spindeln i nätet" och kommunicerar med en stor del av företaget.*
94.	**Ni jobbar efter kundfeedback, för att skapa ett värde för kunden om jag har förstått det rätt. På vilket sätt får ni feedback på det?**
95.	Nej, jag vet inte riktigt hur den feedbacken kommer tillbaka utan den kommer ju till en produktledning. Så att vi får ju kravställningen från produktledningen och sen kommer den ju tillbaka som feedback till produktledningen igen. Och hur det går till mellan kund och produktledning, kundsupport och organisation det känner jag inte till.
96.	*P5 uppger att det är otydligt hur feedback från kund återkopplas.* P5-Otydligt hur kundfeedback återkopplas.
97.	**Stort tack!**

Appendix J

Intervjutranskript och nyckelpoängskodning av intervju med **P6**.

1.	**Hur länge har du arbetat på F3?**
2.	På F3? 8-9 år nu. Började för ganska länge sedan. Då vi var ganska små också.. DevOps fanns då också skulle jag vilja säga, men du gjorde vi det utan att riktigt vara medvetna om det.
3.	**Kände ni att "det här gör vi" när fenomenet dök upp?**
4.	Ja, typ. Det var liksom "jag behöver fixa bygget" eller jag "jag behöver fixa miljö". Men vad tycker ni ingår i DevOps förresten?
5.	**Det har varit en intressant fråga om man ser tillbaka till undersökningen, men vi har väl satt en definition för undersökningen där vi säger att DevOps är en strategi för att uppnå frekventa, snabba och säkra leveranser. Vad tycker du om denna definitionen?**
6.	Det är intressant, för jag har inte hört den dra hela vägen till leverans om jag ska vara ärlig. Eller min uppfattning har aldrig varit så. Men det är ju klart att DevOps är där för att, i slutändan, kunna leverera mjukvara. Med stabilitet och repeatability. Så det tyckte jag var en bra beskrivning faktiskt.
7.	**Vad är DevOps för dig?**
8.	Ja.. DevOps för mig är kanske det jag sitter dagligt med.. Det är build-script, miljöer för att stödja bygget eller task-kedjan, konfiguration... Det är dom puckarna framför allt.
9.	*P6 uppger att DevOps för honom involverar byggscript, miljöer för att stödja bygget och konfiguration.* P6-1 Byggskript är en del av DevOps. P6-2 DevOps involverar miljöer för att stödja byggen
10.	**Vad är din roll på Företaget? Hur ser din titel ut.. Är det utvecklare eller..?**
11.	Min titel är väldigt generisk, jag är arktitekt för ett gäng som heter release readiness. Våran uppgift är att stötta resten av organisationen när det gäller releasearbete. Och där ingår ganska många puckar.. Det är att tillhandahålla system som stödjer releaseprocessen.. När vi pratar releaseprocessen då är jag med på din beskrivning, jag hade kallat den min releaseprocess som ni beskriver DevOps. Det är alla system som stödjer flödet för att kunna leverera ett teams leverans, och det behövs en viss typ av infrastruktur där, och DevOps. Byggsystemet, byggmiljö, testmiljö.. Sen tillkommer fler realeaseförfaranden...

	Jag sitter med de system och de stödmetoderna för att vi kan leverera oavsett vilken typ av leverans det är.
12.	*P6 uppger att hans avdelning stödjer Development med verktyg..* P6-3 Utvecklarna stödjs av verktyg för produktionssättning
13.	**Om vi nu tar definitionen, är det någonting du skulle vilja lägga till där för att mer passa din uppfattning med det där dagliga?**
14.	Nu är jag inte riktigt med.
15.	**Om vår definition passade releaseprocessen, känner du att det saknas en komponent till den definitionen, att man skulle lägga till något för att mer passa?**
16.	Nej, Det är mest att dom blir ungefär samma sak. När ni sade att ni tycker att DevOps är den här säkert ta mjukvara till leverans, det jag ser som releaseprocess, inte som DevOps-process... Jag ser DevOps som en stödjande aspekt till releaseprocessen. Så ni har ett bredare scope än vad jag hade haft för DevOps. Jag säger inte att det är fel, jag har bara inte tänkt så mycket.
17.	**Vi säger inte att vår är rätt heller...**
18.	Det borde ni göra. Det finns många olika definitioner.
19.	*P6 uppger att det finns många sätt att tänka kring DevOps.* P6-4 Det finns många sätt att se DevOps.
20.	**Det här stödjande.. Vad skulle du se som det man måste trycka mest på för att kunna stödja?**
21.	Pratar ni nu om realeaseprocessen eller DevOpsprocessen eller en blandning?
22.	**Mer DevOps som helhet. Hur går det till?**
23.	Mycket är väl att se behovet. Det är verkligen där vi ska börja, det är inte bara att hacka script. Speciellt när organisationen börjar växa. Då är det ofta att en snabb lösning känns bra men inte är det långsiktigt. Sen kommer nästa person, och nästa team, och nästa chef... Så har man en likadan lösning och helt plötsligt skalar den inte. Så det är ofta att förstå grundproblemen. Hur kan vi applicera något som andra kan återanvända? Och det spelar inte i väldigt många DevOps-frågor skulle jag säga. Hela vägen från byggmiljön, till vårat byggsystem till teststrategin och hur DevOps skall stödja dom.
24.	*P6 uppger att det är viktigt att se behovet hos andra avdelningar. Och att skalbarhet för verktygen är viktigt.*

	P6-5 Viktigt att se behov hos andra avdelningar.
25.	**Hur får ni förståelse för deras behov?**
26.	Jag ska inte påstå att vi är fantastiskt duktiga på det. Men.. dom kommer att springa.. Nu kanske jag ska backa ett steg tillbaka, för nu låter det som att jag och mitt gäng äger DevOpsprocessen.. Det gör vi inte, vi äger releaseprocessen, och där ingår mycket DevOps som vi måste göra själv eller få stöd från ett annat gäng som heter "Trunk". Dom sysslar mycket med configuration management och verktyg, som också stödjer DevOps. Däremot så hamnar mycket av DevOpsaktiviteterna nuförtiden hos teams och enskilda utvecklare. Dom ska själva åtminstone kunna bygga egna bygg-skript i byggsystemet, och dom ska själva kunna konfigurera egna jobb i byggmiljöerna, Jenkins och TeamCity om det är någon.. Ofta när man får en sådan problembeskrivning som jag började prata om, att man skall försöka förstå grunderna i att tillhandahålla något man skall försöka fungera till många, så handlar det snarare om mer konceptuella begrepp. Okej, vi behöver ta mjukvaran från code till binära. Det finns många sätt att göra det, och vad vi gör är att vi tillhandahåller ett byggsystem, ett standardsätt för att transformera det. Men den faktiska koden som ingår är något som ett team skall äga själv. Är ni med på vad jag menar? Tillbaka till din fråga, hur förstår vi? Vi tittar på utvecklingsprocessen framför allt, vad behöver vi av kod som skrivs någonstans, hela vägen till mjukvara som har business värde ut till kunderna. Inne i den processen finns det ett antal stora boxar, och det är dom vi försöker stödja med. Och dom går in i dom här grejerna, som byggsystemet, byggmötena, testmiljöer.. configuration management, issue-hantering. Den typen av grejer. Då kommer vi hela vägen.
27.	*P6 uppger att de kunde varit bättre på att förstå varandras behov och att det är viktigt.* *Jenkins nämns som ett verktyg för byggmiljö.* *Business-värde är viktigt att se i allt man utvecklar.* *P6 uppger att de tillhandahåller verktyg för continuous delivery-processen.* P6-6 Viktigt att se behov hos andra avdelningar. P6-7 Jenkins används för byggmiljöer. P6-8 Viktigt att se business value i allt som utvecklas. P6-9 Verktyg för automatisk byggning tillhandahålls utvecklare
28.	**Er organisation.. stödjer den resten av verksamheten?**
29.	Ja det är en stödorganisation inne i organisationen.
30.	**Hur ser er kontakt ut med utvecklarna? Pratar ni ofta och diskuterar behov och så?**
31.	Ja det gör vi hela tiden. De är alltid uppe hos oss eller så är vi nere hos dem. Eller andra kommunikationskanaler såklart. Det finns väldigt aktiva grupper i.. Vi anv änder ett verktyg som heter Slack för vår kommunikation. Där inne har vi specifika kanaler för.. Det heter inte DevOps men det är många av dom frågorna helt enkelt ..

	Hela vägen från "hur löser vi den här conceptual problemet" till "bygget fungerar inte". Då kommer man ofta in med expertkompetens för att försöka hjälpa dom att komma vidare.
32.	*P6 uppger att kontakt mellan hans organisation och utvecklare är viktig och ofta sker face-to-face.* *Ett chattsystem används för att snabbt kunna kommunicera problem mellan avdelningar.* P6-10 Kommunikation mellan avdelningar viktigt. P6-11 Face-to-face kommunikation mellan avdelningar viktigt. P6-12 Ett chattsystem (Slack) används för kommunikation mellan avdelningar.
33.	**Ni som stödorganisation, jobbar ni mycket med kunskapsdelning på det sättet att ni lär dom hur det går till eller..?**
34.	Vi borde jobba mer med det om vi säger så.
35.	*P6 uppger att det är viktigt att jobba mycket med kunskapsdelning men att det kunde göras mer.* P6-13 Det är viktigt med kunskapsdelning.
36.	**Okej.**
37.	Jag tycker personligen det iaf. Vi är ofta ganska bra på att komma fram till dom där jättefina konceptuella lösningarna och skapa verktygen för att lösa den där konceptuella pott.. Vi bara låter den naturligt rulla ut. Och det är inte alltid det bästa. Då kommer folk börja arbeta det fast vi inte vet hur och sedan faller vi i problemområde. Jag tycker att vi måste vara mycket bättre på kommunikation.
38.	*P6 uppger att kommunikation mellan avdelningar är viktigt och kunde göras bättre.* P6-14 Kommunikation mellan avdelningar viktigt.
39.	**Har du någon tanke om hur man bäst skulle lösa det?**
40.	Jag tycker att de flesta grejerna börjar med ett kommunikationspaket. Vad har ni ändrat eller introducerat och varför? Vad är grundproblemet. Det är väldigt viktigt, man skall ha en kommunikationsplan. När den träffar verkligheten så finns det ett nytt behov, och det är de praktiska aspekterna. Hur använder man faktiskt den här förändringen? Nu förstår jag att det kommer en ändring, och varför. Men hur använder jag den? Och då börjar man med eller.. Det finns många sätt, det kan finnas training, det har vi ibland när man tar en extern eller intern .. av människor som behöver någon slags djupare kompetens. Champions är något vi använt, det tycker jag är jättebra. Det innebär att olika delar av organisationen får utpeka någon som skall vara lite mer kunnig. Och då stödjer vi dem jättemycket i början och pratar jättemycket med dem i grundfasen. Det hjälper på olika sätt: För det första får vi organisationen intresserad av förändringen, för då

	bidrar dom hela tiden. Och då ger dom kompetens, både om förändringen men också om det vi faktiskt implementerar samtidigt som vi implementerar det. Så, feedback till oss, en kompetens till dem, en förståelse för dem, en förståelse till dem.. Mycket lättare att träffa organisationen när man går ut med en kommunikationsplan också, såklart, för det är redan delvis ute. Så det här är en stor skillnad mot att låta något naturligt rulla sig själv ut och att ha en kontrollerad kommunikationsplan där man byggt ut ett stödnätverk med vissa människor sitter ute redan. Fan vad jag babblar. Det är bara att säga "tyst" om… *skratt*
41.	*P6 ser kommunikationspaket som en lösning på att öka förståelse och sprida kunskap.* *Intern utbildning av de anställda är viktigt.* *“Champions”, någon som i ett tidigt stadie får ta del av och utvärdera verktyg, används för att sprida kunskap och ge feedback till de som utvecklar verktygen.* P6-15 Kommunikationspaket ökar förståelse. P6-16 Kommunikationspaket sprider kunskap. P6-17 Champions ger feedback till de som tillhandahåller verktyg. P6-18 Champions medför kunskapsdelning.
42.	**Väldigt intressant med champions. Var de så de kallades?**
43.	Det är det vi använder som begrepp i vårat fall ja.
44.	**Okej, så man har en Champion i olika team som man bidrar till för att de sedan skall sprida detta inom sina team?**
45.	Ja, och det är guld för oss att få feedback också. Kontrollerat. Vi träffar dom veckovis eller varannan vecka. Det här sker oftast för stora förändringar, det är kanske inte så kostnadseffektivt om man gör det för alla förändringar. När man skall gå från nått stort till nått stort så är det väldigt väldigt bra.
46.	*P6 uppger att Champions används för att ge tillbaka feedback.* *Champions är framför allt viktigt vid stora ändringar.* P6-19 Champions ger feedback till de som tillhandahåller verktyg. P6-20 Champions bra vid stora ändringar.
47.	**I mötena för feedback.. Är det ett möte bara för feedback eller är det att dessa championsen svarar till er i en slags hierariskt kedja och rapporterar till er?**
48.	Det kan finnas sådana aspekter också. Det beror lite på vilken fas man är i. I början är det mycket "vi gör såhär, och såhär långt har vi kommit", har ni feedback på detta? För att se om organisationen kan hantera dom här grejerna. Så får man både positiv och negativ feedback. Då innebär det att vi kan justera våran strategi på väg till implementation. Sen när man har börjat rulla ut saker och ting, då är det viktigt att få mer "Hur går det? Berätta" och inte bara vad vi behöver förändra. ??
49.	*P6 uppger att Champions ger insyn till de som tillhandahåller verktygen till utvecklarna.*

	P6-21 Champions ger insyn till utvecklarna åt de som tillhandahåller verktygen.
50.	**Och detta, det är att ni träffas för att att prata om det?**
51.	Ja.
52.	**Dom här championsen.. behövs det en viss personlighet, att man ser att det behöver vara en viss typ av person för att kunna bli en champion?**
53.	Det har jag inte funderat så djupt på. Ofta plockar man folk som redan är delvis intresserade av ämnet. Det är väldigt mycket lättare för dem att bidra på det sättet vi vill. Företaget har en rekryteringsstrategi att vi ändå försöker få in folk som kan ta lite olika roller, och det innebär att om någon skall ta ett steg upp och kunna ta in tekniska och konceptuella förändringsinformation och sedan dela med sig detta... De flesta klarar det, man behöver inte fundera jättemycket på det. Det är snarare att den har intresse.. och arbetsbelastning som spelar mer roll. Nu gissar jag lite för att vara tydlig, för det är oftast inte jag som väljer ur dessa människor.. Ibland pekar vi ut folk för att vi vet att de är intresserade, och då frågar vi om den personen har tid. Men i andra fall så är det teamet själv eller organisationen själv som tar beslutet.
54.	*P6 uppger att tekniskt intresse är viktigt hos champions.* *Företaget rekryterar gärna folk med bred kompetens.*
55.	**Men är det en del av rekryteringen att man gärna ser någon som kan ta en bred roll?**
56.	Jag tycker att vi.. och nu talar jag väldigt personligt, det kanske inte är hur företaget skulle beskriva det.. hur vi rekryterar, men man är ofta ute efter människor som vill stödja varandra. Teamwork for results är en av våra core values. Där är det en viktigt aspekt, precis som jag säger, att man kan stödja teamet om det behövs. Så länge man rekryterar utifrån sina core values.. vi har fem stycken.. så får man en grupp medarbetare som kan ta lite olika roller under behov. Hur kommer ni att använda den här informationen?
57.	*Teamwork är viktigt i F3 enligt P6.* *Att kunna arbeta i team viktigt vid rekrytering.* P6-22 Samarbetsvilja viktigt vid rekrytering.
58.	**Vi kommer att använda den för att se hur organisationer ser till vad som... i förhållande till litteraturen.. hur DevOps organisationer kan arbeta. Det finns en rätt stor brist utav det för den som vill sätta sig in i det. Det är en deskriptiv studie på det sättet, vi vill beskriva..** **Har ni någon slags avdelning som kör driften för all kod på servrarna också.. drifttekniker och sådär?**
59.	Ja, det har vi. Development Infrastructure. Dom tar hand om vad vi beskriver som

	"compute". Alltifrån compute och OS, så hårdvara och OS då, men sedan så tar de inte så mycket hand om applikationslagret. Det är ofta applikationsägare. Trunk är ett typiskt gäng som sysslar med det som man skulle kunna kalla för DevOps, och där ingår applikationslagret, såsom byggservrarna, och ser till att dom... och källkodshanteringssystem, ser till att det alltid är tillgängligt och bra, sånna puckar. Så det ingår i deras puckar. Det kommer att ni höra mycket mer om när P7 kommer ner. Det är han som sitter där.
60.	*Operations har hand om hårdvara och operativsystem, men inte applikationslagret. Applikationsägare har ansvar för applikationslagret.* P6-23 Operations ansvarar inte för applikationslagret. P6-24 Applikationsägare ansvarar för applikationslagret.
61.	**Om en utvecklare har lagt till en feature och committat sin kod och skickat ut det i produktion, hur ser ansvaret ut för koden då?**
62.	Det är en jättebra fråga. Jag gillar sådana frågor. Det finns inget jättebra svar på den frågan här på F3. Det finns en väldigt bra anledning, att vi historiskt inte har skickat ut kod till produktion direkt. .. blev den integrerad till master, packeterad, skeppat till en downloadsidan och sedan såld som ett färdigt installationspaket. Om det är det du menar, åtminstone för oss, så ligger ansvaret hierarkiskt, först ligger det på R&D och sedan går det ner till teamet, beroende på var problemet ligger. Om det finns ett problem i koden iaf. Tills det finns ett problem i den delen av produkten, i den delen av koden, så är det hands off. Det behövs inget ansvar längre. Förutom kanske när man utvecklar vidare på det paketet.
63.	När man kommer mer till Cloud och Cloud Deployment så har vi mycket mer börjat med att teamet själv äger hela vägen ut till produktion, även när det kommer ut till drift. Anledningen till att det är en bra fråga är att det fortfarande är väldigt suddigt, om det fungerar i verkligheten eller inte. För vi har en blandad ... vi bygger komponenter som kan deployas ut i en cloud-miljö, men också ingå i ett installationspaket. Det innebär att det är lite svårt.. Det är definitivt svårt för ett gäng att äga en komponent som ingår i ett installationspaket och kanske kör i ett företags drift i fem år. Då har man ingen kontroll över det, alls. Men i en cloudmiljö som vi äger, där kan man ha mycket kontroll över sina komponenter. Så det finns olika gränser där. Svarade jag på frågan?
64.	*P6 uppger att ansvar för kod hela vägen till produktion finns vissa delar av organisationen där man arbetar med cloud.* *P6 uppger att det är svårt att ansvara för kod som ingår i ett installationspaket.* P6-25 Ansvar för kod sker i vissa fall hela vägen ut till drift. P6-26 Svårt att ansvara för kod i installationspaket.
65.	**Ja, software as a service möjliggör ägande hela vägen ut i princip?**
66.	SaaS blir nog något vi äger... Jag blandar alltid ihop dessa as a service, för man kan själv köra mjukvara i en cloud-miljö som SaaS, men det är fortfarande företaget som köpte mjukvaran som äger den, men de lägger ut den för andra att konsumera

	som en SaaS. Men en SaaS som vi själva äger, där han man kontroll hela vägen.
67.	*P6 uppger att kod ansvaras för hela vägen ut till drift i deras moln-miljöer.* P6-27 Ansvar för kod i drift sker i cloud-miljöer.
68.	**Hur lägger ni upp prioriteringar.. Vi hörde av P5 att ni använder kanban, men hur väljer teamet hur dem prioriterar?**
69.	Du menar utvecklingsteamet?
70.	**Mm**
71.	Det är en handskakning med produktägaren. Vi har ganska många olika produktägare som äger olika produkter. Vi har flera olika produkter, de flesta har bara hört om x och y, men vi har ett antal fler. Beroende på hur stora de är så ingår det ett antal produktägare. Dom ihop med komponentägare, eller, team som utvecklar som ordnar prioritet. Den processen stödjs av JIRA. Där kan det ingå allt från en feature request till en bugg eller en task eller vadsomhelst.
72.	*P6 uppger att prioriteringar av arbete sköts av komponentägare(team) och produktägare.* *P6 prioriteringar sköts med hjälp av JIRA.* P6-28 Prioriteringar sköts med hjälp av ärendehanteringssystem med Kanban.
73.	**Är det så att de som får tasken ser var det kommer ifrån eller måste man fråga produktägaren var feedbacken kom ifrån från början? Ifall det var en innovation inhouse eller om det var kundfeedback - är det något som når teamet?**
74.	Jag kanske är fel person att svara på det.. Gissningsvis så är det lite blandat. Vissa produktägare ger väldigt mycket information till teamet, tillochmed bjuder in kunder till exempel för att förklara problemet. Jag har sett exempel på det. I vissa fall finns det en väldigt tät koppling till problembeskrivning eller grunden till problemet.. Från kunden ofta.. Men i andra fall misstänker jag att det bara är,....Jag skulle nog säga att det är väldigt blandat.
75.	*P6 uppger att feedback till utvecklarna ofta kommer från produktägaren, uppifrån.* *I vissa fall kan kund bjudas in för att direkt ge feedback till utvecklare.* P6-29 Feedback ges ofta från produktägare. P6-30 Kunder kan bjuda in för feedback vid speciella tillfällen.
76.	**Hur ser era team ut, gällande roller och ansvar, specialiseringar?**
77.	Vårat team eller utvecklingsteamet?
78.	**Utvecklingsteamet överlag.**
79.	Utvecklingsteamet består av åtminstone en ingenjör. Men ingenjören har lite olika

	inriktningar, kan inrikta sig på test/utveckling eller utveckling.. vanlig utveckling. Beroende på vilken del av produkten så har vi olika språkkompetens, och det påverkar också testkompetens skulle jag vilja säga. I vissa team är det väldigt stark manuell testning, i andra andra är det mer api-testning, ingen UI, eller väldigt lite. Sen har man en chef eller någon slags projektledare som hjälper teamet med att prioritera och jobba mot.. Om det behövs så finns det en chef eller projektledare. Det är inte så att alla team har UX människor eller documentalist, det är snarare en shared pool som går ut till teams under behovet. Det är mest en resourcehanteringsgrej. Det har fungerat ganska bra tror jag.
80.	*P6 uppger att det finns många olika roller i ett team.* P6-31 Det finns olika roller i ett team.
81.	**Har de anställda en bredare kunskap utöver sin specialisering?**
82.	Med bredare, menar du att de kan täcka upp för de andra?
83.	**Precis.**
84.	I 2016 vill jag säga att de flesta kan någonting om det mesta. Det vanligaste gränsen går där teamet stödjer med test-aktiviteterna. Det finns fortfarande en uppfattning om att det är det enkla skillsetet i ett teams uppsättning, och då kan alla bidra. Men det är en lång diskussion om vi går in det så jag vill helst inte gör det. Men kanske om UX är upptagen och det är en lite mindre feature så kan utvecklaren ta ansvar själv där och se vad som behövs hos slutanvändaren. Sällan tror jag folk kopplar in utvecklare, men det är inte helt omöjligt.
85.	*P6 uppger att många utvecklare har insikt i varandras roller.* P6-32 Det finns gränser mellan roller, men de är ej skarpa.
86.	**Ah okej. Vi tror att tiden är slut tyvärr. Har vi en avslutande fråga?** **Hur mäter ni hur väl DevOps organisationen fungerar?**
87.	Vi gör nästan inte det skulle jag säga.
88.	*P6 uppger att det inte förekommer utbredd mätning av hur DevOps arbetet fungerar.* P6-33 Det finns ingen stark process för mätning av DevOps-arbetet.
89.	**Stort tack!**

Appendix K

Intervjutranskript och nyckelpoängskodning av intervju med **P7**.

1.	**Hur länge har du arbetat på F3?**
2.	På F3 har jag ganska prick arbetat i 2 år nu
3.	**Och du har alltid arbetat med DevOps under tiden du har varit här?**
4.	Ja det skulle jag vilja säga!
5.	**Har du någon tidigare erfarenhet med DevOps?**
6.	Detta området så har jag jobbat med sen jag började jobba på (tidigare bolag) 97
7.	**97? Det var DevOps redan då? (förvåning)**
8.	Äh, det skulle jag väl inte säga riktigt men, frågan är väl lite vad DevOps är.
9.	**Det har vi också frågat oss**
10.	Haha, det är lite så buzzword, det beror på vem man frågar. Men det är klart att man försökte komma dit hän på den tiden också. Jag började ju jobba med mobila basstationer och de är ju, för den tiden, gigantiska system. Så att bara köra någon kontinuerlig integration tar ju liksom 24h att bygga ihop en av komponenterna.. Svårt!
11.	*P7-1a DevOps är svårt att definiera* *P7-1b DevOps handlar om att integrera enklare* *P7-1c* DevOps buzzword
12.	**Vad är DevOps för dig?**
13.	Ja bra fråga, det handlar väl mer om en organisation där så många som möjligt egentligen har bra insyn och kunskaper om hela flödet utifrån krav till produktion och tillbaka.
14.	P7 beskriver DevOps i en öppen fråga i enlighet med hur vi karaktäriserar DevOps! P7-2 DevOps handlar om kunskap och insyn om hela leveransflödet
15.	**Har skapar man den insynen?**
16.	Haha, det var en jättefråga. Det har ju folk brottats med i decenier, det är ju från Mythical Man-Moth från 60 talet, det all litteratur handlar om.

17.	*Mycket skapar insyn, svårt att ge i ett långt svar.*
18.	**Du ingick i vad ni kallar trunk teamet (referens från tidigare intervjuer, P5, P6)?**
19.	Aa!
20.	**Som vi förstod från Christopher, så är ni en stödorganisation till resten av organisationen**
21.	Ja det är det ju! Jag skulle ju inte riktigt säga att vi är DevOps på riktigt som jag ser det. Det är lite för uppdelat mellan utveckling och de senare bitarna. Vi försöker komma dit. Vissa delar är det ganska mycket DevOps. Så hela organisationen skulle jag inte påstå är DevOps.
22.	*P7 anser att hela organisationen till viss del är DevOps, men ser uppdelningen mellan utvecklare och resterande är för sepererad.*
23.	**Okej!**
24.	Det finns DevOps-öar liksom.
25.	**Det finns alltså ett slags spektrum? (förklaring: spektrum till vilken grad av DevOps)**
26.	Aa, det skulle jag säga!
27.	P7-3 *En organisation kan vara gradvis DevOps*
28.	**Och ni förser utvecklarna med det som behövs för att kunna göra en *delivery pipeline?***
29.	Mm!
30.	**Okej! Vad är det ni gör på Trunk (avdelningen)?**
31.	Aa man blir lätt sådär allt i allo när man håller på med sådant där, för man får ju verkligen jobba med hela kedjan. Så det är allt från att fixa rätt sorts maskiner att köra på och sen utvärdera dem, till konfigurera nätverk och dokumentera processer och sådana grejer. Så det är ju hela vägen från typ kablar till processdokument (skratt). Men om man tittar på tiden som vi lägger så handlar det ganska mycket om att se till att vi har infrastruktur för de olika bitarna på plats, så att vi har fungerande pipeline, så att man kan stoppa in någonting i en änden så kommer det ut någonting i andra. Där emellan är vi och optimerar det flödet. Mäta och utvärdera.
32.	*Stöd grupp till utvecklingen, förser Dev med verktygen. Ops (?)* *Ops uppgift med vad P3 sa.*

	P7-4 *Ops förser utvecklare med verktyg för CD* P7-5 *Mätning och utvärdering en viktig del för att optimera leveranskedjan*
33.	**När ni utvärderar, vad är det man tittar på då? I mätningen och utmaningar?**
34.	Det är som sagt också lite av en smaksak. Rent generellt sett så gillar jag att fokusera på köer. Där det tar stopp av olika anledningar. Där ligger saker och ting och väntar på någonting.
35.	**Okej! Köer som i.. ?**
36.	Vad som helst egentligen! Det kan vara allt från liksom att det läggs issues i Jira som ingen tittar på, till att man har en feature gren som ligger och har inte blivit integrerat på ett halvår. Överallt där det tar stopp i röret egentligen. Där det ligger massa work in progress. För att mäta sådana köer hittar man ganska snabbt var man har kapacitetsproblem, eller var har du process problem.
37.	*P7 menar att man kan mäta på flera variabler och att han gillar att mäta på köer som fastnat*
38.	**Hur sker den mätningen? Är det du som manuellt får leta igenom det?**
39.	Ofta blir det ju att vi försöker pinpointa någonting. “Okej, här tror vi att det är problem”, så börjar man försöka mäta det, börja samla ihop den datan. Hur många commiter har inte blivit integrerade eller hur många issues.. Eller hur lång tid ligger issues innan utan att de åtgärdas. Mycket sådana grejer. Hur lång tid tar det från det att en utvecklare säger att den är klar tills den ändringen är i produktion. Den kölängden är ju intressant
40.	*Hur många commits som inte blivit integrerade och issues (kanban) som inte blivit gjorde ännu är ex. på mättal*
41.	**Vad har ni för minsta längd för den kön?**
42.	Ja! (tänker) .. vad kan det vara teoretiskt sätt. Kortaste tid för att få grejer i produktion är vår molnmiljö, jag tror det är den kedjan som är den snabbaste. Skulle man nog kunna klara på , om det kniper, på dagen iaf. Men ofta vågar man inte göra det, för att man kommer med förstora ändringar på en gång. (skratt). OM det har legat en featuregren och stått och jäst i ett halvår, så vill man gärna testa den. Mycket och länge! Så om någon säger: Nä nu måste vi ha in den här! Tills det att den faktiskt har blivit levererad, skulle jag säga att det gör vi inte på en dag.
43.	**Okej! För då vill man testa den lite mer. Har man tappat tilliten till det för att den inte har varit med på ett tag?**
44.	Jag skulle säga att ju större ändringar man gör, desto sämre översikt får man över vad riskerna är att integrera det. Om det är en liten ändring så kan du ju sätta vilken nisse som helst på att säga liksom: Vad kan gå sönder? Har du liksom 20 000 rader kod.. Vad kan gå sönder? (skratt) Lite vad som helst!

45.	*Större ändringar innebär större risk.*
46.	(diskussion ifall F3 använder en viss produkt för att utvärdera sina mättal)
47.	Vi använder ju mest (PRODUKT). Vi använder det för att analysera oss själva ganska mycket och analysera testresultat, byggfel och allt möjligt sånt. Men frågan är ju vad är viktigaste att titta på just nu. Tittar du på för mycket grejer så tenderar man att strunta i.. .ett enskilt mätvärde säger ingenting. Du har så många mätvärden så det går inte att styra något beteende längre. Får du en rapport där det står 40 olika parametrar, vad är det som man ska titta på, vad ska man reagera på, vad är det man ska fokusera på?
48.	*Pipelinen är svårmätbar*
49.	**Du tittade på köer, va det så?**
50.	Alla vill ha sitt eget mätvärde (skratt) Kan man väl säga!
51.	**Tänkte fråga mer om Insynen. Nu ställde jag en jättebred fråga i början, men ser du kunskapsspridning som en bidragande faktor för att skapa insyn!**
52.	Det är det ju väldigt mycket. Kunskapsspridning kan du ju ha som ett sätt för att få folk att fråga efter mer saker. Istället för att liksom, att folk bara sitter och väntar på att bli matade med insikter. Det bästa är ju om folk förstår så mycket om hela liksom. T.ex DevOps är, vad det kan vara osv? Så att fler börjar fråga efter och se problem med det lilla närmaste som man själv jobbar med.
53.	P7- 6 Kunskapsspridning får folk mer intresserade av hela leveranskedjan
54.	**Att bredda perspektivet?**
55.	Mm! Det är någonting som vi jobbar på en hel del, och jag som jobbat med sånna här grejer i 20 år snart. Det är ju fortfarande så att det här DevOps tänket inte är särskilt utbrett i de företag jag jobbat i. Utan det är öar, kan man säga! Jag tycker att det är för många programmerare! (skratt)
56.	P7-7 Kunskapsspridning skapar ett helhetsperspektiv
57.	**Okej! På det sättat att..?**
58.	Att dom inte ser hela flödet egentligen och inte vill befatta eller inte ser det som sitt problem, att inte känna till hela flödet. Utan det är från: checka ut koden, skriva några rader kod och comitta det igen. Det är flödet man vill titta på och då blir det jättesvårt att optimera hela kedjan, därför att man kan inte styra utifrån. Liksom, vad är optimalt för varje liten del, utan det bästa är om var och en kan fundera över det själv: “Vad är min del i det här flödet”? Att hur kan jag förbättra flödet genom nånting som jag gör. Om man inte vet vad det är, eller inte ens frågar sig vad det är.. Så blir det ju mycket svårare.
59.	*P7-8 Att inte se hela flödet ses som negativt.*

	P7-9 Organisationens kontroll underlättas genom kunskapsspridning.
60.	**Så där är kunskapsspridning..**
61.	Så Kunskapsspridning det handlar ju om att få folk att fundera över “Vad kan man göra, vad gör andra, de som är bra på det här vad gör dem”?
62.	P7-10 Kunskapsspridning skapar nyfikenhet
63.	**Så om en utvecklare skrivit ett par rader kod och committat den raden kod, så är det proiblemet ur världen för den utvecklaren då?**
64.	Ja det blir ju lite så, och man vet inte vad som händer. Man försöker automatisera så mycket som möjligt så att så många som möjligt ska kunna fundera på bara koden egentligen. Den ska inte behöva tänka på så mycket mera, men för att kunna komma dit, så måste man automatisera jättemycket grejer. Och om man inte funderar och tänker hur det ska gå till, så är det väldigt att det man gör passar dåligt in i flödet. Så att även efter man automatiserat så måste man fortfarande hela tiden vara på vakt att det man utvecklar passar! Eller liksom att man förändrar, att det växer ihop hela tiden. Så det perfekta flödet är ju bara perfekt just nu. Det måste man hela tiden anpassa efter den, vi har ju liksom flera hundra personer som sitter och knackar kod… Det växer! Hänger då inte både flödet med och anpassar sig efter nya och/eller växande produkter, så kommer inte det att fungera. Men det är också så att om det går att göra produkter, asså arkitektur och sånt, som är mer eller mindre bra på att passa in i ett sådant flöde. Det är inte för inte som Microservices är på modet. Just därför att det skalar bra. Du har inte ett flöde, utan du har många små. Det är lättare att få det att funka än om du bara bakar på en jätteprodukt hela tiden. För det blir mer och mer komplexitet, utan det är ett sett att hålla nere komplexiteten i en sådan pipe.
65.	*Automation för att slippa tänka så mycket, men samtidigt bör Utvecklaren ha en insyn i organisationen och ett helhetsperspektiv för att kunna skriva mjukvara som passar i leveranskedjan.* *Med en insyn och helhetsperspektiv kan man bättre reagera när saker och ting ändras.* *Arbetar med mindre och fler tjänster med lös koppling för att undvika komplexitet och använder flera pipelines för att underlätta.* P7-11 Med insyn och helhetsperspektiv skrivs bättre produkt/mjukvara P7-12 Insyn och helhetsperspektiv behövs då pipelinen ständigt förändras P7-13 Automation för att minska bördan av att veta mycket P7-14 Tjänstifierar för att minska komplexiteten (SOA?), Microservices starkt anknutet P7-15 Flera pipelines används för att minska komplexitet

66.	**Arbetar ni microservices?**
67.	Mer och mer!
68.	**Bara för att stämma av ifall min uppfattning av Microservices. Är det att man bygger team utifrån en tjänst, att de är frånkopplade?**
69.	Det finns lite olika varianter av det också ju, det kan du också hitta många definitioner på. Det du vill ha är att du har en ganska liten tjänst som täcker in ett affärsproblem, så att du verkligen kan ha ansvar i ett litet team, enda vägen från krav till kundfeedback. Men då gäller det ju att du kan utforma varje service på ett sånt sätt att det faktiskt är gör någonting som är relevant för en slutkund. Och det är inte helt enkelt (skratt) Utan många servicar är ju sånt som används utav andra servicar. Då har du inte hela spannet från krav till slutkund egentligen. Så det är väldigt svårt att göra den perfekta microservices arkitekturen
70.	Bolaget strävar mot att implementera Microservices så att teamen kan ta fullt ansvar hela vägen från utveckling till feedback P7-16 Strävar mot utvecklarteams helhetsansvar
71.	**Men man vill att det teamet som utvecklare ska ta ett helhetsansvar?**
72.	Helhetsansvar ja! Och att man sätter ihop systemet inte när man bygger eller inte ens testar utan i produktion egentligen. Om du måste testa din service i varje sammanhang den kan förekomma i, så för det mesta kommer varje service köra ungefär samma grejer mot din service, så varför skulle du testa i varje samhang. Då bygger det istället att du har väldefinierade APIer i din service och du verifierar att dem API:erna gör vad du säger att du gör. Sen blir det problem någon annans problem.
73.	P7-17a Strävar mot utvecklarteams helhetsansvar
74.	(diskussion mer inriktat på microservices) … **Hur funkar det hos er, delar flera teams på en delivery pipeline eller har de individuella pipelines?**
75.	Vi är i en vad jag skulle kalla en övergrångsfas just nu skulle jag säga. Vi har haft väldigt mycket, man ansvar för en del av koden i någonting som man delar en leverans pipeline. Men vi försöker mer och mer bryta isär dem grejerna så att det verkligen är ett team ansvar en längre bit egentligen. Om man har så långt som möjligt sin egen pipeline egentligen. Någonstans går det ihop till en större produkt, men att dela på det under en längre tid, för att hålla nere komplixiteten för enskilda team och inte behöva tänka på så mycket, utan gå mer mot kontraktstestning och man verifierar sin modul egentligen mot de API:erna som den delan ska supporta. Sen gör man en sorts mindre systemtest efter det som egentligen inte teamen behöver bekymmra sig så mycket om. Klart går det fel hela tiden kommer det ju tillbaka någon som undrar varför det går fel. Men tanken är att man ska slippa mer

	och mer bry sig om vad andra team gör så mycket som möjligt. Slippa koordiner och slippa vänta på andra. Slippa testa andras grejer.
76.	*Strävar för fler pipelines (fler team seperationer) för att team ska kunna ta hehetsansvar. P7 förklarar att man vill hålla pipelinesen så långa som möjligt för det enskilda teamet, men att de tillslut kommer mynna ut i en och samma produkt och pipeline. Detta för att team ska kunna gå vidare så fort som möjligt och slippa långa ledtider.* *P7-17b Fler pipelines något man strävar efter så långt det går.*
77.	**Känner ni att det fungerar i F3?**
78.	Delar! Och det pågår rätt mycket arbete för att göra den situationen bättre!
79.	**Du pratade innan om att det fanns olika öar av DevOps i organisationen. Vad skulle du säga det är för öar?**
80.	Det var ju sedan, när den nu gick live, vår cloud produkt. Där har vi ju verkligen microservices som kombineras med dem lite old school windows applikationer än sålänge. Men där har vi ju verkligen microservices som går ute i produktion. Inte riktigt än sålänge så att när du trycker på merge knappen på GitHub så ploppar det efter en stund upp i produktion ifall allt har gått bra. Och det teamet har TV skärm bakom med statistik från produktion och sånt.
81.	*Cloud produkt bättre anpassat för DevOps och Continuous Delivery.* *Ej fullt automatiserat.* *P7-18 DevOps team har tillgång till statistik från produktion.*
82.	**Okej, så utvecklarna ser statistiken då och de är de som är ägarn på så sätt..?**
83.	Ja vi har en Ops organisation, det behöver man på nått sätt ändån. Man behöver ju 24/7 365. Du kan inte ha ett Dev team som sitter och minitorerar 24/7 365.
84.	*Dev har ansvar, men Ops har jouren* P7-19 Dev och Ops delar ansvaret för att monitorera produkten.
85.	**Okej så Ops har jour på så sätt?**
86.	Ja! De har jour!
87.	**Om Dev har det ansvaret, men Ops har jouren.. Har Ops den kunskapen att fixa till någonting som egentligen..**
88.	Än sålänge inte riktigt så mycket som de kanske borde ha i sådana fall.
89.	*P7-20 Kunskapsdelning mellan Dev och Ops fördelaktigt för drift*

90.	**Ops har first line..**
91.	Mm..
92.	**Men Dev och Ops, är de separerade organisatoriskt eller sitter de i nära anslutning till varandra?**
93.	För de bitarna där det verkligen är Microservices, så är ju Dev teamet tar hand om ända ut till i princip blue deployment
94.	*P7-21 För att Dev ska kunna ta ett helhetsansvar krävs en arkitektur som stödjer det, exv. Microservices.*
95.	**Blue deployment?**
96.	Blue deployment, eller vad kallar Facebook det.. Canary deployment
97.	**Det var nytt för mig. Är det ute i produktion..?**
98.	Ute i produktion men man skickar inte alla användare till den nya versionen utan man tar bara en del av trafiken och skickar en del till en ny version. Man testar i produktion.
99.	(Diskussion om canary deployment)
100.	**Men de är separata (Dev och Ops)? De sitter i sär?**
101.	Ops sitter i en annan del av organisationen.
102.	*Ops sitter i en annan del, någonting P7 tidigare pekade på var negativt för DevOps*
103.	**Hur ser kommunikationen ut mellan dem? Är det någon regelbunden kommunikation?**
104.	De sitter ju på samma site åtminstonde och har telefonmöten och videokonferenser flera gånger i veckan iallafall.
105.	*Trots avståndet sker en frekvent kommunikation* P7-22 Verktyg underlättar frekvent kommunikation mellan avdelningar
106.	**Det sker ett samarbete på så sätt?**
107.	Aa det finns slack kanaler och så. Slack kanaler är att man postar feedback direkt in i en slack kanal så att man kan snacka om det där.
108.	P7-23a Chattverktyget Slack används för feedback. P7-23b Chattverktyget Slack används för direkt kommunikation.
109.	**Du anser att ni inte har DevOps fullt ut?**
110.	Det är för att vi inte kör på det viset överallt typ. Men det är också, det funkar ju bäst

	ifall du har en moln /saas produkt. För då har du en miljö att deploya till, du har helt kontroll över den och kan köra blue och canary deployment och sådana grejer. Du behöver inte ha något stort systemtest på så sätt. Det beror lite på hur mycket du vågar göra fel jämtemot användare, för det kommer ju skita sig någongång ifall du testar på användare (skratt) Så det beror lite på hur tåliga de är och så, vilka typer av kunder. Är det någonting du inte betalar för så kanske folk är mer villiga att bli utsatta för experiment. Betalar man mycket pengar så kanske man är mindre villig till det.
111.	P7-24 SaaS produkter lämpar sig bättre för delivery pipeline.
112.	**Men är det så att det lämpar sig bättre att ha ett DevOps tänk ifall man har en SaaS Produkt.**
113.	Vi har ju andra team som jobbar med andra delar av systemet som inte bara är webbprylar, utan native kod verkligen, jätteoptimerad för att vara jättesnabb. Då vill då göra det för att det är native kod.
114.	(diskussion om assembler kod)
115.	Så de bitarna är ju väldigt väldigt miljöspecifika. Det innebär ju det att, när man säljer det till kund som ska installera det i sitt eget datacenter eller någon molnleverantör någon sannansts, så är det ju väldigt beroende utav att det funkar i just den miljön som just den kunden har. Det har man ingen som helst kontroll över! T.ex. vi vill ha det på både Windows och Linux, då måste man kompilera det två gånger, du måste testa det två gånger. Sen så bygger man på lite grejer och sen sitter du med att försöka få någon IPv6 certifiering.. Ja då måste du testa det där också för att det finns vissa kunder, nu speciellt i sydostasien där IPv4 adresserna är slut... Ja, så du har inte kontroll längre. Då börjar det bli väldigt jobbigt att göra alla dessa testerna i en kontinuerlig pipeline, då den pipelinen väldigt vädligt lång eller väldigt bred, om man vill göra den hela tiden för varje liten ändring. Så man kanske, för sådana delar är det väldigt svårt att ha ett kontinuerligt flöde ut till varje målmiljö, för att det är väldigt svårt och få det att funka för alla dem samtidigt.
116.	*P3 förklarar att de flera miljöerna en produkt kan existera på innebär mer testande och mer anpassning vilket ökar storleken och stegen i pipelinen.*
117.	**Skulle man kunna säga hur testbart det är som avgör..**
118.	Ja! Hur snabbt och hur tillförlitligt du kan verifiera ändringar egentligen.
119.	**Det har en avgörande roll?**
120.	Mycket! Är det dessutom on-premise, så kan du inte göra någon green testing eller liknande.
121.	*Det är testbarheten, dvs hur väl det kan testas och testas automatiskt som kan*

	avgöra ifall en Continuous Delivery pipeline är lämpligt eller inte. Svårigheten ökar kraftigt då testerna måste anpassas till flera olika miljöer. P7-25 Testbarheten av produkten avgör hur väl en pipeline kommer att fungera.
122.	**Om du hittar ett stopp i ett flöde, hur gör du för att se till att det åtgärdas**
123.	Stopp som i..
124.	**Att ett flöde går långsamt**
125.	Det jobbigaste är ju att klura ut vilken delprocess som är viktigast, som ger mest bang-for-the-buck. Allra bästa är att försöka leta upp ställen där man gör saker som man kanske inte behöver göra. Peter Drucker sa en gång i stilen “ theres nothing quite so useless as doing effeciently that which should not be done”. Så det har jag sett många gånger man sitter “Oh det här går sakta”. Så sitter man och optimerar en process och så tänker man: “Varför i hela friden gör man det här? Eller varför ska den här personen göra det här när det sen bara ligger på hög någon annanstans” Därför att det inte finns kapacitet att ta mot det. Man hade massa sådana.. Kommer ihåg kravhanterinsprocessen på (TIDIGARE BOLAG), då hade man ändå haft ett stort projekt i ett år för att förbättra den och så skulle man rulla ut den nya kravprocessen och så skulle man gå en halvdagskurs för att lära sig hur den fungerar. Så när vi kom till slutet där vi gått igenom alla steg och började rita upp och skriva på tavlan själv för att visa att man kunde. Ett antal delsteg och så räknade vi. Så frågade vi kursledaren: “Det här review mötet, hur ofta kör man det, cheferna som ska träffas som ska ge klartecken eller inte.. Hur ofta sitter dem där?” Vi la ju ihop alla stegen och det tog ju minst 8 veckor för att komma igenom, men tiden arbete man la ner under dem 8 veckorna var ungefär en man vecka. Så varför tog det 8 veckor? Var verkligen alla de här review stegen nödvändiga, måste man batcha allt det där? Så man tyckte att man hade gjort en jättefin process det tog bara 8 veckor och sådär, otrolig effektivt. Men tittade man på det så var det ju 87% dötid. Det hände ingenting . (skratt) När jag började ställa sådana frågor var det liksom “ojdå, nä tänkte inte riktigt på det”. Det är så många som är så ovana att tänka på det viset.
126.	**Men då ritades processen om?**
127.	Ja det blev väl lite mer så organiskt arbete så småningom. Man förstod ju det att det är dumt att om man vill kapa tid här, att vänta på att.. Liksom lägga krav på hög och inte göra nått med dem 4 dagar utav 5. Då går det inte fort! Det är typiskt sånt som jag lägger tid på att försöka förklara för folk. (skratt) Det kostar att vänta! (berättelser från tidigare arbetsplatsen fortsätter)
128.	**Den problemlösningen säger man att det är DevOps eller är det mer Lean?**
129.	Det är Lean tänk med DevOps! (brister ut) Anledningen, hela liksom idée historien

	som har lett fram till DevOps är ju Lean tänket, det är ju TPS som man har anpassat för mjukvara egentligen. Det är liksom lean i mjukvara är DevOps. Det handlar ju mycket om att ta bort dem här vallarna där det samlas skräp och DevOps var ju mer en lösning på att just den här friktionen där Ops alltid ville att det skulle funka. Dev hade ingen aning om vad som hände där på andra sidan så de försökte ju trycka på.
130.	*P7 förklarar att DevOps är Lean applicerat på mjukvaruutveckling, där det inte ska förekomma några vänttider och tar exempelet att Dev traditionellt sett producerat mängder när det inte fanns kapacitet att ta mot det på Ops sidan.* *P7-26 DevOps är Lean för mjukvara*
131.	**Ops stabilitet och..**
132.	Och Dev, hastighet liksom
133.	**Är det att alla har samma uppfattning?**
134.	Njä det är ju lite olika inriktning fortfarande. Men DevOps är ju ett sätt att göra det till allas problem hur det funkar i produktion. Det är det som är intressant. Det spelar ingen roll hur snabbt du kan få fram det till Ops ifall de inte kan deploya det. Det har liksom ingen betydelse. Då är det ju det här igen, då optimerar man någonting som sen bara ligger och väntar på att komma i produktion. Så DevOps är ju ett sätt att lösa just det köproblemet. Så man ser det ju som ett problem när man börjar titta på det ur Lean perspektivet.
135.	*P7 ser att uppfattningen inom organisationen kan variera och berättar att DevOps är att göra produktionen till allas problem och att det är huvudintresset.* *P7-27 Produktionen ska vara allas problem*
136.	**Sprider ni detta i ert bolag, att man ska fostra den tanken att alla ska se..**
137.	Jag försöker, men ingen sån corporate uppmaning.
138.	**Okej! Stöd från ledning? Är det en viktig faktor?**
139.	Det måste jag säga att det är! Ska man göra det på riktigt så handlar det ju ofta om att göra stora organisatoriska förändringar och verkligen sprida ut ansvaret av vad som händer i produktion på flera händer. Det räcker inte att säga det på pappret utan det måste vara. Det handlar jättemycket om utbildning av hela organisationen egentligen för att få det att fungera bra.
140.	*P7 ser att stöd från ledning är av stor vikt för att produktionen ska uppfattas som allas problem då detta kan innebära stora organisatoriska förändringar. Menar sedan att utbildning är viktigt för att driva igenom förändringen* *P7-28 DevOps kan innebära en stor organisatorisk förändring*

	P7-29 Stöd från ledning viktigt för implementationen av DevOps
141.	**Var vänder organisationer sig för att få en sådan utbildning. Finns det expert for hire?**
142.	Tror faktiskt inte det finns.. Kanske finns några experter man kan låna in.
143.	**Tror du att det är någonting som kommer att uppkomma**
144.	Jag är lite skeptisk! Det är ju hur jag är också, jag gillar ju att man försöker förstå, inte att DevOps är det viktigaste, utan det viktigaste är varför så man kan försöka lära folk varför. Jag tycker ju man ska köra mycket mer Lean utbildningar. Det är ju ungefär samma diskussion som i Agile och SCRUM. Hälften av alla inlägg är ju liksom "Är det här mer eller mindre agile?". Så totalt meningslös diskussion. Eller "Är detta agile? Teamet som gör såhär, är de agile?" jaa.. Hur långt är ett snöre? Det är så enormt onödig diskussion, det beror ju helt på kontexten, hur dem jobbar och vilka andra problem de har. Utan det är att leta upp det viktigaste problemet för din organisation och lös det, ja då är du agile!
145.	*P7 anser att DevOps lider av en liknandes diskussion som förs inom Agile att utövare fokuserar på fel saker och att man glömmer bort varför man gör det. Pekar på likheten mellan Agile och DevOps och att det är någonting som är svårt att bedömma ifall man är agil eller ifall man är DevOps.* P7-30 Utövare kan ha fel fokus P7-31 DevOps liknat med Agile P7-32 DevOps abstrakt och svårbedömt
146.	**Skulle man kunna säga samma sak om DevOps eller är det ett annat?**
147.	DevOps tycker jag är påväg att bli precis likadant! Det är ett buzzword som man använder för att pracka på någon en kurs eller ett verktyg.
148.	P7 ser att DevOps blivit ett buzzword P7-33 DevOps buzzword
149.	**Istället så är det mer..**
150.	Istället för att ta till sig vad som vi försöker åstadkomma. Så blir det "Ja vi ska ha DevOps, då ska vi ha det verktyget och vi måste hyra in den här föreläsaren, då är vi DevOps. Men det är ju inte det som är poängen utan det är ju något helt annat som är det viktiga egentligen och det är att hela tiden titta på det här flödet egentligen. Så du blir inte agile eller DevOps och bra på det, genom att bara kopiera någonting och säga "Tar vi till oss de här verktygen och de här handgreppen, då är vi DevOps sen". Ja då är du bra på det som någon annan redan har slutat göra för det var inte så bra och optimalt längre. Utan det som man måste bli bra på är att hela tiden bli bättre. Att haka upp sig för mycket på DevOps är bara att titta på den här barriären som var på vissa ställen mellan utveckling och produktion. Ja, du har inget optimalt flöde, har du löst det problemet så har du ett problem någon

	annanstans istället. Då är det något annat som är det viktigaste att ta tag i, men är du fokuserad endast på “oh DevOps” då tittar man kanske för mycket på den bromsklossen och tappar fokus på andra grejer som kan vara problemet nu. Likadant med Agile och SCRUM
151.	P7 menar att DevOps är mer än verktyg och handgrepp ochatt det inte nödvändigtvis innebär att organisationen är DevOps. P7-34 DevOps mer än verktyg
152.	**Har du ett liknande exempel till DevOps?**
153.	Tycker att det kan vara lite farligt med den där monitorerings TV’n till exempel. Den ser jag lite som ett symbolvärde, den ska man ha där med mätdata från serverhallen ifall man är DevOps. Men det är ju inte den som gör dig till DevOps, utan det är hur snabbt du kan göra något åt problem som uppstår i produktion liksom. Det viktiga är ju inte om det är Dev eller Ops som gör det, utan hur lång tid tar det? Det som skulle stå överallt!
154.	**Du ska ha stort tack!**

Appendix L

Sammanställning av nyckelpoänger kategoriserade efter CAMS-värdegrunderna.

Culture	
P1-1	DevOps naturligt för utvecklande IT-bolag.
P1-3	Många ska kunna göra mycket.
P1-5	DevOps faller naturligt för mindre bolag.
P1-6	Fritt mandat för förändring ifall personen tror det är nödvändigt.
P1-7	Inget större beslutsorgan.
P1-8	Delat ansvar kan innebära missar.
P1-11	Vill undvika nyckelpersoner.
P1-12	Inga tydliga roller.
P1-13	Svårt att se saknad kompetens iom. Otydliga roller.
P1-14	Problematik löses ad-hoc framför planering.
P1-15	Ser bred kunskap framför nyckelkompetens.
P1-16	Ser personlighet och intresse framför nyckelkompetens.
P1-21	För mycket feedback kan vara överväldigande.
P1-22	Utvecklingen är beroende av kundfeedback.
P1-23	Trender kan styra utvecklingen.
P1-24	Utvecklare står för stor del av innovationen.
P1-25	Större förändringar kräver mer planering.
P1-26	Personligt Intresse kan avgöra ledarrollen.
P1-27	Team byggs upp efter de resurser som krävs.
P1-28	Team kan innehålla personal utanför Development och Operations.
P1-30	Utvecklare tar ett helhetsansvar för produkten.
P1-31	Det sker en dialog mellan Dev och Ops vilka miljöer som behövs sättas upp.
P1-32	Utvecklaren har jouren för att produkten fungerar, medan Ops fungerar som secondline.
P1-37	Finns inga tydligt definierade roller.
P1-38	Otydliga roller kan innebära att problem eskalerar för långt.
P1-39	Grupper kan utgöras utifrån en spec. applikation.
P1-40	Applikationsgrupperingen kan ibland ge upphov till att problem inte går att placera.
P1-45	Initial skeptism, men visat sig fungera bra för P1’s organisation.
P1-46	Prioriterar snabba ändringar framför extensiv planering och fokuserar på att reagera snabbt.

P1-47	P1-46 Kan orsaka svårigheter i hur prioritering ska ske.
P1-50	DevOps kan komma naturligt eller som en nödvändighet vid få anställda.
P1-54	Separationen mellan roller ska vara så liten som möjligt.
P1-55	Utvecklare ansvarar för sin produkt konstant.
P2-1	Det existerar olika ansvarsområden.
P2-2	DevOps kan ha olika strukturer.
P2-4	Öppen miljö, alla får tycka till om prioriteringar.
P2-7	Många spontana arbetsuppgifter som stör de ordinarie arbetsuppgifterna.
P2-8	Inga krav på tidsrapportering.
P2-9	Balans mellan kortsiktiga och långsiktiga mål.
P2-16	Större releases diskuteras med Operations genom DevOps chatt.
P2-17	DevOps fungerar väl trots växande utvecklingsteam.
P2-18	Det är svårt att hitta bra personal till ett DevOps team.
P2-19	DevOps ställer höga krav på social kompetens hos utvecklare.
P2-20	Att distribuera DevOps till internationella teams kan ses problematiskt.
P2-21	Operations och Dev jobbar på samma kontor.
P2-22	Utvecklingsavdelningen utför delar av driftsättningen av mjukvara.
P2-23	Det finns ofta en person i varje team som sköter den huvudsakliga kontakten med driften.
P2-24	Utvecklarna sköter själva många av de uppgifter som typiskt utförs av drift, såsom produktionssättning.
P2-25	Jour-ansvar för drift läggs på (vissa) utvecklare.
P2-26	Driftavdelningen har ansvar för problem kopplade till hårdvara och nätverk.
P2-27	Utveckling och drift arbetar jämte varandra.
P2-28	Utvecklare ansvarar för sin kod hela vägen tillochmed drift.
P2-29	Kommunikation viktigt mellan support och dev.
P2-30	Det kan vara svårt för dev och support att få förståelse för varandras situationer.
P2-32	DevOps är att sudda ut gränserna mellan Utveckling och Drift.
P2-34	Support, Dev och Ops sitter nära varandra rent fysiskt.
P2-35	Vid brådskande ären finns möjlighet att direkt kommunicera face-to-face.
P2-38	Det är viktigt att förebygga problem genom att förvarna varandra (Dev <-> Ops).
P3-5	DevOps / CD kan tas emot på ett bra eller dåligt sätt.
P3-6	DevOps omfattande nog att behöva coaching för.
P3-9	Produktion kräver ett bra samarbete med Operations.
P3-10	Utvecklare ska veta direkt ifall deras förändring inneburit framgång eller fel.
P3-11	Feedback till utvecklaren beror till viss del på samarbetet mellan Ops.

P3-13	DevOps handlar om kultur.
P3-16	Tilliten är ofta det som brister som ett resultat av dålig insyn i andra avdelningar.
P3-18	Ops ansvarar för plattformen, Dev för produkten som kör på plattformen.
P3-19	Med scrum så får alla alla roller.
P3-20	Att Ops ingår i team ses som en förlägning av redan agila team (SCRUM).
P3-24	Utvecklarna ska efterlinka Operations processer.
P3-27	DevOps är involverat i hela utvecklingsarbetet.
P3-28	Snabbare feedback lättare att åtgärda problemet.
P3-30	Tidig involvering främjar snabb feedback.
P3-32	Finns en ovilja att samarbeta.
P3-33	Operations nivå(mogenhet) av kontroll och rutiner påverkar samarbete.
P3-36	Ser nytta i att ha en ledarroll i initiativet.
P3-37	Geografiska avstånd har påverkan på kommunikation och tillit.
P3-38	Organisationens storlek påverkar möjligheter för fysisk närhet.
P3-39	Traditionell projektledning ställer sig motstridigt till förändringen.
P3-41	DevOps för med sig en större insikt i organisationen för individen.
P3-42	Frekventare och mindre ändringar för minskad risk.
P3-43	Minskad risk i ändringar innebär en minskad risk för osämja.
P3-44	Dev och Ops tillsammans ser till business mål.
P3-45	Bra DevOps innebär kommunikation och gemensamma mål.
P3-46	Gemensamma mål anknyter prioritering.
P3-47	Verksamheten bidrar till gemensam prioritering.
P3-49	Bolag med traditionella metoder ser mer motstridighet.
P4-1a	DevOps innebär en mindre definitiv skiljning på Dev och Ops.
P4-1b	Ses organisatoriskt som en och samma avdelning.
P4-2	Team är funktionsorienterade runt applikationer.
P4-2a	Team ska vara korsfunktionella.
P4-3	Organisationen saknar stora avdelningar efter arbetsuppgift.
P4-4	Team är funktions & applikationsorienterade.
P4-5	Rollernas fördelning i framtiden är oviss.
P4-6	Otydlighet kring rollernas framtid.
P4-7	Otydlighet kring rollernas framtid kan vara problematisk.
P4-8	Ansvarsroller kan bli otydliga vilket kan vara problematiskt.
P4-11	Siloeffekter motarbetas med kommunikation.
P4-13	Veckovis möten mellan support och utvecklare hjälper utvecklare att förstå kundernas behov.

P4-18	Operations övervakar det applikationer som körs.
P4-19	Utvecklingsdrivet företag.
P4-20	Utvecklare bytte ansvarsuppgift för att det fanns ett behov.
P4-24	Organisationer tjänar på att ha mindre avskiljningar.
P4-25	Utvecklare har ansvar för kod i drift.
P4-26	Att lämna över ansvaret för mjukvaran till driftavdelningen ses som något som i många fall är sämre.
P4-27	DevOps fungerar i både små och stora organisationer.
P4-28	Att övergå till DevOps kan vara svårare för en stor organisation.
P4-29	DevOps ses som naturligt i väldigt små organisationer.
P4-30	Akut kommunikation sker face-to-face eller via telefon.
P4-31	Icke-akuta ärenden förmedlas via ärendehanteringssystem.
P4-33	Det är ofta diskussioner över hur mycket prestanda Ops skall upplåta till Dev för olika projekt.
P4-34	Diskussioner mellan avdelningar hjälper organisationen att effektivisera saker.
P4-38	Operations involveras i ett tidigt skede av projekten.
P4-47	Alla utvecklare testar själva sin kod.
P4-48	Det existerar ingen test-roll på företaget.
P4-53	Ett ärendehanteringssystem används för att spåra alla aktiviteter.
P4-63	Direktiv kommer oftast från andra avdelningar och inte uppifrån.
P4-64	Det är möjligt att själva prioritera sina arbetsuppgifter.
P4-65	Tidskrävande att prioritera sina arbetsuppgifter.
P4-66	Kundnytta i fokus vid prioriteringar.
P4-67	Platt organsation, ej toppstyrt.
P4-68	Personlighet ganska viktigt inom DevOps.
P4-69	Det är viktigt med möten för att prioritera vad som bör göras.
P4-70	Kommunikation viktigt inom DevOps.
P4-72	Med eget ansvar finns risk att utvecklare jobbar på projekt med låg kundnytta.
P4-74	Svårt att se framtida kundbehov kan ge upphov till problem vid prioritering.
P4-75	Kommunikation mellan Support och Dev är viktig.
P4-76	Kommunikation mellan Support och Dev är svår att prioritera.
P4-77	Säljavdelningen vill ha nya och bättre produkter.
P4-78	Supportavdelningen vill ha stabila och fungerande produkter.
P4-79	Support- och säljavdelningen har olika mål.
P4-80	Beslut tas med fördel genom överenskommelse i grupp.
P4-81	Det är viktigt att kunna anpassa sin kommunikation till mottagaren.

P4-82	DevOps naturligt för organisationer med öppen kommunikation.
P4-85	DevOps upplevs som uppskattat bland de anställda i organisationen.
P4-88	Otydliga ansvarsroller kan vara problematiskt.
P4-89	Otydliga ansvarsroller har fler fördelar än nackdelar.
P4-90	Direktiv kommer inte uppifrån vilket uppskattas.
P4-91	Platt organisation ger bättre produkter.
P4-92	DevOps ger högre kvalitet på arbetet.
P4-102	Avdelningarna får synergieffekter genom att kommunicera med varandra.
P4-108	Prioritering av arbetsuppgifter viktigt och svårt.
P4-111	Verktyg för kommunikation viktigt.
P4-112	Gemensam chatt främjar kommunikation.
P4-114	kommunikation den största utmaningen inom DevOps.
P4-115	Öppna kommunikationskanaler kan ge upphov till överflödig kommunikation.
P4-116	Mycket spring och prat i öppna landskap.
P4-117	Kommunikation kan vara distraherande.
P4-118	Kräver stort personligt ansvar.
P4-119	Platt organisation.
P4-120	Direktiv kommer sällan uppifrån.
P4-122	Anställda ägnar sig aldrig åt bara en specialiserad uppgift.
P5-1	Release Managers ansvariga för godkännande av release ut mot kund.
P5-2	Kundsupport bär ansvaret att distribuera en release till slutanvändaren.
P5-4	Release Readiness ansvarar för att produkten är färdig och färdigtestad.
P5-10	Cloudtjänsten och desktoptjänsten har geografiskt separerade teams.
P5-12	Kodgranskning sker innan release.
P5-13	Kod granskas innan en commit går vidare.
P5-14	Kanban används för prioritering av arbetsuppgifter.
P5-16	Viktigt att omvärdera prioriteringar veckovis.
P5-17	Prioriteringar av arbetsuppgifter sker i första hand top-down.
P5-18	Utvecklare kan själv påverka prioritering av sina arbetsuppgifter.
P5-20	Möjligheten att se andras Kanban-projekt utnyttjas ej frekvent av P5.

P5-21	Möjligheten att se andras Kanban-projekt upplevs som värdefullt.
P5-22	Cloud-lösningen har ett operations team som är separat från Utveckling.
P5-23	Driftproblem delegeras till komponentleverantör.
P5-27	Kundvärde är i fokus vid utvecklingen.
P5-29	Kundvärde i fokus vid priorteringar i utvecklingsarbetet.
P5-31	Ingen inre utvärdering av team.
P5-32	Behovet av att mäta sjunker när anställda tycker det är roligt att arbeta.
P6-5	Viktigt att se behov hos andra avdelningar.
P6-6	Viktigt att se behov hos andra avdelningar.
P6-8	Viktigt att se business value i allt som utvecklas.
P6-10	Kommunikation mellan avdelningar viktigt.
P6-11	Face-to-face kommunikation mellan avdelningar viktigt.
P6-12	Ett chattsystem (Slack) används för kommunikation mellan avdelningar.
P6-14	Kommunikation mellan avdelningar viktigt.
P6-15	Kommunikationspaket ökar förståelse.
P6-20	Champions bra vid stora ändringar.
P6-23	Operations ansvarar inte för applikationslagret.
P6-24	Applikationsägare ansvarar för applikationslagret.
P6-25	Ansvar för kod sker i vissa fall hela vägen ut till drift.
P6-26	Svårt att ansvara för kod i installationspaket.
P6-27	Ansvar för kod i drift sker i cloud-miljöer.
P6-28	Prioriteringar sköts med hjälp av ärendehanteringssystem med Kanban.
P6-31	Det finns olika roller i ett team.
P6-32	Det finns gränser mellan roller, men de är ej skarpa.
P7-2	DevOps handlar om kunskap och insyn om hela leveransflödet.
P7-8	Att inte se hela flödet ses som negativt.
P7-11	Med insyn och helhetsperspektiv skrivs bättre produkt/mjukvara.
P7-12	Insyn och helhetsperspektiv behövs då pipelinen ständigt förändras.
P7-14	Tjänstifierar för att minska komplexiteten,Microservices starkt anknutet.
P7-16	Strävar mot utvecklarteams helhetsansvar.
P7-19	Dev och Ops delar ansvaret för att monitorera produktion.
P7-21	För att Dev ska kunna ta ett helhetsansvar krävs en arkitektur som stödjer det, exv. Microservices.
P7-23b	Chattverktyget Slack används för direkt kommunikation.
P7-24	SaaS produkter lämpar sig bättre för delivery pipeline.

P7-27	Produktionen ska vara allas problem.
P7-28	DevOps kan innebära en stor organisatorisk förändring.
P7-29	Stöd från ledning viktigt för implementationen av DevOps.
P7-30	Utövare kan sakna fokus på produktionskedjan.

Appendix M

Sammanställning av koncept och konceptkategorier.

INSYN	
(CORE) DevOps organisationer karaktäriseras utav en organisatorisk transparens, där varje medarbetare ska ha en insyn till andra delar av organisationen utanför sitt eget område och ha ett helhetsperspektiv av organisationens värdekedja:	
P1-48	Undvika individberoende genom att alla har en helhetsbild.
P2-30	Det kan vara svårt för dev och support att få förståelse för varandras situationer.
P2-32	DevOps är att sudda ut gränserna mellan Utveckling och Drift
P3-41	DevOps för med sig en större insikt i organisationen för individen.
P4-95	Insyn i andra roller ökar förståelse.
P4-109	DevOps metod för att få bredare synsätt på sitt arbete.
P5-3	Insyn saknas från distributionsansvarig till kundsupport. (Se P5-8, P5-9, P7-,)
P6-5	Viktigt att se behov hos andra avdelningar
P6-6	Viktigt att se behov hos andra avdelningar.
P6-8	Viktigt att se business value i allt som utvecklas.
P6-21	Champions ger insyn till utvecklarna åt de som tillhandahåller verktygen.
P7-7	Kunskapsspridning skapar ett helhetsperspektiv
P7-8	Att inte se hela flödet ses som negativt.
P7-11	Med insyn och helhetsperspektiv skrivs bättre produkt/mjukvara.
P7-12	Insyn och helhetsperspektiv behövs då pipelinen ständigt förändras.
P7-13	Automation för att minska bördan av att veta mycket.
P7-19	Dev och Ops delar ansvaret för att monitorera produktion.
P7-26	DevOps är Lean för mjukvara.
P7-27	Produktionen ska vara allas problem.
P7-30	Utövare kan sakna fokus på produktionskedjan.
DevOps karaktäriseras av en organisationsstruktur och ledning stödjer insyn	
P1-5	DevOps faller naturligt för mindre bolag.
P4-29	DevOps ses som naturligt i väldigt små organisationer.
P1-50	DevOps kan komma naturligt eller som en nödvändighet vid få anställda
P4-91	Platt organisation ger bättre produkter.
P4-90	Direktiv kommer inte uppifrån vilket uppskattas.
P4-67	Platt organsation, ej toppstyrt.

P4-68	Personlighet ganska viktigt inom DevOps.
P4-24	Organisationer tjänar på att ha mindre avskiljningar
P4-27	DevOps fungerar i både små och stora organisationer.
P4-28	Att övergå till DevOps kan vara svårare för en stor organisation.
P7-15	Flera pipelines används för att minska komplexitet.
P7-17	Fler pipelines något man strävar efter så långt det går.
P7-21	För att Dev ska kunna ta ett helhetsansvar krävs en arkitektur som stödjer det, exv. Microservices.
P7-28	DevOps kan innebära en stor organisatorisk förändring.
Arbetsfördelning och helhetsansvar	
DevOps-organisationer karaktäriseras utav att lägga mindre vikt till roller och titlar:	
P1-12	Inga tydliga roller.
P1-37	Finns inga tydligt definierade roller.
P1-54	Separationen mellan roller ska vara så liten som möjligt.
P4-1a	DevOps innebär en mindre definitiv skiljning på Dev och Ops.
P7-27	Produktionen ska vara allas problem.
Eftersom:	
P1-43	Självbalans av personligheter och kunskaper.
P1-49	Effektiverare resursanvändning genom breda roller och kunskap.
P4-97	Nyckelkompetens skall jobbas bort.
Detta kan innebära utmaningar:	
P1-8	Delat ansvar kan innebära missar.
P1-13	Svårt att se saknad kompetens iom. Otydliga roller.
P1-38	Otydliga roller kan innebära att problem eskalerar för långt.
P4-8	Ansvarsroller kan bli otydliga vilket kan vara problematiskt.
P4-72	Med eget ansvar finns risk att utvecklare jobbar på projekt med låg kundnytta.
P4-88	Otydliga ansvarsroller kan vara problematiskt.
P4-89	Otydliga ansvarsroller har fler fördelar än nackdelar.
P4-98	Nyckelkompetens svårt att undvika i små organisationer.
P4-118	Kräver stort personligt ansvar.
Inom DevOps-organisationer finns fortfarande olika ansvarsområden:	
P1-30	Utvecklare tar ett helhetsansvar för produkten.
P1-32	Utvecklaren har jouren för att produkten fungerar, medan Ops fungerar som secondline.
P1-55	Utvecklare ansvarar för sin produkt konstant.
P2-1	Det existerar olika ansvarsområden.

P2-24	Utvecklarna sköter själva många av de uppgifter som typiskt utförs av drift, såsom produktionssättning.
P2-24a	Skiljer på driftproblem. Utvecklare löser en viss sorts problem, medan Operations tar mer övergripande infrastruktur problem.
P2-25	Jour-ansvar för drift läggs på (vissa) utvecklare.
P2-26	Driftavdelningen har ansvar för problem kopplade till hårdvara och nätverk.
P2-28	Utvecklare ansvarar för sin kod hela vägen tillochmed drift.
P3-7	Utvecklare får möjlighet att ta ansvar med hjälp av snabb feedback.
P3-17	Ops ansvar att förse Dev med verktygen och kunskapen för att kunna släppa själv.
P3-18	Ops ansvarar för plattformen, Dev för produkten som kör på plattformen.
P4-18	Operations övervakar det applikationer som körs.
P4-25	Utvecklare har ansvar för kod i drift.
P4-26	Att lämna över ansvaret för mjukvaran till driftavdelningen ses som något som i många fall är sämre.
P4-44	Operations är inte involverade i mindre deployments som inte påverkar nätverk.
P4-94	Operations tillhandahåller automatiserade verktyg till Development.
P4-110	DevOps ger mer input från olika håll på sitt arbete.
P7-19	Dev och Ops delar ansvaret för att monitorera produktion.
Dev överlämnar inte ansvaret av produkten, utan tar ett helhetsansvar för produktens hela livslängd	
P1-30	Utvecklare tar ett helhetsansvar för produkten.
P1-32	Utvecklaren har jouren för att produkten fungerar, medan Ops fungerar som secondline
P1-55	Utvecklare ansvarar för sin produkt konstant.
P2-24	Utvecklarna sköter själva många av de uppgifter som typiskt utförs av drift, såsom produktionssättning.
P2-24a	Skiljer på driftproblem. Utvecklare löser en viss sorts problem, medan Operations tar mer övergripande infrastruktur problem.
P2-25	Jour-ansvar för drift läggs på (vissa) utvecklare.
P2-28	Utvecklare ansvarar för sin kod hela vägen tillochmed drift.
P3-18	Ops ansvarar för plattformen, Dev för produkten som kör på plattformen.
P4-18	Operations övervakar de applikationer som körs.
P4-25	Utvecklare har ansvar för kod i drift.
P4-26	Att lämna över ansvaret för mjukvaran till driftavdelningen ses som något som i många fall är sämre.
P5-1	Release Managers ansvariga för godkännande av release ut mot kund. (Se P5-8, P5-9)
P5-2	Kundsupport bär ansvaret att distribuera en release till slutanvändaren. (Se P5-8, P5-9)
P6-24	Applikationsägare ansvarar för applikationslagret. (se P6-27)
P6-25	Ansvar för kod sker i vissa fall hela vägen ut till drift.
P6-26	Svårt att ansvara för kod i installationspaket. (Se P6-27)

P6-27	Ansvar för kod i drift sker i cloud-miljöer.
P7-16	Strävar mot utvecklarteams helhetsansvar.
P7-19	Dev och Ops delar ansvaret för att monitorera produktion.
Ops ansvarar för den underliggande plattformen, men inte produkten som sådan	
P1-32	Utvecklaren har jouren för att produkten fungerar, medan Ops fungerar som secondline.
P2-25	Jour-ansvar för drift läggs på (vissa) utvecklar.
P2-26	Driftavdelningen har ansvar för problem kopplade till hårdvara och nätverk.
P3-17	Ops ansvar att förse Dev med verktygen och kunskapen för att kunna släppa själv.
P3-18	Ops ansvarar för plattformen, Dev för produkten som kör på plattformen.
P4-44	Operations är inte involverade i mindre deployments som inte påverkar nätverk.
P6-23	Operations ansvarar inte för applikationslagret.
P7-19	Dev och Ops delar ansvaret för att monitorera produktion.

Referenser

Allan, G., 2003. A critique of using grounded theory as a research method.*Electronic journal of business research methods*, *2*(1), pp.1-10.

Bajec, M., & Krisper, M. (2005): *A methodology and tool support for managing business rules in organisations*. *Information Systems*, *30*(6), [figur] sid. 428.

Beck, K., 1999. *Embracing change with extreme programming*. Computer,32(10), pp.70-77.

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J. and Thomas, D. (2001): *Manifesto for Agile Software Development*. http://www.agilemanifesto.org/ (hämtad 2016-05-03)

Brandtner, M., Giger, E. and Gall, H., 2014, February. Supporting continuous integration by mashing-up software quality information. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on* (pp. 184-193). IEEE.

Brunnert, A, van Hoorn, A, Willnecker, F, Danciu, A, Hasselbring, W, Heger, C, Herbst, N, Jamshidi, P, Jung, R, von Kistowski, J, Koziolek, A, Kroß, J, Spinner, S, Vögele, C, Walter, J, & Wert, A (2015), *'Performance-oriented DevOps: A Research Agenda*', arXiv, EBSCO*host.* (hämtad 2016-04-05)

Charmaz, K (2006): *Constructing Grounded Theory. [Elektronisk Resurs] : A Practical Guide Through Qualitative Analysis*, n.p.: London: SAGE, 2006, Library catalogue (Lovisa), EBSCO*host*, (hämtad 2016-04-24)

Claps, G.G., Svensson, R.B. and Aurum, A., 2015. On the journey to continuous deployment: Technical and social challenges along the way.*Information and Software technology*, *57*, pp.21-31.

Debois, P. (2008), *Agile Infrastructure and Operations: How Infra-gile are You?*, Agile, 2008. AGILE ’08. Conference, pp.202,207, 4-8 Aug.

Debois, P. (2009). *Devopsdays Ghent 2009*. http://www.devopsdays.org/events/2009-ghent/ (hämtad 2016-04-13)

DeGrandis, D. (2011), *Devops: So you say you want a revolution?* Cutter IT J. 24(8), sid. 34–39.

Denning, S., 2015. How to make the whole organization Agile. *Strategy & Leadership*, *43*(6), pp.10-17.

DevOpsDictionary (2016). *CAMS*. http://devopsdictionary.com/wiki/CAMS (hämtad 2016-05-14)

Dyck, A, Penners, R, & Lichter, H (2015), '*Towards Definitions for Release Engineering and DevOps*', *2015 IEEE/ACM 3Rd International Workshop On Release Engineering*, p. 3

Ducy, M. (2015): *We don't need no manifesto.* https://goatcan.do/2015/01/16/we-dont-need-no-manifesto/ (hämtad 2016-05-05)

Duvall, P.M., Matyas, S. and Glover, A., 2007. *Continuous integration: improving software quality and reducing risk*. Pearson Education.

Dybå, T. och Dingsøyr, T., 2008. Empirical studies of agile software development: A systematic review. *Information and software technology*,*50*(9), pp.833-859.

Ensor, P.S., (1988). *The functional silo syndrome*. AME Target, 16, p.16.

Erich, F., Amrit, C., & Daneva, M. (2014), *Report: DevOps Literature Review*,*University of Twente, Tech. Rep.* http://www.utwente.nl/bms/iebis/staff/amrit/devopsreport.pdf (hämtad 2016-04-20)

Erich, F., Amrit, C. and Daneva, M., (2014): *A mapping study on cooperation between information system development and operations*. In *Product-Focused Software Process Improvement* (pp. 277-280). Springer International Publishing.

Feitelson, D, Frachtenberg, E, & Beck, K (2013): '*Development and Deployment at Facebook*', *IEEE Internet Computing*, 17, 4, p. 8, Publisher Provided Full Text Searching File, EBSCO*host*, (hämtad 2016-05-12)

Fitzgerald, B., Stol, K. (2015) *Continuous software engineering: A roadmap and agenda*, *Journal of Systems and Software*, ISSN 0164-1212, http://dx.doi.org/10.1016/j.jss.2015.06.063. (hämtad 2016-05-03)

Fowler, M. (2006). *Continuous Integration*. http://martinfowler.com/articles/continuousIntegration.html (hämtad 2016-05-02).

Gandomani, T.J., Zulzalil, H., Ghani, A.A.A., Sultan, A.B.M. and Parizi, R.M., (2015). *The impact of inadequate and dysfunctional training on Agile transformation process: a Grounded Theory study.* Information and Software Technology, 57, pp.295-309.

Google (2016): *Interest in "DevOps" search term over time.* https://www.google.com/trends/explore#q=DevOps%20engineer&cmpt=q&tz=Etc%2FGMT-2 [figur](hämtad 2016-05-12)

Hendren, J. (2016). *The Problem with Defining DevOps*. [online] Upguard.com. Available at: https://www.upguard.com/blog/the-problem-with-defining-devops (hämtad 2016-05-02)

Humble, J. (2010). *Continuous Delivery vs Continuous Deployment - Continuous Delivery*. http://continuousdelivery.com/2010/08/continuous-delivery-vs-continuous-deployment/ (hämtad 2016-05-07)

Humble, J, & Farley, D (2011), *Continuous Delivery*, n.p.: Upper Saddle River, NJ : Addison-Wesley.

Humble, J, & Molesky. (2011) *"Why enterprises must adopt devops to enable continuous delivery." Cutter IT Journal* 24.8.6

Huo, M., Verner, J., Zhu, L. and Babar, M.A., 2004, September. Software quality and agile methods. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International* (pp. 520-525). IEEE.

Hüttermann, M (2012), *Devops For Developers. [Elektronisk Resurs]*, n.p.: Berkeley, CA : Apress : Imprint: Apress, 2012., Library catalogue (Lovisa), EBSCO*host*, (hämtad 2016-05-08)

Jacobsen, D, Sandin, G, & Hellström, C 2002, *Vad, Hur Och Varför : Om Metodval I Företagsekonomi Och Andra Samhällsvetenskapliga Ämnen*, n.p.: Lund : Studentlitteratur, 2002 (Lund : Studentlitteratur), Library catalogue (Lovisa), EBSCOhost, viewed 7 June 2016.

Lee, G, & Xia, W (2010): *'Toward Agile: An Integrated Analysis of Quantitative and Qualitative Field Data on Software Development Agility*', *MIS Quarterly*, 1, p. 87, JSTOR Journals, EBSCO*host*, (hämtad 2016-04-24)

Lei, H., Ganjeizadeh, F., Jayachandran, P.K. and Ozcan, P., 2015. A statistical analysis of the effects of Scrum and Kanban on software development projects. *Robotics and Computer-Integrated Manufacturing*.

Lwakatare, L. E., Kuvaja, P, Oivo, M,. (2015), *Dimensions of DevOps*, p. 212-217 Agile Processes, in Software Engineering, and Extreme Programming: 16th International Conference, XP 2015, Helsinki, Finland, May 25-29, 2015, Proceedings

Maxwell, J. A. (1996). *Qualitative research design: An interactive approach* (Vol. 41): Sage.

Miles, M, & Huberman, A 1994, *Qualitative Data Analysis : An Expanded Sourcebook*, n.p.: Thousand Oaks, CA : Sage, cop. 1994, Library catalogue (Lovisa), EBSCOhost, viewed 7 June 2016

Minick, E. (2015). *Surprise! Broad Agreement on the Definition of DevOps - DevOps.com*. http://devops.com/2015/05/13/surprise-broad-agreement-on-the-definition-of-devops/ (hämtad 2016-05-02)

Olsson, H, Alahyari, H, & Bosch, J (2012): '*Climbing the "Stairway to Heaven" -- A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software*', *2012 38th Euromicro Conference On Software Engineering & Advanced Applications*, p. 392

Phifer, B., 2011. Next-generation process integration: CMMI and ITIL do devops. *Cutter IT Journal*, *24*(8), p.28.

Rivera, J. & van der Meulen, R. (2015). *Gartner Says By 2016, DevOps Will Evolve From a Niche to a Mainstream Strategy Employed by 25 Percent of Global 2000 Organizations*. http://www.gartner.com/newsroom/id/2999017 (hämtad 2016-05-03)

Roche, J. (2013), *'Adopting DevOps Practices in Quality Assurance*', *Communications Of The ACM*, 56, 11, pp. 38-43, Business Source Complete, EBSCO*host*, (hämtad 2016-04-24)

Rodríguez, P, Haghighatkhah, A, Lwakatare, L, Teppola, S, Suomalainen, T, Eskeli, J, Karvonen, T, Kuvaja, P, Verner, J, & Oivo, M (2015), *'Continuous deployment of software intensive products and services: A systematic mapping study*', *The Journal Of Systems & Software*, ScienceDirect, EBSCO*host*, (hämtad 2016-05-07)

Shang, W. (2012), *Bridging the Divide between Software Developers and Operators using Logs*. In: 34th International Conference on Software Engineering, pp. 1583–1586. IEEE Press, New York.

Smeds, J., Nybom, K., Porres,I,(2015): *DevOps: A Definition and Perceived Adoption Impediments*, Agile Processes, in Software Engineering, and Extreme Programming, 16th International Conference, XP 2015 Helsinki, Finland, May 25–29, 2015 Proceedings. sid. 166-177.

Smeds, J., Nybom, K., Porres,I,(2015): *DevOps: A Definition and Perceived Adoption Impediments*, Agile Processes, in Software Engineering, and Extreme Programming, 16th International Conference, XP 2015 Helsinki, Finland, May 25–29, 2015 Proceedings [figur]. sid. 171.

Steen, O. (2016). *Uppsatsarbetet SYK02 / INFK11 F3 UPPSATSENS DELAR [Power-Point Presentation]*. https://liveatlund.lu.se/departments/Informatics/SYSK02/SYSK02_2016VT__-99___/Examensarbete(kandidatuppsats)/CourseDocuments/F3%20VT16.pdf (hämtad 2016-05-02)

Stolberg, S (2009), *'Enabling agile testing through continuous integration'*, Proceedings - 2009 Agile Conference, AGILE 2009, Proceedings - 2009 Agile Conference, AGILE 2009, p. 369-374, Scopus®, EBSCOhost, (hämtad 2016-05-06)

Swartout, P., (2014). *Continuous Delivery and DevOps–A Quickstart Guide*. Packt Publishing Ltd.

Wahaballa, A., Wahballa, O., Abdellatief, M., Xiong, H., & Qin, Z. (2015), *'Toward unified DevOps model'*, Proceedings Of The IEEE International Conference On Software Engineering And Service Sciences, ICSESS, 2015-November, ICSESS 2015 - Proceedings of 2015 IEEE 6th International Conference on Software Engineering and Service Science, p. 211-214

Weiss, R. S. (1994). *Learning from strangers: The art and method of qualitative interviewing*. New York: Free Press

Wettinger, Johannes., Breitenbücher, Uew., Leymann, Frank. (2014) *'DevOpSlang – Bridging the Gap between Development and Operations'*, Service-oriented and cloud computing : third European Conference, ESOCC 2014, Manchester, UK, September 2-4, 2014 : proceedings. Heidelberg: Springer, 2014. Print. Pp. 108-123.

Willis, J. (2010). *What Devops Means to Me | Chef Blog*. https://www.chef.io/blog/2010/07/16/what-devops-means-to-me/ (hämtad 2016-05-02)

Virmani, M., 2015, May. Understanding DevOps & bridging the gap from continuous integration to continuous delivery. In *Innovative Computing Technology (INTECH), 2015 Fifth International Conference on* (pp. 78-82). IEEE.