LUNDS UNIVERSITET
Ekonomihögskolan

Institutionen för informatik

Kritiska framgångsfaktorer vid införandet av Continuous Delivery

Kandidatuppsats 15 hp, kurs
SYSK02 i informationssystem

Författare: Niklas Lundkvist
Sebastian Sirviö

Handledare: Björn Johansson

Examinatorer: Bo Andersson
Markus Lahtinen

Kritiska framgångsfaktorer vid införandet av Continuous Delivery

Författare: Niklas Lundkvist och Sebastian Sirviö

Utgivare: Inst. för informatik, Ekonomihögskolan, Lund universitet

Dokumenttyp: Kandidatuppsats

Antal sidor: 52

Nyckelord: Continuous Delivery, Kritiska framgångsfaktorer, CD, Kontinuerlig Leverans

Sammanfattning (Max. 200 ord):

På senare tid har Continuous Delivery ökat kraftigt i popularitet på grund av att flera IT-jättar börjat använda och förespråka tillvägagångssättet för att effektivisera leveransen utav mjukvarusystem. Begreppet Continuous Delivery är relativt nytt men har rötter som sträcker sig tillbaka till agila metoders tillkomst. För att bidra till forskningsunderlaget för Continuous Delivery samt underlätta för ett införande av tillvägagångssättet har vi valt att ta fram vilka kritiska framgångsfaktorer som finns vid ett införande. Vi har valt att intervjua fyra IT-konsulter med god erfarenhet av att införa Continuous Delivery som arbetar i olika roller, tillsammans har de en erfarenhet av ungefär 30 införanden. För att ta fram vilka kritiska framgångsfaktorer som finns vid ett införande av Continuous Delivery har vi utfört intervjuer där vi diskuterar vilka utmaningar som är vanliga vid ett införande av Continuous Delivery. Genom att analysera resultatet av intervjuerna har vi tagit fram fem kritiska framgångsfaktorer som säkerställer ett framgångsrikt införande. De kritiska framgångsfaktorer vi har tagit fram är följande.

- Skapa en förändringsvillighet hos medarbetare
- Utveckla en oberoende struktur
- Försäkra att kodarkitekturen lämpar sig för testbarhet
- Stöd från företagsledningen
- Definiera ansvarsfördelning

Innehåll

Figurer

Tabeller

1

Introduktion

I det här kapitlet presenterar vi bakgrunden till vårt ämne, vi presenterar problemområdet samt definierar syftet.

Bakgrund

Continuous Delivery har sina rötter i den agila systemutvecklingsmetodiken Extreme Programming där en av reglerna uppmanar till att integrera kod ofta (Wells, 1997, 1999). Några år senare kom Agile Manifesto med en liknande uppmaning, ”Vår högsta prioritet är att tillfredsställa kunden genom tidiga och kontinuerliga leveranser av värdefull mjukvara” (Fowler och Highsmith, 2001). År 2006 skrev Martin Fowler, en av författarna av Agile Manifesto, en introduktion och guide till införandet av Continuous Integration vilket är en av delkomponenterna som ligger till grund för Continuous Delivery (Fowler, 2006).

Humble och Farley (2010) menar att paketeringen, leveransen och driftsättningen av mjukvarusystem traditionellt ofta kräver invecklade och specifika instruktioner som orsakar bekymmer för både utvecklare och tekniker. Eftersom att den slutgiltiga processen för paketering och leverans traditionellt är ett manuellt arbetsflöde som sker relativt sällan ökar det risken för att instruktionerna blir föråldrade mellan leveranser och att dessa därför kan behöva rättas flertalet gånger under leveransprocessen. Enligt Chen (2015) låter Continuous Delivery mjukvaruutvecklare snabbt och effektivt leverera uppdateringar till mjukvarusystem, vilket leder till att tiden från att en funktion är färdigutvecklad tills att den kan finnas i produktion minskar kraftigt.

En snabbare väg från utveckling till produktion innebär att utvecklarna kan få feedback ifrån användare snabbare, vilket ger utvecklare möjlighet att forma utvecklingsarbetet utifrån vad som efterfrågas (Chen, 2015). Dessutom kan Continuous Delivery bidra till en ökad kundnöjdhet på grund av att tiden mellan leveranser samt frekvensen av buggar minskar (Chen, 2015). Företag vars leveransfrekvens ökade med minst 10% jämfört med föregående år är 250% mer sannolika att se en årlig ekonomisk tillväxt på 10% eller mer för samma period, enligt en undersökning utförd av Enterprise Management Associates (Craig, 2016).

Problemområde

Continuous Delivery har nått vidd spridning och flera internetgiganter som Netflix (Bukoski et al., 2016), Google (Slatkin, 2013) och LinkedIn (Tate, 2013) använder sig utav tillvägagångssättet. Även den svenska Pensionsmyndigheten har börjat använda sig utav tillvägagångssättet (Danielsson, 2015). Nyligen beskrev en reporter på TechWorld Continuous Delivery som ”det kanske hetaste begreppet inom systemutveckling idag”

(Danielsson, 2016). Intresset för att utveckla stödsystem som understödjer Continuous Delivery låg i år på samma nivå som intresset låg på för år 2014, på en fortsätt förstaplats, när Enterprise Management Associates undersökte vilka stödsystem som var mest önskvärda att investera resurser i (Craig, 2016). Man kan tolka det som en indikation på att Continuous Delivery inte bara är en tillfällig trend utan är här för att stanna.

Trots att Continuous Delivery har nått vidd spridning och sägs ge en rad förbättringar som effektiviserar leveransen av mjukvarusystem finns det en klar brist på information och forskning på området, vilket är något som påpekas av Chen (2015). Vidare menar Chen (2015) att sådan forskning skulle kunna vara till stor hjälp för företag som tänkt sig implementera Continuous Delivery.

Frågeställning

Vilka kritiska framgångsfaktorer finns vid ett införande av Continuous Delivery?

Syfte

På grund av den begränsade mängd forskning som finns tillgänglig, den ökande popularitet samt det potentiella affärsvärdet hos Continuous Delivery har vi valt att fokusera på vilka kritiska framgångsfaktorer som finns vid införandet av Continuous Delivery. Valet föll på teorin om kritiska framgångsfaktorer på grund av att vi vill underlätta införandet av Continuous Delivery hos organisationer som idag använder antingen agila utvecklingsmetoder eller mer traditionella utvecklingsmetoder. Vi avser även att bidra till forskningsunderlaget för Continuous Delivery med den här uppsatsen.

Struktur

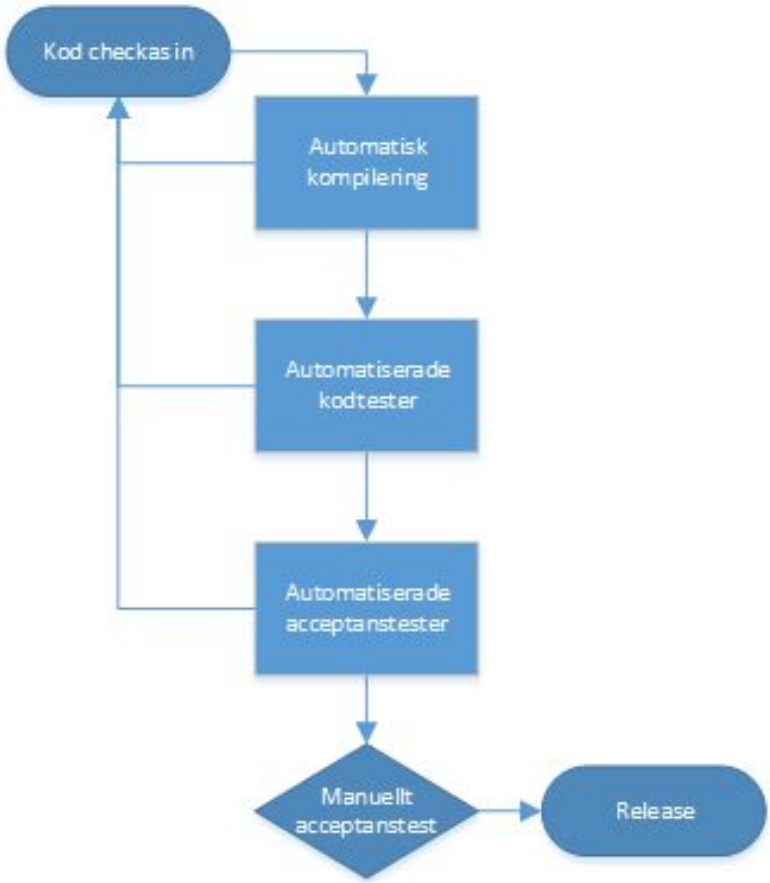
Uppsatsen är primärt strukturerad i kapitel och underkapitel med tillhörande listor över figurer, tabeller samt referenser.

2

Litteraturgenomgång

I det här kapitlet börjar vi med att definiera Continuous Delivery och de delkomponenter som Continuous Delivery består utav. Efter det tar vi upp principer för effektiv leverans utav mjukvarusystem och vilka former av utmaningar som kan uppstå vid ett införande av Continuous Delivery. I nästa del klargörs kritiska framgångsfaktorer och hur de identifieras. Vi redogör även för framgångsfaktorer som tidigare har tagits fram för införandet av Continuous Delivery. Slutligen presenterar vi det teoretiska ramverk som vi utgår ifrån.

Continuous Delivery



Figur 1

Continuous Delivery är ett tillvägagångssätt för att effektivisera leveransen av mjukvarusystem genom att automatisera paketering, testning och driftsättning, både i testmiljöer och i produktionsmiljöer (Karlsson, 2015). Detta gör man genom att använda sig utav ett versionshanteringssystem, Continuous Integration, enhetstester, utförliga automatiserade tester, ett manuellt test samt automatiserad driftsättning på begäran (Humble och Farley, 2010). Alla dessa delmoment genererar feedback som sammanställs i ett centralt

system där utvecklarna kan övervaka statusen av kodbasen. Detta skapar en teknisk helhet som ryms under begreppet Continuous Delivery (Humble och Farley, 2010).

- Versionshanteringssystem

Ett versionshanteringssystem är en central plats där kod samt andra filer och skript som är nödvändiga för att mjukvarusystemet ska kunna kompileras, testas och driftsättas sammanställs. Man kan också använda ett versionshanteringssystem för att ångra förändringar i kodbasen och få tillbaka en tidigare version (Humble och Farley, 2010).

- Continuous Integration

Continuous Integration ligger som grund till Continuous Delivery. De andra delmomenten är beroende av att Continuous Integration-systemet ser när förändringar görs i versionshanteringssystemet, utför enhetstester och gör mjukvarupaket tillgängliga (Duvall et al., 2007).

- Utförliga automatiska tester

Automatiserade tester som testar mjukvarusystemets funktionalitet i en testmiljö, dessa tester är betydligt mer ingående än enhetstesterna och ger viss försäkran om att regression i funktionalitet inte har skett (Humble och Farley, 2010).

- Manuella Tester

Innan slutgiltig leverans och driftsättning i produktionssystem sker ofta ett manuellt test eftersom att det är svårt att testa all funktionalitet automatiskt (Humble och Farley, 2010). Populärt kallas tillvägagångssättet för Continuous Deployment istället för Continuous Delivery om man inte använder sig utav det här steget, då driftsätter man systemet direkt när de automatiska testerna är lyckade och inte efter en manuell utlösare (Fowler, 2013).

- Automatiserad driftsättning

Skript för att automatiskt driftsätta systemet så att en leverans enkelt och effektivt kan ske. Den slutliga driftsättningen startas manuellt men processen är automatiserad när den är startad (Humble och Farley, 2010).

Principer för effektiv leverans av mjukvarusystem

Humble och Farley (2010) beskriver i sin bok om införandet av och fördelarna med Continuous Delivery att det finns åtta principer för lyckad mjukvaruleverans. Dessa principer är baserade på författarnas personliga erfarenheter och är något som författarna har funnit att vara gemensamma för samtliga mjukvaruprojekt de har varit delaktiga i. Principerna som tas

upp nämner författarna som ”de saker som vi inte kan föreställa oss att klara oss utan om vi vill att vår leveransprocess ska vara effektiv” (Humble och Farley, 2010).

- Skapa en pålitlig och upprepningsbar process

Humble och Farley (2010) menar att syftet med att skriva boken ”Continuous Delivery - Reliable Software Releases Through Build, Test och Deployment Automation” var att underlätta mjukvaruutveckling och att syftet kan förklaras med hjälp av endast en mening, ”Skapa en pålitlig och upprepningsbar process för mjukvaruleverans”. Författarna menar att en pålitlig och upprepningsbar process krävs för att göra driftsättningar av mjukvarusystem så enkla som möjligt, driftsättningen ska bli enkel genom att man har en väl beprövad process som man upprepar ofta. Tre grundstenar möjliggör enligt Humble och Farley (2010) en lyckad driftsättning: installation och upprätthållning av en miljö där mjukvaran kan köras, installation av en uppdaterad version av mjukvarusystemet och konfiguration, inklusive eventuell data som systemet kräver för att fungera. Alla dessa grundstenar, förutom hårdvaruinstallation, kan enligt författarna automatiseras.

- Automatisera nästan allt

Humble och Farley (2010) menar att vissa delar av leveransprocessen och driftsättningen inte kan automatiseras, t.ex. godkännanden från olika avdelningar eller manuell och specifik testning kräver manuell interaktion. Författarna menar att fler delmoment än vad många tror kan automatiseras, som databasuppgraderingar, brandväggskonfiguration, nätverkskonfiguration och acceptanstester (Humble och Farley, 2010). Författarna påstår att man ofta inte automatiserar så mycket som man kan i ett utvecklingsprojekt eftersom att det verkar som ett stort och skrämmande åtagande. Även om det vid första anblick ser ut som manuell testning, installation, konfiguration och driftsättning är enklare så blir automationen i slutändan enklare beroende på hur många gånger processen upprepas (Humble och Farley, 2010). Författarna poängterar att man ska se automatiseringen av befintliga processer som en identifikation av flaskhalsar och ett iterativt arbete för att automatisera bort dessa allt eftersom.

- Förvara allt i ett versionshanteringssystem

Alla filer som används för att specificera, bygga, konfigurera, testa och implementera ett mjukvarusystem bör finnas sparat på en central plats i ett versionshanteringssystem, enligt Humble och Farley (2010). Vidare bör varje bygge kunna härledas till en specifik uppsättning filer med t.ex. ett nummer som identifikation (changeset) vilket sedan kan användas för att se vilka versioner av olika filer som använts.

- Gör det som är svårt ofta

Humble och Farley (2010) påpekar vikten av att utföra processer som skapar bekymmer eller leder till problem ofta. Författarna menar t.ex. att om testfasen skapar stora bekymmer i slutet

av ett utvecklingsprojekt, sikta på att testa kontinuerligt under projektets livscykel istället. Om integrationer skapar stora problem när de t.ex. sällan görs eller är försenade, sikta på att integrera kontinuerligt istället. På så sätt bygger man enligt Humble och Farley (2010) bort problemen allt eftersom, även om vägen dit är svår och iterativ. Essentiellt är Extreme Programming resultatet av arbete efter den här principen (Humble och Farley, 2010).

- Bygg in kvalitet

Enligt Humble och Farley (2010) var ett av W. Edwards Demming, en av LEAN-rörelsens föregångsmän, valspråk ”Bygg in kvalitet”, och med detta valspråk menar författarna att det är lättare att åtgärda problem ju tidigare de hittas. Tekniker som Continuous Integration, omfattande automatiserad testning och automatiserad driftsättning är designade att hitta defekter tidigt så att man kan åtgärda dessa billigt (Humble och Farley, 2010). Vidare påpekar författarna att testning inte är en fas i slutet av utvecklingsfasen utan en kontinuerligt pågående process under hela utvecklingsfasen, om man lämnar testning till slutet av ett utvecklingsprojekt blir det svårt och eventuellt kostsamt att åtgärda problem (Humble och Farley, 2010). Humble och Farley (2010) tillägger slutligen att testning är allas ansvar, inte bara testarnas, och att alla är ansvariga för mjukvarusystemets kvalitet.

- Klar betyder integrerad

Enligt Humble och Farley (2010) är utvecklad funktionalitet inte helt klar innan den är redo för en slutanvändare att använda men man poängterar också att det kan vara både opraktiskt och orimligt att kompromisslöst applicera den här principen beroende på situation. Författarna menar att definitionen av klar i det här sammanhanget kan ändras till att innebära att funktionaliteten är integrerad och kan demonstreras i en produktionsliknande miljö (Humble och Farley, 2010).

- Gemensamt ansvar för leveransprocessen

Humble och Farley (2010) menar att för att säkerställa en effektiv mjukvaruleverans bör alla sträva efter samma mål och jobba tillsammans för att uppnå det gemensamma målet men att så inte alltid är fallet inom ett utvecklingsprojekt eller organisation. Författarna menar att i vissa fall växer en kultur fram där t.ex. testare skyller på utvecklare och utvecklare på testare istället för att man arbetar tillsammans för att lösa eventuella problem som uppstår.

- Kontinuerlig förbättring

Den första leveransen av en mjukvara är inte slutet på utvecklingsarbetet, mjukvara utvecklas och fler versioner kommer att behöva levereras. Enligt Humble och Farley (2010) måste leveransprocessen fortsätta utvecklas allt eftersom att utvecklingen av mjukvaran fortgår. Författarna skriver att hela organisationen behöver involveras för att motverka att man optimerar lokalt på bekostnad av generell optimering.

Utmaningar vid införande av Continuous Delivery

Chen (2015) har fått en artikel om sina erfarenheter av att införa Continuous Delivery på företaget Paddy Power publicerad i IEEE Software. Innan införandet av Continuous Delivery hade man på Paddy Power problem med att leverera mjukvara effektivt och tidsenligt, på grund av att flera manuella och felkänsliga delmoment krävdes av leveransprocessen (Chen, 2015). Baserat på sin erfarenhet med att införa Continuous Delivery inom flera utvecklingsprojekt på Paddy Power skriver Chen (2015) om de tre formerna av utmaningar som påträffades.

- Organisationella utmaningar

Enligt Chen (2015) har den största utmaningen varit organisationell vid införandet av Continuous Delivery metodiken, vid leveranser har många aktörer ifrån olika delar av verksamheten varit inblandade och alla har egna intressen, arbetssätt och kontrollområden. För att underlätta införandet av Continuous Delivery var man tvungen att omorganisera företagsstrukturen för att bidra till en samarbetsvillig företagskultur med bättre kommunikation mellan avdelningar.

- Processutmaningar

Den andra stora utmaningen var enligt Chen (2015) traditionella processer när leveranscykler ska ske ofta och effektivt. På Paddy Power behövde en ny version av ett mjukvarusystem godkännas av de berörda aktörerna från flera olika avdelningar vilket kunde försena en leverans med upp till fyra dagar (Chen, 2015). Chen (2015) menar att när utvecklingstiden av en ny version av mjukvaran bara var några dagar bestod en alltför stor del av den totala tidsåtgången av godkännandet för driftsättning.

- Tekniska utmaningar

Chen (2015) kunde vid tiden av införandet utav Continuous Delivery på Paddy Power inte hitta någon robust, anpassningsbar och färdig mjukvarulösning för Continuous Delivery och således bestämde man sig att utveckla en helt egen mjukvarulösning. Att få olika verktyg, tekniker att samverka som byggstenar i mjukvarusystemet som utvecklades kom att bli en utmaning (Chen, 2015). Vidare beskrivs införandet av Continuous Delivery i utvecklingsprojekt för monolitiska mjukvarusystem som en teknisk utmaning.

Kritiska framgångsfaktorer

Bakgrund och introduktion till kritiska framgångsfaktorer

Bullen och Rockart (1981) beskriver kritiska framgångsfaktorer som de begränsade antalen områden som ett lyckat resultat resulterar i framgång för en individ, en avdelning eller en organisation. kritiska framgångsfaktorer är de saker som måste bli rätt för att mål ska kunna nås. Den kritiska framgångsfaktormetoden blev framtagen på 70-talet av Rockart som ett sätt för direktörer att ställa krav på system så att sådant som var kritiskt för organisationen skulle tas med, närliggande koncept inom andra områden är Key Success Factors. Kritiska framgångsfaktorer används idag flitigt inom forskning samt hos konsulter för att hjälpa med strategisk planering (De Sousa, 2004). Ellis (2008) beskriver hur 68% av alla IT-projekt är troliga att misslyckas, detta gör det viktigt att ha koll på alla faktorer som kan spela in i huruvida ett projekt blir lyckat eller ej. Tillgång till och identifiering av kritiska framgångsfaktorer underlättar och hjälper företag att undvika misslyckanden och nå framgång.

Definition av kritiska framgångsfaktorer

Kritiska framgångsfaktorer beskrivs av Caralli et al. (2004) som faktorer som är kritiska för en organisations framgång. Freund (1988) skriver att det finns ett antal faktorer som krävs för att något ska kunna sägas vara kritiska framgångsfaktorer, dessa är följande:

- Viktiga för nå övergripande mål
- Mätbara och kontrollerbara
- Enbart faktorer som faktiskt är kritiska
- Saker att göra, inte slutmål
- Generella, inte företagsspecifika
- Hierarkiska

(Freund, 1988)

Bullen och Rockart (1981) delar inte helt denna syn på kritiska framgångsfaktorer utan skriver hur kritiska framgångsfaktorer bör ses utifrån en managers synvinkel, på en avdelning under en viss tid. Här skiljer sig litteraturen då Bullen och Rockart (1981) använder sig av en version där de kritiska framgångsfaktorerna kan vara företagsspecifika och subjektiva, tillskillnad från Freund (1988) som anser att de bör vara generella och ej företagsspecifika.

Aspekter av kritiska framgångsfaktorer

Kritiska framgångsfaktorer kan delas in i ett flertal olika kategorier av aspekter där kritiska framgångsfaktorer kan identifieras. Chow och Cao (2008) beskriver hur de delar in kritiska framgångsfaktorer i fem olika aspekter för agila projekt.

- **Organisationsaspekter**

Under organisationsaspekten ingår problemkomplex som t.ex. ledningsstöd, och företagskultur.

- **Människoaspekter**

Människoaspekten handlar om frågor som t.ex. att ha kompetenta och motiverade medarbetare.

- **Processaspekter**

Processaspekterna handlar om processerna, t.ex. att följa agila arbetsmetoder.

- **Teknikaspekter**

Teknikaspekten innehåller t.ex. saker som att ha en väldefinierad kodstandard.

- **Projektaspekter**

Projektaspekten handlar om projektensutformning

(Chow och Cao, 2008)

Vikten av kritiska framgångsfaktorer

Leidecker och Bruno (1984) beskriver hur kritiska framgångsfaktorer är av vikt för att direktörer och företagsledare ska kunna beskriva vilken information som är viktig för dem. Leidecker och Bruno (1984) skriver även att kritiska framgångsfaktorer är ett viktigt verktyg för strategisk planering och företagsstrategi. Då IT-projekt misslyckas i 68% av fallen blir all kunskap som kan underlätta för ett IT-projekt av värde. Bullen och Rockart (1981) skriver om hur det för varje manager endast finns ett begränsat antal riktigt viktiga områden som en manager måste fokusera på, dessa områden är de faktorer som är kritiska för framgång. Syftet med kritiska framgångsfaktorer är enligt Bullen och Rockart (1981) att få managers att fokusera på dessa kritiska framgångsfaktorer istället för att lägga resurser på områden som inte är lika kritiska.

Dessa områden är i regel redan kända av de som arbetar med dem enligt Bullen och Rockart (1981). Däremot är de oftast outtalade, vikten av arbete med kritiska framgångsfaktorer är att få dessa faktorer att gå från att vara outtalade till att bli uttalade. Efter att de kritiska framgångsfaktorerna blivit uttalade kan de enligt Bullen och Rockart (1981) börja användas av organisationer i deras planeringsprocesser och för att stödja det praktiska arbetet med utveckling och införande. Då kritiska framgångsfaktorer visar de områden som där bra resultat måste nås, är det till stor hjälp för en organisation om dessa tas med i planeringen av projekt, enligt Bullen och Rockart (1981) så är det en bra idé att för en organisation eller

individ bestämma sig för sina kritiska framgångsfaktorer och därefter allokera resurser för att se till så att de uppfylls.

Identifiera kritiska framgångsfaktorer

Leidecker och Bruno (1984) beskriver flera olika kategorier av metoder för att identifiera kritiska framgångsfaktorer samt hur och när de bör användas, Vi har valt att lyfta fram en specifik kategori av metoder som av Leidecker och Bruno (1984) kallas för branschexperter för det här arbetet.

Att använda sig utav branschexperter innebär att man hämtar in åsikter och erfarenheter från experter inom det område som undersöks. Leidecker och Bruno (1984) skriver att den information som hämtas in från en experts insikt i ett ämne ofta är en fantastisk källa för kritiska framgångsfaktorer, framförallt kan denna metod användas för att få fram sådana kritiska framgångsfaktorer som lätt missas av mer objektiva och analytiska identifieringsmetoder. Enligt Leidecker och Bruno (1984) utförs denna typ utav kritisk framgångsfaktor identifikation genom att ställa de rätta frågorna till lämpliga personer och sedan tolka informationen korrekt. Risken finns dock att den information som samlas in från experterna är subjektiv och eventuellt inte är trovärdig.

Framgångsfaktorer för Continuous Delivery

Craig (2016) framförde i webbseminariet ”Release Automation as a Catalyst for Continuous Delivery” fem framgångsfaktorer för införandet av Continuous Delivery. Hon har härlett dessa ur intervjuer med klienter till branschanalys- och konsultföretaget Enterprise Management Associates där hon är verksam som forskningsdirektör.

1. Ledningens engagemang

Enligt Craig (2016) är ledningens engagemang viktigt, över 60% av Continuous Delivery införanden sker på ledningens initiativ. Den här framgångsfaktorn förklaras inte djupare än så.

2. Förståelse för att DevOps bidrar till verksamheten och inte bara är IT

Förståelse för affärsvärdet av förändringarna gör så att det inte bara ses som en IT-investering och organisationer blir därmed mer öppna till förändring.

3. Övergång till väldefinierade, övervakade och upprätthållna processer

En dålig process är en dålig process oavsett om den utförs snabbt och effektivt med hjälp av automation, man behöver förändra processer istället för att blint automatisera befintliga (Craig, 2016).

4. Villighet att anamma samarbete och planering mellan avdelningar

Avdelningar baserat på kompetens kommer alltid att behövas och att bryta ner organisationsstrukturen helt är inte önskvärt, däremot kan man samarbeta bättre mellan avdelningar.

5. Sikta på att automatisera över hela livscykeln

Automation ökar möjligheterna till repetition, granskning, förändringskontroll och flexibilitet vilket över tid leder till mätbar reduktion för förändringsrelaterade produktionsincidenter (Craig, 2016).

Teoretiskt ramverk

Det teoretiska ramverket är både den litteratur, forskning och information som finns tillgänglig angående Continuous Delivery samt den forskning vi anser vara relevant för arbetets syfte gällande Kritiska framgångsfaktorer. I den undersökning som görs ämnar vi att jämföra vilka Kritiska framgångsfaktorer vi hittar i praktiken med vad litteraturen, forskningen och informationen som finns tillgänglig för Continuous Delivery säger. Listan över Kritiska framgångsfaktorer framtagna av Craig (2016) är också något vi vill återkoppla till med de Kritiska framgångsfaktorer vi påträffar.

Continuous Delivery är ett relativt nytt begrepp men har en historia som sträcker sig tillbaka till agila metoder som användes på sent 90-tal (Wells, 1997, 1999). Vi har valt att, både med bakgrund av det och på grund av att snarlika tillvägagångssätt som Continuous Integration och Continuous Deployment finns tillgängliga, återge en klar bild av vad Continuous Delivery faktiskt innebär. Vi anser att det är viktigt att göra på det sättet, dels för att undvika att de olika Continuous termerna förväxlas men också för att försäkra att en klar bild av vad Continuous Delivery innebär finns tillgänglig. Vidare har vi valt att återge en uppfattning av vad förespråkare av Continuous Delivery menar är viktiga områden att fokusera på under ett införande, i form av principer och upplevda utmaningar.

Kritiska framgångsfaktorer är faktorer som måste bli uppfyllda för att ett projekt ska kunna framgångsrikt genomföras. Fördelen med kritiska framgångsfaktorer är att de ger beslutsfattare en klar bild över vad som måste utföras för att projektet ska bli lyckat och de kan därmed använda detta i planeringen. Vi har valt att ge en kortare introduktion och bakgrund till kritiska framgångsfaktorer, definierat vad kritiska framgångsfaktorer är, visat på de olika aspekter kritiska framgångsfaktorer kan bestå utav samt redogjort för kategoriseringar av de områden där kritiska framgångsfaktorer kan identifieras. Vidare har vi presenterat den metod för identifiering av kritiska framgångsfaktorer som vi kommer att använda oss utav.

För att ta fram kritiska framgångsfaktorer för införandet av Continuous Delivery måste vi definiera vad framgång i den här kontexten betyder. Vi har valt att definiera framgång för ett införande av Continuous Delivery som ett införande där man framgångsrikt inför alla tekniska delkomponenter som utgör Continuous Delivery samt använder sig utav dessa för att

möjliggöra en frekvent leverans av ett mjukvarusystem efter ett införande. Vi menar att ett införande som inte uppnår dessa kriterier alltså inte kan anses vara framgångsrikt.

3

Metod

I det här kapitlet presenterar vi den metod vi har valt för att samla in och analysera våra empiriska data. Vi går även igenom hur urval och genomförande har gått till. Slutligen diskuterar vi undersökningskvalitet och etiska aspekter.

Metodval

Jacobsen beskriver hur en kvalitativ metod ger få begränsningar, och passar bra för att samla in information om detaljer och nyanser. Vi har på grund utav detta valt att använda oss utav en kvalitativ snarare än en kvantitativ metod eftersom att vi är ute efter djupgående svar på vilka problemområden och utmaningar som Continuous Delivery för med sig så att vi kan utgå ifrån dessa för att ta fram vilka kritiska framgångsfaktorer som finns vid ett införande. En av fördelarna med en kvalitativ metod är att svaren ligger öppna och eftersom att vi inte vet vilka svar vi kommer att få anser vi att en kvalitativ metod är att föredra över en kvantitativ sådan (Jacobsen, 2002).

Vi vill genomföra ett antal intervjuer för att sedan analysera och jämföra dessa med varandra, för att se om det finns några synvinklar och åsikter som tas upp gemensamt av flera. Dessa kan då rimligtvis antagas vara användbara även för andra företag som vill införa Continuous Delivery. När vi sedan granskar den insamlade informationen med bakgrund av litteraturen kan vi härleda vilka de kritiska framgångsfaktorerna kan tänkas vara för projekten som våra intervjupersoner har varit delaktiga i.

Urval

Vi har valt att intervjua två operativa/strategiska configuration managers, en förändringsledare och en person som skiftar roll mellan arkitekt, koordinator och utvecklare. Alla intervjupersoner har erfarenhet av att arbeta med införanden av Continuous Delivery. Vi anser att det här urvalet ger en mångfald baserat på skild projekterfarenhet samt olika roller inom införandet. Trots att vi endast använder oss utav fyra intervjupersoner anser vi att bredden på undersökningen är god med bakgrund av dessa intervjupersoners roller i införandeprojekten och deras samlade erfarenhet som sträcker sig över ett 30-tal projekt utförda på svenska företag av varierande storlek. En av våra respondenter önskade anonymitet, både för företaget och för sig själv, vilket är något vi tagit hänsyn till.

Företag	Funktion	Namn	Metod	Bilaga
Anonymt företag	Utvecklare/Arkitekt /Koordinator	Anonym Respondent	Telefon	IP1
Softhouse Consulting	Operativ/Strategisk CM	Pendleton, Christian	Fysisk	IP2
Softhouse Consulting	Operativ/Strategisk CM	Rigo, Albert	Fysisk	IP3
Diabol	Förändringsledare	Rehn, Andreas	Skype	IP4

Tabell 1

Det anonyma företaget är ett IT-konsultföretag främst fokuserat på konsult- och utvecklingsverksamhet inom IT-systemintegration. Företaget anpassar och implementerar integrationssystem baserade på den arkitektoniska integrationsmodellen Enterprise Service Bus.

Softhouse Consulting är ett konsultföretag inriktat på mjukvaruutveckling, Softhouse har cirka 200 anställda och kallar sig själva ett av de ledande företagen i Skandinavien på Lean & Agile. Softhouse har kontor i ett flertal svenska städer, med huvudkontoret beläget i Malmö.

Diabol är ett IT-konsultföretag främst inriktat på systemutveckling i Java men med en särskild fokusering på Continuous Delivery. Företaget har omkring 15 anställda och är verksamma främst i Stockholmsområdet.

Genomförande av intervjuer

Vi väljer att huvudsakligen genomföra fysiska intervjuer istället för telefonintervjuer eller liknande eftersom att vi vill hålla intervjuer som förlitar sig utav öppna frågor. Enligt Jacobsen (2002) har vissa undersökningar visat att det kan bli svårare att erhålla ärliga svar i en telefonintervju, speciellt om intervjufrågorna är mestadels öppna. Vi har dock varit tvungna att utföra en intervju över telefon och en intervju över Skype på grund av geografiska avstånd. Hela intervjuerna spelas in digitalt för att försäkra oss om att ingen information går förlorad. Anteckningar under intervjun kommer även att tas till en viss grad. Vi avser att hålla intervjun till ungefär en timme, eventuella variationer kan uppstå beroende på vilken information vi kan erhålla på den tiden. Enligt Jacobsen (2002) är mellan en timme och en och en halv timme den optimala tidsramen för en intervju och vi planerar att hålla oss inom den.

Där det är möjligt försöker vi att hålla intervjun på en neutral plats eftersom att intervjupersonen enligt Jacobsen (2002) kan känna sig bekväm med att uttrycka sina åsikter friare där, jämfört med t.ex. hos en kund eller på intervjupersonens konsultföretag. Vi hoppas att detta leder till att kontexteffekten hålls så låg som möjligt. Avsikten med intervjun bör hållas öppen eftersom att vårt ämne är i sin natur ett okänsligt ämne och risken för att intervjupersonen förvränger sina svar anses vara minimal (Jacobsen, 2002).

Intervjuguidens utformning

Intervjuguiden börjar med mer generella frågor innan vi ställer specifika frågor om vilka utmaningar som upplevs vid ett införande av Continuous Delivery, på grund av att vi vill få en bakgrund av vad svaren för de senare frågorna grundar sig i. Jacobsen (2002) skriver att det är bra att inleda med mer generella och öppna frågor då detta ger intervjupersonen chansen att säga mer istället för att endast ge svar på våra mer specifika frågor.

Intervjuguiden för våra intervjuer är utformad för att fånga intervjupersonens personliga erfarenheter, åsikter och lärdomar. Istället för att fråga direkt om vilka områden som uppfattas som kritiska framgångsfaktorer har vi valt att utforma intervjuguiden med öppna och generella frågor som startar en konversation om Continuous Delivery. Tanken är att vi ska kunna komma in på utmaningar och problemområden naturligt.

Trots att frågorna om utmaningar, vilket är vad vi med undersökningen vill lära oss mer om, utgör endast en liten del av intervjuguiden har vi lagt stor fokus där under intervjuerna. Genom att ställa frågan om utmaningar öppet vill vi få veta vad intervjupersonen anser vara viktigt att nämna, istället för att ställa frågor som indirekt leder intervjupersonen till att svara som vi tror att han borde.

Bearbetning och analys av intervjuer

Under intervjuerna gjordes vissa anteckningar som vi sedan använde som grund när vi lyssnade igenom intervjuerna och transkriberade vad som sades. Jacobsen (2002) skriver att det oftast är nödvändigt att förenkla insamlad information för att göra det enklare att arbeta med. Till viss del har vi därför parafraserat svaren och för ett fåtal frågor, som en följdfråga eller ett förtydligande, har vi lagt förtydligandet i svaret för huvudfrågan. Vi har även valt att ersätta svordomar med mildare ord som har liknande betydelse när dessa har förekommit.

Eftersom att vi valde att hålla en intervju baserad på öppna frågor är både intervjuerna och transkriptionerna någorlunda ostrukturerade. Efter transkriberingsarbetet valde vi att sammanställa citat från olika personer om utmaningar baserat på ämne för att bättre få en översikt över vem som tagit upp vilka utmaningar. Maxwell (2004) skriver hur ett vanligt misstag vid kvalitativa studier är att låta transskript och anteckningar samlas på hög, för att undvika detta har vi kontinuerligt analyserat den insamlade informationen samtidigt som vi samlat in ny information, detta har gett oss en bra bild över den information vi haft tillgänglig och underlättat för analysen. För att analysera har vi valt att använda oss av en kategoriserande strategi. Maxwell (2004) beskriver hur kodning är lämpligt för att bryta ner den insamlade informationen och arrangera den i kategorier för att sedan jämföra informationen i dessa kategorier.

Undersökningskvalitet

Validitet och Reliabilitet

När en intervju genomförs finns det två grundkrav som måste uppfyllas för att säkerställa att intervjun är riktig och ger riktiga resultat, informationen som samlas in måste vara giltig och relevant men också trovärdig och tillförlitlig (Jacobsen, 2002).

Dialogisk validering sker i form av följdfrågor och förtydliganden under intervjun att försäkra att svaret har uppfattats korrekt (Jacobsen, 2002). För att ge undersökningen en god validitet har vi både använt oss utav dialogisk validering samt bett om intervjupersonens godkännande av transkriptet i efterhand. Intervjuareffekten som beskrivs av Jacobsen (2002) är något vi har haft i åtanke både innan och under intervjuerna. Genom att strukturera intervjuerna öppet vill vi öka trovärdigheten och tillförlitligheten så att vi inte indirekt leder intervjupersonerna att svara som vi eventuellt tror att de kommer att göra. Intervjuerna har genomförts och planerats mer som samtal än strikt strukturerade med stängda frågor som ställs i följd. Enligt Jacobsen (2002) ger en mindre formell intervju möjlighet för intervjupersonen att öppna upp mer.

Det finns två problem för validiteten och reliabiliteten av den här undersökningen. När det kommer till konsekvens mellan intervjuer har vi varit tvungna att utföra två intervjuer över telefon och Skype istället för fysiskt som för de andra två. Dett kan ha konsekvenser för kvaliteten på undersökningen genom att svar skiljer sig baserat på kontexteffekten och den minskade tillförlitligheten hos en telefonintervju (Jacobsen, 2002). Det andra problemet härstammar ifrån vår användning av branschexperter för att ta fram kritiska framgångsfaktorer. Leidecker och Bruno (1984) menar att man genom att använda branschexperter för framtagning av kritiska framgångsfaktorer riskerar att erhålla mer subjektiva kritiska framgångsfaktorer.

Etiska aspekter

Det finns alltid etiska aspekter inom undersökningar vilka är relevanta för en intervjuare att ha i hågkomst. Tillförlitligheten hos en intervju kan i värsta fall äventyras på grund av olika etiska aspekter. Enligt Jacobsen (2002) består etiska aspekter för en undersökning av tre delar, informerat samtycke, rätt till privatliv och riktig presentation av data.

Jacobsen (2002) skriver att informerat samtycke innebär att intervjupersonen ska vara medveten om att det sker en intervju samt frivilligt delta. Informerat samtycke består utav fyra delar, kompetens, frivillighet, full information och förståelse. I det här sammanhanget betyder kompetens att intervjupersonen måste kunna förstå vad han väljer att delta i. Frivillighet innebär att intervjupersonen självmant tar beslutet att delta, utan påtryckningar. Full information innebär att intervjupersonen får full information rörande undersökningens syfte. Förståelse innebär att intervjupersonen också ska förstå den fulla informationen (Jacobsen, 2002).

Rätt till privatliv innebär att en intervjuperson ska ha rätt till att hålla viss information privat, den viktigaste aspekten av detta är att det ska vara frivilligt från respondentens sida att ingå med identifierbar information i den insamlade empirin (Jacobsen, 2002). Vår undersökning är inte privat i sin natur och vi ämnar att både erbjuda rätt till anonymitet och att inte inkräkta på intervjupersonernas privatliv. Alla respondenter blev erbjudna anonymitet men endast en respondent tackade ja till erbjudandet.

Riktig presentation av data innebär att data ska presenteras så fullständigt som möjligt, och ej manipuleras. Att plocka citat och resultat ur sitt sammanhang är något som ej bör göras. Förfalskning av resultat är självklart även det helt förbjudet. För att motarbeta allt detta bör informationen som rör undersökningen vara tillgänglig för andra, därför har vi gjort transkripten av våra intervjuer tillgängliga som bilagor (Jacobsen, 2002).

4

Resultat

Resultat från undersökningen sammanställs här, vi har valt att organisera detta kapitel utifrån vilka utmaningar som har nämnts under intervjuerna.

Automatiska tester

Under intervjuerna nämner flera respondenter att införandet utav automatiska tester är en utmaning och den här utmaningen målas upp som den största (IP3, F15; IP2, F17; IP4, F14). Respondenterna som nämner utmaningen är också överens om att det här är både en organisationell och teknisk utmaning (IP3, F16; IP2, F12; IP4, F14). Den organisationella utmaningen bottnar i att det finns en testorganisation och att dessa personer får en ny roll i utvecklingsteamet där de ska lära utvecklare att själva testa koden som de skriver, rollfördelningen blir annorlunda (IP3. F16; IP2, F17; IP4, F14). Förändringen innebär att både testarnas och utvecklarnas roller förändras. En övergång till automatiska tester innebär en ansvarsförändring där utvecklarna bär ansvaret för mjukvarusystemets kvalitet, vilket kan vara svårt att hantera (IP4, F14).

På ett tekniskt plan är införandet av automatiska tester i ett befintligt projekt en stor uppgift (IP3, F15; IP2, F17; IP4, F14). Svårigheten av den tekniska aspekten beror mycket på hur väl mjukvarusystemets befintliga kod lämpar sig för automatisk testning (IP2, F12; IP4, F13). En respondent menar att det tekniskt är en stor utmaning men att de organisationella aspekterna är svårare (IP4, F14). Ett mjukvarusystems testbarhet har kodarkitekturen som grund, en ogynnsam arkitektur hos t.ex. ett legacy-system försvårar utveckling av automatiska tester vilket kan innebära enorma kostnader (IP2, F26; IP4, F13). Om inte kodarkitekturen är god kan det innebära en stor utmaning omarbeta denna (IP2, F17; IP4, F7). Enligt en av våra respondenter lämpade sig automatisk testning illa för utveckling av integrationssystem och således användes inte den här delkomponenten av Continuous Delivery för dennes införanden (IP1, F7).

Vad som kan hända när det föreligger både organisationella och tekniska hinder är att motivationen brister, målet ligger för långt borta och känns onåbart vilket kan leda till att införandet misslyckas (IP2, F12; IP3, F12-13). En respondent nämner även att problematik kan uppstå på grund av att testcheferna eller testarna ser sitt inflytande sjunka och sin roll förändras och därför motsätter sig förändringen (IP2, F17).

Struktur

Alla respondenter har poängterat vikten av struktur eller arkitektur på olika sätt och en av respondenterna nämner den här utmaningen som ”egentligen den största utmaningen” (IP4,

F16; IP2, F17; IP1, F11). Den här utmaningen sträcker sig över flera delar och handlar om kodens struktur, namnkonvention och att beroenden mellan olika team bör lösas upp så att de kan leverera enskilt (IP2, F17; IP1, F11; IP3, F16). Enligt en respondent bygger automatiseringen av driftsättning mycket på att det finns en gemensam struktur, om den gemensamma strukturen finns kan man med lätthet automatisera (IP1, F12).

Motivation

Samtliga respondenter anser i olika grad att det innebär en utmaning att övertyga om nyttan med ett införande av Continuous Delivery inom en organisation (IP2, F17; IP3, F14; IP1, F14; IP4, F8). Eftersom att förändringen berör andra avdelningar än utvecklingsavdelningen, t.ex. testningsavdelningen och driftsavdelningen, bör nyckelpersoner som kan understödja förändringen övertygas (IP2, F17; IP3, F7; IP4, F8). Annars blir det svårare och man riskerar att införandet fördröjs eller helt avbryts på grund av bristande övertygelse (IP3, F7-8; IP2, F20; IP4, F8).

I Sverige har ingenjörer mycket inflytande över arbetet, för att förändringen då ska fungera måste alla arbeta efter samma vision, då blir kommunikation, att prata om hur det ska bli och vilka problem som finns på vägen viktigt för att få med sig medarbetare (IP2, F19). För att få med sig utvecklare visar man var problemen med traditionella tillvägagångssätt ligger och visar dem lösningen (IP2, F19; IP1, F14). I annat fall kan verktygen upplevas som ett hinder för arbetet och man riskerar att de slutar användas (IP1, F14).

> *”[Att bara] trycka ner en lösning i halsen fungerar nästan aldrig, utan jag måste först visa vad problemet är och då kan jag visa lösningen!”*
> *(IP2, F19)*

Flertalet respondenter är överens om att det behövs en vision att arbeta efter, någon som kan prata om varför det är bra eller någon som har framgångsrikt infört Continuous Delivery tidigare och vet att det kan fungera (IP2, F19; IP3, F9; IP4, F8).

Något annat som är viktigt för införandet av Continuous Delivery är att man måste kunna lita på att det faktiskt går att testa automatiskt (IP3, F17; IP2, F17). I annat fall riskerar man att fortfarande förlita sig för mycket på manuella tester, även om det finns automatiska tester, och på så sätt uteblir en effektivitetsförbättring (IP2, F12).

Ansvar och kontinuerligt arbete

”De som bara för in någonting och sedan låter det vara, det är oftast dom som misslyckas i längden”
(IP1, F20)

Flertalet respondenter menar att Continuous Delivery inte är något man inför en gång och sedan är klar med (IP2, F26; IP3, F14; IP1, F20). Att hela tiden effektivisera och förbättra verktygen som Continuous Delivery består utav är något som kontinuerligt måste utvecklas i takt med mjukvarusystemet som utvecklas (IP2, F17; IP3, F14; IP1, F20; IP4, F11). Dessutom menar respondenterna att det kan finnas en svårighet i att få någon på företaget att ta ansvar för och underhålla verktygskedjan, det är oklart var det här ansvaret hamnar organisationellt (IP3, F21; IP2, F13; IP1, F13; IP4, F11). En respondent understryker speciellt svårigheten i att få någon att ta ansvar för verktygskedjan på större företag (IP1, F13). Vissa respondenter underströk också vikten av att visualisera flödet för att hitta flaskhalsar och effektivisera allt eftersom (IP3, F22; IP1, F20; IP4, F15-16).

Processförändringar

Hela införandet av Continuous Delivery är en processförändring, först arbetar man på ett sätt med många manuella moment och sedan omarbetar man processerna och automatiserar (IP4, F15). Processförändringar sker ofta i kombination med kontinuerlig förbättring, man arbetar för med att få de viktiga delarna på plats och sedan arbetar man med att automatisera eller omarbeta mer och mer (IP1, F20; IP3, F14; IP4, F15-16). Vissa av våra respondenter har nämnt vikten av att minska påverkan av manuella godkännanden vid införandet av Continuous Delivery (IP3, F18-20, IP4, F15). En av våra respondenter menar att godkännanden kan fördröja leveransprocessen av ett projekt vilket kan leda till att andra delar av Continuous Delivery fungerar väl, men att viktiga nyckelpersoner inte ser nyttan med förändringen på grund av tidsåtgången (IP3, F18-20). Manuella godkännanden är dock inte nödvändigtvis omöjliga att använda i kombination med Continuous Delivery enligt samma respondent, om godkännandena är tidsenliga och effektiva.

5

Diskussion

I det här kapitlet förenar vi resultaten från den empiriska undersökningen med teorin från litteraturen och våra egna tankar för att ta fram vilka de kritiska framgångsfaktorerna för ett införande av Continuous Delivery är, hur de kategoriseras samt jämför dessa mot befintliga framgångsfaktorer.

Kritiska framgångsfaktorer

- Skapa en förändringsvillighet hos medarbetare

Flera respondenter påpekade vikten av att det fanns någon som drev förändringen, men inte nödvändigtvis endast på beslutsfattande nivå. Det handlar om att motivera medarbetare till förändringen och att visa vad som kan göras bättre. Vidare är det enligt respondenterna av vikt att arbetet fokuseras och att man jobbar mot samma mål, att ett fåtal är motståndiga är utgör inte ett problem men merparten måste deltaga i arbetet för förändring.

För att kunna införa automatisk testning behöver testningsorganisationen och utvecklingsorganisationen förändras, istället för att vara separata avdelningar ska dessa integreras i utvecklingsavdelningen. Det här innebär att både medarbetare som arbetar med testning och medarbetare som arbetar med utvecklings får sina arbetsuppgifter förändrade. Dessutom förflyttas ansvaret för mjukvarusystemets kvalitet från testningsavdelningen till utvecklingsavdelningen. Det här är en stor förändring och flera av våra respondenter samt Chen (2015) tar upp organisationella förändringar som en svår utmaning. Även intressenter som inte är direkt involverade i utvecklingsarbetet får eventuellt mindre inflytande i form av manuella godkännanden och driftsavdelningens arbete påverkas av förändringarna.

Vi anser att införandet av alla tekniska delkomponenter som utgör Continuous Delivery samt tillvägagångssättets fortlevnad efter ett införande är direkt beroende av den här faktorn. Om inte förändringsviljan finns riskerar man enligt våra respondenter att inte alla tekniska delkomponenter införs, att omorganiseringen inte genomförs och att de tekniska delkomponenterna slutar att användas, upplevs som ett hinder eller faller i glömska. Vi hittar även stöd i litteraturen från Humble and Farley (2010) som talar om vikten av automatisering och processförändring vilket i förlängningen innebär en förändring av arbetssätt och organisation.

- Utveckla en oberoende struktur

Vi anser att en välplanerad och gemensam struktur är en kritisk framgångsfaktor vid införandet av Continuous Delivery. Vi har i vår studie funnit att en välplanerad struktur är en förutsättning för att lyckas effektivisera leveransprocessen. Strukturen mellan olika projekt inom företaget måste vara utformad för att möjliggöra att separata leveranser kan ske, detta

innebär att beroenden mellan olika projekt bör lösas upp för att separata team ska kunna gå igenom leveransprocessen oberoende utav varandra.

Chen (2015) beskriver hur de för att underlätta införandet av Continuous Delivery var tvungna att förändra företagsstrukturen på det företag han arbetade för att göra samarbete över avdelningar lättare i en miljö där det finns ett flertal olika aktörer med egna intressen. Även för att uppnå den princip som Humble and Farley (2010) skriver om som handlar om att automatisera nästan allt krävs det enligt vår empiriska studie att strukturen är både välplanerad och gemensam för de olika teamen, när strukturen finns på plats blir det enligt vår empiriska studie enkelt att automatisera test och driftsättning. En välplanerad och gemensam struktur är ett krav för att nå den definition av framgång vid ett införande av Continuous Delivery som vi använder oss utav, utan en oberoende struktur kan man ej möjliggöra en frekvent leverans eftersom att beroenden försvårar både leverans samt utvecklingen av automatiserad driftsättning.

- Försäkra att kodarkitekturen lämpar sig för testbarhet

I vår empiriska studie har vi funnit att en testbar arkitektur för ett mjukvarusystem är en kritisk framgångsfaktor vid införandet av Continuous Delivery. Enligt våra respondenter påverkar en kodarkitektur införandet av både den automatiserade driftsättningen och införandet och utvecklingen av automatiska tester. Chen (2015) nämner att det på Paddy Power var en teknisk utmaning att införa Continuous Delivery för utveckling av monolitiska mjukvarusystem. Även flertalet utav våra respondenter nämner arkitekturella utmaningar i ett införande utav Continuous Delivery som en viktig faktor för ett framgångsrikt införande. Särskilt inom införande av Continuous Delivery i äldre mjukvarusystem är arkitekturen en svåröverkomlig utmaning enligt våra respondenter. Våra respondenter menar att om kodarkitekturen inte är utformad med testbarhet i åtanke så kan det innebära att införandet blir svårt, dyrt eller i värsta fall omöjligt. För att framgångsrikt och tidsenligt kunna införa alla de delkomponenter som Continuous Delivery består utav så krävs det att arkitekturen stödjer detta. För att uppnå den definition av framgång som vi bestämde i det teoretiska ramverket är en testbar kodarkitektur en nödvändighet.

- Stöd från företagsledningen

I den empiriska studien fann vi att för att möjliggöra de organisationella, procedurella och tekniska förändringar som ett införande av Continuous Delivery enligt Chen (2015) kräver måste det finnas ett stöd från företagsledningen. Kommunikation mellan avdelningar behöver underlättas, samarbete mellan avdelningar behöver ske i högre grad och utvecklings- och kvalitetssäkringsorganisationen behöver omorganiseras. Humble och Farley (2010) menar att en mer generell optimering är starkt att föredra framför lokala optimeringar. Utan ett stöd från företagsledningen begränsas möjligheterna för en mer generell optimering.

Stöd från företagsledningen är av speciellt viktig innebörd för att ett införande utav Continuous Delivery ska uppnå våra kriterier för framgång. När stödet inte finns riskerar ett

införande att bli inkomplett eller avbrytas helt vilket, enligt vår definition av framgång, skulle innebära ett misslyckat införande.

- Definiera ansvarsfördelning för delkomponenterna

Flera av våra respondenter påpekade vikten av att delkomponenterna som utgör Continuous Delivery måste tilldelas ansvar för och vissa beskrev även att få någon att ta ansvar för dem som en svårighet. Det kan vara oklart var ansvaret för dessa delkomponenter hamnar, t.ex. om ansvaret hör hemma hos utvecklarna eller hos teknikerna på driftsättning och underhåll. Även Humble och Farley (2010) påpekar vikten av att dessa delkomponenter underhålls och utvecklas i takt med att mjukvarusystemet utvecklas. I annat fall riskerar man att dessa system försvårar utvecklingsarbetet istället för att underlätta det eller att de inte är relevanta för mjukvarusystemets nuvarande status. Enligt en respondent skulle detta kunna leda till att man slutar använda sig utav dessa delkomponenter. Det skulle i så fall innebära att införandet misslyckats enligt vår definition av framgång för ett införande. För att försäkra sig om att Continuous Delivery fortlever efter ett införande behöver ansvaret för att underhålla och utveckla de delkomponenter som Continuous Delivery består av tilldelas.

Aspekter av våra kritiska framgångsfaktorer

- Skapa en förändringsvillighet hos medarbetare

På grund av att den här kritiska framgångsfaktorn handlar om en vilja eller motivation hos medarbetarna att förändra sitt arbetssätt och förändra hur olika arbetsuppgifter utförs anser vi att den passar in under Människoaspekter i de aspekter som Chow och Cao (2008) har tagit fram för agila införanden.

- Utveckla en oberoende projektstruktur

Denna kritiska framgångsfaktor faller in under två utav de aspekter utav kritiska framgångsfaktorer som Chow och Cao (2008) beskriver, det är dels en processaspekt, på grund av att det handlar om att ändra arbetsprocesser genom att ändra projektstrukturen så att de stödjer Continuous Delivery. Framförallt rör sig detta om en projektaspekt då det faktorn handlar om är hur projekten ska utformas vilket är själva kärnan i vad Chow och Cao (2008) kallar för en projektaspekt.

- Försäkra att kodarkitekturen lämpar sig för testbarhet

Den här kritiska framgångsfaktorn handlar om ett rent tekniskt område, och inbegriper bland annat att hålla sig till en väldefinierad kodstandard. Den här kritiska framgångsfaktorn faller tydligt in under de tekniskaspekterna enligt den indelning som Chow och Cao (2008) har gjort.

- Stöd från företagsledningen

Den här kritiska framgångsfaktorn berör ledningsstöd direkt, vilket är en av de kritiska framgångsfaktorerna som Chow och Cao (2008) har definierat Organisationsaspekter med hjälp utav. Därför hör den tydligt hemma under organisationsaspekter.

- Definiera ansvarsfördelning för delkomponenterna

Att definiera en ansvarsfördelning hör hemma under organisationsaspekter som definierat av Chow och Cao (2008) eftersom att en tilldelning av arbetsuppgifter bör ske.

Jämförelse mot tidigare framgångsfaktorer

De kritiska framgångsfaktorer som vi har tagit fram skiljer sig ifrån men har även en del gemensamt med faktorerna som Craig (2016) har tagit fram. Våra framgångsfaktorer tar upp vikten av struktur och ansvarsfördelning men hennes fokuserar mer på vad Continuous Delivery utgörs utav. En processförändring och en övergång till automatisering är naturligtvis nödvändig när man förändrar hur leveransen av mjukvarusystem sker enligt Continuous Delivery. Vi anser att om man inte har för avsikt att göra det så är det inte Continuous Delivery som man inför och därför återfinns inte dessa framgångsfaktorer bland våra kritiska framgångsfaktorer.

Istället har vi fokuserat på förutsättningar för att ett införande ska vara möjligt, framgångsrikt och leda till en effektivisering av mjukvarusystemsleverans. Våra kritiska framgångsfaktorer handlar om teknisk arkitektur, projektstruktur och ansvarsfördelning vilket vi inte har kunnat återfinna bland hennes. Vissa av framgångsfaktorerna som Craig (2016) har tagit fram rör gemensamma områden och vi kan återfinna en av våra framgångsfaktorer bland hennes, den som berör vikten av ledningsstöd. Även villighet att anamma samarbete och förståelse för DevOps bidrag till verksamheten är liknande vissa av våra faktorer men överensstämmer inte fullt ut.

6

Slutsats

Vårt syfte var att ta fram vilka kritiska framgångsfaktorer som finns vid införandet av Continuous Delivery. Våra respondenter är valda på grund av en bred erfarenhet av införanden av Continuous Delivery. För att identifiera kritiska framgångsfaktorer har vi valt att använda oss utav kvalitativa intervjuer med fokus på vilka utmaningar som upplevs under införanden. Genom att ta fram vilka utmaningar som upplevs vid ett införande har vi kunnat härleda dessa till fem kritiska framgångsfaktorer som finns vid ett införande av Continuous Delivery.

- Skapa en förändringsvillighet hos medarbetare

Att det finns en förändringsvilja hos medarbetare är av särskild vikt. En motvilja mot förändringen eller en oförståelse för syftet av förändring kan innebära ett avbrutet eller inkomplett införande.

- Utveckla en oberoende struktur

Strukturen för projekt måste både vara strukturerad och oberoende för att ett införande ska lyckas enligt vår definition. Automatisering av processer och effektiv leverans av mjukvarusystem förlitar sig på en god struktur.

- Försäkra att kodarkitekturen lämpar sig för testbarhet

För att kunna erhålla en god täckning av automatiserade tester måste kodarkitekturen lämpa sig för automatisk testning. I annat fall kan utvecklingen av automatiska tester bli svårt, dyrt och i värsta fall omöjligt.

- Stöd från företagsledningen

För att möjliggöra de förändringar som ett införande av Continuous Delivery innebär måste det finnas ett stöd från företagsledningen. Vid ett uteblivet stöd blir omorganiseringar och förändring utav ansvarsområden kraftigt försvårat.

- Definiera ansvarsfördelning

Under ett införande behöver ansvar tilldelas för att underhålla de delkomponenter som utgör Continuous Delivery. Mjukvarusystem utvecklas och delkomponenterna måste utvecklas tillsammans med det för att även i fortsättningen kunna användas vara till hjälp vid utveckling.

Vi har genom att ta fram dessa fem kritiska framgångsfaktorer besvarat vår forskningsfråga samt bidragit till forskningsunderlaget för Continuous Delivery. På grund av att vi använde metodkategorin branschexperter för att ta fram dessa kritiska framgångsfaktorer innebär det

en ökad risk för subjektivitet för våra kritiska framgångsfaktorer. Eftersom att vi inte utförde alla intervjuer på samma sätt finns det även en risk för att undersökningen har en försvagad reliabilitet.

Appendix Intervjuguide

Hur länge har du arbetat med utveckling?

Hur länge har du arbetat på företaget?

När kom du först i kontakt med Continuous Delivery?

Hur många projekt har du varit med i där Continuous Delivery införts?

Hur anser du att införandet av Continuous Delivery påverkar ett utvecklingsprojekt?

Har du varit med om att Continuous Delivery införs inom ett redan påbörjat projekt?

Har du varit med om att ett projekt påbörjas där man inför Continuous Delivery direkt?

Har du varit i ett projekt där införandet av Continuous Delivery har varit misslyckat?

Har du varit i ett projekt där införandet av Continuous Delivery har varit lyckat?

Vem var initiativtagare till att börja införa Continuous Delivery?

Vilka fördelar med införandet utav Continuous Delivery?

Vilka nackdelar med införandet utav Continuous Delivery?

Har du stött på några utmaningar inom införandet utav Continuous Delivery?

Hur kunde dessa utmaningar lösas?

Hur skulle dessa utmaningar kunna förebyggas?

Fanns det risk att utvecklingsprojektet misslyckades med bakgrund av tidsplanering, budget eller dylikt?

Hur har utvecklingsarbetet behövt förändras på grund av Continuous Delivery?

Finns det något som du har upplevt som extra viktigt för att ett införande ska bli lyckad?

Vilka är de viktigaste fokuseringspunkter för ett lyckat införande av Continuous Delivery?

Vad anser du om principerna för frekvent mjukvaruleverans?

Appendix IP1

1:Hur länge har du arbetat inom it eller utveckling?

Det blir tio elva år.

2:Hur länge har du arbetat på ENFO?

Lite över sex år har jag jobbat på Enfo.

3:Vad är din roll?

Högst skiftande. Allt möjligt, utvecklare, arkitekt, koordinator osv.

4:När kom du först i kontakt med Continuous Delivery?

Oj, kanske.. ganska många år sedan, 5-6 år sedan kanske.

5:Hur många projekt har du varit med i där Continuous Delivery har införts?

Hur många som helst, mellan tummen och pekfingret kanske 10, där jag har varit med och införa det alltså.

6:Hur anser du att införandet Continuous Delivery påverkar ett utvecklingsprojekt?

Hur det påverkar ett utvecklingsprojekt.. Oftast i min erfarenhet så handlar det inte så mycket om det projekt specifika utan snarare mer om process och organisation. Mm man kommer till en plats där man inte har har jobbat med detta tidigare så vill man inte göra så stora ingrepp i dom nuvarande bygg och deploy processerna utan vill oftast anpassa sin uppsättning till någonting de känner igen. Så för själva projektet har det oftast bara en fördel med att man kan leverera med kvalité och ah lite mer effektiva processer, i det stora hela, att förändra den nuvarande uppsättningen till något mer dynamiskt.

7:Har du varit med om att Continuous Delivery införs inom ett redan påbörjat projekt?

Absolut.

8:Har du varit med om att ett projekt påbörjats med hjälp utav Continuous Delivery?

Jepp, dock i intergrationsvärlden, problemet inom integration är oftast automatiska tester, oftast är det, jag ska inte säga omöjligt men det är svårt att få något värde i de testerna, du behöver extrem bra ramverk och referensarkitektur för att kunna sätta upp testcase som ger något mervärde mer än att bara bocka av att det är testat. Oftast när vi pratar om Continuous Delivery och Continuous Intergration inom integration så handlar det om hela kedjan, automatiska byggen, automatiska deployer, policy violations osv, men ibland och ibland inte ingår det automatiska tester, beror mycket på kund och var dom är i sin livscykel. I varje projekt där vi är med och implementerar någonting sånt här så är hela implementationen en iterativ process där vi hela tiden arbetar med förändring och utvecklar och förbättrar din process.

9:Har du varit i ett projekt där införandet av CD varit misslyckat?

Nej, faktiskt inte, snarare tvärtom, varje projekt har tillfört något mervärde, och inte bara för projektet utan för organisationen och verksamheten som helhet. Vi har alltid lite rolig statistik där vi tittar på en process innan införandet av t.ex. automatiska byggen, vissa kunder har processer där själva bygget av ett deploy-paket kan ta allt från en timme till en halv dag. Och när du då automatiserar bort det så sparas den tiden och istället kan det göras på sekunder eller minuter. Vid alla projekt jag har varit med om vid införandet så har det ansetts vara lyckade.

10:Vem brukar vara iniativtagare till Continuous Delivery?

Jätte olika beroende på vilken organisation det är och hur den är strukturerad, i de senare fallen kan det röra sig om allt från tekniskt ansvariga för platformen, eller platformsansvariga, ibland är det högre upp i kedjan så att säga. Ofta så är ju vi iniativtagare och försöker föklara fördelarna med att implementera något sånt här. Vi kan också påvisa att själva införandeprojektet inte är ett jättestort projekt utan vi kan applicera oss på deras nuvarande bygg och deployprocess i ganska stor utsträckning vilket gör att själva ingreppet för att göra detta inte påverkar deras day to day business.

11:Vad har du stött på för utmaningar när du försökt införa Continuous Delivery?

Det handlar ju mycket om att indoktrinera hur, min inställning till automatiseringen bygger egentligen på fyra komponenter, struktur, struktur, struktur struktur, utan struktur har du en rätt stor utmaning för att kunna automatisera någonting. Den största utmaningen är att försöka få de olika domänansvariga att ha den insikten av en namnkonvention, projektstrukturen, kodstruktur och så vidare. Du vinner ganska mycket på att sätta en sådan struktur och ibland är det helt nödvändigt för att kunna införa det, det är väl egentligen den största utmaningen.

12:Struktur alltså?

Nja, eller indoktrinering för att kunna sätta den strukturen. På vissa företag har du massor av olika utvecklare olika bakgrunder och inställningar, och olika struktur, det innebär att om du går till t.ex. versionhantering så har de strukturerat kod och projekt på olika sätt, att då automatisera någon form utav byggprocess på det när det är olika struktur är väldigt svårt, därför bygger oftast automatisering på att du sätter en gemensam struktur. Finns det kan vi med lätthet automatisera någonting.

13:Kommer du i kontakt med fler utmaningar?

Absolut, ansvarsbiten är alltid, eller kanske inte alltid men det kan vara en utmaning beroende på vilken organisation och verksamhet du är hos, vem ansvarar för servern? Vem äger komponenterna? Tittar vi på olika organisationer så är de oftast uppsplittade på olika domäner, det är då man börjar prata om dev ops och vilken kultur företaget har. Vem tar vilket ansvar? Vi promotar alltid att en från operations och en från utveckling ska vara nyckelpersoner, de ska vara ansvariga för själva lösningen, oftast är inte själva integrationsmotorn själva problemet utan allt kringliggande. Men ansvarsbiten blir alltid ett problem, och speciellt i större organisationer blir det en utmaning. Man tycker att det är inte ens ansvar, dock när man kört detta och de förstår vad det är man försöker lösa så är det oftast inte ett problem, men det är oftast ett motstånd innan man har visat på mervärdet av det.

14:Är det också en utmaning, att kunna visa värdet?

Nja, inte riktigt, men du måste ju kunna påvisa mervärdet från det. I vissa mindre organisationer även om jag skulle promota det så kanske de inte ser någon större vinning av det, nu, men om vi då kan förklara vinningen över tid, oftast växer ju en verksamhet, och deras integrationslandskap kommer med all säkerhet växa. Det är därför det även är viktigt att få en mindre organisation att investera i det även om de inte har den största vinningen initialt utan över tid, det är så du kan få dem att första mervärdet, genom att sälja in varför, annars har du problemet att du för in någonting som de inte förstår varför de ska ha, och då kan de bli så att de över tid slutar använda då de inte ser vinningen. Jag har varit på många ställen där man tidigare använde t.ex. maven för automatiska byggen, det har man liksom slutat med över tid då det är mer komplext och kräver mycket konfiguration för att få det att fungera på ett tillfredsställande sätt. Då blir verktyget bara i vägen för det man vill åstadkomma, så över tid slutar man använda och då ligger det bara där som legacy utan någon som ansvarar eller har kunskap över det.

15:Skulle du säga att det är viktigt att man väljer rätt verktyg så att det passar dom som jobbar på företaget?

Nej, det beror på hur du menar med verktyg, vi har alltid inställningen att för att det här ska bli lyckat behöver du ingen projektspecifik konfiguration för att det ska fungera, det ska fungera dynamiskt, sen vilken verktygslåda du vill stoppa in ska ju egentligen inte spela någon roll. Vi vill ju inte göra dem verktygstrogna, hos en del kunder är det open-source verktyg som gäller, hos en del kunder har de krav från management att köpa verktyg

och så vidare. Där kanske de har en specifik leverantör, av ett specifikt verktyg och då handlar det om att anpassa oss dit. Så själva processen och för att nyttja processen ska vara ganska flexibelt och dynamiskt.

16:Har du någon mer utmaning du kommer att tänka på?

Nej inte på rak arm, sen finns det alltid mer specifika utmaningar i införandeprojektet. All form av förändring leder ju alltid till någon slags motstånd. Personer är alltid motståndsbenägna när det kommer till förändring, återigen är det därför vi är ganska benägna att anpassa oss efter deras nuvarande situation istället för att införa något stort på en och samma gång, hellre ta det i de mest basala och enkla stegen i små steg långsamt, så att det faktiskt är dom som är med och förbättrar och förändrar än att vi kommer in från sidan och för in något monster som de inte riktigt vill ha.

17:Vi tänkte prata lite om automatiska tester, du sa att det var svårt att testa automatiskt?

En integration består av flera komponenter, att testa vissa komponenter i sig självt ger liksom inget mervärde, det berättar ingenting om din integration, om du tänker dig att din integration är en kedja utav olika händelser så om du testar händelse ett, två och tre så kan de vara helt okej, men tillsammans i kedjan kan den bli bruten. Därför blir tester av enskilda artefakter inte lika värdefulla i ett integrationssammanhang, sen har vi problemet att olika plattformar hanterar olika protokoll beroende på vilken input de har. Säg till exempel att du exponerar en webservice som tar ett soap-meddelaned med någon salgs security setup, dels måste du då ha tillgång till det, du måste ju på något sätt kunna autogenerera på ett bra sätt, soap-kuvert och payload har du ju förmodligen, sen är det det fine om du kan göra det, sen vill du gå igenom hela din kedja och egentligen exkludera inserts på olika fält i meddelandet för att se att din payload t.ex. har förväntade värden. Sen kan du också ha beroenden till externa saker, en databas osv. Det jag vill komma till är att det oftast inte är omöjligt, det är bara det att sätta upp själva testet är en för stor effort med tänkte på värdet för att göra det, att uveckla en integration kan ta en timme, men sätt upp ett testcase kan ta fem timmar, oftast vill inga verksamheter betala för det. Dock egentligen brukar man göra tvärtom idag, du testar inte ditt flöde automatiskt utan du spelar in trafik, ett förväntat input och förväntat output, och i ett test så spelar du upp trafiken. Och jämför den med ett facit, lite mer som ett regressionstestfacit. Istället för att hålla koll på alla de specifika delarna. Detta gör dock att om du jämför data mellan olika tidpunkter kan datan skilja sig, du måste ha något smart system för att kunna pinpointa den datan då. Oftast med integration så brukar problemen vara av mer basal nivå, t.ex. att ett datumfält har en felaktig formattering eller något liknande. Det gör att din testreferens måste vara extremt tydlig. Vilket då sätter ett ganska högt krav på värdet av din validering eller vad man ska säga. Vad man hela tiden vill göra, är att sätta upp ett testcase ska inte ta mer tid, eller det ska i alla fall vara rimligt i förhållande till tiden det tar att sätta upp integrationen.

En annan sak som också inte är extremt svårt men när vi pratar om t.ex. enhetstester så kan faktiskt själva testet göra, när du mockar eller lägger på data så kan du faktiskt förvanska datan på ett sätt som gör att det fungerar, till skillnad från där integrationen är i sin riktiga miljö, där du har ett avsändarsystem som kanske egentligen är i ett testsystem som inte forcerar så att säga. Vilket gör att när det går i ett riktigt case kommer det inte fungera, men när du gör ditt testcase kommer det fungera. Du har liksom inte de miljöspecifika delarna med i testet, det är väldigt svårt att få upp en sån typ av testmiljö där du faktiskt kan mocka både dina avsändande och end system och så vidare.

17:Arbetar du någonting med kontinuerlig förbättring i samband med Continuous Delivery?

Varje dag. För varje införande projekt vi har så promotar vi våran, jag och en kollega har tagit fram något vi kallar en build and deployprocess som egentligen är en mognadsstege på 7 eller 8 steg, du kommer iterativt gå igenom stegen hela tiden, tittar vi på kunder som har implementationer som är 5 år gamla så kommer du förbättra, förändra och utveckla nästan vad processen ska göra eller innehålla varje dag eller vecka.

18:Vad säger du om godkännanden, dvs check-points osv och processförändringar, är det någon utmaning du stött på?

Jag skulle säga att oftast är godkännanden inte en utmaning, utan snarare ett krav från verksamheter att de vill ha någon form av manuellt steg eller godkännande för att ta vidare processen. Sen sa du även processförändring va?

19:Ja, exakt

För oss ingår det lite i vår mognadsstege,det handlar även om att titta på nuvarande processer för att antingen förbättra eller förändra dom.

20:Så det här egentligen ihop med kontinuerlig förbättring?

Absolut, för oss spelar det egentligen ingen roll vilken form utav continuous kärna du försöker implementera, det handlar hela tiden om att förbättra, förändra, göra det mest optimala, något vi lärt oss över tid, dom som bara för in någonting och sedan låter det vara, det är oftast dom som misslyckas i längden. Det handlar om att hela tiden försöka göra det smidigare, bättre och effektivare. Oftast kan du göra en stor sak i små steg, det handlar om att inte göra allt på engång, små steg, hur gör vi det så att alla är med på tåget, vad är målbilden och hur tar vi oss enklast dit? Och återigen har vi då team, då någon är med från utvecklarna, någon från operations, och kanske någon från verksamheten, då är det mycket enklare att förbättra, alla förstår värdet av det, du har commitment från alla delar.

21:Har du varit med om att motivationen för ett införande har försvunnit när man börjat arbeta på det?

Nej, tvärtom, jag var hos en kund för någon månad sen som hade ett väldigt stort motstånd, eller icke förståelse för varför man skulle föra in det, att föra in Continuous Integration där från scratch och anpassa sig efter deras dåvarande bygg och deployprocesser tog en arbetsdag, så på 7 timmar var alla verktyg och allt som behövdes för att automatisera byggen och deployer införda, och det är snarare på det sättet som vi kan övertyga kunder om hur lättviktigt det kan vara att föra in ett sådant verktyg även om de inte riktigt förstår vinningen med det, och i det här fallet så handlade det mer om att visa och få dom att förstå för de hade inte en aning om vad det var eller varför de skulle ha det, de var en väldigt liten och tight grupp på två personer. Men de hade ett krav på sig från verksamheten att utöka teamet, och det är nog först då de förstår vinningen då.

Appendix IP2

1: Hur länge har du arbetat inom IT?
Christian: Sedan 94, så.. 22 år, började på eriksson, oftast internationella projekt. Kort anställd på Sony.

2: Hur länge har du arbetat på Softhouse?
Christian: Sedan 2008, så 8 år.

3:När kom du först i kontakt med Continuous Delivery?
Christian: När kan det ha vart.. Beror lite på, jätte svårt att säga när man hörde det först, men säg att det var sex sju år sedan, som begrepp som en glidande skalafrån nightly builds, hourly builds, nightly builds började redan på 90 talet, bygga varje commit började typ 2008, när jag började på Softhouse ungefär.

4: Hur många projekt har du varit med i där Continuous Delivery införts?
Christian: Ohoho, inte så många faktiskt, suttit mycket på strategiska roller,har suttit mest vid sidan om projekten, vad ska vi säga.. men säg att det är 8, kanske 10 projekt.

5: När du haft dem strategiska rollerna, har du då haft någon kontakt med Continuous Delivery då?
Christian: Absolut, det är en av frågorna jag drivit, just att få upp det här flödet och leveranstakten liksom, försöka enabla det så mycket som möjligt, just därför jag tycker att om man sitter vid sidan av projekten kan man förändra hela infratstrukuten på företaget istället, så slipper man vara projektbunden.

6: Ger det en stor fördel för projektet, Continuous Delivery tycker du?
Christian: Ja absolut

7: Hur tycker du att det ger fördelar?
Christian: Får man till det här med automatiskt testning så att det verkligen fungerar bra, och att man litar på dom testerna då får man en konstant kvalité. Då slipper man alla stabileringsfaser och kan scrappa en massa onödig branchning, statusen blir bättre, alla vet vad de ska basera sin kod på, och det kommer uppdateringar hela tiden. Jag baserar min nya ändring på någonting som jag vet fungerar, har man alltid ett fungerande system så blir det mycket lättare att koda, och att underhålla sin arkitektur, finns mycket fördelar.

8: Är det några nackdelar som du ser?
Christian:
När man automatiserar testningen så i en organisationen med mycket blamekultur, så drar man ner brallorna på alla, man kan inte gömma sig, det blir tydligt vilka som är bättre och sämre, en automatiskt testning tar inte hänsyn till vem som är vem och vem som gör vad, det blir opersonligt. Med manuell testning kan testarna hjälpa utvecklarna. Det hänger lite grann på de mjuka värdena i organisationen. Det kan påverka människor negativt då det känns dåligt att bli anklagad. Kan vara kulturella skillnader också, i Sverige är vi rätt bra på att ta det, men räcker att åka till sydeuropa så ser de mer allvarligt på att bli anklagade, på Sony så märkte de att de från asien hade svårt att ta resultaten från kodgranskningen. Tog ett tag innan de accepterade att folk betygsatte dem. Finns ett Jenkins plugin som ger -1 och +1, de kan rösta på commitsen, testsystemet sätter helt enkelt -1 och +1.

9: Har du varit med om att Continuous Delivery införs inom ett redan påbörjat projekt?
Jaoja, i många stora företag så är projekten ofta så långsamma att de snarare är att betrakta som verksamheten, fem sex år, de blir en del av strukturen på företagen, händer en massa förändringar under projektens gång.

10: Har du upptäckt några skillnader när det införs i ett redan påbörjat projekt eller verksamhet, gentemot när man startar upp med det?
Nej inte så mycket på projekt, men däremot stor skillnad om det är en produkt som redan finns och där projektet kanske är en vidare utveckling av produkten, införa automatiskt testning och förändra leveranssättet på en existerande produkt är fantastiskt jobbigt, det finns en teknisk skuld att arbeta hem som mycket tung att ta. Mycket jobbigare att göra det i efterhand.

11: Skulle du säga att hela kvaliteten på projektet påverkas positivt om man börjar med Continuous Delivery tidigt?

Christian: Den dagen man ska börja med automatisering, man hittar ofta fel som varit dolda. Projekten/produkten blir bättre om CD blir infört med en gång. Kvaliten på testningen höjs o, används, manuella tester släpper ibland igenom fel, då det är lättare att fuska, vi ska ändå inte släppa till kund nu, kör du CD är alltid redo att leverera.

12: Har du varit i ett projekt där införandet av Continuous Delivery har varit misslyckat?
Christian: Ja, det som har hänt varje gång är att när man ska införa automatisk testning på gamla system och så orkar man inte med. Automatisk testning på gamla system är svårt, det är en stor utmaning att få till de automatiska testerna, både organisationell och tekniskt. Själva kompileringen och bygget det brukar sällan vara ett problem. Men just att börja testa automatiskt är en puckel som man måste komma över, kan vara svårt att hinna med under ett projekt. Måste lita på testerna, både kund och utvecklare, utvecklare vill ofta testa manuellt. Effektiviseringen uteblir om utvecklare testare manuellt trots automatisk testning.
Halvfärdig införande har skett. Försökt få igång kunder, kunder bryter ibland innan de börjat för att det är för stort steg att införa automatiska tester. Där det går segt, eller där man känner att man inte riktigt kommer igång. Håller inte visionen igång vart ska man komma, vad ska vi åstadkomma, är det stora system så tar det flera år att komma upp i tillräcklig volym för de automatiska testerna. Det kan vara svårt att på förhand se hur mycket man kommer att tjäna in. Finns en brist på data. På en konferens på nittiotalet fanns det en endast en rapport om CM. Det är svårt att jämföra med eller utan Continuous Delivery, måste köra identiska projekt ett med ett utan.

13: I ett av de arbeten vi läste hade de lyckats minska frekvensen av buggar i sitt system med 90%.
Christian: Det kan jag tänka mig, vi på Softhouse var med på ett projekt på Eriksson, gick från integrationsprocess på två veckor till varje dag, gick från 60% passrate till 90%, bara med den ändringen. På två veckor, om det är tusen personer i projektet, fatta hur många rader kod de hinner skriva, hur många comitter, utvecklarna börjar nya features på två veckor gammal kod. Det låg tusentals commiter däremellan.

14: Vem var initiativtagare till att börja införa Continuous Delivery?
Christian: Det är väldigt varierande, ibland chefer ibland en hög chef som varit på seminarie, chefer på lägsta nivå som känner att de måste öka effektivitet. Det kan vara vi konsulter, kan vara utvecklare som tycker att leveransrutinerna suger som vill ha snabbare feedback, kan inte säga att jag sett något mönster där.

15: Vilka fördelar med införandet utav Continuous Delivery har du sett?
För det första en konstant kvalite som är känd, som man har på sin produkt, om du testar sällan så går kvaliten ner, skriver alltid sönder när man ändrar, ju längre det går mellan test destå sämre blir det. Testar man eller integrerar sällan så går det djupt ner, kan man hålla konstant testning kontinuerligt, små fall istället för stora. Feedback fort, en utvecklare som får feedback efter två veckor håller ju på med något annat, som baseras på något som redan är trasigt, får han feedback efter femminuter så går han på muggen, sen när han är tillbaka så fixar han vad som är fel direkt.

Hela informationsflödet blir bättre, man pratar om ett fungerande system istället för ett trasigt. Mer progressivt tänkande då man alltid har ett fungerande system. Ger en mer positiv bild på företaget. Pratar om att lägga till features istället för fixa problem. Rollback försvinner mycket, blir istället roll forward, leverar rättning snabbt som fan. Hela mentaliteten i utvecklingen blir mer positiv, annorlunda och bättre.

16: Vilka nackdelar med införandet utav Continuous Delivery ser du?
Finns alltid risker att folk inte förstår varandra, oavsett vilken förändring man inte får. Annars inga riktiga nackdelar som jag ser, strömförbrukningen kanske går upp då man har mer testmaskiner.

17: Har du stött på några utmaningar inom införandet utav Continuous Delivery?
Största som vi pratar om mest är testautomatiseringen, när våra kunder frågar var dem ska börja så säger vi att dem ska börja skriva automatiska tester. Stort problem då det tar lång tid. Passar vår arkitektur för att bli testbar, måste den förändras? Sen handlar det om att ändra mindset, utvecklare måste lära sig om testning, testarna måste sättas ut hos utvecklingsteamen och lära utvecklare. Rollfördelningen blir annorlunda.

Sen är det det här med trust, lita på att det faktiskt går att kontrollera kvaliten automatiskt.

En annan utmaning är att man måste liksom testa sina tester, underhålla hela testsystemet. Så att man kan lita på att det fungerar.

När det handlar om att bygga infrastruktur på företag så finns det massor med hinder, måste blanda in olika avdelningar. För att göra förändringar behöver man infrastruktur grejer från IT-avdelningen. Vi har sådana där yttre krav, var på uppdrag från ett företag med FDA krav, så de har en valideringsprocess på ett år, då tänker de att varför ska vi då ha testning under projektetsgång? Men validering är inte verifiering. Validering ska man säkerhetsställa att systemet fungerar som det är tänkt, verifiering ska man hitta fel och se till att det vi har gjort är rätt.

Organisatoriska förändringar, chefslager som måste övertygas, så fort man ändrar informationsflödet så finns det risker som man smäller på.

Ansvarsfördelningar som ändras, testchefer som tycker att deras roller försvinner. Kan vara svårt att acceptera att sin arbetsroll förändras. Rollförändringar, samma problem som när vi inför agilt arbete. Expertroll istället för sin tidigare roll. Agila transformationen ger mer makt till utvecklarna, andras roller blir mycket att bygga infrastruktur kring utvecklarna så att det blir så enkelt som möjligt.

18: Hur kunde dessa utmaningar lösas?
Kulturförändringar tar mycket lång tid, nystartade företag går ganska enkelt, men femtioåriga trygghetsnarkomaner på eriksson har det svårare att acceptera att sin roll förändras än nyexade i tjugoårsåldern.

19: Finns det något man göra för att få förändringarna att accepteras?
Christian: Man måste måla bilden så att det blir tydligt, visionärt arbete. Ingenjörer, i Sverige särskilt, tar mycket beslut, de måste arbeta mot samma mål. Väldigt mycket självbestämmande så alla måste arbeta efter samma vision, kommunicera om vad vi vill uppnå, hur vill vi att det ska vara, vad ser vi för problem. Trycka ner lösning i halsen fungerar nästan aldrig, utan jag måste först visa vad problemet är och då kan jag visa lösningen!

20: Måste det vara hela organisationen, eller räcker det med nyckelpersoner?
Christian: Hela organisationen, nästan alla. Räcker inte med nyckelpersoner. Det finns alltid personer som inte vill förändra, en viss andel kan man kanske acceptera, men man bör ha med sig bulken. Har du inte nyckelpersoner så blir det svårare. Vill gärna hitta nyckelpersonerna först men när man väl kör vill man ha med sig alla. Se bilderna och flödena.

Chefer som är icke-tekniker eller bara inte kan de tekniker som är nu måste även de komma med på båten och förstå vad som ska göras. Ju högre upp man kommer desto lättare blir det. Om kunden visar tydligt att det är vad de vill ha blir det lättare. Då känner utvecklarna sig uppskattade.

21: hur har utvecklingsarbetet behövt förändras på grund av Continuous Delivery?
Christian: Utvecklarna måste bli medvetna om hur de hänger in i flödet, det enskilda och teamansvaret blir stort, att inte skylla ifrån sig, ta feedbacken på allvar, bryter man master får man fan fixa det på en gång. Ansvarskänsla, liknar mycket hur vi inför agilt. Det är det här teamansvaret, de hänger ihop väldigt mycket. För mig handlar det mycket om att vara stolt över det man levererar. Måste förstå sammanhanget man hänger in i, see the whole picture.

22: finns det något som du har upplevt som extra viktigt för att ett införande ska bli lyckad?
Christian: Jag skulle nog säga visionen, hålla den levande, vi ska hit, gärna får en med ledarroll och någon med informell makt! Stå längst fram och säga THIS IS THE SHIT, det här är bra, någon som vågar ta de jobbiga besluten. Det är nog det viktigaste. Sen kan man snacka hur mycket som helst om att jenkins måste funka osv.

23: något annat?
Christian: Våga börja, inte väntar och väntar på ett perfekt tillfälle eller det perfekta projektet. Bara kavla upp armarna och sätt igång. Finns inget som hindrar dig från att starta. Mycket en mental grej, måste våga.

Något som är bra är om det finns folk som har sett det lyckas tidigare, både kompetens, vilja och erfarenhet. De hjälper till att måla bilden visionen.

24: Vi har hittat ett arbete med ett gäng principer för Continuous Delivery. Vi vill bara höra med dig om det är någon som är extra relevant, eller någon som inte är så viktig.
Skapa en pålitlig och upprepningsbar-process.
Automatisera nästan allt.

Förvara allt i versionshanteringssystem.
Gör det som är svårt ofta, det vill säga, hitta flaskhalsarna och gör dem ofta.
Bygg in kvalitet, bygg ett system som går att testa.
Klar betyder integrerad, när släpper man sitt ansvar, när får jag lämna ifrån mig det här?
Gemensamt ansvar för leveransprocessen över avdelningarna.
Kontinuerlig förbättring, hela tiden bli lite lite bättre.

Håller du med om dessa?
Christian: Principerna är viktiga allihopa, behöver jag rangordna?

26: Nej, det behöver du inte, men om det finns någon som du tycker är extra viktig får du gärna peka ut den.

Christian: Att bygga in kvalité är viktigt, en bra arkitektur är viktigt, får man inte in den så kommer man lida av det i det långa loppet, testerna ska vara enkla att skriva, det blir enorma kostnader om arkitekturen är dålig. Arkitekturen är väldigt viktig men dess betydelse glöms ibland. Mycket erfarenhet krävs, duktiga arkitekter är svårt att hitta, nya tankar och nya koncept blandat med erfarenhet. Enbart ett par arkitekter som jag träffat under 22 år har haft vad som krävs. Det är som viktigast när det gäller stora system. Kontinuerlig förbättring och automatisering är självklart viktigt.

DoD är viktigt när man arbetar, annars kan man inte riktigt lita på att andra gör sina jobb. Ska du jobba i på ett agilt sätt med olika team så kan man inte jobba med teambyggnad, alla måste kunna lita på andra gör sina jobb, därför måste man ha definition of done på plats.

Appendix IP3

1: Hur länge har du arbetat inom IT?
Tretton, fjorton år

2: Hur länge har du arbetat på Softhouse?
Ett halvår på Softhouse

3: Var jobbade du tidigare?
Sony

4: När kom du i kontakt med CD?
Ett par år sedan, men jag har tidigare jobbat åt det hållet, 8-9 år, först senaste åren visste jag vad det hette. Att automatisera flödet alltid varit viktigt.

5: Hur många projekt har du varit en del av där Continuous Delivery använts?
Jag jobbar nu med tre ställen där det införs och ett till innan dess, halvdant.

6: Hur påverkar det ett projekt när man inför eller försöker införa Continuous Delivery?
Med ett stort projekt med mycket gamla vanor, "vi har alltid gjort så här", så blir det en stor impact, man får snabbare feedback tillbaka. Det påverkar positivt.

7: Har du varit med om att det påverkat negativt?
Nja, inte negativt. Däremot har jag varit med om att chefer tror att det är dåligt. Vissa försöker hålla kvar sin domän och kanske försöker motverka införandet.

8: Har det blivit problematiskt att införa Continuous Delivery när chefer försöker motverka införandet?
Det är svårt eftersom att initiativet till införandet ofta kommer nedifrån och då har det kanske inte tagits något beslut än. Det leder ofta till att det tar längre tid. Jag är nu på ett företag i Helsingborg och där tror jag att det blir väldigt svårt att införa det eftersom att det är en gammal kultur och de har jobbat på ett helt annat sätt men jag hoppas på att det går.

9: Har du med om att man ifrån starten använder Continuous Delivery i ett projekt?
Från start, inte själv men jag har kollegor som har gjort det ifrån ett tidigt skede. Jag har ett projekt just nu där vi i och för sig har startat det ifrån ett tidigt skede. Det gäller att någon i projektet kan prata om det och som vet varför det är bra är med från början.

10: Har du varit med om att buggar slunkit igenom de automatiserade testerna?
Jag har inte varit med om att de automatiserade testerna har varit så bra att de ersatt manuell testning helt. Ofta säger man att det här är så speciellt att det inte kan automatiseras. Visst slinker det igenom men då har inte testerna varit så bra. Hur bra de automatiserade testerna är beror på vad det är man utvecklar.

11: Är det ifrån utvecklarna som förslaget att införa Continuous Delivery kommer oftast?
Dels ifrån utvecklarna men också om man har modellerare och systemtänkande personer, det är mest ifrån dom. Det beror väldigt på vilken sorts utvecklare det är, om man tänker på helheten. Många utvecklare tycker att det är skönt att jobba på sin grej och sen kastar man iväg det, klart, och sedan behöver man inte bry sig om det längre, med dem är det svårare. Som införare av Continuous Delivery måste man se till att det finns automatiska

byggen, automatiska tester, automatisk deploy och delivery beroende på var man kommer så att de får snabb feedback tillbaka. Det får inte ta veckor.

12: Har du varit i något projekt där Continuous Delivery blivit helt misslyckat, där det inte gått att införa?
Ja, men då har det att göra med politiken i företaget, att vissa sätter stopp.

13: Kan du ge något mer exempel på ett rent politiskt problem som du har varit med om eller hört talas om?
Det handlar ju om att man vill hålla kvar sitt mandat. Continuous Delivery är ju agilt och när man vill införa agilt i ett företag så har teamet mer att säga till om och cheferna mindre. Det blir en sorts maktbalans som skjuts bort ifrån cheferna. Det brukar man kalla för manager's dilemma. Vad ska de göra när man har agilt och teamet bestämmer? Många personer klarar inte av det, däri ligger ett problem och det kan yttra sig på många sätt.

14: När du har stått inför den sortens politiska problem, vad kan man göra för att komma runt det?
Jag sitter som konsult på ett par ställen och försöker införa det. Man får försöka påverka ifrån sidan, ta fram data som visar vilka förbättringar som Continuous Delivery ger. Det är svårt eftersom att det inte finns så mycket data på vad som blir bättre. Man får försöka övertyga genom att ta fram det, kanske med ett referens-case. När man pratar om Continuous Delivery är det många som inte vet vad det är och det består av många olika delar där man måste ta det steg för steg och varje steg ger fler förbättringar. Utan stegen ligger visionen väldigt långt borta. Det är ett förbättringsprojekt som man aldrig blir helt klar med. Det går hand i hand med Continuous Improvement.

15: Vilka utmaningar har du stött på när du inför Continuous Delivery?
Den största utmaningen är att få till automatiska tester. Om man går till ett företag som har jobbat länge med mjukvara så har de inte så mycket automatiska tester. De testar allting manuellt varje kvartal t.ex.. Att skifta till att testa varje dag är ett jättestort steg. Om man kommer in till ett företag och de vill ha Continuous Delivery så kan man säga så här, automatisera testerna så kommer vi göra resten sakta men säkert. Det kommer ta så lång tid för de att göra det.

16: Är det en teknisk utmaning eller är det organisationellt?
Det är dels en teknisk utmaning, att man förstår att det går att automatisera testerna. Organisationellt har ofta företag en testorganisation, och vad ska de göra då? Ta in testarna i utvecklingsteamen istället där de kan se till att det blir testbar kod, teststrategier och sådant. Bara för att man har automatiska tester så betyder det inte att man inte behöver testkompetens.

17: Finns det fler utmaningar?
Svår sak att få folk att förstå att det här är bra och så här gör man, om de förstår det så är det bara att göra stegen. En sak som oftast är svår att få folk att förstå är att om man gör en kodändring och det går igenom automatiska tester och är redo för leverans direkt, att det inte är farligt? De känner sig otrygga med det. Kan vi lita på att det fungerar? Att det är bra? Ju snabbare en utvecklares förändring går och ju längre ut den kommer, desto mer ansvar tar den personen. Får du feedback en halvtimme efter att du gjorde ändringen istället för efter två veckor så kommer du ihåg vad du gjorde bättre. På nyare ställen förstår de att det är ett modernt sätt att jobba. Continuous Delivery är lika revolutionärt som att införa agila arbetssätt som SCRUM t.ex.. För företag som inte arbetar agilt redan är det ett väldigt stort steg att ta. Det är en stor faktor.

18: Är det cheferna eller utvecklarna som behöver inse nyttan med CD?
Båda delar, det beror på hur "up to date" de är men det tar ungefär lika lång tid. Utvecklare kan vara lättare att övertyga eftersom att de vill ta det extra ansvaret som Continuous Delivery medför egentligen, chefer tappar

ansvar. De som håller hårt i manuella checkpoints tycker inte om Continuous Delivery. Vi vill att det ska vara automatiska checkpoints, de kan vara kvar men de ska vara automatiska och hoppa vidare till nästa steg.

19: Är det många stakeholders ifrån olika delar av verksamheten som måste godkänna?
Om inte alla delar är med på förändringen så kanske vissa delar är kvar i det manuella. Det kan leda till att de andra stegen i Continuous Delivery fungerar jättebra men det tar ändå väldigt lång tid på grund av det manuella. Då gör man det halvdant och då blir det dåligt och då får det dåligt rykte. Då tappar cheferna intresset och tycker att det är dåligt.

20: Kan man få in en vana att stakeholders manuellt kan snabbt godkänna istället för automatiska godkännanden?
Det beror väl på vad som är i tid, det ska gå tillräckligt snabbt. Om det går tillräckligt snabbt så går det ju bra men det beror helt på vilket företag man är på. Det är svårt att svara på, eventuellt kan det fungera om det är tillräckligt snabbt.

21: Kommer du på någon mer utmaning?
Jag är inte på ett företag så länge och när man inför Continuous Delivery så vill man att någon där ska ta ansvar för verktygskedjan och flödet. Att välja verktyg är viktigt så att det passar de som jobbar där. Jag har varit med om företag som använder 5-6 olika versionshanteringssytem för samma kodbas och två-tre olika byggsystem och det försvårar väldigt mycket att ta hand om det, det är viktigt att konsolidera det. Det är dock specialfall, precis som företag som inte använder versionshanteringssystem. Då får man börja från steg noll. Företag som inte har någonting ser förbättringarna snabbare och följer med mycket lättare. Annars tycker folk gärna att det nuvarande systemet fungerar och man vill ogärna ändra på det. Dåliga vanor måste man först ta bort.

22: Finns det något som du har upplevt som extra viktigt för att ett införande ska lyckas?
Det finns en hel del saker som är extra viktigt. Om du ska få till ett bra Continuous Delivery flöde så behöver du något som kollar när har en ändring gjorts. Du behöver ett byggsystem som kan bygga ihop det . Du behöver testa så mycket som möjligt automatiskt helst, du kan testa varje check-in, commit. Du behöver kunna deploya på en test-environment. Det viktigaste tror jag är att man måste ha en value stream mapping, man får se hela värdekedjan, utvecklare tills att det kommer ut till kunden. Var någonstans finns det flaskhalsar att laga där? Visualisera flödet är det viktigaste. Gör man inte det kan man förbättra hur mycket som helst men det blir inte bättre. Ska man införa det på ett befintligt projekt ska man göra det först och sedan förbättra steg för steg.

De slutliga testerna är ofta manuella.

Systemtänkande personer oftast de som börjar införa, de som tänker på flödet. Helhetstänkande, många utvecklare tycker det är skönt att göra sin grej, och vänta på feedback (inte de som tycker det är bra)

23: Vi: Vi har hittat ett arbete med ett gäng principer för Continuous Delivery. Vi vill bara höra med dig om det är någon som är extra relevant, eller någon som inte är så viktig.
Skapa en pålitlig och upprepningsbar-process.
Automatisera nästan allt.
Förvara allt i versionshanteringssystem.
Gör det som är svårt ofta, det vill säga, hitta flaskhalsarna och gör dem ofta.
Bygg in kvalitet, bygg ett system som går att testa.
Klar betyder integrerad, när släpper man sitt ansvar, när får jag lämna ifrån mig det här?
Gemensamt ansvar för leveransprocessen över avdelningarna.
Kontinuerlig förbättring, hela tiden bli lite lite bättre.
Håller du med om dessa?

Man är inte klar om det inte är testat. Kontinuerlig förbättring.

Appendix IP4

1: Hur länge har du arbetat med utveckling eller inom IT överlag?

Sedan 1999, vad blir det, 17 år.

2: Hur länge har du arbetat på företaget?

Sedan 2007, nio år.

3: När kom du först i kontakt med Continuous Delivery?

Det var tvåtusen eh,. begreppet Continuous Delivery kom jag i kontakt med 2010, när det kom ut en bok som heter så. Den är bra.

4: Idéerna bakom är lite äldre menar du?

Ja absolut, man kan säga så här, jag och några kollegor vi började tittade på det här några år innan , kanske runt 2008, men då fanns inte Continuous Delivery som begrepp, men i princip samma idéer om snabba leveranser och så där, sen dök dev ops rörelsen upp där i samma veva runt 2009 ungefär som också är besläktad.

5: Hur många projekt har du varit med i där Continuous Delivery införts?

en två, tre, fyra fem, beror lite på, kanske sju åtta stycken.

6: Hur anser du att införandet av Continuous Delivery påverkar ett utvecklingsprojekt?

Själva införandet är ganska smärtsamt många gånger, det handlar om stora förändringar i organisationen, både i arbetssätt, organisation, ansvarsområden, verktyg och arkitektur. Förändring är alltid jobbigt, alla vill ha förändring men ingen vill förändras. Så själva införandeprojektet är smärtsamt är jobbigt många gånger, men också otroligt kul när man ser att det börjar ta fort och blir bra, när man ser resultat då blir det ett jätte stort uppsving i organisationen, inte bara resultatmässigt det vill säga att man kan gå snabbare till market, utan framförallt att det upplevs som ett mycket roligare sätt att jobba på, hos nästan alla involverade, utvecklare, testare operations människor. Så att det var en bred fråga men det påverkar organisationen mycket, gör man det på rätt sätt så är det stora förändringar. Sen har jag sett många fall där man kanske inte gått hela vägen, utan där man bara gjort det tekniska perspektivet utan att ändra arbetssätt, då kommer man inte hela vägen, bara för att man automatiserar massa saker så inför man inte Continuous Delivery, utan det handlar om ett radikalt förändrat mind-set, både från ett kravställande håll och framförallt utvecklingsteam om hur man förändrar sina system.

7: Är det en organisationell utmaning då, eller teknisk eller procedurell?

Det är utmaningar på alla håll och kanter, men den tekniska utmaningen är förhållandevis enkel, men det är de organisationsmässiga aspekterna som är de svåraste att förändra. Aspekter som hur ska man sätta upp sina team, hur ska man strukturera sig, hur ser gränssnittet mot kravställande verksamhet ut, hur budgeterar man, hur bedriver man affärsutveckling på ett annat sätt som utvecklar Continuous Delivery. Jobba med att förändra beteenden, snabb feedback, långsiktigt kvalitetstänkande och alla de aspekterna som är icke-tekniska då. Organisation och företagskultur är bland det svåraste att jobba med. En annan svårighet att jobba med då är ju arkitekturen, om man inte börjar från ett blankt papper då, vilket inte så många har förmånen att göra, så krävs det förändringar i arkitekturen. Behöver titta på gränssnittet, behöver kanske tillämpa feature-hiding teknik och annat.

8: Har du varit i ett projekt där införandet av Continuous Delivery har varit misslyckat?

Misslyckats är ett väldigt definitvt ord, men jag har varit med i projekt där det har lyckats mindre bra, och andra projekt där det har lyckats mer bra.

Ett exempel, kanske inte misslyckat, men vi nådde inte effekterna som vi ville, helt och hållet, jag tror att om man ska analysera kärnproblemen så handlar det rätt mycket om att vi inte hade buy-in, vi hade inte mandat från management att göra det hela vägen, det är otroligt viktigt, it-ledningen måste ha förståelse för vad innebär och att man har mandat att göra de här förändringarna ganska brett i IT-organisationen då. I det här fallet handlade det om att vi rörde oss i en lite för snäv box, vi fick inte gå hela vägen till produktion. De hade en extern drift partner, och det gränssnittet var komplicerat. Vi fick inte automatisera deployment i produktion. Vi fick inte förståelse för ledningen vad det innebär, och hur man kan uttnyttja continuous, vad det innebär att göra releaser oftare och snabbare.

Den enskilt mest kritiska faktorn är när nyckelpersoner slutar, då tappar man momentum, vi kallar dem champions, de som driver på och sponsrar själva införandet, kanske framförallt budgetmässigt men även icke budgetmässigt, om de personerna försvinner utan att det har tagit fart då är det tufft.

9: -har varit lyckat?

På alla mina projekt har vi alltid fått ut effekt, men kanske inte fått ut de fantastiska effekterna som man kan, det har aldrig blivit sämre, även om effekten bara blivit att man ökat förståelsen och medvetenheten om vad man kan åstadkomma så är det väldigt nyttigt. Kommer man inte hela vägen är det svårt att få dom tydligt märkbara effekterna, du kanske kan bli lite mer produktiv, öka trivseln, jobba mycket med interna värden, men inte så mycket extern, utifrån IT sett.

Förtydligande via epost:

Jag har varit med i flera lyckade projekt och har väldigt tydliga mätvärden på resultaten att visa upp men vi kom aldrig in på det. Här än några uppmätta effekter från ett av mina projekt som löpte under 11 månader 2013:

* Fördubbling av throughput , från 1000 stories till 2000 stories på ungefär samma storlek på oraganisaitonen (ca 120 pers på IT)

* Halvering av incidenter i produktion

* Ledtiden för leverans (från incheckning till produktion) gick från ca 70 dagar till ca 14 dagar i genomsnitt och man kan göra release på mindre än en dag

* Från 1 release per månad till 20+ releaser om dagen

10: Vem var initiativtagare till att börja införa Continuous Delivery oftast?
Väldigt blandat, nu, på senare år så är det IT-lednings personer, ledande personer inom IT-organisationen, för några år sedan så var det inte det utan då kom det från utvecklarhåll. Medvetenheten om det här har ökat generellt på marknaden, det man vill är ju att medvetenheten kommer hela vägen upp på företagsledningshåll, och där är den inte riktigt än.

11: Har du stött på några utmaningar inom införandet utav Continuous Delivery?

Massor, det är bara en stor hög utmaningar. Icke-tekniska utmaningar, en utmaning är att göra ett business-case på det, vilket krävs många gånger för att få buy-in från ledning, ska man göra förändringar måste man visa vad nyttan blir. Det är ganska lätt att göra i lite större organisationer, genom att bara räkna waste, alltså hur mycket man sparar på att inte hålla på med onödiga saker. Men i lite mindre organisationer blir det svårare, då får man titta på kvalité, och saker som är svårare att mäta, innovationskraft och sådana saker, time to market, cost of delay brukar man kunna prata om också.

Den största utmaningen vi haft är att få kunden självgående, kontinuerlig förbättring, att oavsett vad de använder för metod så måste de skapa kunskapen och förmågan att få till det. Så själva överlämningen är svår. Det är svårt på grund utav flera saker, det ena är att det är svårt att hitta kompetens, det växer inte på träd. I många fall har man försökt rekrytera rätt länge utan att hitta någon som kan området och som kan ta över och driva det vidare. En annan är att man har svårt att hitta en plats i organisationen för det här arbetet. Det finns några olika modeller,

man kan skapa ett tools-team, eller support dev-ops team som kan göra det här, det är inte helt lätt att veta var det ska ligga, under utveckling eller operations?

12: Vad är det ni förordar?

Framförallt är det att man inte har en klassisk uppdelning med utveckling och drift, utan att hela driftansvaret ligger hos utvecklingen, och att driftexpter formerar sig i team som stöd åt utvecklings teamen, eller ja, ingår i utvecklingsteamen, så att inte driftansvaret ligger hos någon annan än de som skriver koden. Det är en sak men det finns flera. Även hur man tänker att man ska organisera utvecklingsteamen utifrån affärsflöden istället för en teknisk uppdelning.

Vilken fråga var vi inne på? Utmaningar?

13: Ja exakt.

Det finns ju tekniska utmaningar, det finns gamla legacy system som faktiskt inte alls är byggda för det här sättet att jobba på, det kan vara en utmaning. De kanske har licensmodeller, som inte går att hantera i testmiljöerna. Det kan vara gränssnitt. Testbarheten är ju viktig att det går att automatisera test. De tekniska utmaningar är oftast lösbara.

Andra utmaningar.. Det är väl den här förändringsbiten helt enkelt, man fårstånga huvudet i väggen många gånger med personer i organisationen som har andra sätt att se på saker och ting och övertyga dom. Finns bättre sätt att jobba på då. Det kan handla om att övertyga utvecklare att checka in sin kod, så enkelt. Det handlar om förändringsledning, kunna leda sina kunder genom förändring, det är alltifrån ledningsgrupper till utvecklare, testare och drift, konfiguration management osv.

14: Vid tidigare intervjuer har många nämnt automatiserade tester som en stor utmaning, både organisationellt och tekniskt, håller du med?

En stor del av ett införande såklart, men den tekniska aspekten är inte så.. Det är klart det kan vara gamla system som är svåra att automatisera, men återigen mest en organisationell och kultur fråga, det vill säga vem har ansvar för kvalitén? Det är utvecklingsteamet som har ansvar för kvalitén, i ett CD scenario så är det inte QA utan det är faktiskt utvecklingsteamet som har ett långsiktigt ansvar för kvaliteten. Det blir ett skifte i ansvar, och det är smärtsamt att hantera. Visst, det finns stora utmaningar där, för att se effekterna, om man göra alla de här delarna kan det fortfarande ta ganska lång tid innan man kan se effekterna om man inte kan bygga upp en hyfsad testautomatisering, och det går inte på en dag eller två, man måste helt enkelt börja, och där tror vi också att testautomatiseringen är en del av kvalitetsarbetet, behöver göra mer för att snabba upp processen. Det ena är att få ordning på testmiljö och ledtiderna för att hantera dom. Det kan handla om testdata, så även om man gör manuella tester så om man har bättre hantering och automatisering kring sin testdata så kan man snabba upp processen oerhört. Andra saker såsom information, det vill säga att man kan snabbt och enkelt jobba med mindre inkrement av koden och veta exakt vad som ändrats då kan man göra riskbaserad manuell testning, så även om man inte har full automatiserad test täckning så kan man snabba upp processen rejält med de här principerna. Men visst är det en utmaning sen att bygga upp den här automatiserade testbiten så att man kommer dit man vill, det vill säga att man har i princip full automatiserad testtäckning.

Test är ett stort område och det finns utmaningar där, en av de största utmaningarna vi ser är hur högnivå tester ska utföras och ansvaras för, om man nu organiserar sig i utvecklingsteam efter vissa affärsflöden eller hur man nu väljer, och de här teamen har ansvar för vissa tjänster eller produkter var och en, och de kan testa dom och releasa dom enskilt, så finns det fortfarande större flöden som spänner över flera av de här teamen som faktiskt är det kanske kunderna använder. Hur ska man hantera kvalitén och säkerhetsställa att de överliggande flödena fungerar? Det är en utmaning, det handlar väldigt mycket om att teamen måste kommunicera och testa i liksom helt integrerade miljöer då hela tiden.

15: Processförändringar vid införandet av Continuous Delivery, är det en utmaning?

Det är ju en stor processförändring hela införandet, först jobbar man på ett sätt, då har man en massa manuella moment och onödiga saker, avstämningsmöten och planeringsmöten och go live möten och allt det kan vara i sin process. Vi brukar titta på det hela från att en ide föds någonstans, ett krav uppstår någonstans hos en kund, hos

verksamheten eller hos en användare till dess att den finns i produktion och är tillgänglig, och den processen är ganska lång och komplicerad många gånger, hela införandet handlar om att slimma den processen dels slimma ner den och ta bort onödiga moment och sen också allt som man kan i den processen. Så processförändringar är ju det som vi jobbar med.

16: Visualiserar ni flödena då?

Ja det tycker vi är en jätteviktig aspekt i det hela, det handlar mycket om att få snabb feedback, det är ju ett ledord i CD, snabb feedback kan man ju ordna på flera olika sätt, men det visuella är viktigt för att skapa en transparens, så att vem som helst kan se vad som pågår i processen, vilka features är påväg ut? Vilka pullrequsts ligger för test just nu? Det är viktigt med det visuella för att skapa en transparens i organisationen tycker vi. Sen få den snabba feedbacken också, men där finns det ju olika sätt för teamen, men en klar trend är att fler väljer att visualisera på dashboards.

En annan utmaning är att lösa upp beroenden mellan team, så de får långsiktigt ägandeskap av vissa kompontenter och automatiserar sitt release flöde då är det viktigt att de kan releasa dom utan att vara beroende av andra team, så de här beroendena är krångliga att lösa upp många gånger. Men det finns två aspekter på det, den ena är en teknisk aspekt, gränssnitten sitter tight ihop, är beroende av förändringar hos en annan komponent, det kan man jobba bort med en långsiktig arkitektuell roadmap, sen finns det mer kravmässiga beroenden, där får man helt enkelt jobba upp förmågan att uttnyttja sin organisation som är uppsatt efter CD på bästa sätt, dvs lösa upp kraven på ett sätt som gör att man kan releasa dom enskilt.

17: Är det checkpoints och godkännanden som du tänker på lite grann?

Nja, säg att det finns ett projekt som det ofta finns från affärsutvecklingshåll då som säger att vi ska göra dom här och dom här förändringarna för att vi ska ut på en ny marknad, och då kokar det ner till förändringar i en massa system, och dom här olika systemen ägs då av olika team, och för att realisera ett av dom här kraven då så krävs det att båda dom här teamen har levererat. Kan man då lösa up det här på ett smidigare sätt så att de kan leverera enskilt så blir det bättre, det är svårt att generalisera här för det skiljer mycket från fall till fall, får titta på hur de jobbar med kravställningen osv. Typiskt är att man har acceptansgodkännanden där, men det är en utmaning.

18: När man har börjat införa har flera av våra tidigare respondenter sagt att det är möjligt att testorganisationen motsäger sig den här förändring för att de kanske vill behålla sina roller, är detta något du har sett?

Ja absolut, den organisationsmässiga aspekten är viktig, du kan inte som jag sa i början bara för att du automatiserar en massa saker så får du inte Continuous Delivery, utan du måste förändra organisationen också, testavdelningen ska bort helt enkelt, och det är inte så att man sparkar en massa folk oftast, utan kompetensen behövs utan vi måste utnyttja den på ett annat sätt. Det vi förordar är att man lyfter in test mycket närmare i utveckling, i teamen, så att test kan agera coacher till teamen och vara involverade i teamen, men det är inte så att de gör jobbet. Det är den modellen vi sett fungerat bäst, men det är inte helt trivialt att få till när man har en klasissk testorganisation. Inte minst för att kompetensen är ofta för låg, folk som manuellt suttit och testat i alla år, inte helt enkelt att börja skriva kod. Den delen av branschen ligger lite efter mognadsmässigt, deras kurser, certifikat och utbildningserbjudanden tycker jag inte matchar Continuous Delivery och det sättet man kommer jobba på framöver, utan de förstärker snarare gamla strukturer, kommer väl dit men tar lite längre tid.

19: Vad är det viktigaste punkterna att fokusera på?

När jag är ute och föreläser brukar jag säga så här, det viktigaste för att starta förändringsresan är att skaffa sig buy-in, från management. För att göra det måste du sätta upp vilka mål och effekter som förväntas. Sedan finns det jätte mycket jobb, men det som är viktigt att tänka på är att man ska skapa ett scope, som man kan leverera, där man kan mäta förändringen så att man inte tar allting på en gång. Det beror på vad man har för organisation. Har man en stor organisation är det svårt att göra allt samtidigt. Det här scopet, då är det viktigt att inte bara ta det lågt hängande frukterna, dvs de här nya webbsystemen som är skrivna med modern teknik, utan tar med den delen som gör mest ont, den som är trögast, långsammast och tar mest tid. Så att man börjar faktiskt jobba med det, plus andra lågt hängande frukter runtom det för att driva den här mognaden, för man kommer inte lika långt snabbt på de svåra delarna. Det kan vara t.ex. ett gammalt databassystem skrivet i databasteknologi, och det är

inte så lätt att automatisera, få utvecklare som suttit och skrivit databaskod i 25 år att ändra sitt arbetssätt, men det måste göras. Så det är väl det, sedan andra tips är att det är viktigt att ha mål, målen ska vara aggressiva, de ska vara snudd på omöjliga, det finns en mening med det, för det första så går det att nå dom, det är otroliga effekter när man gör det hela vägen. När man har fått buy-in för att göra det hela vägen med målet i sikte så har man en stor murbräcka att göra stora förändringar med, det kan handla om till exempel ett projekt som vi körde där man hade, från att det var utvecklat och klart till leverans, en ledtid på 60-70 dagar, som bara handlar om kvalitetssäkring, olika test och hit och dit , vårt mål med projektet var att ta det till under en dag, väldigt aggressivt men det går och det gick. I och med att vi hade det målet så hade vi en stor murbräcka att göra stora förändringar med, annars kan man bara fila i kanten lite och då kommer man inte lika långt. Sen måste man även vara agil i sitt arbetssätt, dom förutsättningar man sätter upp i början av ett sånt här projekt de gäller inte efter tre fyra månader. Man kanske säger, vi ska köra de här teamen, de här systemen, men det förändras, det kommer nya team, då får man helt enkelt titta på förutsättningarna, man kanske till exempel inte hade tänkt införa cloudteknik, men det uppstår ett behov.

20: Vi glömde fråga tidigare, men vad är din roll i införandeprojekten?

Förändringsledare eller Coach

Referenser

BUKOSKI, E., MOYLES, B. & MCGARR, M. 2016. *How We Build Code at Netflix* [Online]. Netflix. Available: http://techblog.netflix.com/2016/03/how-we-build-code-at-netflix.html [Accessed 10 maj 2016].

BULLEN, C. V. & ROCKART, J. F. 1981. A primer on critical success factors.

CARALLI, R. A., STEVENS, J. F., WILLKE, B. J. & WILSON, W. R. 2004. The critical success factor method: establishing a foundation for enterprise security management. DTIC Document.

CHEN, L. 2015. Continuous delivery: Huge benefits, but challenges too. *Software, IEEE,* 32**,** 50-54.

CHOW, T. & CAO, D.-B. 2008. A survey study of critical success factors in agile software projects. *Journal of Systems and Software,* 81**,** 961-971.

CRAIG, J. 2016. *Release Automation as a Catalyst for Continuous Delivery* [Online]. Automic; Enterprise Management Associates. Available: http://offers.automic.com/release-automation-as-a-catalyst-for-continuous-delivery-webcast-reg [Accessed 16 maj 2016].

DANIELSSON, L. 2015. *Alltid driftstart hos Pensionsmyndigheten* [Online]. IDG. Available: http://computersweden.idg.se/2.2683/1.608510/alltid-driftstart-hos-pensionsmyndigheten [Accessed 10 maj 2016].

DANIELSSON, L. 2016. *Därför är automatiserade tester nyckeln till kontinuerliga leveranser av mjukvara* [Online]. IDG. Available: http://techworld.idg.se/2.2524/1.655426/automatiserade-tester-mjukvara [Accessed 11 maj 2016].

DE SOUSA, J. M. E. 2004. *Definition and analysis of critical success factors for ERP implementation projects*. Universitat Politècnica de Catalunya, Barcelona, Spain.

DUVALL, P. M., MATYAS, S. & GLOVER, A. 2007. *Continuous integration: improving software quality and reducing risk*, Pearson Education.

FOWLER, M. 2006. *Continuous Integration* [Online]. MartinFowler.com. Available: http://martinfowler.com/articles/continuousIntegration.html [Accessed 11 maj 2016].

FOWLER, M. 2013. *Continuous Delivery* [Online]. MartinFowler.com. Available: http://martinfowler.com/bliki/ContinuousDelivery.html [Accessed 11 maj 2016].

FOWLER, M. & HIGHSMITH, J. 2001. The agile manifesto. *Software Development,* 9**,** 28-35.

FREUND, Y. P. 1988. Critical success factors. *Planning Review,* 16**,** 20-23.

HUMBLE, J. & FARLEY, D. 2010. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, Pearson Education.

JACOBSEN, D. I. 2002. *Vad, hur och varför?,* Lund, Studentlitteratur.

KARLSSON, T. 2015. Automated build service to facilitate Continuous Delivery. *LU-CS-EX 2015-27*.

LEIDECKER, J. K. & BRUNO, A. V. 1984. Identifying and using critical success factors. *Long range planning,* 17**,** 23-32.

MAXWELL, J. A. 2004. Qualitative Research Design: An Interactive Approach.

SLATKIN, B. 2013. Continuous Delivery at Google. *In:* MOZILLA (ed.). Mozilla Air.

TATE, R. 2013. *The Software Revolution Behind LinkedIn's Gushing Profits* [Online]. Condé Nast Publications. Available:

http://www.wired.com/2013/04/linkedin-software-revolution/ [Accessed 11 maj 2016].

WELLS, D. 1997, 1999. *Lessons Learned* [Online]. Available: http://www.extremeprogramming.org/rules/integrateoften.html [Accessed 10 maj 2016].