LUNDS UNIVERSITET
Ekonomihögskolan

Institutionen för informatik

Agila mjukvaruprojekts största utmaning

Kandidatuppsats, 15 högskolepoäng, SYSK02 i informatik

Framlagd: Maj, 2015

Författare: Anton Hollerup
Joakim Hermansson
Marcus Green

Handledare: Anders Svensson

Examinatorer: Odd Steen
Markus Lahtinen

Abstrakt

Titel Agila mjukvaruprojekts största utmaning

Författare Anton Hollerup
Joakim Hermansson
Marcus Green

Utgivare Institutionen för informatik, Lunds Universitet

Handledare Anders Svensson

Examinatorer Odd Steen
Markus Lahtinen

Publicerad Maj, 2015

Uppsatstyp Kandidat

Nyckelord Agilt, mjukvara, projekt, metod, utmaning, Scrum, Kanban

Sammanfattning

Denna studie fokuserar på agila projektmetoder men utesluter inte traditionella metoder, då de fortfarande används och de kan ställas mot det agila. De tre vanligaste agila metoderna (Scrum, Kanban och XP) samt den mest välkända traditionella metoden (Vattenfallsmetoden) behandlas. De agila metoderna fokuserar på att leverera värde till kunden och ett projekt anses avslutat när användaren är nöjd, medan det traditionella förespråkar att en plan sätts före projektet påbörjas och därefter följs. Trots att det finns flertalet olika metoder som följs i olika projekt så misslyckas en stor andel av IT-projekt. Syftet med studien är därför att identifiera de utmaningar som har störst påverkan på mjukvaruprojekt och således är viktigast att adressera för att förbättra utkomsten av projekten. Granskning av tidigare litteratur har gett en teoretisk bakgrund som visat att agila metoder står inför utmaningar med användarinvolvering och -acceptans, skalbarhet, samt förutsägbarhet och estimering. För att fördjupa förståelsen för dessa utmaningar och eventuellt identifiera fler utmaningar har kvalitativa intervjuer genomförts med sju personer som tillsammans har över 100 års erfarenhet från mjukvarubranschen. Informanternas val av metoder visade sig variera stort. Vi fann att informanterna i stort var överens med litteraturen om att framförallt förutsägbarhet och estimering samt användarinvolvering och -acceptans är stora utmaningar i agila mjukvaruprojekt, vilket även leder till utmaningar med kontrakt och överenskommelser. Vidare fann vi dock att utmaningen med estimering inte beror på huruvida en agil eller traditionell metod används, utan hur komplext och föränderligt projektet är. Med anledning av detta bör traditionell metodik endast användas i projekt som hanterar avgränsade och förutsägbara problem, i motsats till det agila som bör användas i föränderliga miljöer och för komplexa problem utan kända lösningar.

Innehåll

Figurer

Tabeller

1 Inledning

1.1 Bakgrund

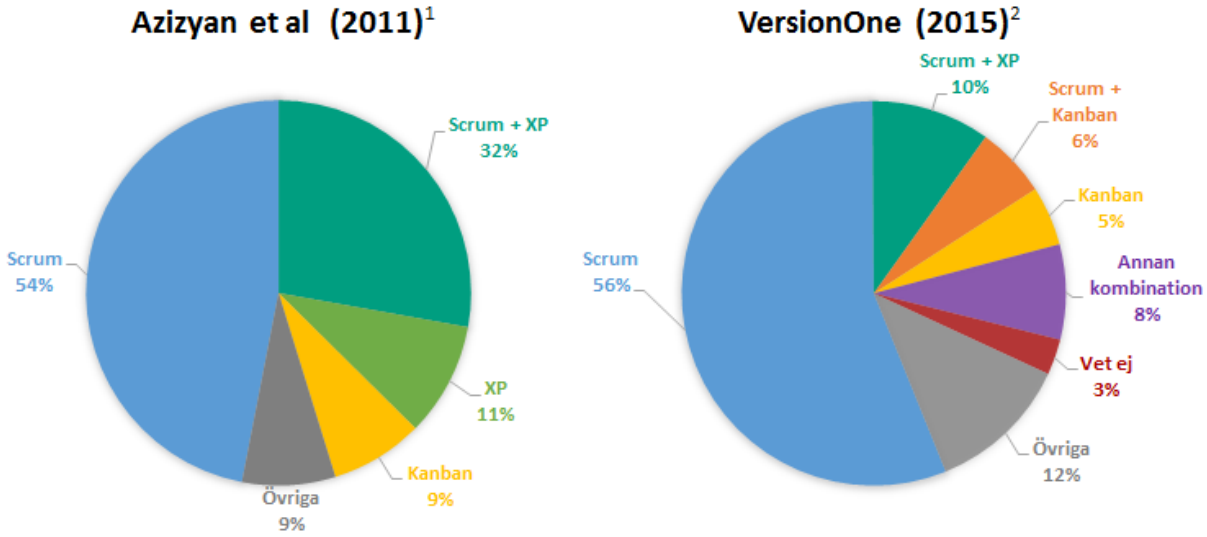
Under de senaste decennierna har marknadskrafter, systemkrav och implementering av teknologi förändrats i ökande takt. Dessa förändringar tvingar mjukvaruutvecklingen att ske i samma takt för att uppnå nya krav. Svenska företags totala utgifter för mjukvara, inklusive investeringar, uppgick år 2013 till hela 22,8 miljarder kronor (SCB 2014).

I samband med denna tillväxt och förändring har ett nytt arbetssätt uppkommit, det agila. Begreppet agil kommer från engelska "agile", som översatt till svenska betyder smidig och lättrörlig. Kärnan hos det agila är föränderlighet i form av iterationer. Varje iteration innehåller en förhållandevis liten del av det totala arbetet och löper vanligtvis ett par veckor. En styrka hos det agila är att en tänkt lösning på ett problem kan provas, utvärderas och därefter ändras vid behov innan nästa iteration påbörjas. Arbetet i ett agilt projekt anpassas således, baserat på vad som händer och upptäcks längs med vägen från start till mål (Cockburn & Highsmith, 2001). I kontrast till agila arbetssätt finns så kallade traditionella arbetssätt, vilka bygger på att en komplett och detaljerad plan tas fram för det totala arbetet innan det påbörjas och därefter följs genom hela projektet (Cho 2008).

Ständiga förändringar och förbättringar inom teknik och arbetsmiljöer leder löpande till att organisationer ställs inför nya utmaningar. Detta kan göra det problematiskt att använda traditionella, icke-agila metoder för att skapa en fullständig kravspecifikation för att klara av framtiden (Cho 2008). Med anledning av detta kan traditionella utvecklingsmetoder upplevas frustrerande och medföra svårigheter, vilket har lett till uppkomsten av olika agila metoder (Awad 2005). De agila metoderna och dess praxis är byggda på iterativa förbättringar, ett koncept som introducerades redan 1975 (Awad 2005).

Det finns ett flertal olika agila metoder. Azizyan, Magarian och Kajko-Mattson (2011) undersökte vilka metoder som användes av 121 företag i 35 länder. Företaget VersionOne (2015) utförde under 2014 en liknande studie hos 3 925 företag i primärt Nordamerika och Europa. Gemensamt för båda studierna är att de undersökte företag i flera olika industrier, samt att de är begränsade till företag som uppgett sig arbeta agilt. Scrum visade sig i båda studierna vara den mest använda agila metoden; både som enskild metod samt i kombination med andra metoder (varav eXtreme Programming (XP) och Kanban var vanligast).

Figur 1.1: Användning av olika agila metoder



[1] Till vänster: Summan av procentenheterna är 115 % på grund av att vissa respondenter uppgav att de använde olika kombinationer av metoderna, utöver kombinationen "Scrum + XP".
[2] Till höger: "Annan kombination" omfattar alla andra kombinationer än "Scrum + XP" och "Scrum + Kanban" (inklusive "Övriga" metoder).

Traditionella och agila metoder används i olika stor utsträckning i olika organisationer. Gemensamt för de flesta projekt är att de betraktas som misslyckade då de överskridit budgeterad kostnad, tid, och/eller inte nått uppsatta mål. I Tabell 1.1 illustreras en sammanställning av fyra av varandra oberoende undersökningar. Det framgår i samtliga undersökningar att mer än hälften av de IT-projekt som studerats har misslyckats.

Tabell 1.1: Andel av IT-projekt som misslyckas

Källa	Antal projekt	Antal organisationer	Andel misslyckade projekt	Studerade år
Standish Group (2013)	50 000	---	61 %	2003-2012
KPMG (2013)	---	200	67 %	2012-2013
McKinsey & Oxford (2012)	5 400	---	50 % [1]	2012
Riksrevisionen (2001)	---	73	55 % [1]	2009
[1] Omfattar uteslutande "Stora IT-projekt".				

I Sverige har flera misslyckade IT-projekt fått medial uppmärksamhet under de senaste åren. Ett av dessa var Polisens avrapporteringssystem, SiebelPUST, som resulterade i att polisens arbete tog längre tid att utföra än innan dess införande. Ett grundläggande problem i projektet

visade sig vara bristande beställarkompetens (Rikspolisstyrelsen 2013). Ett annat omtalat IT-projekt var Försvarsmaktens resurs- och ekonomiledningssystem, PRIO. Det projektet skapade frustration bland användarna och enligt Statskontorets slutrapport har projektet dragit över både tid och kostnad (Statskontoret 2010).

1.2 Problem och utmaningar

Som presenterats ovan misslyckas en stor andel IT-projekt, vilket lett till en undran om vilka faktorer som ligger bakom. Med anledning av att IT-projekt ofta innehar många uppgifter som ska lösas och att ständiga förändringar sker vilka kan påverka projektens framgång har det agila arbetssättet växt fram (Awad 2005). Trots att Scrum användes redan i början av 1990-talet (Schwaber & Sutherland, 2013) och att det agila manifestet utkom år 2001 (Beck et al, 2001) så används även traditionella metoder än idag. Om agila metoder enkelt hade kunnat användas för att lösa alla utmaningar som leder till att så många projekt misslyckas vore det rimligt att utgå från att alla projekt redan skulle drivas agilt och bli lyckade. Eftersom så inte är fallet kan frågan om vilka anledningar som ligger bakom valet av metodik ställas.

Att de agila metoderna har vuxit fram från de traditionella metoderna ser vi som ett tecken på att de uppfyller vissa behov som de traditionella metoderna inte uppfyller. Med anledning av detta betraktar vi inte det agila som ett problem i sig, utan snarare som något som uppfyller vissa behov men samtidigt står inför en eller flera stora utmaningar. Detta har lett oss till att undra vilken eller vilka dessa utmaningar är samt vilken av dem som har störst påverkan på projektens framgång och således är att betrakta som störst, vilket leder oss till vår forskningsfråga.

1.3 Frågeställning

Forskningsfrågan lyder enligt följande:

Vilken är den största utmaningen för agila mjukvaruprojekt?

1.4 Syfte

Uppsatsen ämnar att identifiera utmaningar som agila mjukvaruprojekt ställs inför, samt påvisa vilken av de identifierade utmaningarna som har störst påverkan på projekten och således är viktigast att adressera.

1.5 Avgränsningar

Uppsatsen behandlar de tre vanligaste agila metoderna Scrum, eXtreme Programming (XP) och Kanban (se Figur 1.1) samt den välkända traditionella metoden Vattenfallsmetoden (Munassar & Govardhan, 2010) (även känd som *Vattenfallsmodellen, Waterfall model*). Projektmetoder utöver dessa är avgränsade från denna studie. Vi kommer även avgränsa studien till att behandla hur dessa metoder används inom just mjukvaruprojekt. Studien kommer inte vara inriktad på de styrkor som de olika metoderna har men vi kommer inte heller bortse från dem; dels för att det inte går att förklara och behandla skillnaderna mellan metoderna utan att förklara anledningarna till att de finns, och används och dels för att de styrkor som identifieras kan vara av nytta för att bemöta utmaningarna.

2 Arbetsmetoder i mjukvaruprojekt – en teoretisk överblick

I detta kapitel definieras och förklaras olika arbetsmetoder som används i mjukvaruprojekt. Detta sker i form av genomgång och granskning av samlade teorier, där agil systemutveckling inledningsvis presenteras. Vidare beskrivs grunderna för tre agila metoder: Scrum, XP och Kanban. Dessa metoder utgör tillsammans över 70 % av användningen av agila metoder i projekt (se Figur 1.1). Vår definition av "agila metoder" är därför begränsad till just Scrum, XP och Kanban, som tillsammans med det Agila Manifestet (Bilaga 1) anses utgöra en grund för vad det agila innebär. Slutligen presenteras den traditionella metoden Vattenfallsmetoden, för att påvisa skillnader mellan agila och traditionella metoder och således fördjupa förståelsen kring vad som utmärker de agila metoderna.

2.1 Agila metoder

År 2001 samlades 17 programutvecklare för att diskutera utvecklingsmetoder, vilket ledde till skapandet av Manifesto for Agile Software Development (Beck et al, 2001), på svenska känt som det Agila Manifestet. Det Agila Manifestet kan betraktas som en sorts definition av vad som menas med agil mjukvaruutveckling. De vanligaste agila metoderna (Scrum, XP & Kanban) (Figur 1.1) presenterades i sina ursprungsformer innan det Agila Manifestet utkom men uppfyller genomgående principerna i det Agila Manifestet.

Det Agila Manifestets prioritering är att kunden blir tillfredsställd genom att ständigt leverera fungerande programvara, vilket ses som det bästa sättet att mäta framsteg (Beck et al, 2001). Under utvecklingen ska kundens ändrade krav kunna tas emot och appliceras i projektet. Detta ska ske genom god samverkan, dagliga arbeten mellan verksamhetskunniga och utvecklare, där traditionell kommunikation ansikte-mot-ansikte anses vara den bästa kommunikationsformen (Beck et al, 2001). Vidare krävs också reflektioner över tidigare arbeten för att kunna förbättra och effektivisera arbetet i fortsättningen. Till sist krävs enkelhet, att projektet ska hålla jämn utvecklingstakt samt med fokus på bästa möjliga teknik och design (Beck et al, 2001). Det Agila Manifestets tolv punkter finns i sin ursprungliga form i Bilaga 1.

Att agila metoder bygger på iterationer, återkopplingar och enkelt kan anpassas efter förändringar under projektets gång har det lett till att agila metoder kallas "light-weight methods" (lättviktsmetoder) (Awad 2005). De agila utvecklingsmetoderna passar bäst till små och mellanstora projekt (Boehm & Turner, 2003).

En kravspecifikation i agila projekt är flexibel och utökas löpande under projektets gång (Stoica, Mircea & Ghilic-Micu, 2013). Därmed finns det ingen fullt detaljerad kravspecifikation eller planering för ett agilt mjukvaruprojekt, istället bearbetas befintliga krav

efter att de har definierats och satts ihop till en kravspecifikation. Kraven avklaras var för sig och testas löpande, vilket minimerar risken för fel senare i utvecklingen (Stoica et al, 2013). Vidare ska det ske interaktion och öppen kommunikation med kunden under projektet. Andra kännetecken för agila mjukvaruprojekt är enligt Stocia et al (2013) att dokumentationen är minimal i utvecklingsfasen samt att det sker ett nära samarbete inom teamen.

Enligt Stocia et al (2013) behöver planering ske då nuläget är analyserat och börläget definierat, för att kunna visa på hur projektet ska fortgå. Då planering sker behöver kontrakt skrivas; detta sker enligt Turk, France och Rumpe (2002) ofta i upphandlingar genom flexibla kontrakt, med målpris och ungefärliga tidsuppskattningar, med anledning av att grundprincipen med agila metoder är att hålla projektet flexibelt och anpassningsbart till förändringar och tillägg.

Det är dessa grunder som de agila metoderna följer under mjukvaruprojekt. Det finns dock skillnader mellan olika agila metoder. Nedan följer presentationer av de tre vanligaste agila metoderna idag: Scrum, XP och Kanban.

2.1.1 Scrum

Vad är Scrum?

Termen Scrum kommer enligt Cho (2008) från den brittiska bollsporten rugby, där det är en strategi för att returnera en förlorad boll i spelet som innebär att spelarna samarbetar för att föra bollen framåt på planen. Detta kan likställas med hur team i agila mjukvaruprojekt samarbetar för att bearbeta och slutföra ett projekt.

Enligt Schwaber och Sutherland (2013) ska inte Scrum likställas med en process eller teknik för att bygga produkter, utan kan istället förklaras med begreppet ramverk, där olika processer och tekniker används. Inom ramverket behandlas arbetsteam, aktiviteter, artefakter samt regler som håller ihop helheten.

Scrumprocessen

Scrumprocessen kommer i följande stycke att definieras utifrån Schwaber och Sutherland (2013), vilka tillsammans presenterade Scrum för första gången år 1995. I Scrum arbetar man efter en produktbacklogg; en lista med allt nödvändigt för projektet. Detta inkluderar vad produkten behöver för egenskaper, krav och funktioner. En produktbacklogg ska vara dynamisk och kunna förändras under projektets gång, vilket innebär att den aldrig blir fullständig. Fram tills att produkten är färdigställd blir en produktbacklogg kontinuerligt större, genom nya tillägg för att nå bästa resultat (Schwaber & Sutherland, 2013).

Utvecklingsteamet tar även fram en sprintbacklogg; en prognos för vad nästa inkrement ska ha för funktionalitet och vad som krävs för att kunna nå målen (Schwaber & Sutherland, 2013). Ett inkrement är alla färdigställda uppgifter från produktbackloggen under en sprint. En sprintbacklogg innehåller detaljerad information om förändringar som dyker upp under sprinter (Schwaber & Sutherland, 2013).

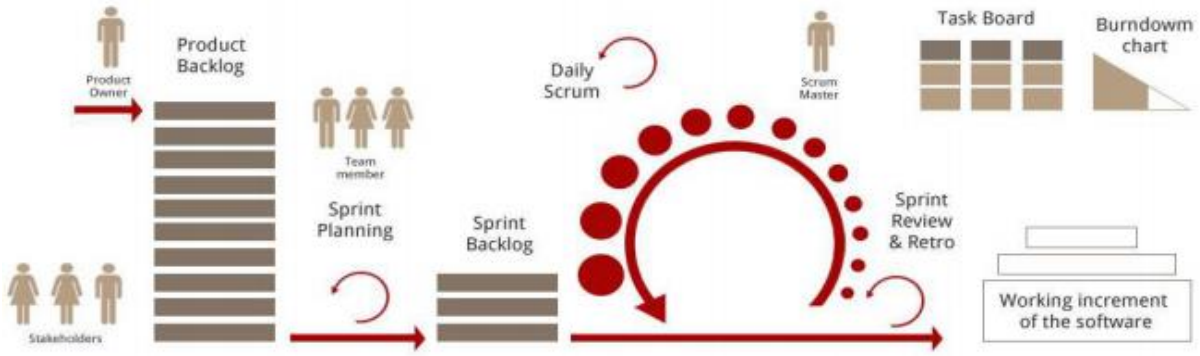
Sprintbackloggen ligger till grund för de dagliga scrummötena (även känt som *Daily Scrums* eller *Stand-up-meetings, Stå-upp-möten*) (Schwaber & Sutherland, 2013). Mötena hålls vanligtvis på samma tid och plats varje dag. Under dessa möten granskas föregående dags arbete och planeringen för vad som ska göras härnäst baseras till stor del på detta. Genom dagliga möten kan utvecklingen i projektet bedömas samt att uppdateringar angående hur projektet ligger tidsmässigt (Schwaber & Sutherland, 2013).

Enligt Rubin (2013) medför scrumprocessen flera fördelar. Några av dessa menar Rubin (2013) är: kortare cykeltider (vilket kan leda till snabbare feedback), minskad risk, lägre kostnad, och ökad motivation hos utvecklingsteamet.

Cyklerna går under namnet sprint och utgör en stor del av Scrum (Schwaber & Sutherland, 2013). Varje sprint är 1-30 dagar. I början av en sprint görs något som kallas sprintplanering; här planeras och klargörs vad målet med sprinten är, vilken information den ska innehålla, vad som ska levereras, hur resultatet ska bli samt en beskrivning på hur målet ska nås (Schwaber & Sutherland, 2013).

Vid slutet av varje sprint planerar scrummastern en sprintgranskning, vilken är en granskning för att se inkrementen (Schwaber & Sutherland, 2013). Scrummastern planerar tillfällen för dessa granskningar och då ska överenskommelser göras över vilka uppgifter som behöver göras härnäst för att förbättra produkten. Mellan en sprintgranskning och sprintplanering ska en sprintåterblick genomföras, det är en återblick där samtliga involverade ska granska vad som gjorts samt vad som kan ändras till det bättre inför nästa sprint (Schwaber & Sutherland, 2013). Efter varje avslutad sprint påbörjas en ny sprint, tills projektet når sitt slut.

Figur 2.1: Scrumramverket



(Sverrisdottir, Ingason & Jonasson, 2014, s. 259)

Roller i Scrum

Scrum förespråkar tre roller:

1. **Produktägaren**: Produktägaren är ansvarig för finansieringen av projektet under dess livscykel och sätter projektets krav och mål. Sverrisdottir, Ingason och Jonasson (2014) hävdar att denna roll är en av de viktigare i Scrum. Produktägaren ansvarar även för hantering, ordning och prioritering av uppgifter i produktbackloggen. Produktägaren ska inte vara ett team, utan en enskild person, och fungerar som en representant för alla intressenter i projektet (Sverrisdottir et al, 2014). Därför är det vanligt att produktägaren är en person från kunden som projektet ska levereras till.
2. **Scrummaster**: Scrummasterns huvudsakliga uppgift är att se till att arbetet sker enligt Scrum och att deltagarna följer dess teori och regler (Schwaber & Sutherland, 2013). Scrummastern hjälper både produktägaren och utvecklingsteamet att öka förståelse och interaktioner mellan samtliga aktörer. Scrummastern säkerställer även att produktägaren utformar en produktbacklogg på rätt sätt samt tar fram lösningar på hur hanteringen ska ske. En scrummaster coachar även utvecklingsteamet och underlättar deras arbete genom vägledning samt hittar och eliminera svagheter (Schwaber & Sutherland, 2013).
3. **Utvecklingsteam**: Utvecklingsteamet består av yrkesmänniskor som arbetar med att omvandla uppgifterna i produktbackloggen till inkrement (Schwaber & Sutherland, 2013). Teamet arbetar på ett självorganiserat sätt och deltagarnas kompetenser slås samman för att nå uppsatta mål. Det finns inga uppdelningar inom detta team – alla bär lika ansvar för arbetet.

Scrums begränsningar

Cho (2008) lyfter flera utmaningar med Scrum, varav en är kommunikation. Cho (2008) upptäckte i sin studie att eftersom de dagliga mötena endast hålls separat inom varje team, existerar inte övergripande kommunikation mellan de olika teamen på samma vis. De dagliga mötena lägger en bra grund för att inte dubbelarbete utförs inom samma team, men då

övergripande kommunikation saknas kan samma arbete utföras utav två team, utan att detta uppmärksammas i ett tidigt skede. I en studie som Sutherland, Viktorov, Blount och Puntikov (2007) har genomfört påvisas att bristen på effektiv kommunikation är ett stort problem med Scrum. Parnas (2006) påpekar att ineffektiv kommunikation är en av de vanligaste orsakerna till misslyckade projekt.

En annan utmaning Cho (2008) påvisar är att få kunden involverad i projektet. Kunden är ofta upptagen med ordinarie uppgifter som sker jämsides med utvecklingsprojekten. Vidare hävdar Cho (2008) att sprintplaneringen och sprintgranskningen inom Scrum även kan leda till ineffektivitet på grund av att de endast är givna saker som tas upp på dessa möten. Detta leder till både tids- och arbetsspill.

Tanner och Mackinnon (2013) kunde i sin undersökning se djupare på problemen med Scrum samt förklara hur de kan undvikas. Ett problem visade sig vara att arbetsbelastningen kunde bli för hög inom teamen. Det leder till att exempelvis utvecklarna tvingas arbeta övertid för att hinna med arbetsuppgifterna inom respektive sprint (Tanner & Mackinnon, 2013). Vidare visar undersökningen att detta i sin tur kan bero på att arbetsuppgifternas omfattning uppskattades fel vid planeringen och att hög arbetsbelastning kan leda till förvirring och misstag. Det kan hända att en utvecklare arbetar inom flera olika sprintar samtidigt, vilket inte ska ske i Scrum då det kan leda till att flera uppgifter kan röras ihop och bli ogjorda.

Sammanfattning av Scrum

Scrum är en iterativ och inkrementell metod som lämpar sig främst när det är svårt att genomföra initial planering. Mjukvaran utvecklas av självorganiserade team och det finns uppsatta mål att arbeta efter. Syftet med Scrum är att minimera nedlagd tid och arbete på mindre viktiga uppgifter. Arbetet sker genom sprintar; varje sprint innehåller samma struktur och påbörjas med planering samt avslutas med bedömning. Det är kunden som utformar kravlistan och vad som ska genomföras i nästkommande sprint. Teammedlemmarna har dagliga möten för att samordna verksamheten och arbetar därefter självständigt för att lösa uppgifterna på eget vis. Scrummastern ansvarar för att respektive team arbetar effektivt genom att lösa problem som uppstår under projektet.

2.1.2 eXtreme Programming (XP)

Vad är XP?

Beck & Andres (2004) förklarar eXtreme Programming (XP) som en lättviktig metod för utvecklingsteam i alla storlekar. Metoden behandlar risker på alla nivåer i utvecklingsprocessen och anpassar sig till vaga eller snabbföränderliga krav. Metoden togs fram på grund av att många mjukvaruprojekt avbröts eller inte höll projektplanen. XP är produktiv och levererar högkvalitativ mjukvara, samtidigt som den går ut på att bemöta begränsningarna inom mjukvaruutveckling och endast fokusera på de delar som behövs för att skapa värde för kunden

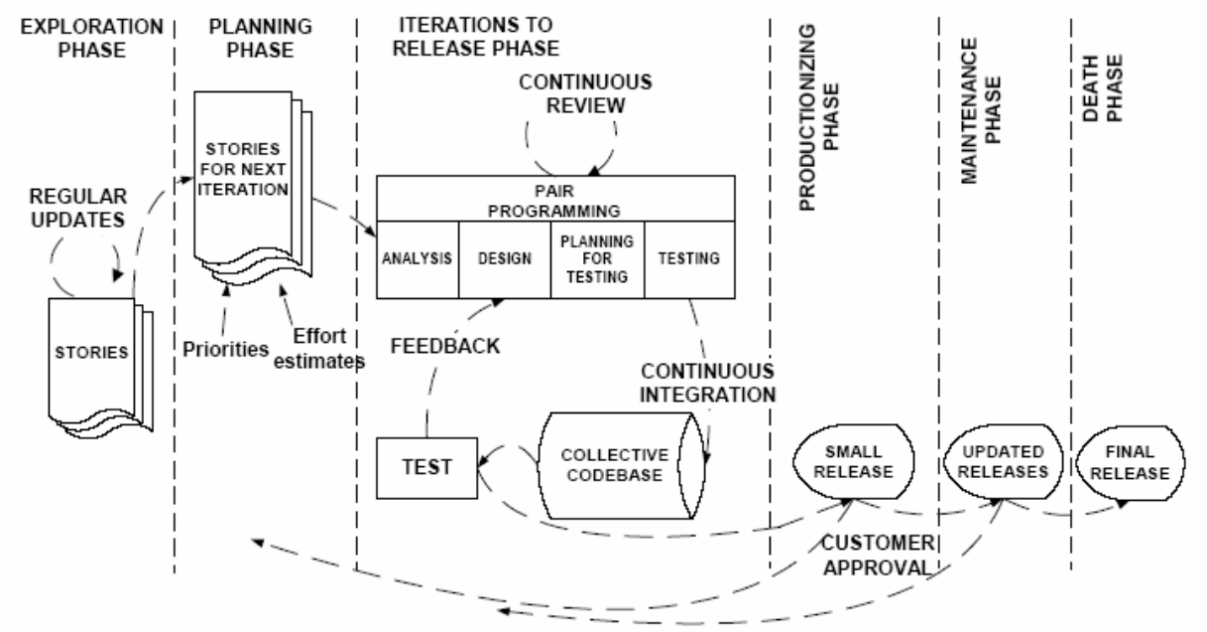
(Beck & Andres, 2004). XP bygger på fem värderingar: kommunikation, enkelhet, mod, respekt och återkoppling (Beck & Andres, 2004) (se Bilaga 2).

XP-processen

Processen bygger på små cykler, inkrementell planering, kontinuerlig feedback, kommunikation och revolutionerande utformning (Awad 2005). Respektive cykel bör vara en till två veckor (Shiohama, Washizaki, Kuboaki, Sakamoto & Fukazawa, 2012).

XP har sex faser: utforskning, planering, iteration och release, produktion, underhåll, och dödsfas (Awad 2005). I utforskningsfasen skriver kunden user stories, vilket leder till planeringsfasen där en prioritering och en tidsestimering sker. I iteration- och releasefasen skapar utvecklingsteamet en del av systemet medan de kontinuerligt testar och integrerar koden. I produktionsfasen sker ytterligare testning och prestandakontroller av systemet innan det överlämnas till kunden. Idéer och förslag som uppkommer i fasen dokumenteras och implementeras sedan i underhållsfasen. Den sista fasen Awad (2005) nämner är dödsfasen, den börjar när kunden inte har mer att tillägga och all nödvändig dokumentation av systemet skrivs ner då ingen mer kod ska skrivas.

Figur 2.2: XP-processens livscykel



(Beck & Andres, 2004)

XP innehåller 13 primära praktiker (Bilaga 3) och tolv följdsatser (Wake 2000). De primära praktikerna är avsedda att vara oberoende av varandra och de övriga följdsatserna, detta även om interaktionen mellan de olika tillvägagångssätten kan stärka deras effekt (Beck & Andres,

2004). Stand-up meetings förekommer i samband med XP, trots att det inte finns med som en del av den "officiella" XP-praktiken (Wake 2000).

Roller i XP

XP har sex primära roller: Chef, Coach, Tracker, Programmerare, Testare och Kund (Wake 2000) (Bilaga 4).

XP:s begränsningar

Munassar och Govardhan (2010) fann i sin jämförande studie av mjukvaruutvecklingsmetoder att XP har utmaningar; bland annat är det svårt att applicera på stora projekt där dokumentation är viktigt. Det krävs även expertis och erfarenhet för att få projektet att flyta på samt för att kunna konstruera testfall.

Sammanfattning av XP

XP lämpar sig för projekt med dynamiska krav eller projekt vilka inte är definierade från början. I metoden XP anses programutveckling i störst utsträckning innebära att skriva program. Teammedlemmarna ska utbyta kunskap och bli duktiga programmerare. XP ska inte generera mycket dokumentation, istället behöver det finnas ett samarbete mellan kund och programmerare.

2.1.3 Kanban

Vad är Kanban?

Kanban kommer ursprungligen från biltillverkaren Toyota, där det togs fram för att styra materialflödet in på fabriksgolvet; när de nödvändiga delarna börjar ta slut ska nya skjutas in (Toyota 2015). Detta bidrar till att minsta möjliga mängd nödvändiga delar finns tillgängliga på fabriksgolvet, vilket även leder till kortare ledtider (Kniberg & Skarin, 2009). Denna bakgrund med Kanban inom Toyota är grunden till användandet av Kanban som utvecklingsmetod.

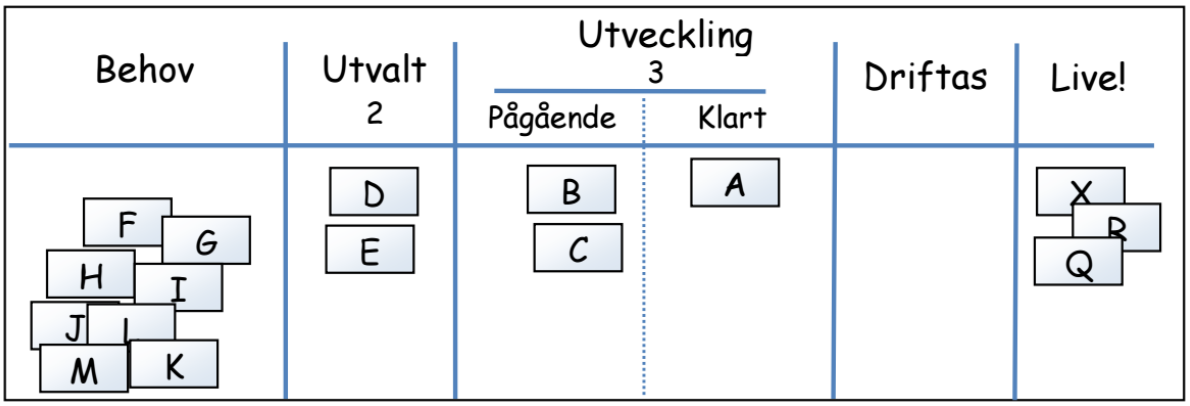
Kanbanprocessen

Kniberg och Skarin (2009) skriver att Kanban är ett dragsystem för planering och fungerar på så vis att teamet väljer när och hur många uppgifter de ska förhålla sig till. Vidare skriver de att i Kanban är visualisering av arbetsflödet en stor del och att detta görs genom att dela upp arbetet i mindre delar och skriva varje post på ett kort. Därefter sätts korten på en tavla som har namngiva kolumner för att illustrera var i flödet posten befinner sig (Kniberg & Skarin, 2009).

Enligt Kniberg och Skarin (2009) ska det finnas en person med god insyn och som ska veta vilka arbetsposter som bör prioriteras. Den personen ser till att förflytta arbetsposter från specifikationslistan till nästa steg (se Figur 2.3 nedan) vilket är "Utvalt". Detta betyder dock

inte att uppgiften är påbörjad, utan endast att den ska påbörjas härnäst. När det finns tid tas denna uppgift över till nästa steg av teamet i fasen "Pågående" (under utveckling). I det steget finns flera personer inom olika områden med många olika kompetenser. När arbetsposten är slutförd flyttas den över till sista steget, "Live!". Det är enligt Kniberg och Skarin (2009) först då nästa arbetspost tas in i steget "Pågående" och beroende på projektet sätts en begränsning för hur många arbetsposter som får vara pågående samtidigt. Den begränsningen bidrar till att minsta möjliga arbete sker i stegen. Desto fler uppgifter som utförs samtidigt, desto längre tid tar det att slutföra samtliga uppgifter. Därför finns det en begränsning vilket ökar effektiviteten och minskar totaltiden av projektet. Under tiden är det viktigt att mäta cykeltiden, den genomsnittstid det tar att genomföra en arbetspost. Detta optimerar processen och maximerar förutsägbarheten angående den totala ledtiden (Kniberg och Skarin, 2009).

Figur 2.3: Kanbantavlan



(Kniberg & Skarin, 2009, s 41)

Roller i Kanban

Kanban föreskriver inte roller (Kniberg & Skarin, 2009).

Kanbans begränsningar

Inom Kanban är målet att försöka undvika att utföra icke värdefullt arbete. Ikonen, Kettunen, Oza och Abrahamsson (2010) visar dock i sin forskning att det förekommer flertalet aktiviteter som kan anses vara slöseri med tid och arbete. Bland annat visar de att arbeten ibland endast blir delvis slutförda, vilket leder till problem senare i ledet. Om en uppgift blir halvfärdig eller försenas stoppar det inflödet av andra uppgifter, vilket i sin tur leder till förseningar i projektet. Det bestämda antalet uppgifter som finns inom Kanban finns för att minimera det pågående arbetet. Detta kan dock ställa till det då flera parter behöver invänta uppgifter som befinner sig i tidigare steg. Väntetiden kan inom Kanban uppgå till flera veckor (Ikonen et al, 2010).

Sammanfattning av Kanban

Med Kanban kan ett projekt visualiseras och styras på ett tydligt sätt. Syftet med Kanban är att minsta möjliga arbete ska ske samtidigt vilket leder till minskade ledtider. En arbetspost får inte gå vidare i projektet om det finns andra arbetsposter som inte är slutförda, då behöver befintliga arbetsposterna avklaras alternativt byta plats.

2.1.4 Jämförelse av de tre agila metoderna

I Tabell 2.1 nedan är identifierade skillnader mellan de tre agila metoderna Scrum, XP och Kanban sammanställda. Jämförelsens syfte är att påvisa och skapa förståelse för vad som skiljer de olika metoderna från varandra.

Tabell 2.1: Jämförelse av Scrum, Kanban och XP

Område	Scrum	Kanban	XP
Typ	Ramverk (Schwaber & Sutherland, 2013)	Dragsystem (Just-in-Time) (Kniberg & Skarin, 2009)	Ramverk och fördjupad praktik för kodleverans (Beck & Andres, 2004)
Roller	Föreskriver tre roller: Produktägare Scrummaster Utvecklare (Schwaber & Sutherland, 2013; Sverrisdottir et al, 2014)	*Ej specificerat* (Kniberg & Skarin, 2009)	Föreskriver sex roller: Chef, Coach, Tracker, Programmerare, Testare, Kund (Wake 2000) (Bilaga 4)
Iterations-längd	Medium (upp till 30 dagar) (Schwaber & Sutherland, 2013)	Kort (1 vecka) *Kanban föreskriver dock inte iterationer; kan väljas själv.* (Kniberg & Skarin, 2009)	Kort (cirka 1-2 veckor) (Shiohama et al, 2012)
Kund-involvering	Medium (efter varje sprint) (Schwaber & Sutherland, 2013)	Låg (Kniberg & Skarin, 2009)	Hög (daglig input) (Beck & Andres, 2004)
Teamstorlek	Små team (Helst mindre än nio personer) (Schwaber & Sutherland, 2013)	*Ej specificerat* (Kniberg & Skarin, 2009)	Team i alla storlekar (Beck & Andres, 2004)
Uppgifts-fördelning	Uppgiftslistan för sprinten ägs av ett specifikt team (Kniberg & Skarin, 2009)	Kanbantavlan delas av flera team eller individer (Kniberg & Skarin, 2009)	En uppgiftslista för iterationen ägs av kunden (Wake 2000)
Produkt-ägare	En representant från kunden (Sverrisdottir et al, 2014)	*Ej specificerat* (Kniberg & Skarin, 2009)	En representant från antingen kunden eller utvecklingsorganisationen (Wake 2000) (Bilaga 4)

2.2 Traditionella metoder

De traditionella metoderna – som även går under namnet plandrivna metoder (Boehm & Turner, 2003) – anses vara det traditionella sättet att utveckla programvara. Traditionella utvecklingsmetoder är i allmänhet enkelt utformade och består övergripande av fyra steg i form av frågor och agerande: (1) Vad krävs? (2) Hur ska det genomföras? (3) Genomför enligt föregående steg (4) Kontrollera att allt genomfördes enligt föregående steg (Gugenberger 2007). Enligt Awad (2005) bygger metoderna på en sekventiell serie av steg, såsom kravställning, lösningsbyggnad, testning och driftsättning. Dessa utvecklingsmetoder kräver därför att en tydlig och omfattande definition och dokumentation görs, eftersom en hel uppsättning krav redan beslutas i uppstarten av projektet (Awad 2005). Traditionella metoder skapar bättre förutsättningar för större projekt; de planer, dokument och processer som föreskrivs ger bättre kommunikation och samordning mellan stora grupper. Detta kan dock vara mycket ineffektivt i små projekt (Boehm 2002).

Boehm (2002) påpekar att metoderna använder planer för att förankra sina processer. Planerna utgör en stor del av den dokumentation som krävs i de flesta plandrivna strategier. Vidare skriver de att traditionella metoder är väldigt beroende av dokumenterade processplaner som scheman, milstolpar och rutiner samt produktplaner såsom krav, arkitektur och standarder för att koordinera ett projekt.

2.2.1 Vattenfallsmetoden

Vad är Vattenfallsmetoden?

Vattenfallsmetoden är en strategisk utvecklingsmodell, där utvecklingen kan ses som en ström ner i fyra faser: kravanalys, design, implementation och testning (Sharp, Rogers & Preece, 2007). Vattenfallsmetoden kallas tungviktsmetod på grund av den omfattande planering, dokumentation och design som metoden förespråkar (Awad 2005).

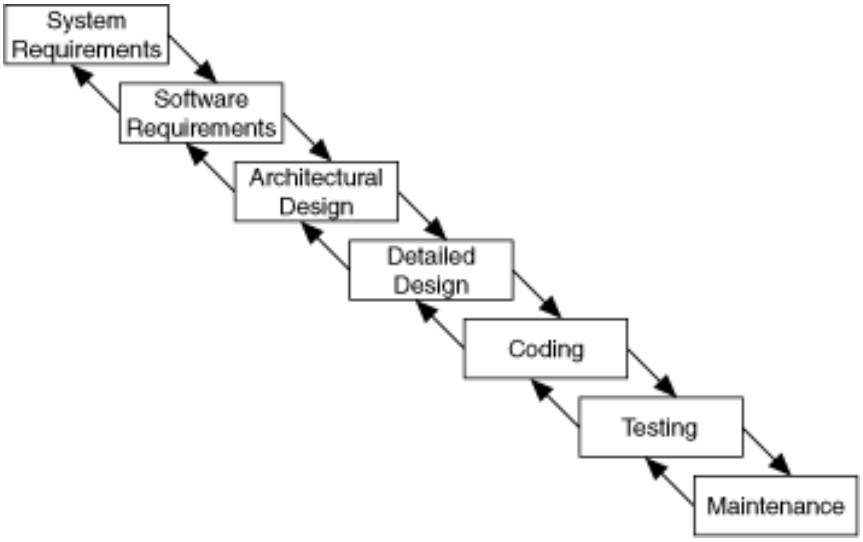
Dess intensiva dokumentering och planering gör att metoden fungerar väl för projekt där kvalitetssäkring är en viktig del (Sharp et al, 2007). Vidare hävdar författarna att Vattenfallsmetoden inte är lika flexibel jämfört med de agila metoderna, på grund av att nästa steg inte påbörjas förrän det tidigare steget är helt färdigställt och har verifierats. Metoden förbjuder dock inte möjligheten att gå tillbaka och ändra i en tidigare fas men detta involverar ofta en kostsam omarbetning (Munassar & Govardhan, 2010).

Vattenfallsprocessen

Vattenfallsmetoden är en av de äldsta metoderna för systemutveckling (Munassar & Govardhan, 2010). De fyra faserna som Vattenfallsmodellen består av kan utvecklas och presenteras tydligare genom följande sju steg enligt Munassar och Govardhan (2010):

- Systemkrav: Bestämmer komponenterna för uppbyggnaden av systemet, detta inkluderar, hårdvara, mjukvara och andra nödvändiga komponenter.
- Programkrav: Sätter förväntningarna för mjukvarufunktionalitet och identifierar systemkrav som programvaran påverkar. Analysen inkluderar beslut om vilken interaktion som behövs med andra applikationer och databaser, prestandakrav, krav på användargränssnitt.
- Arkitektur: Bestämmer ramen för systemet för att möta de specifika kraven. Designen definierar de viktigaste komponenterna och samspelet mellan dem men det definierar inte strukturen av varje komponent. De externa gränssnitt och verktyg som används i projektet kan bestämmas av konstruktören.
- Detaljerad design: Undersöker mjukvarukomponenter som definieras i arkitekturen och skapar en specifikation för hur varje komponent ska implementeras.
- Kodning: Genomförandet av specifikationerna som tagits fram i detaljerad design.
- Testning: Bestämmer om programvaran uppfyller de specifika kraven och hittar de eventuella fel som kan finnas i koden.
- Underhåll: Påpekar problem och förbättringsförslag efter de olika programversionerna.

Figur 2.4: Vattenfallet



(Munassar & Govardhan, 2010)

Vattenfallsmetodens begränsningar

Munassar och Govardhan (2010) ser vissa problem och begränsningar med metoden:

- Idealiserad, matchar inte verkligheten
- Speglar inte den iterativa naturen hos utforskande (exploratory) utveckling
    - Det är orealistiskt att förvänta noggranna krav i så tidigt skede
- Mjukvara levereras sent i projektet, vilket förskjuter upptäckten av allvarliga fel
- Svårt att integrera riskhanteringen
- Svårt och dyrt att göra ändringar i dokument, att ”simma uppströms” (se Figur 2.4)
- Stor administrativ overhead-kostnad, dyrt för små team och projekt

Vattenfallsmetoden gör det även svårt att se resultat, då utvecklingsdelen kommer i ett så pass sent skede. Detta kan i sin tur bli förvirrande för ledning och kunder. Enligt Munassar och Govardhan (2010) tycker många även att all den dokumentation som görs i Vattenfallsmetoden är överdriven och oflexibel.

Sammanfattning av Vattenfallsmetoden

Vattenfallsmetoden är en metod som passar sig bäst till fasta och definierade krav. Metoden är inte flexibel utan utgår strikt efter fasta faser, dock säkerställer metoden leverans av kvalitet. Metodens styrkor är att den är lätt att förstå och tillämpa, är välkänd, stärker goda vanor samt att metoden sätter och identifierar delresultat och milstolpar (Munassar & Govardhan, 2010).

2.3 Jämförelse av agil och traditionell mjukvaruutveckling

I Tabell 2.2 nedan presenteras skillnaderna mellan de traditionella och agila utvecklingssättet. Jämförelsen ämnar att ge en bättre bild och förståelse för var och vad det är som skiljer metoderna åt.

Tabell 2.2: Jämförelse av traditionellt och agilt

Område	Traditionellt	Agilt
Användarkrav	Detaljerat och definierat innan kodning och implementation	Interaktiv input
Kostnad för omarbete	Hög kostnad	Låg kostnad
Utvecklingsriktning	Fast	Flexibel
Testning	När kodningen är klar	Under varje iteration
Teamstorlek	Stora team	Små team
Kundinvolvering	Låg involvering	Hög involvering
Skalar sig bäst till	Stora projekt	Små till mellanstora projekt
Arkitektur	Designad för nuvarande och förutsatta krav	Designad för nuvarande krav
Huvudmål	Hög säkerhet	Snabbt värdeskapande

(Stoica et al, 2013, s. 71, modifierad)

2.4 Utmaningar för agila mjukvaruprojekt

Även om många positiva effekter har uppkommit genom de agila metoderna (såsom kortare utvecklingscykler, högre kundtillfredsställelse, mindre buggar och snabbare anpassning till föränderliga behov (Cho 2008)) har det även framkommit utmaningar och svårigheter.

2.4.1 Användarinvolvering och -acceptans

Boehm (2002) anser att de agila metoderna har en stark betoning på användarinvolvering och att användarna är att betrakta som en del av utvecklingsteamet under hela utvecklingsprocessen:

> "Agile methods work best when such customers operate in dedicated mode with the development team, and when their tacit knowledge is sufficient for the full span of the application." (Boehm 2002, s 4)

Boehm och Turner (2003) påvisar även att om agila projekt ska lyckas måste kundens representanter vara i fas med både systemanvändarna och utvecklingsteamet. Vidare menar de att problem kan uppstå när representanterna inte beskriver de behov och önskemål som användarna har på ett korrekt sätt. Detta på grund av bristande förståelse och på så vis kan representanter istället bli en bristningspunkt i agila metoder.

Awad (2005) hävdar att de agila utvecklingsmetoderna inte alltid kräver att alla inblandade ska ha lika djup kunskap eller kapacitet, då metoderna förlitar sig på att tyst kunskap ska utformas inom teamet, snarare än att kunskapen ska dokumenteras. Detta kan enligt Awad (2005) anses vara en begränsning hos de agila utvecklingsmetoderna. Författaren menar att kunskap som inte dokumenteras kan leda till arkitektoniska misstag och att det inte kan upptäckas av externa granskare i ett senare skede.

2.4.2 Skalbarhet

Både Boehm och Turner (2003) samt Boehm (2002) hävdar att de agila metodernas snabbhet och smidighet till stor del kommer från avsiktlig teamplanering, vilket möjliggör att projektet kan drivas med tyst kunskap istället för dokumenterad kunskap (i form av planer eller specifikationer). De pekar även på att detta kan bli svårt att efterleva efterhand som projektet skalas upp.

Cockburn och Highsmith (2001) skriver att anpassningsmöjligheter till större team blir problematiskt när de agila utvecklingsmetodernas skalas upp till större projekt: *"Agile development is more difficult for larger teams […] as size grows, coordinating interfaces become a dominant issue."* (Cockburn & Highsmith, 2001, s. 133)

Turk et al (2002) uppger att då agila mjukvaruprojekt tillråder god samordning med kommunikation inom teamen, lämpar sig agilt bäst på små till medelstora team. Vidare säger Turk et al (2002) att genom större team kan de kraven på samordning och kommunikation istället minska den positiva effekten av aktiviteter såsom möten ansikte-mot-ansikte samt granskningsmöten.

Skalbarheten i agila projekt är något även Solinski och Petersen (2014) påstår är svårt och säger att de är en av de största begränsningarna för agila projekt. Vidare skriver de att skalbarhet är en erkänd fråga inom agila metoder och det antas ofta att agilt endast lämpar sig för små grupper och projekt. Även om vissa lösningar för detta existerar, har majoriteten av deltagarna i deras forskningsstudie erkänt att skalbarheten är en betydande begränsning (Solinski & Petersen, 2014).

2.4.3 Förutsägbarhet och estimering

Armour (2014) skriver i sin artikel att en utmaning i agila mjukvaruprojekt är estimering. Då ett agilt arbetssätt tillämpas sker leveranser i flera omgångar; det sker förändringar under projektets gång vilka läggs in i en kravspecifikation. Armour (2014) skriver att kraven inte kan definieras till hundra procent i ett agilt mjukvaruprojekt och att det inte heller bör eftersträvas i de flesta fall. Armour (2014) anser att någon sorts estimering behöver göras inför projekt, men att det är viktigt att förstå och vara medveten om vilken typ av projekt det rör sig om. Att planering och estimering är svårt påpekas även av Cohn (2005), som därför hävdar att estimering istället övergår till en förbindelse mellan leverantör och kund.

Planering behövs även för att veta vem som behöver vara tillgänglig vid vilket tillfälle, samt för att veta om projektet överhuvudtaget är på rätt spår:

> ”If we do not know what we want to do we cannot expect to figure out precisely how long it will take or what it will cost. But if we do not have a baseline of some expectations we will not even know where we are or where we are going.” (Armour 2014, s 43)

Vidare trycker Armour (2014) på att estimering är viktigt och att det är gemensamt för samtliga projekt. Om ingen estimering finns kan inte ett projekt drivas, även om det är ett agilt projekt. Boehm och Turner (2003) uppskattar att 20 % av tiden läggs på initial planering eller omplanering i agila projekt.

Cockburn och Highsmith (2001) säger att friheten inom de agila metoderna kan skapa bättre arbetsflöden, men samtidigt försvåra tydligheten i vad det slutgiltiga målet är och hur projektet ska genomföras. Viktiga faktorer för att projekt ska lyckas är de mänskliga faktorerna såsom talang, kommunikation och skicklighet (Cockburn & Highsmith, 2001). Då de mänskliga faktorerna brister, till exempel om de involverade deltagarna saknar driv och inte är samarbetsvilliga, förpliktigade eller kunniga finns risken att kraven blir ofullständiga (Awad 2005).

3 Undersökningsmetod

3.1 Tillvägagångssätt

Forskningsfrågan vi ämnar att besvara i denna uppsats är: Vilken är den största utmaningen för agila mjukvaruprojekt? För att besvara forskningsfrågan har användningen av agila metoder i mjukvaruprojekt undersökts genom kvalitativa intervjuer med informanter från sju olika IT-företag.

För att inte påverka rapportens utgång i förväg har inget förväntat resultat av forskningen redovisats. Mot denna bakgrund beslutades även att inte på förhand utforma frågor som kunde antas ge förväntade svar.

3.2 Intervjuer

För att beröra forskningsfrågan ur ett brett perspektiv samt för att ämnet agila metoder är komplext och dess användning kan ske i stor variation, valde vi att utföra kvalitativa intervjuer framför kvantitativa intervjuer. Att utföra kvantitativa intervjuer där ett formulär skickas ut eller genom att följa en intervjumall med frågor av begränsad omfattning (Jacobsen 2002) bedömde vi inte skulle tillföra det empiriska material som detta uppsatsämne kräver. Att styra och forma det empiriska resultatet i en given riktning är inget vi avsåg att göra. Därmed valde vi att genomföra kvalitativa intervjuer med sju personer, vilket resulterade i stora mängder data i form av synpunkter och erfarenheter från de utvalda informanterna.

3.2.1 Intervjuguide

Vår intervjuguide (Bilaga 5) fokuserade på tre områden:

(1) Informanten, (2) Metodik, (3) Mjukvaruutveckling i allmänhet.

Första området fokuserade på informanten ifråga, vilken roll och vilka tidigare erfarenheter personen har. Detta för att påvisa relevansen och legitimiteten i informanternas svar. Vidare har intervjun övergått till den mest omfattande delen – metodvalet – där fokus låg på hur informanten valde att använda metoderna. Vidare fortsatte intervjun mer allmänt inom ämnet mjukvaruutveckling och vad informanten ansåg fungera bra eller mindre bra med de olika metoderna. Samtliga delar hölls öppna för att informanten skulle besvara våra frågor så omfattande som möjligt och därmed ge oss svar på våra mer specifika funderingar, utan att ställa ledande frågor eller rikta fokus direkt dit.

Innan intervjutillfället skickades ett förberedande dokument ut till informanterna (Bilaga 6). Det förberedande dokumentet var till viss del förberedande för oss själva såväl som för informanterna, dock var inte tanken att informanterna skulle förse oss med helt förbereda svar under intervjuerna utan mer ha fått chansen att fundera kring ämnet. Det förberedande dokumentet var också för att vi skulle få en bild av vilka utvecklingsmetoder informanterna använder, för att bättre kunna förbereda följdfrågor samt vara förberedda utifall en, för oss, ny utvecklingsmetod kommer på tal. Vidare berättade vi i det förberedande dokumentet inom vilka områden vi skulle rikta fokus på under intervjuerna.

Genom att ha en intervjuguide har inte en helt öppen intervju hållits. Vi strävade dock efter att inte ha strukturerade intervjuer eller att styra informanternas svar. Därför valde vi att ha semi-strukturerade intervjuer utformade enligt Jacobsen (2002). Vi följde en intervjuguide med tema och stödfrågor vi önskade svar på; dock utan fast ordningsföljd och informanten fick styra svaren i sin egen riktning. Det ledde i sin tur till en levande dialog med informanterna, där informanternas kunskap och åsikter har insamlats. Att vi valde bort en helt öppen intervju utan intervjuguide och ordningsföljd beror på att den insamlade datan skulle bli så komplex och inte hanterbar för vår vidare analys (Jacobsen 2002).

3.2.2 Urval

Urvalet av informanter utgjordes av ett bekvämlighetsurval (Jacobsen 2002). Då uppsatsens undersökning har fokus på agilt arbetssätt är urvalet riktat mot personer med lång erfarenhet av detta. Urvalet har skett genom att tillfråga IT-företag efter personer vilka har ansetts kunna tillbringa studien god och relevant information. Samtliga informanter nåddes via författarnas kontakter i branschen. Ingen av dem är dock närstående (släkt eller vänner) till författarna.

Gemensamt för samtliga informanter är följande:

- Lång erfarenhet från IT-branschen i allmänhet och mjukvaruutveckling i synnerhet
- Har ansvarsroller inom IT, såsom projektledare och chefer

För att undvika svar från endast en sida har variation i följande kriterier prioriterats:

- Levererar egenutvecklade respektive befintliga lösningar
- Ansvarsroller

3.2.3 Datainsamling och analysmetod

Datainsamlingen till uppsatsen har skett genom kvalitativa intervjuer. Samtliga intervjuer har spelats in efter samtycke från informanterna. Vi önskade att hålla intervjuerna på deras arbetsplatser eftersom det oftast blir en bättre kontakt och genom forskning även bevisats ge mer öppna och sanningsenliga svar (Frey & Oishi, 1995). Vi har dock varit tvungna att i några fall ha intervjuer över Skype samt telefon, då vi befunnit oss på olika geografiska platser. Trots

nackdelen att inte träffas ansåg vi att personerna i fråga skulle ha såpass värdefulla erfarenheter och kunskaper kring forskningsfrågan att vi inte ville exkludera dem.

Vi försökte hålla intervjuerna till en timme, vilket vi ansåg vara tillräckligt för att få önskat omfång på svar samtidigt utan att stressa fram.

Under transkriberingsprocessen upptäcktes att en av intervjuinspelningarna inte uppnådde den kvalité som behövdes för att kunna transkribera varje ord, på grund av tekniska problem vid inspelningen. De delar som inte lyckats tydas har därför markerats med hakparenteser med streck i, "[---]". Vidare har den nästan fullständiga transkriberingen skickats till informanten för godkännande, varpå informanten fyllt i tomrummen, samt omformulerat några svar för förtydligande.

De andra intervjuerna samt transkriberingarna genomfördes utan hinder och skickades tillbaka till informanterna för godkännande. Vidare analyserades de insamlade svaren för att sedan kategoriseras. Analys och kategorisering gjordes genom att läsa igenom samtliga transkriberingar och i uppsatsen lyfta de delar där informanterna nämnt svårigheter eller stött på hinder. Då delar från olika informanter var sammanhängande placerades de tillsammans under en övergripande rubrik, exempelvis "Estimering". Detta eftersom samtliga intervjuer omfattade stora områden och gav svar på specifika frågor i ett ofta mer inkluderande svar. Då samtliga intervjuer i transkriberad form uppgått till cirka 100 sidor har styckesindelning och kategorisering varit nödvändigt för vidare analys samt större tydlighet vid presentation av resultaten i kapitel 4. Transkriberingarna återfinns i Bilagor 7 till 13.

3.2.4 Etik

Informerat samtycke

Vi har varit tydliga inför informanterna med vad vår undersökning ämnar att undersöka och att det kommer bli en allmän handling vid examination. Vi informerade även informanterna i förväg om vilka ämnen vi önskade diskutera under intervjuerna och att vi önskade spela in dem.

Fullständig transkribering med godkännande

Fullständig transkribering gav möjlighet att ytterligare öka reliabiliteten i den insamlade informationen (Jacobsen 2002). När respektive transkribering var färdig sändes den till informanten, för att få godkännande och ett slutgiltigt bevis på att inget missuppfattats eller oönskat tagits med. Om informanterna har gjort ändringar utöver stavfel har det tydliggjorts genom att skrivas inom måsvingar, "{ }".

Anonymitet av känslig data

I samråd med informanterna har valet att inte inkludera deras namn eller namn på företag som de tidigare arbetat för gjorts. Däremot har namnen på deras nuvarande företag presenteras, då det bidrar till att visa skillnaderna mellan hur olika typer av företag arbetar med projekt. För att vidare säkerställa informanternas anonymitet är rollbeskrivningarna och exakt antal års erfarenhet generaliserade (en specifik titel i kombination med ett specifikt företagsnamn, den månad då uppsatsen läggs fram, samt exakt antal år i branschen kan leda till att anonymiteten inte längre kan säkerställas). Gemensamt för rollerna med "Brett IT-ansvar" (Tabell 4.1) är ett stort ansvar för hela eller stora delar av företagens verksamhet, med direkt koppling till utveckling och/eller leverans av företagets mjukvaruprodukter och tjänster.

Följdfrågor vid otydligheter

Vi har genom att ställa följdfrågor och följt upp otydligheter undvikits att framställa insamlad information som kan ha vinklats felaktigt. Detta syftar också på att informationen inte ska ha delats upp för att sättas in i andra sammanhang, utan vi tar all information ur sitt sammanhang och på så vis undviker missförstånd.

3.3 Validitet och reliabilitet

Att kritiskt granska undersökningar är viktigt; genom att granska undersökningar minimeras problem där validitet och reliabilitet kan ifrågasättas. Vi har med detta beaktat validitet och reliabilitet på data vi samlat in till uppsatsen. Enligt Jacobsen (2002) kan validitet delas i intern och extern validitet. Intern validitet avser hur väl undersökningen mäter det som är av intresse att mäta, extern validitet avser istället mäta i vilken utsträckning resultaten från undersökningen kan generaliseras till andra sammanhang.

Genom att vi valt en kvalitativ undersökning har antalet intervjuer varit få, vilket kan ge det en felaktig bild om resultatet generaliseras till ett annat sammanhang. Det är dock inte heller avsikten med en kvalitativ metod, syftet med en kvalitativ undersökning är istället att bibehålla en intern validitet (Jacobsen 2002). Det har vi uppnått genom att välja informanter med noggrannhet, de har goda erfarenheter, olika synpunkter och är oberoende då de inte känner varandra.

3.4 Kritik mot tillvägagångssätt

För att genomföra den kvalitativa undersökningen tillämpades semi-strukturerade intervjuer. Vi utformade vår intervjuguide med öppna frågor och det har gett oss omfattande svar vilka kan gett oss mer arbete att analysera. Vi inser att med mer specifika och inriktade frågor skulle vi få empiri vilket kunde varit enklare att analysera. Vi anser dock att de öppna frågorna har

gett oss information vi annars inte skulle kunna täcka in genom specificerade frågor samt följdfrågor. Vi har genom detta tillvägagångssätt fått en bredare syn på forskningsområdet (Jacobsen 2002). För att säkerställa att vi fått svar på det önskade under intervjuerna har vi haft följdfrågor klara vilka vi ibland använt för att få ut all önskad empiri.

Val av utvecklingsmetoder i teorin har skett utifrån forskningsområdet, till en början samlade vi information om flertalet agila utvecklingsmetoder men efter vidare fördjupning insåg vi att de inte används i större utsträckning i verksamheten. Vi begränsade därför antal metoder i vår teoridel för att endast beröra relevant teori för vidare analys i uppsatsen.

3.5 Språk- och namngivningsval

Utformningen av denna uppsats i form av språk- och namngivningsval har skrivits till följd av ett antal principer, vilka också följts av översättarna till Scrumguiden (Schwaber & Sutherland, 2013). Detta bidrar till att skapa en enhetlig uppsats. Principerna är något modifierade av oss gentemot de som kan återfinnas i Scrumguiden. Våra principer för språk- och namngivningsval i denna uppsats är:

- Språket ska vara väl begripligt och svenskt, detta ska uppnås utan att avvika mer än nödvändigt från de ursprungliga namngivelserna.
- Att översätta samtliga scrumtermer till svenska. Engelska ord som används vid uttryck även på svenska har dock inte översatts, för att ge en följsam och naturlig läsning genom uppsatsen.
- Då Scrum skrivs ihop med annat ord i uppsatsen skrivs det med liten begynnelsebokstav, enligt svensk praxis. Enligt samma praxis används stor begynnelsebokstav då Scrum skrivs ensamt.

3.6 Referenshantering i empirimaterialet

Samtliga sju intervjuer finns i transkriberad form och är styckesindelade, för att enkelt kunna visa i vilken del av respektive intervju som det refererade sades och utav vilken informant. Då vi refererar till en informant i empirin sker det i form av (x.y), där x = Informant och y = Stycke i transkriberingen. Då vi inte refererar till ett specifikt citat eller stycke refererar vi enligt “INFx” (Tabell 4.1, s. 30).

4 Informanternas syn på mjukvaruprojekt

Resultatet av vår empiriska studie presenteras i kapitlet nedan. Sju informanter från olika IT-företag har deltagit i intervjuerna. Transkribering av samtliga intervjuer har genomförts och dessa finns bifogade i Bilagor 7 till 13. Med anledning av transkriberingarnas omfattning och intervjuernas semi-strukturerade natur behandlas inte allt som sagts under intervjuerna inom ramen för uppsatsen. Istället ligger fokus på de aspekter som flera informanter berört samt till viss del aspekter som endast enstaka informanter berört men som de har ansett vara av särskilt stor vikt. Vidare granskning av transkriberingarna rekommenderas för mer djupgående information om de aspekter som endast kortfattat behandlas i uppsatsen.

4.1 Informanterna

Informanternas erfarenheter kring användandet av olika agila utvecklingsmetoder skiljde sig åt; sex av sju informanter har dock mer än 15 års erfarenhet inom IT-branschen vilket medför en god förståelse för när olika arbetsmetoder fungerar och inte fungerar. Vi vill poängtera att studiens omfång endast berör en liten del av alla systemutvecklare och kan därför inte generaliseras i större sammanhang. I Tabell 4.1 nedan återfinns en presenteras av informanterna.

Tabell 4.1: Studiens informanter

Informant	Roll[1]	Erfarenhet[1]	Företag	Antal anställda	ERP/ Övrigt[2]	Egen produkt/ Konsult[3]
INF1	Utvecklare	1-5 år	Infor	12 700 (Infor 2015)	ERP	Egen produkt
INF2	Projektledare	15-20 år	Vattenfall IT[4] på Ringhals[5]	1 650 (Ringhals 2014)	ERP & Övrigt	Egen produkt & Konsult
INF3	Brett IT-ansvar	15-20 år	Fortnox	124 (Fortnox 2015)	Övrigt	Egen produkt
INF4	Brett IT-ansvar	15-20 år	Acando	1 800 (Acando 2015)	ERP & Övrigt	Konsult
INF5	Brett IT-ansvar	25-30 år	EG	1 600 (EG 2015)	ERP	Konsult
INF6	Brett IT-ansvar	15-20 år	Qlik	2 000 (Qlik 2015)	Övrigt	Egen produkt
INF7	Brett IT-ansvar	15-20 år	Tretton37	90 (Tretton37 2015)	Övrigt	Konsult

[1] Generaliserat för informantens anonymitet (se kapitel 3.3.3)
[2] "ERP": Företaget sysslar primärt med ERP-mjukvaror.
"Övrigt": Företaget sysslar primärt med andra sorters mjukvaror än ERP.
[3] "Egen produkt": Företaget levererar och underhåller egenutvecklade mjukvaror.
"Konsult": Företaget agerar leverantör av mjukvaror från andra företag (såsom SAP eller Microsoft Dynamics).
[4] Företagen "Vattenfall" och "Vattenfall IT" har inga kopplingar till Vattenfallsmetoden.
[5] Informanten är i praktiken inhyrd konsult från ett stort IT-konsultföretag, på uppdrag hos Vattenfall IT, som i sin tur levererar IT på Ringhals. Personen har dock uteslutande haft uppdrag hos Vattenfall på plats på Ringhals under flera års tid och intervjun behandlar arbetet på just Ringhals, varför det aktuella konsultföretaget inte nämns.

4.2 Informanternas val av projektmetoder

> "Det finns ingenting som är komplett och som har allt du behöver i varje given situation, utan det handlar om att ha ett set av verktyg och veta när du ska använda dem." (3.90)

Flera informanter ansåg att Scrum eller andra agila metoder i allmänhet är att betrakta som ramverk eller verktyg, snarare än regler att följa under alla omständigheter (3.90; 6.4; 6.10; 7.2). Trots att samtliga informanter har uppgett att de antingen arbetar helt eller delvis agilt följer endast två av sju informanter Scrum i stort sett fullt ut (3.14; 6.4). De andra fem informanterna blandar istället olika metoder – både agila och traditionella (1.26; 2.20; 4.14;

5.16; 7.2). INF7 förklarar valet av att inte ha en specifik arbetsmetod med att det inte skulle fungera i deras verksamhet och att mycket framförallt beror på vad kunden är mogen för. De uppger dock att de arbetar helt agilt och att deras förhållning till agila metoder går bredare och djupare än till en process, vilket innebär att de kan arbeta agilt utan att följa en viss process (7.2). Informanternas företagsinriktning skiljer sig mellan anpassning och implementering av ERP-lösningar samt nyutveckling av programvara (se Tabell 4.1 ovan). Detta har visat sig spela en betydande roll för deras val av metoder.

Tre informanter pekar på att ett traditionellt arbetssätt fungerar bäst då de försöker lösa ett specifikt problem och att en förutsägbar lösning finns, och drar paralleller till att ERP ofta är av den naturen (4.10; 5.58; 6.22). En annan anledning till att traditionella metoder kan fungera bättre än agila metoder vid implementationen av ERP säger INF1 och INF4 vara att ERP har så många beroenden till andra delar av systemet att det är svårt att på ett agilt vis ersätta en del åt gången (1.134; 4.10). INF4 förklarar det genom att en agil metod innebär på att börja väldigt litet och sedan bygga på det. När ett nuvarande system ska bytas ut kan inte en tiondel plockas ut och utvecklas agilt för att sedan sättas tillbaka. I sådana tillfällen när allt behöver ersättas samtidigt måste det ske enligt de traditionella metoderna (4.10). INF6 håller dock inte med om den bilden utan uppger att det förvisso (som nämnts tidigare) kan fungera att jobba mer traditionellt när ett väl avgränsat problem finns och med en förutsägbar lösning, men att det aldrig är optimalt att arbeta helt vattenfallsbaserat (6.30).

En ytterligare anledning till informanternas olika metodval är hur kundens organisation ser ut, vad kunden vill ha och vad de faktiskt behöver (1.24; 2.26; 4.10; 5.8; 6.22; 7.2). Det kan vara så att kunden vill arbeta på ett visst sätt och med en viss förbestämd metod som är vanligt förekommande inom deras organisation. Vissa kunder kan på grund av detta ha svårare för att använda sig av den mest lämpliga metoden för projektet.

INF4 och INF7 argumenterar för att projekt alltid bör drivas agilt då förutsättningarna finns (4.34; 7.2). INF4 motsätter sig dock denna uppfattning och menar att ibland kan det bli tvunget att arbeta traditionellt, till exempel då ett visst system ska ersättas och komplexiteten gör att det rent kostnadsmässigt inte fungerar att ha två system igång parallellt; att budgeten skulle överskridas då krav på integration mellan systemen behövs för att samexistera. Därför väljer INF4 att arbeta med olika metoder som både inkluderar traditionella och agila metoder för olika typer av projekt samt av kunder (4.56). Avslutningsvis belyser INF7 vikten av att vara väl insatt i de metoder som används (7.48); utan ordentlig förståelse för vad som metoderna förespråkar kommer projektet inte lyckas oavsett hur mycket tid och resurser som läggs på projektet. Vidare påstår INF7 att många kunder saknar denna förståelse:

> "Har du inte förstått tankarna bakom ett push-pull system ska du inte hålla på med Kanban. Har du inte förstått syftet med WIP-limits [Work In Progress-limits] ska du inte hålla på med Kanban. Har du inte förstått syftet med interaktion eller den typen av sprintplanering, så ska du inte köra Scrum." (7.18)

INF6 menade att medan framgång i projekt som följer traditionella metoder är att "kryssa i boxar" enligt en tidigare satt plan (vilken egentligen är fylld med hypoteser, snarare än garanterade lösningar) (6.30) så fokuserar det agila på att leverera så mycket värde som möjligt (4.18; 6.30; 7.22). Oavsett om agila eller traditionella metoder används lyfte flera informanter att människorna i projekten – med sina specifika kunskaper och förmågor – anses vara den viktigaste framgångsfaktorn (3.22, 3.40, 3.74-76, 3.80, 3.86; 6.14, 6.18; 7.6). För att skapa värde med det agila arbetssättet är det således viktigt att inte enbart fokusera på att utveckla mjukvaran, utan även att utveckla människorna.

4.3 Visualisering

Två informanter belyser att användningen av agila metoder bidrar till ökad transparens (2.86; 7.2, 7.18) vilket leder till förbättrad visualisering (2.98; 4.14; 7.10). Både Scrum (2.98) och Kanban (4.56) anses kunna bidra till ökad visualisering. INF4 ansåg att Kanban är bäst av de olika agila metoderna på just visualisering och att det hjälper att påvisa vad som faktiskt har genomförts i projektet. Den vanligaste kombinationen av agila metoder hos informanterna visade sig vara Scrum och Kanban (4.32; 6.4-6; 7.10). INF7 tillade att ett moget scrumteam automatiskt även använder en del av Kanban-metoden. Det är inte obligatoriskt att arbeta visuellt med en tavla i Scrum men INF7 menar att mogna scrumteam använder detta för att visualisera projektets framdrift (7.10).

4.4 Användarinvolvering och -acceptans

Det är viktigt att löpande stanna upp och granska hur projektet ligger till gentemot förväntningarna och planerna (6.10; 7.6). För att lyckas väl med detta och få en så sann bild av verkligheten som möjligt krävs god involvering från kunden (2.28, 2.62-64; 4.20; 5.32, 5.36; 7.28, 7.36). Samtliga informanter som belyste kundinvolvering var eniga om att kundinvolvering även är viktigt för att projektteamet ska kunna få löpande input från de blivande användarna, men att det kan vara svårt att lyckas med kundinvolveringen (2.28-2.34, 2.56-64; 4.20; 5.44; 6.22, 6.30; 7.32); särskilt i stora företag (4.20, 4.52, 4.56):

> "Och i vissa företag gillar de inte att [vara delaktiga i projektet] av någon anledning. De tycker det är skönt att fixa kravspecifikationen och sen släppa det och 'fokusera på det jag ska göra så löser ni mitt problem' […] För stora företag spelar det ingen roll ifall systemet kostar 500 miljoner eller 600 miljoner […] för många andra företag är den diffen på 100 miljoner rätt tung." (4.20)

Vidare uppger INF4 att de kan gå in och kräva att kunden avsätter tid för projektet. Kunderna har i vissa fall blivit ställda på grund av att de inte har den tiden som krävs, vilket i så fall skrivits i kontraktet (7.40). Vidare kan det lösas genom att kunden får anställa exempelvis en ekonomiassistent för att frigöra tid.

INF2 berättade om samma problematik med att få kunden och användarna att avsätta tid till projektet och nämnde ett exempel på när det krävts att en person hos kunden officiellt fick avsätta 50 % av sin arbetstid under en viss tidsperiod för delaktighet i projektet (2.60). Enligt INF5 kan brister i användarinvolveringen gå så långt att projekt som tar fram ett system som egentligen är bra och uppfyller alla krav kan bli misslyckade, enbart på grund av att användarna aldrig förstod hur eller varför de skulle använda det (5.44). INF7 lyfte att produktägarrollen är så pass viktig för kundinvolveringen att om ingen hos kunden kan eller är lämpad för den så kan någon hos dem axla rollen (7.16, 7.32-7.38).

Löpande användarinvolvering kan även fungera som ett verktyg för att öka användaracceptansen, genom att hjälpa organisationen att anpassa sig innan det nya systemet implementeras. För detta förändringsarbete är det viktigt med tydlig kommunikation och god utbildning av användarna (4.52-54; 5.8). INF4 belyste att Kanbans styrkor inom visualisering är till hjälp vid denna användarinvolvering (4.56); att visa projektets Kanban-tavla med samtliga uppgifter för användarna kan göra det lättare för dem att få en uppfattning om projektets framfart istället för att peka på avtalsdokumentets olika punkter och vilka som kan bockas av (4.56). Vidare menade INF4 även att involveringen och förändringsarbetet kan minska risken för att introduktionen av systemet kommer som en chock för användarna (4.12, 4.22).

4.5 Förutsägbarhet och estimering

> "Du kan vända på en sten, och där ligger det något riktigt läskigt, och det finns ingen process i världen som kan skydda dig från det." (3.72)

Flera av informanterna lyfte det faktum att mjukvaruprojekten ständigt förändras; exempelvis förändras lösningens utformning och de externa faktorerna som inte inkluderas i projekten men ändå påverkar dem (2.98-102; 3.24-28; 4.10; 6.14, 6.22, 6.28; 7.2, 7.6-8):

> "There's still a lot of hypothesizes in [the backlog/to-do list] and a lot of assumptions in there, which often aren't perceived as hypothesizes or assumptions; they are perceived as the truth. It's a problem that really exists out there and we need to solve it – we need to solve it completely." (6.22)

INF2 har själv erfarenhet från att ha suttit på den mottagande sidan av ett mjukvaruprojekt vid dess leverans och upplevt att "*Oj, det var inte det här vi hade tänkt oss!*", vilket kan motverkas med hjälp av den löpande uppföljningen inom Scrum (2.98). En annan faktor som påverkar förändringarna under projektets gång är dess tidsspann; ju längre ett projekt pågår desto fler saker hinner ändras – framförallt olika externa faktorer (exempelvis vid omorganisationer inom företaget) (4.10). INF3 menar även att ett problem med Vattenfallsmetoden är att den inte tar höjd för förändring och anser att den iterativa utvecklingsmetoden är det bästa sättet att tackla förändringarna på (3.24). INF6 gör gällande att Vattenfallsmetoden är byggd för en värld där ett projekts förutsättningar och planer kan ligga oförändrade i sex månader, vilket inte fungerar i den värld vi lever i (6.14). Informanten menar att denne på grund av detta skulle välja att

arbeta agilt då det hjälper till att skapa mer värde genom ständiga förbättringar under varje iteration istället för att endast utgå från vad som kändes till innan projektet inleddes (6.14).

Flera informanter betraktade IT som komplext och svårt på grund av att det används för att försöka lösa olika komplexa utmaningar som ständigt förändras (3.28-30; 4.22; 5.24; 6.10, 6.14, 6.28, 6.34): *"[IT] is even more difficult than brain surgery. And we say that because it is intangible, what we are doing. You cannot really see it in your hands."* (5.24)

INF3 betraktade att IT-boomen (som inträffade runt år 2000) skapade en uppfattning om att mjukvaruutveckling skulle vara enkelt, vilket det inte är (3.26-32). Vidare ansåg INF3 att *"[...] egentligen så borde man inte starta de allra flesta projekten, och det hade man inte gjort om man hade tänkt till lite till.*" och belyser att ett problem med det agila kan vara att ett ordentligt förarbete aldrig görs (3.32).

INF6 betraktade även att det agila i sig inte är svårt eller dåligt, utan att det är just problemlösning som är det svåra (6.18). Att vissa trots denna komplexitet inte ens följer någon metod eller några givna riktlinjer ser INF5 som paradoxalt (5.24).

Samtliga sju informanter är till olika grad överens om att denna komplexitet och föränderlighet leder till att estimering och förutsägbarhet är svårt i mjukvaruprojekt; vissa menar till och med att det är det allra svåraste (1.52; 2.60-62, 2.98-103; 3.32, 3.72, 3.82; 4.10; 5.22-24; 5.28, 5.34, 5.40; 6.30; 7.20). På frågan om huruvida estimering och att hålla tidsplanen kan vara svårt svarade INF7: *"Alltså, 'hur långt är ett snöre?' Eftersom att kunderna inte vet vad de vill ha... De tror de vet det, men kommer inse om någon vecka att det gör de inte."* (7.20)

INF3 (som arbetar med nyutveckling) anser att varken Scrum eller någon annan metod innehåller någon lösning för utmaningen med estimering, varför de valt att komplettera just den förberedande planerings- och estimeringsfasen med en tydlig definition-of-done (krav som ska vara uppfyllda för att projektet ska anses vara färdigt) (3.64). INF5 (som arbetar med ERP) har istället försökt lösa utmaningen genom att spendera resurser på att anlita managementkonsulter som – redan innan kunden fattat beslut om att anlita dem eller ej – tillsammans med kunden analyserar dennes organisation och olika lösningar. Medan detta kan leda till att en veckas arbete utförs helt gratis så ser INF5 det som den enda lösningen; alternativet skulle vara att delta i projekt utan att veta om projektet går att leverera eller inte (5.30).

INF4 (som främst arbetar med ERP) betonar att det är vanligt att stora och gamla företag väljer att beställa affärssystemsprojekt enligt Vattenfallsmetoden, trots att *"[...] affärssystemsprojekt alltid blir för dyra, tar för lång tid och man når aldrig budget [...]*" (4.10). En av anledningarna till detta sägs vara att de är vana vid att arbeta och beställa efter traditionell metod och anser det för svårt att ändra sig. Även INF6 (som arbetar med nyutveckling och är en stark förespråkare av agila arbetssätt) ser hanteringen och administrationen av stora projekt som en stor utmaning som är viktig att lösa (6.18). Mindre projekt undgår dock inte nödvändigtvis

problematik relaterad till dess storlek; INF7 påstår att Scrum förespråkar mer administration än vad som behövs för små projektteam (7.4).

Flexibiliteten hos agila metoder – att det löpande ges möjlighet att kontrollera arbetet och anpassa det därefter – lyfter flera informanter som en styrka (4.56; 6.10-12; 7.18, 7.40). Enligt INF4 bidrar detta till att det är lätt att komma igång med projekt, att utveckla och sedan kontrollera efter en iteration. Det innebär att om något visats sig vara fel kan det slängas, *"[...] tre veckors jobb var ju jobbigt men det var inte ett års jobb.*" (4.56)

Flexibiliteten behöver med andra ord inte nödvändigtvis hjälpa till vid estimeringsarbetet i början av projektet men kan bidra till en bättre löpande estimering eller "riskhantering" som INF7 uttryckte det (7.18). INF6 anknöt denna flexibilitet till vikten av att utvecklas som person och lära sig mer:

> "We've got to learn faster. Think of it as an organism – it's evolution. The faster you learn, the more chance you have of surviving; the more chance you have of absorbing and responding to different changes, which were either in your control or outside your control." (6.14)

4.6 Kontrakt och överenskommelser

> "Många tror fortfarande idag att om man skriver allt på ett papper, en stor backlog så kommer det vara exakt samma förutsättningar om sex månader. Och det är ju bara att få dem att inse: så funkar det inte. Man kan inte veta allting! Mjukvaruutveckling är inte som att bygga ett hus; det är en hemsk metafor folk använder, att det är som att bygga ett hus, men det är långt ifrån det." (7.2)

Flera av informanterna lyfte att det är mycket viktigt att som leverantör vara överens med kunden om exakt vad som ska utföras (4.18, 4.40; 5.44; 6.30-32; 7.2). Ingen av informanterna har uttryckt att projekt eller kontrakt som bygger på Vattenfallsmetoden är att föredra men (som nämnts tidigare) anser INF4 att det ibland trots allt kan vara nödvändigt om kunden inte är villig att arbeta agilt (4.10). INF7:s företag har istället valt att inte göra några fastprisprojekt alls; med andra ord att inte redan i planerings- och estimeringsfasen fatta beslut om hur slutprodukten ska se ut och vad slutpriset ska bli (7.20). Ett annat sätt att skapa kontrakt som INF7 och INF4 nämner är kontrakt som har ett "målpris" (exempelvis att projektet kommer kosta mellan 1-2 MSEK) (4.46; 7.46). Det förtydligas av INF7 att det är ett kontrakt som inte går ut på att kunna visa exakt det här levererar vi på det här datumet, utan det visar att ändringar kan ske men att vi kommer hamna någonstans mellan givna punkter (7.46). INF4 påpekar dock att om ett sådant kontrakt ska fungera måste antaganden och förutsättningar om projekt och mjukvara bestämmas med kunden i förväg (4.46).

INF7 och INF4 är inte ensamma om att förespråka tillämpningen av agila kontrakt (4.46; 6.32; 7.2, 7.20-22, 7.41-46). Generaliserat bygger dessa kontrakt på partnerskap och att leverantören kontinuerligt träffar kunden och tillsammans bedömer vad som har utförs och planerar vad som

ska göras framöver. Det tillför tillräckligt mycket värde för att ett samarbete ska fortsätta istället för att parterna redan innan arbetet påbörjar beslut om hur slutprodukten ska se ut.

INF6 beskrev hur ett tecknande av ett agilt kontrakt såg ut med sitt nuvarande företag då han/hon först började där som inhyrd konsult:

> “I, as a consultant, said ‘I’ll work in seven two-week iterations [...] I’ll work each iteration almost like it’s agile; we’ll plan together and we’ll sit and work together, we’ll review how things are going, and we’ll do a little retrospective on the things that need to change. We’ll do that seven times [...] and while we’re doing that, we’ll be making results and I’ll be feeding back.” (6.32)

5 Analys och diskussion

I detta kapitel kommer resultaten från vår undersökning att analyseras och ställas mot teorin som presenterats i litteraturgenomgången (kapitel 2).

5.1 Informanternas val av projektmetoder

Gemensamt hos flera informanter var att Scrum beskrivs som ett ramverk. Att Scrum ska benämnas som en metod har även dementerats av Schwaber och Sutherland (kapitel 2.1.1), vilket informanterna således inte talat emot. Vi har funnit att informanterna betraktar det mer som just ett ramverk, en stomme innehållande olika verktyg, snarare än en fast metod som måste följas till fullo för att vara användbar, så som Schwaber och Sutherland (kapitel 2.1.1) även förklarar Scrum. Vidare visar empirin att informanternas åsikter kring agila metoder delvis går isär. Vi har funnit vissa kopplingar mellan de olika åsikterna och vad för sorters programvara det är som utvecklas i informanternas respektive organisationer.

Vår studie visar att användningen av metoder skiljer sig åt på flera olika vis bland informanterna. De tydligaste skillnaderna mellan metodval visas mellan projekt som omfattar utveckling av nya program och moduler, jämfört med projekt som involverar utrullning eller implementering av befintliga produkter, såsom ERP-system. Inom detta fann vi skillnader mellan huruvida intressenterna förespråkar agilt eller traditionellt arbete. Detta påvisar att agil projektmetodik inte alltid anses vara bäst för ett mjukvaruprojekts framgång bland informanterna, vilket kan återkopplas till den befintliga litteraturen om att traditionell utveckling lämpar sig bäst för stora projekt (Stoica et al, kapitel 2.3), vilket en utrullning av ERP-system kan anses vara.

Vidare anser flera informanter och Stoica et al (kapitel 2.3) att projekt som försöker lösa ett avgränsat problem med en förutsägbar lösning är lämpat för Vattenfallsmetoden. Vidare klargör Stoica et al (kapitel 2.3) att ett sådant problem möjliggör en detaljerad plan och inte kräver någon hög kundinvolvering. Att de olika metoderna lämpar sig olika, inte bara på grund av projektets typ utan även beroende på projektets storlek, dess team och kundens begäran är något som vidare har nämnts av en del informanter. INF7 anser att allt för små team kan innebära att de krav agila arbetssätt har på samordning och kommunikation kan anses vara överflödiga och onödiga i sin form, just för att den nödvändiga kommunikationen sker i den dagliga verksamheten. Detta tas även upp av Cho (kapitel 2.1.1) som påstår att sprint-planering, -granskning samt dagliga möten kan bidra till både tids- och arbetsspill. Turk et al (kapitel 2.4.2) uppger dock att just den positiva effekten av samordning och kommunikationen tvärtom kan minska i stora team, och Scrum förespråkar små team (kapitel 2.1.4). Vi föreställer oss att det är viktigt att hålla teamets storlek inom gränsen för antingen för stort eller för litet.

5.2 Visualisering

En fördel med agila metoder i allmänhet och Kanban i synnerhet är visualiseringen, vilket har förklarats djupare av Kniberg och Skarin (kapitel 2.1.3). Två av våra informanter har också nämnt detta, INF4 utvecklar fördelen med god visualisering till att det är hjälpsamt vid användarinvolveringen. En annan fördel INF4 talar om är möjligheten att påvisa hur ett projekt fortgår. Möjlighet till att enkelt visa kunden hur progressen ser ut kan öka acceptansen hos kunden.

Vi bedömer att värdet i god visualisering är stort och i agila metoder kan det kopplas till involveringen samt acceptansen. Att kunden enkelt kan följa projektets gång då de är involverade och lära sig redan innan projektets mjukvara levererats. Detta är även något som sparar tid vid utvecklingstillfället, då kunskapen om mjukvaran som de involverade personerna får i sin tur även kan användas för att utbilda övrig personal. Att visualisering inte belysts av fler informanter kan bero på att både vi och de har fokuserat på utmaningar med agila metoder under intervjuerna och därför inte diskuterat detta mer ingående.

5.3 Användarinvolvering och -acceptans

Sverrisdottir et al (kapitel 2.1.1) anser att en av de viktigare rollerna i Scrum är produktägarrollen, vilken kräver involvering från kundens sida. Stoica et al (kapitel 2.3) menar att även nivån av andra former av användarinvolvering behöver vara hög i samtliga agila projekt. Däremot säger de inget om hur de ser på att den som innehar rollen ska kunna avsätta den tid som krävs för att projektet ska kunna fortskrida vilket har visat sig vara en utmaning för flera informanter.

Boehm och Turner (kapitel 2.4.1) bedömer att utvecklingsteamet måste ligga i fas med användarna för att teamet ska kunna leverera något värdefullt. Samtidigt påpekar Cho (kapitel 2.1.1) att involvering av användare är ett problem i Scrum, då den involverade från kunden har sina egna arbetsuppgifter som löper parallellt med utvecklingsprojektet. Den utmaningen ser även flera av informanterna, att vissa kunder har en tendens att endast fokusera på utvecklingsprojektet när kravspecifikationen skapas, för att sedan återgå till sina ordinarie arbetsuppgifter. Utmaningen med användarinvolveringen är enligt informanterna att kunden behövs under projektet men kan sakna möjlighet att avsätta tiden. Att löpande under hela projektet involvera användare och intressenter kan således hjälpa dem att se vilka förändringar som projektet kommer att ha på deras roller och verksamheten i övrigt, vilket kan öka användarnas förståelse och acceptans för projektet och förändringarna som det medför. En förklaring till utmaningen med involvering kan vara okunskap hos kunderna, att de inte förstår vikten av involvering och därmed inte vill avsätta tid till projektet.

INF4 gör gällande att om inte kunden har tid att avsätta måste det lösas genom att kunden sätter in mer personal som kan assistera för att ge kunden möjlighet att avsätta mer tid, vilket kunden

inte alltid inser. I en situation då involveringen brister från kundens sida, just angående produktägarrollen, uppger INF7 att de själva axlar rollen. Det innebär att projektet fortgår med mindre kundinvolvering än vad Stoica et al (kapitel 2.3) förespråkar för agila mjukvaruprojekt, i vilka författaren hävdar att användarinvolveringen ska vara hög. Boehm och Turner (kapitel 2.4.1) uppger att användarinvolvering ibland kan försvåra projektet när kunder inte kunnat beskriva kraven för projektet ordentligt, på grund av brist på förståelse eller okunnighet. Den bristande förståelsen och kunskapen har också nämnts av INF7.

Vi bedömer att en orsak till utmaningarna med involvering kan vara att kunden inte har fått tillräcklig förståelse för vad som krävs av dem, snarare än att de inte är intresserade av hur projektet går. Vi tror därför att utmaningen ligger i att få kunden att förstå vad som förväntas av dem. Att användarinvolveringen är så pass viktig att företagen kan kräva detta av kunderna anser vi vara fullt rimligt och det framgår i intervjuerna att en del av informanterna redan gör detta. Att göra som INF7 och ibland tillsätta den rollen i form av produktägare internt, vilket motsätter en av de agila metodernas grunder - användarinvolvering, kan ses som en nödlösning ifall kunden inte är tillgänglig. För att ha någon som agerar kund och fungerar som mellanhand mellan kund och leverantör tror vi fortfarande är bättre än att inte ha någon produktägare överhuvudtaget. Detta är något som Wake (2000) tar upp i rollbeskrivningen för XP (se Bilaga 4), där metoden föreslår att en individ från utvecklande organisationen kan agera produktägare. Vi tror dock att lösningen är labil på grund av den markanta begränsning utav användarinvolvering som sker, just för att användarinvolveringen är såpass viktigt för att projektet ska lyckas. Vi tycker att lösningen inte involverar kunden i lika hög grad som det ska göras, vilket kan leda till att det agila i projektet, det vill säga återkopplingarna och förändringarna blir lidande.

5.4 Förutsägbarhet och estimering

Både Armour och Cohn (kapitel 2.4.3) och samtliga informanter anser att förutsägbarhet och estimering är en utmaning i mjukvaruprojekt, vissa ser det till och med som den största utmaningen. De två främsta anledningarna till denna utmaning har vi funnit vara komplexiteten hos mjukvaruprojekt samt att både interna och externa faktorer ständigt förändras under projektens gång.

Några vanliga interna förändringar i projekten är att oförutsedda problem visar sig och att tänkta lösningar inte fungerar som planerat (vilket även kan bero på att problemet i sig hunnit ändras). Några exempel på externa förändringar som lyfts av informanterna är omorganisationer i företaget, uppköp och sammanslagningar med andra företag, och att personer som ingår i projektet (såsom teammedlemmar eller nyckelpersoner i andra delar av organisationen) blir tvungna att plötsligt prioritera andra arbetsuppgifter. Andra externa förändringar som vi finner rimliga är förändringar i projektets tillgängliga resurser (såsom pengar och personal) och förändringar i regelverk, rutiner och lagar.

Flera informanter belyste att komplexiteten hos mjukvaruprojekt i direkt eller indirekt jämförelse med andra sorters projekt i hög utsträckning påverkar förutsägbarheten och således estimeringen. INF5 menar att denna komplexitet i kombination med mjukvaruprojekts ”intangible” (immateriella eller icke greppbara) natur leder till att mjukvaruprojekt är svårare än hjärnkirurgi.

Hur uppfattningarna kring estimering ser ut i andra sorters projekt än just mjukvaruprojekt ligger utanför denna uppsats undersökningsområde men Armour (kapitel 2.4.3) lyfter just att estimering är svårt i alla projekt. Vi finner detta rimligt då vi kan anta att det finns andra projekt än just mjukvaruprojekt som anses vara av komplex och oförutsägbar natur.

Att vissa företag trots komplexiteten hos mjukvaruprojekt inte följer någon slags metod, befintlig eller egen, betraktar INF5 som paradoxalt. Vi instämmer i den kritiken, att ta sig an så svåra utmaningar utan att följa en tydlig struktur bedömer även vi som en onödig risk och vi har svårt att föreställa oss att det kan vara en god idé, utom möjligen när ett väldigt väl avgränsat och begränsat problem ska lösas.

Den oförutsägbarhet som föränderligheten och komplexiteten medför leder till att det är svårt att utföra korrekta estimat; dels för vad som faktiskt kommer behöva utföras och dels för hur mycket resurser som kommer krävas för att utföra det. Våra informanter har valt att försöka hantera utmaningen på olika sätt, dels genom förberedande faser och att påbörja grundarbete redan innan kunden anlitat dem.

I deras svar ser vi en trend i form av att fasta estimat som sätts innan projektet börjar (enligt traditionell metodik) endast bör sättas för projekt som är korta, väl avgränsade, och till en mycket liten grad påverkas av interna och externa faktorer under projektets gång. Vidare bör projekt som kan förväntas komma att ändras och stöta på många problem (vilket i hög grad inkluderar mjukvaruprojekt) inte få något fast estimat, utan bör istället drivas agilt och estimeras först när problemet har utforskats vidare.

Vi tolkar dock inte detta som att det aldrig skulle gå att starta ett projekt om man inte vet det exakta problemet eller vad dess lösning är, utan snarare att det inledningsvis bör undersökas kring problemet och olika tänkbara lösningar noggrant innan estimering sker om vad som kommer krävas för att lösa problemet. Alternativet till detta är att se problemet och tänka sig en lösning, för att därefter estimera ett tillvägagångssätt som baserar estimatet på en tänkt lösning som inte har testats alls. Medan INF3 bedömer det som en nackdel hos det agila, att det aldrig utförs något ordentligt förarbete, så tror vi snarare att förarbetet bör vara en eller ett par agila iterationer. Det för att få en mer realistisk bild av problemen och hur de kan lösas. INF5 har valt att försöka hantera problemet på ungefär det sättet, genom att redan innan ett estimat görs granska kundens organisation och vilka problem de vill få lösta och därefter hur de tror att problemen kan lösas tillsammans med kunden under en kort period.

Medan vi tror att detta tillvägagångssätt kan hjälpa till att skapa mer realistiska estimat finner vi det nödvändigt att kort belysa en tänkbar problematik med det: upphandlingar. Vi tror dock

att det är upphandlingen i sig och inte hur estimatet hanteras som är problemet, då det ligger i en upphandlings natur att ha ett avgränsat och bestämt problem som ska lösas, som tidigare nämnts i denna del samt av informanterna är det vanligt att den initiala problemformuleringen och backloggen med tänkta lösningar inte stämmer överens med verkligheten och/eller att verkligheten till stor del hinner ändras under projektets gång.

5.5 Kontrakt och överenskommelser

Att kontrakt och andra överenskommelser i hög grad påverkas av hur väl estimering om vad som kommer behövas för ett projekt betraktar vi som naturligt. Det kan därför vara en utmaning att skriva kontrakt för agila projekt som vi alltså tror behöver påbörjas innan ett realistiskt estimat kan göras.

Flera informanter belyste att det är mycket viktigt att leverantör och kund är överens om vad som ska göras. För att lyckas med detta förespråkade flera informanter användandet av agila kontrakt. Genom att arbeta med ett agilt kontrakt kan kunder och leverantörer tillsammans se över vad som kommer tillföra mest värde, samt bestämma ett målpris. På så vis får både kund och leverantör ett pris att sikta på, vilket både INF4 och INF7 förespråkar. Cohn (kapitel 2.4.3) förespråkar också att den ursprungliga estimeringen bör övergå till en förbindelse mellan leverantör och kund. Förutsägbarheten och estimeringen för ett agilt projekt är svårt att sätta fingret på och blir därför några av faktorerna som försvårar ett kontrakt med fastpris. Det kan bli tydligare ifall ett fast kontrakt upprättas, men då ökar istället risken att slutprodukten inte blir det som kunden tänkt och risken att budgeten spricker är stor. Ett agilt kontrakt kan låta kunderna bestämma vilken riktning projektet ska gå mot och avgöra när projektet är klart. Medan INF7 har gått så långt att de inte tar sig an fastprisprojekt (med motiveringen att det inte fungerar i deras väldigt agila verksamhet) så tar INF4 sig fortfarande an fasta kontrakt om kunden inte är villig att arbeta agilt trots att denne anser att sådana projekt “alltid” blir försenade eller dyrare än förväntat.

Att upprätta ett kontrakt som täcker alla aspekter anser vi vara en utmaning. Då det råder skiljaktigheter i hur utmaningarna med kontrakt tas an. De agila kontrakten ger mer frihet och värde till kunderna, men inget tydligt pris eller avslut. Medan ett fast kontrakt visar kunden ett bestämt pris och avslut, kan det samtidigt begränsa värdet på leveransen vid projektets slut. Det är en avvägning som måste förtydligas för kunder så att de kan fatta ett så korrekt beslut som möjligt, baserat på deras situation.

6 Slutsats

Forskningsfrågan som denna uppsats baserats på lyder:

"Vilken är den största utmaningen för agila mjukvaruprojekt?"

För att kunna besvara denna frågeställning har vi undersökt olika utmaningar som är vanliga och i hög grad påverkar agila mjukvaruprojekt. Vidare har vi i enlighet med studiens syfte bedömt vilken av de funna utmaningarna som har störst påverkan på projekten och således kan anses vara viktigast att adressera. Medan studiens inriktning är agila metoder har vi inte bortsett från traditionella metoder, då dessa ofta ställs mot de agila samt fortfarande används inom mjukvaruprojekt.

Inom ramen för vår undersökning drar vi slutsatsen att mjukvarans tänkta användare och deras acceptans för den är viktig för projektets framgång. Vi har funnit att det faktum att användarna ofta är upptagna med ordinarie arbetsuppgifter som en utmaning för projektet, vilket behöver deras input för att mjukvaran ska uppfylla deras behov. Vidare har vi även sett en korrelation mellan god användarinvolvering under projektets gång och användarnas acceptans för mjukvaran när den väl implementeras i verksamheten; dels för att möjligheten att mjukvaran uppfyller deras behov kan öka och dels för att de kan hinna anpassa sig till systemets implementering. Med anledning av detta betraktar vi det som viktigt att kunden avsätter den tid för användarna som krävs för att projektet ska få tillräckligt mycket input från dem, samt för utbildning av dem i både nyttan av systemet och hur det används.

Våra resultat visar även att estimering genomgående är en stor utmaning för mjukvaruprojekt. Både den behandlade litteraturen och samtliga informanter är medvetna om detta och delar uppfattningen. Vi har dock funnit att tillämpningen av det agila i sig inte behöver vara anledningen till detta, utan att de två största bidragande orsakerna är komplexiteten samt föränderligheten hos mjukvaruprojekt. Detta antyder att även projekt som drivs efter traditionella metoder står inför samma utmaning. De traditionella metoderna bygger dock på förutsägbarhet, att redan innan projektet startar känna till vad problemet egentligen är och hur det ska lösas. Detta har lett oss till uppfattningen att traditionell metodik är lämpad för projekt som ska lösa väl avgränsade problem med en befintlig lösning, och inte komplexa problem utan kända lösningar, eftersom projektplanen läggs innan projektet påbörjas.

Att komplexiteten och föränderligheten är de största bidragande faktorerna till utmaningen med estimering leder oss till uppfattningen att projekt som är tänkta att lösa komplexa problem utan kända lösningar och äger rum i en föränderlig miljö, pågår under lång tid och/eller projekt som är stora och omfattande inte bör estimeras fullt ut innan projektet påbörjats och problemen och lämpliga lösningar undersökts vidare. Intressant är att detta står i kontrast till vår litteraturstudie, som påvisade att stora projekt bör bedrivas efter traditionella metoder och estimeras innan de påbörjas. Vi fann även att vissa informanter följer litteraturens rekommendation, men att det snarare beror på att kunden ber om fastpris än att de själva

föredrar det. Detta antyder att hur kunderna beställer projekt i hög grad kan påverka estimeringens överensstämmelse med verkligheten och därmed hur väl projektet lyckas hålla sig inom budgeterad tid och kostnad samt uppnå satta mål.

Denna påverkan på beställande som estimering verkar ha leder även till utmaningar med kontrakt och överenskommelser. Vissa av våra informanter förespråkar agila kontrakt som ett sätt att hantera utmaningen. Agila kontrakt fokuserar på att parterna löpande undersöker om projektet fortskrider som planerat och tillför tillräckligt mycket värde till kunden för att det ska vara värt att fortsätta, istället för att ett fast pris och fasta mål sätts innan projektet påbörjas.

Eftersom estimeringen av projekt direkt eller indirekt påverkar hela projektet samt har visat sig vara svårt betraktar vi detta som den största utmaningen för agila mjukvaruprojekt. Vidare har vi funnit att denna utmaning inte är begränsad till just agila mjukvaruprojekt, utan att den finns även för mjukvaruprojekt som drivs efter traditionell metodik. Vi tror även att andra sorters projekt som är tänkta att lösa komplexa problem utan kända lösningar kan stå inför samma utmaning.

6.1 Sammanställning av resultat

- De tre största utmaningarna som agila mjukvaruprojekt ställs inför är (ordnade störst till minst):
  1. Estimering
  2. Användarinvolvering och -acceptans
  3. Kontrakt och överenskommelser

- Traditionell metodik är lämpad för projekt som är tänkta att lösa väl avgränsade problem med kända lösningar.

- Agil metodik är lämpad för projekt som är tänkta att lösa komplexa problem utan kända lösningar och som bedrivs i föränderliga miljöer.

7 Bilagor

Bilaga 1: Det Agila Manifestet

1. Vår högsta prioritet är att tillfredsställa kunden genom tidig och kontinuerlig leverans av värdefull programvara.
2. Välkomna förändrande krav, även sent under utvecklingen. Agila metoder utnyttjar förändring till kundens konkurrensfördel.
3. Leverera fungerande programvara ofta, med ett par månaders mellanrum, ju oftare desto bättre.
4. Verksamhetskunniga och utvecklare måste arbeta tillsammans dagligen under hela projektet.
5. Bygg projektet kring motiverade individer. Ge dem den miljö och det stöd de behöver och lita på att de får jobbet gjort.
6. Kommunikation ansikte mot ansikte är det bästa och effektivaste sättet att förmedla information, både till och inom utvecklingsteamet.
7. Fungerade programvara är främsta måttet på framsteg.
8. Agila metoder verkar för uthållighet. Sponsorer, utvecklare och användare ska kunna hålla jämn utvecklingstakt under obegränsad tid.
9. Kontinuerlig uppmärksamhet på förstklassig teknik och bra design stärker anpassningsförmågan.
10. Enkelhet – konsten att maximera mängden arbete som inte görs – är grundläggande.
11. Bäst arkitektur, krav och design växer fram med självorganiserande team.
12. Med jämna mellanrum reflekterar teamet över hur det kan bli mer effektivt och justerar sitt beteende därefter.

(Beck et al, 2001)

Bilaga 2: XP:s fem värderingar

1. Kommunikation ger viktig återkoppling då kunden som finns på plats löpande ger feedback på systemet.
2. Enkelhet nås via kontinuerlig omstrukturering av kod samt minimering av dokument som inte är en del av det slutliga systemet.
3. Mod när något behöver ändras som inte är speciellt populärt.
4. Respekt för medarbetare och deras kod. Undvik att göra ändringar i medarbetares kod utan att först genomfört ordentligt med testning.
5. Återkoppling skapas genom många delleveranser och löpande enhetstester.

(Beck & Andres, 2004)

Bilaga 3: XP:s 13 praktiker

1. **Sit together**: Att hela teamet utvecklar på samma plats.
2. **Whole team**: Utnyttja ett tvärfunktionellt team av alla de som är nödvändiga för att produkten ska lyckas.
3. **Informativ workplace**: Placera synliga grafer runt arbetsytan så att teammedlemmar och andra intressenter kan se få en uppfattning om hur projektet går.
4. **Energized work**: XP-team fungerar inte med övertid under långa perioder. Motiveringen bakom denna praxis är att hålla hög kvalitet på koden.
5. **Pair Programming**: All producerad kod är skriven av två programmerare vid samma maskin.
6. **Stories**: Teamet ska skriva korta berättelser för funktionalitet som önskas i produkten, detta ska vara synligt för kunden. Utvecklarna uppskattar historien; kunden prioriterar berättelsen.
7. **Weekly cycle**: I början av varje vecka hålls ett möte för att granska framstegen. Kunden får välja historier som ska genomföras den veckan och för att bryta ner berättelserna till små uppgifter som ska göras i veckan. I slutet av veckan ska acceptanstester för de valda berättelserna vara igång så att kunden kan köra igång nästa veckas cykel.
8. **Quarterly cycle**: Hela teamet bör sätta ett eller flera teman till berättelser för en fjärdedel av berättelserna. Teman hjälper teamet att reflektera över de i större bild.
9. **Slack**: Planera in några lägre prioriterade uppgifter i varje iteration, som kan läggas om ifall laget skulle bli efter tidsplanen. På så sätt kommer kunden fortfarande få deras viktigaste funktioner levererade.
10. **Ten-minute build**: Strukturera projektet och dess tillhörande tester så att hela systemet kan byggas och alla tester kan köras i tio minuter. Då kan systemet byggas och testas oftare.
11. **Test-first programming**: Alla berättelser har minst ett acceptanstest, vilket helst ska vara automatiserat. När testet för en användares berättelse godkänns, ses dessa som klara.
12. **Continuous Integration**: Integrera och bygg systemet dagligen så fort en ny uppgift slutförs.
13. **Incremental design**: Snarare än att utveckla en djupt detaljerad utformning före genomförandet, investera i utformningen av systemet varje dag. Att förbättra utformningen av tidigare skriven kod är väsentlig. Team med robusta enhetstester kan med säkerhet experimentera med refaktorisering eftersom ett skyddsnät är på plats.

(Beck & Andres, 2004)

Bilaga 4: XP:s sex roller

1. **Manager**: Äger teamet och dess problem. Manager har hand om teamet, resurser, hantering av människor och problem samt gränssnitt med externa grupper.
2. **Coach**: Lär teammedlemmarna om XP-processen. Ingriper vid problem och ser över om processen följs. Coachen är vanligtvis en programmerare och inte en chef.
3. **Tracker**: Är en programmerare som regelbundet samlar in användarberättelser och acceptanstest från utvecklare. Samt skapar de synliga graferna.
4. **Programmerare**: Skriver tester, design och kod. Programmeraren har i uppgift att identifiera och uppskatta uppgifter och berättelser (den här personen kan också vara en testare).
5. **Testare**: Hjälper kunderna att utveckla tester (den här personen kan också vara en programmerare).
6. **Kund**: Skriver berättelser och acceptanstester, plockar berättelser för en release och iteration. En vanlig missuppfattning är att kunden måste spelas av en person från kundorganisationen. En grupp av kunder vara inblandade eller en representant kan väljas inifrån utvecklingsorganisationen (men i så fall en som är utanför utvecklingsteamet).

(Wake 2000)

Bilaga 5: Intervjuguide

Informanten

- Titel och nuvarande roll/er?
- Berätta lite om din karriär!
    - Hur lång erfarenhet har du från IT-branschen?

Metoder

- Vilken metod använder ni er av?
- Hur länge har ni använt er av metoden?
- Hur mycket erfarenhet har du av att arbeta med respektive metod?
- Varför har ni valt att använda den metoden?
    - Om det inte är Scrum eller om den inte ens är agil: Varför inte?
    - Om det är en egen metod: Kan du beskriva metoden för oss?
        - Om den innehåller delar av populära metoder:
            - Vilka och varför?
            - Vad gjorde att ni inte valde de andra delarna också?
- Guida oss igenom ett projekt som använder/har använt metoden (Planering, Genomförande, Leverans, Uppföljning)
    - Roller
    - Processer/faser
    - Aktiviteter/möten
    - Regler
    - Dokumentation
- Har ni funderingar på att övergå till en annan utvecklingsmetod?
- Vilka styrkor ser du med användningen av metoden gentemot [agil/vattenfall, beroende på vad de använder nu]?
- Vilka svagheter och problem ser du med användningen av metoden gentemot [agil/vattenfall, beroende på vad de använder nu]?
- I vilka projekt passar metoden bäst/sämst? (stora/små, utveckling/utrullning, nyutveckling/förbättring, många/få personer)

Breda frågor

- Varför tror du att så många IT-projekt misslyckas med att hålla sig inom tidsramen och/eller budgeten?
    - Hur ofta har du själv stött på de problemen?
        - Hur har ni lyckats lösa/undvika de problemen?
            - Hur väl bemöter metoden ni använder de problemen?

Bilaga 6: Intervjuförberedande mejl

Hej [intervjupersonens namn],

Stort tack för att vi får möjligheten att intervjua er om mjukvaruutvecklingsmetoder för vår C-uppsats.

Inför intervjun vill vi gärna få svar på några korta frågor, för att kunna förbereda oss:

- Går det bra att vi spelar in intervjun och inkluderar den i transkriberad form i vår uppsats? *
- Vilken eller vilka utvecklingsmetoder använder ni för närvarande?
    - Om ni använder flera metoder: Vilken av dem är er primära?
- Använde ni någon annan metod innan?
    - Om ni gjorde det: När bytte ni till den nuvarande?

Under intervjun kommer vi fokusera på följande områden:

- Din roll i företaget och dina tidigare erfarenheter från mjukvaruutvecklingsprojekt.
- En genomgång av den/de utvecklingsmetod/er ni använder (exempelvis styrkor/svagheter, hur tillämpningen av metoden ser ut från start till slut under ett projekt, och dess roller).
- Din syn på olika aspekter hos IT-projekt, såsom vanliga problem och hur du anser att man kan lösa dem.

Tveka inte att fråga om något är oklart; annars hörs/ses vi [avtalad tid].

Med vänlig hälsning
[uppsatsförfattarens namn]

** Ni kommer inte nämnas vid namn utan endast ert företagsnamn kommer användas, men vi kan göra undantag från detta om ni så önskar. Ni kommer även få möjlighet att godkänna transkriberingen innan den läggs in i uppsatsen.*

Bilaga 7: Transkribering #1 – Infor

1.1	**Väldigt schyst att vi fick prata lite med dig.**
1.2	Ja det är ingen fara, klart jag ställer upp! Jag har ju själv varit i den sitsen, så det är klart.
1.3	**Ja, okej, härligt. Då tänkte vi börja här med lite allmänna frågor som vi har. Vi funderar lite på vad för position det är som du har i företaget?**
1.4	Jag är konsult inom Infor Consulting Services, och jobbar främst med utveckling av modifieringar för Infor M3.
1.5	**Okej, så du jobba bara med det egna - såklart - det egna affärssystemet som ni har?**
1.6	Ja, precis.
1.7	**Är det någon speciell del där som du ansvarar för, eller är det utveckling av hela affärssystemet?**
1.8	Nej, alltså i projekt så brukar det vara vissa delar som man är med i i så fall.
1.9	**Ja, okej.**
1.10	Exempelvis, om man håller på och kanske - där jag jobbar väldigt tätt med affärssystemskonsulterna, som jobbar mer standard med business-grejerna, jobbar de exempelvis med logistik, och så kanske det blir att jag är med och jobbar med logistikproblem eller så vidare som vi inte kan lösa i standard, så hjälper jag till med extra funktionalitet eller något som de vill ändra på och så vidare.
1.11	**Okej.**
1.12	Eller att lägga till, som de känner att de har behov av, som inte finns i standard då.
1.13	**Kan man fråga vad du har för tidigare erfarenheter av detta? Lärde du dig allting när du började på Infor eller…?**
1.14	Nja, vi hade ju en del programmering i skolan, hade vi ju, men det är klart att sen så höll jag på lite på fritiden, och sen så har jag ju lärt mig större delen här.
1.15	**Ja, okej. Hur länge har du jobbat på Infor nu?**
1.16	I [1-5 år].
1.17	**Ja, okej då. Men vad bra, då har vi ju lite bakgrundsinformation där då.**
1.18	Japp.
1.19	**Vi skriver ju en uppsats; ja du vet kanske… Vi skickade ju lite information där [om] vad vi skriver om.**
1.20	Ja, jag har ju egentligen rubriken på mailet där då.
1.21	**Ja, precis. Så vi är lite i uppstartsfasen här och vi tänkte ju skriva om lite olika mjukvaruutvecklingsmetoder.**
1.22	Ja, okej.

1.23	**Och jag funderar ju då först på vilka mjukvaruutvecklingsmetoder ni använder på Infor?**
1.24	Ja, vi använder ju - vad ska man säga - vi… I och med att det är ett så pass stort när vi implementerar så är det ju inte just - eller vad ska man säga… Utvecklingen eller så, utan vi ska ju försöka få ihop hela, sätta in ett helt nytt affärssystem kanske hos en kund, och då är det ju liksom väldigt stort och komplicerat, ofta. Så då har de ju haft, så att säga, egna eller, om de har tagit fram en best practice eller så vidare, men innan har vi haft en som heter StepWise.
1.25	**Har ni haft det innan sa du?**
1.26	Ja, men nu har vi en som heter Infor Deployment Method, som är liksom framtagen förmodligen från något eget företag skulle jag anta, som liksom har kommit fram, så vi använder väl en egen projektmodell kan man säga.
1.27	**Okej. När var det ni övergick till denna metoden då?**
1.28	Den senaste, Infor Deployment Method, det var… Den kom in förra året.
1.29	**Så du var med lite när det övergicks?**
1.30	Ja, precis.
1.31	**Okej, intressant. Men då vet vi ju ungefär hur länge ni har använt den nya metoden. Men vet du varför ni gick över till denna nya metoden? Var det några större problem eller var det några stora faktorer som gjorde att ni var tvungna att gå över?**
1.32	Tyvärr vet jag inte, men jag kan tänka mig att man… Infor Deployment; men jag antar att, jag känner mer att det kanske blir mer… Vi har ju inte en produkt heller, utan vi har ju flera produkter, inom själva den… Så jag antar att i Infor Deployment Method står väl att den här ska användas över hela bolaget, förstår jag det som. Att den är mer, liksom, global inom bolaget, än kanske den andra som hette StepWise, så har haft en… Även innan - det började ju som Intentia, det här egentligen, och då hade de en annan projektmodell också, som de hade liksom, Implex tror jag den hette, så grunden kommer kanske från länge, så de har väl byggt på den länge antar jag.
1.33	**Okej, där ser man. Men kan du... vet du hur övergången gick? Var det enkelt, eller var det någonting du upplevde, att det var problematiskt att gå över till en annan metod?**
1.34	Nej, det har jag inte upplevt, någon problematik ännu.
1.35	**Nej, okej.**
1.36	Och de här två skiljer sig väl inte jättemycket antagligen, den ena bygger väl på den andra såklart, så har den förbättrats förmodligen.

1.37	**Okej. Är det så att den här metoden ni använder nu, använder ni den under alla projekt som ni då har?**
1.38	Nej, saken är ju då att exempelvis som, just nu är jag på ett projekt där kunden vill ha en annan modell. Alltså, som kunden vill använda. Och där jobbar jag faktiskt för första gången med… Alltså vår Deployment Method är ingen agil metod egentligen. Men nu jobbar jag i ett projekt där de vill köra projektet i en agil metodik då - Scrum.
1.39	**Varför vill de göra det, istället för att köra på er metod?**
1.40	Nej vad jag förstår så vill de, liksom, eftersom vår metod inte är agil så vill de se mer av det här med working software, som är ett mer agilt framtagande.
1.41	**Ja, okej. Ser du några fördelar med att det är agilt? Eller varför har inte ni en agil metod?**
1.42	Det var en bra fråga! Vad jag ser är väl liksom… Vad jag tycker om agilt är ju att endast förändringen är så pass stor i en agil utvecklingsmetod, att man liksom från en vecka till en annan kan ha ändrat så mycket, och som jag ser det när man arbetar med ett affärssystem är att flera delar måste hålla ihop hela tiden. I ett agilt projekt; vad ska man säga… Man ska utveckla en ny app till en telefon, säger vi. Då är det ju den produkten man ska ta fram. Men i ett affärssystem har man så många delar. Man har ju liksom lager/ekonomi, man har produktion och ”det” och ”det”, och allting måste ju liksom komma samman och mötas på samma ställe. Har man inte satt upp det ena så fungerar ju inte det andra, i ett affärssystem, oftast.
1.43	**Nej, okej. Men den metoden ni hade innan ni övergick till eran egenutvecklade, var det en agil metod?**
1.44	Nej, ingen av dem har varit det. Den har väl varit mer en sån, vad heter det, en Vattenfalls-modell egentligen, fast inte riktigt en Vattenfall-modell heller, men mer i stuk åt det hållet i så fall.
1.45	**Ja, okej. Då tänkte jag, har ni dokumenterat ner riktlinjer för den metoden ni använder nu?**
1.46	Ja, där finns ordentliga riktlinjer.
1.47	**Okej, följer du denna dokumentationen - kollar du på denna dokumentationen - varje dag för att veta exakt vad som ska göras?**
1.48	Nej, det gör jag inte. Vi har ju även projektledare och så vidare, och det är ju egentligen de som jobbar mest med den, och följer liksom upp projektet med den egentligen.
1.49	**Ja, okej, men då är det inte så att... men du har tillgång till dokumentet?**
1.50	Ja, det har jag.

1.51	**Ja, okej. Men då tänkte vi att du har sån insyn att du skulle kunna ge oss - eller kunna guida oss igenom - ett projekt där det är mjukvaruutveckling. Alltså hur går det till inför, under och vid leverans; om det är möjligt?**
1.52	Ja, då blir det väl alltså så som jag jobbar. Jag har, som sagt… Oftast har vi en lösningsarkitekt, en lösningsansvarig. Och vi har en affärssystemskonsult, och så jag som jobbar kanske som programmerare eller teknisk systemerare. Då jobbar väl de mer med kunden och tar fram en teknisk specifikation, på vad [som] behöver göras och så vidare. Och sen så börjar han skriva på den, och så kanske jag är med på några möten och säger att "Ja, det här går väl att göra och det här är svårare att göra; kan man tänka på något annat sätt?" och så vidare. De kanske inte har mer av den här biten som jag då tänker på, "hur ser det ut bakom egentligen; kan man göra allt det här?", i och med att man, i ett affärssystem… Jag utvecklar ju med den grunden som vi har i vårt affärssystem, och där är man ju begränsad också ibland, tyvärr. Och sen så börjar jag väl utveckla då, efter den här specen som han har gett mig då; hans - vad ska man säga - krav, och skapar fram det. Sen gör jag mina tester och tycker att det fungerar, och sen testar affärssystemskonsulten, och tycker inte han att det är som han vill så kan jag få tillbaka det, och så kan det vara så. Och sen så kan ju ha olika miljöer då, i ett projekt. Vi har kanske en development-miljö, där vi jobbar mycket, och så har vi ha en test, och så kan man ha en verifikation, och sen en produktion; ibland så har man ännu fler. Men det är ganska vanligt att man har en sån, och vi brukar ju ge - när affärssystemskonsulten är redo så skickar han iväg den till test, och då testar kunden där, om de är nöjda. Skulle de inte vara det så går vi tillbaka igen och så fortsätter vi i development då.
1.53	**Okej.**
1.54	Och det vi oftast brukar göra, det svåraste är ofta att få fram, "vad kostar en sån här lösning?", utöver att…
1.55	**Men är det ett pris som ni kan sätta efter att ni har fått ihop en kravlista då eller?**
1.56	Precis, så den sätter vi ju egentligen, att jag räknar ju på det som affärskonsulten har tagit fram egentligen. Och då kommer ju det här in med det agila, vilket jag tycker - när man jobbar agilt - att det är svårt och - har jag uppmärksammat, och jag har ju inte jobbat jättemycket med det - men jag uppfattar att det är svårt att räkna, då man ju inte har något slut oftast. Det innebär i agilt - vad jag fattar det som - att man ska arbeta fram något som kallas Working Software ofta, och det är svårt att sätta ett stopp; för då kan man komma med förändringar, att "detta vill vi göra och detta vill vi göra" och alla förändringar är väldigt dyra ofta; det påverkar mer än vad man tror. De här små grejerna blir oftast större.

1.57	**Är det alltid möjligt att lägga in förändringar under projektets gång?**
1.58	Ja det är det ju, men oftast sätter vi ju ett pris på den specifikation som vi har skrivit; "det här gäller", och så sätter vi ett pris på en modifiering exempelvis.
1.59	**Var det så att ni… Hur sa du att det va?**
1.60	Alltså, om vi har då en specifikation som de har skrivit, då skriver vi ju ett pris på den specifikationen och ger till kunden. Och då så får de ju skriva "ja" eller "nej" eller så, så att säga. Den är ju bestämd innan då.
1.61	**Men om de sen vill lägga till nya grejer till det, eller ändra saker…?**
1.62	Ja då brukar det ju komma som en Addition eller som ett tillägg, att så kan det vara.
1.63	**Men hur funkar det med den metoden som ni har; är det så att ni arbetar med delmål och levererar - vad ska man säga… - delmål, eller är det så att ni levererar allt på en och samma gång?**
1.64	Nej det är, det är delmål. Så det är ingen sån här Big Bang, utan oftast så är det ett delmål.
1.65	**Okej. Sen funderar jag lite på det här med aktiviteter och så; är det inplanerat, till exempel med uppföljning, att ni har möten - dagliga möten - hur funkar det med det; under projektets gång då?**
1.66	Just nu - i och med att jag jobbar med… Detta är ett av de större fallen där jag har kommit in och det har varit mer agilt, eftersom kunden vill ha det, då är det ju varannan vecka. Men i och med att den nya där, har jag inte riktigt jobbat med ett agil-projekt tidigare, så kan jag inte riktigt säga där faktiskt, men jag tror i alla fall att det är månadsvis - om inte mer. Och där är även - som du säger - för varje del, som jag har sett det som, på den här har du en del som ska göras, och där i varje del så har du en massa grejer som är uppradade som ska göras. Att den är väldigt strukturerad, vår modell.
1.67	**Så det är på det sättet ni också, som ni följer upp att ni håller tid då; att ni måste klara av varje delmål?**
1.68	Ja, precis.
1.69	**Att ni vet att annars blir ni försenade med projektet?**
1.70	Ja, det stämmer.
1.71	**Så att ni håller det lite flexibelt, slutdatum?**
1.72	[Instämmer.]
1.73	**Du berättade ju där, om lite olika roller och så. Vet du mer vilka roller - alltså vilka bestämda roller - som finns i ett sånt här projekt?**
1.74	Ja vi har ju, projektledare har vi ju. Och vi har lösningsarkitekter. Vi brukar ha en teknisk projektledare också, som är lite mer på installationer och teknik och sådär. Sen har vi ju konsulter…

1.75	**Okej, interna konsulter eller…?**
1.76	Ja, vi kör ju på det först, men vi har även partners och så vidare, men det brukar ju vara Infor-konsulter främst.
1.77	**Sen tänkte jag på… Är det alltid så att ni har samma roller då; det är inte så att olika projekt kan ha olika roller?**
1.78	Det skulle de väl egentligen säkerligen kunna ha. Det är efter behov också, men i största del så brukar det i större projekt ingå sådana roller som jag nämnde innan.
1.79	**Men du kör ofta samma roll i dina projekt, eller?**
1.80	Ja, det gör jag ju; som sagt, det gör jag.
1.81	**Hur långa brukar projekten vara i genomsnitt, ungefär?**
1.82	Jag kan ju ha allt från en modifiering på bara några dagar egentligen, och upp till - nu har jag ju suttit på ett projekt i över ett halvår.
1.83	**Ja det är klart att det kan variera väldigt mycket där.**
1.84	Ja, precis, nu är jag ju med i en ny kund där, så att… Ja, precis, så det blir ju nog nästan ett år i alla fall, minst.
1.85	**Okej. Hur skulle du säga - med den metoden som ni har använt - är det lätt att leverera på tid, eller brukar det dra över på tid, de projekten ni har?**
1.86	Den frågan kan jag inte riktigt svara på, faktiskt.
1.87	**Okej, [men i] de som du vet att du har varit med i?**
1.88	Nej men det har fungerat.
1.89	**Jag tänkte – det fanns ju en kravlista, den berättade du ju om - men sen finns det… Har ni någon checklista då som det är projektledaren kanske som ansvarar för att checka av att allting sker?**
1.90	Ja, det skulle jag säga att det är.
1.91	**Okej, så ni har daglig kontakt med projektledaren, eller vet du hur det funkar; hur ofta måste ni rapportera till projektledaren?**
1.92	Jag pratar med projektledaren kanske… Det är lite olika, i och med att jag jobbar nu i ett agilt så har vi ju en Scrum Master och det är lite annorlunda. Men en projektledare - även i det här projektet - pratar jag med kanske en gång i månaden i alla fall.
1.93	**Men vet du, har projektledaren i vanliga fall följer upp vad som görs? Är det endast att han tar kontakt med alla som jobbar inom projektet, eller är det så att han måste följa upp och verkligen se framgången?**
1.94	Nja, han pratar nog; och även… Exempelvis lösningsarkitekten följer väl upp ganska mycket och säger liksom vad som… De har väl - kan jag tänka mig - ännu större kontakt där och följer upp med varandra.

1.95	**Men den metoden som ni har nu, är det mycket kommunikation där emellan de olika rollerna, skulle du säga; alltså, sitter ni alla på Infors kontor eller är det vissa som är ute hos kunden och projektledaren sitter på sitt kontor; eller…?**
1.96	Det är nog väldigt olika, men min roll är oftast på kontor; man kan ju jobba så pass bra remote idag, så jag sitter ju mycket på kontoret. Och det sparar ju mycket reskostnader och så vidare; men det är klart att man behövs på plats ibland, och då är man ju ute. Men det är klart att affärssystemskonsulten är väl mer ute hos kunden och liksom diskuterar lösningen och så vidare och kanske demar [demonstrerar] och så vidare.
1.97	**Då ska vi se…**
1.98	Så försöker väl egentligen hålla den, att jag som utvecklare då ska inte försöka ha direktkontakt med kunden, om det är möjligt att vi har en…
1.99	**Vad sa du nu; "att vi ska inte försöka ha en…"?**
1.100	Nej vi jobbar väl lite så att affärssystemskonsulten har lite mer tight med kunden, så att affärssystemskonsulten blir en mellanhand mellan mig och kunden egentligen; eller att de tar fram en lösning och det är jag som utför den. Men ibland sitter väl jag också med, men det är väl så vi försöker hålla det lite.
1.101	**Ja, okej.**
1.102	Och det innebär väl också att - i och med att vi har en standard som vi jobbar mycket efter, så kan vi ju - eller jag kan ju - inte ner på detaljnivå på exakt varje flöde och sådär, utan då lite mer övergripande bild tar ju de då fram.
1.103	**Efter att ni har levererat, är det så att ni har kvalitetskontroll då, utefter någon mall - någon standardmall - så att ni vet att ni har levererat det ni ska?**
1.104	Ja vi har ju acceptenstester, som kunden gör, och liksom verkligen att… "Såhär är det tänkt att fungera" och de kör verkligen stora tester och så vidare, innan vi har gått live, som sagt. Som sagt; i och med att vi har så himla många olika miljöer - som vi kör grejerna i - så kvalitetssäkrar även kunden - samt vi - så att det fungerar när vi flyttar. Så ingenting ska egentligen komma in i produktion [förrän] det är riktigt testat.
1.105	**Okej; men fortsätter ni testa även när det har kommit in i produktion?**
1.106	Alltså då hör ju kunden ganska fort av sig om det inte fungerar.
1.107	**Är det något problem som du har stött på någon gång; att det har blivit problem när ni har levererat?**
1.108	Nej, det har jag inte varit med om - än så länge - så jag hoppas inte att det blir så.
1.109	**Efter ni har levererat, brukar ni ha att ni följer upp er själva också; att ni har självkritik, så att ni kan förbättra era projekt i framtiden?**

1.110	Ja, det har projektledaren och sådär, och vi får även - på oss konsulter - som projektledaren fyller i.
1.111	**Vad menar du nu?**
1.112	Alltså en utvärdering på oss själva; projektledare utvärderar vårt arbete, och så vidare.
1.113	**Okej, hur gör ni det? Sitter ni och pratar med varandra då?**
1.114	Nej vi har ett formulär som kommer utifrån den nya metoden, med färdiga frågor och sådär.
1.115	**Och det samlar projektledaren ihop sen då, och utvärderar?**
1.116	Ja, precis, det så skickar han ut det till…
1.117	**Så du känner att det mesta, eller allting fungerar bra med den metod ni använder nu?**
1.118	Ja, jag ser inga [problem] just nu. Men som sagt, jag har inte kört så mycket på den, i och med att det blev en annan metodik på den här senaste; jag har varit där i 17 månader nu. Men det tycker jag, att den verkar fungera.
1.119	**Är det så att ni bara jobbar… Är det bara konsulter från Sverige som ni jobbar med, eller blir det att ni jobbar med konsulter från andra länder också?**
1.120	Ja, det gör vi också.
1.121	**Hur funkar det då med att man kan ha olika språk och olika kulturer i den här metoden; är det några större problem? För i vissa metoder där det sker väldigt mycket samarbete kan det bli mer problem om man pratar olika språk.**
1.122	Okej, nej det har jag inte uppmärksammat - i alla fall inte nu. Vi har off-shore också, och där jobbar väl jag ganska tight ibland med vissa från andra länder, men då blir det väl egentligen… Eller jag blir - inte ansvarig för dem - men när det kommer tillbaks så testar jag först och levererar vidare till affärskonsulten och så vidare - eller - den som har beställt det.
1.123	**Nu kom vi ju in lite mer på allmänna frågor, som du kanske uppfattade det. Du visste ju lite om vad Scrum var - du är ju lite insatt i det.**
1.124	Ja det har jag ju fått lite nu, i och med att jag jobbar i ett projekt som använder Scrum. Och jag är väl inte proffs på det, men i det som jag jobbar med så tycker jag att jag har en del koll.
1.125	**Upplever du några större skillnader mellan Scrum och det som ni har på Infor?**
1.126	Ja, det gör jag. Jag tycker att det är lite mer… Det är mer planering.
1.127	**I Scrum, eller?**
1.128	Ja, precis. I och med att man… Varannan vecka så har vi - varje vecka har vi uppföljning - men varannan vecka så spelar man med poäng, gör vi. Poäng är inte tid eller så specifikt, utan det är komplexiteten egentligen, över en viss grej; man har så kallade User Stories. Nu skriver ju inte jag dem så ofta - i och med att de andra gör det - men det tar ju en hel del tid

	också. Jag menar, om vi då egentligen har hela vår spec - som är vårt då - så vill de bryta ner specen i olika user stories; att "skapa den här Tabellen" eller "skapa den är funktionaliteten", så spelar man då om den i en grupp; och det tar ju lite tid ibland då, att få ihop alla de här.
1.129	**Så vad skulle du föredra då: Scrum eller er egna metod, om du fick välja vad du skulle arbeta med?**
1.130	Jag har svårt att säga… Alltså i just det som jag gör så tycker jag att det agila fungerar, i och med att vi har "den här modden [modifikationen], den ska tas fram"; att "vi ska ta och bygga den här logistikgrejen", kan vi säga - en funktionalitet som de vill ha - och då ska vi ju få fram den. Men för att få testa den här grejen så behöver vi ju jättemycket data, jättemycket ordrar, jättemycket artiklar - eller jättemycket såna grejer - och då är vi ju kanske begränsade av andra team. Och i och med att den är så pass agil så har man kanske inte riktigt den långsiktiga planen, utan man har det här med Working Software - alltså att "testa det här i två veckor", men så är man inte färdig med det, utan "okej", eller "lägg till det här och det här och det här"; det tycker jag, att den här förändringen kan bli svår, i och med att, vad ska man säga… Att implementera ett affärssystem är ju ganska komplicerat i sig, och det gäller ju att alla delar hänger med varandra, och det tycker jag är lite svårare då man inte har den långsiktiga planen alltid [i Scrum].
1.131	**Jag förstår. Men skulle du säga att det är värt den extra tiden som det tar med att använda Scrum, som metod?**
1.132	Hur menar du? Att applicera den på hela vårt implementeringsprojekt; att byta ut den och göra den mer agil?
1.133	**Ja i det projektet som du är inne i nu; alltså att istället för att använda den egna metoden, att ni… Att det skulle vara värt att använda Scrum istället för den egna metoden, i och med att den tar mer tid och att den innebär mer planering.**
1.134	Jag har lite svårt att se att [man kan] implementera ett helt affärssystem i Scrum, på det sättet. Det har jag. I alla fall kostnadsmässigt. Det känns - i alla fall för mig - känns det som att Scrum-projekt har lätt att bli dyrare, tack vare den stora… För man vill ju enligt Scrum ha en ganska enkel förändring; att man ska kunna ändra på saker som man har bestämt, ganska så intensivt och enkelt. Man är rädd för de här stora leveranserna, att "nu är det färdigt - pang, här är det". Att man har Working Software, att man får se vad det är som händer varje vecka i princip, eller varannan vecka; att man ska dema [demonstrera] och så vidare.
1.135	**Jamen det låter faktiskt bra! Vi har ju fått lite information om vad det finns för begränsningar, och du har ju berättat lite vad du ser för fördelar och så med era**

	metoder också. Det känns egentligen som att jag har fått svar på mina frågor. Jag ska bara kolla med mina kollegor här, om det är något jag har missat.
1.136	Om det är några frågor som du känner att du inte har fått svar på, så kan du ju egentligen maila mig sen, och i så fall så kan jag ju vidarebefordra dem till en av våra projektledare här; om det är något fråga där du känner att "här fick vi nog inga svar som vi skulle vilja ha haft", så kan jag ju maila dem vidare och se om du kan få ett bättre svar från en projektledare - om ni vill det.
1.137	**Det hade varit jättebra, absolut! Men vi känner verkligen att det har varit givande med den här intervjun, men absolut - om vi vet att det dyker upp så kan vi skicka dem.**
1.138	Ja självklart, gör det, så ska jag se vad jag kan göra. I och med att jag inte jobbar som projektledare så är ju vissa frågor väldigt svåra att svara på.
1.139	**Jomen det förstår vi också.**
1.140	Ni har ju fått… Min roll efter [tid på företaget] så har jag ju jobbat i [1-2 år] på ett större [projekt], och det har ju inte gått live ännu, utan vi håller fortfarande på och det är säkert ett år kvar; eller kanske inte ett år, men åtta månader kvar innan det ska gå live, så att det är väl den bilden jag har fått. Och som sagt, är det något annat så maila över de och så kan jag maila dem vidare och trycka på att ni får något bra svar där.
1.141	**Okej, då gör vi så. Tack så jättemycket för tiden!**
1.142	Lycka till, ska jag säga också!

Bilaga 8: Transkribering #2 – Ringhals

2.1	**Så, tänkte börja lite kort om dig, vad har du för titel och för roll i företaget?**
2.2	Jag är konsult, jobbar på [ett konsultföretag] och jag jobbar som Projektledare och Business Analyst, och har varit konsult på Ringhals i [runt fem] år. Jag har jobbat som konsult på flera andra bolag innan dess, bland annat på [storkommun], [telekomföretag], [medicinteknikföretag], [fordonstillverkare].
2.3	**Okej, men började du på [nuvarande konsultföretag] i samma veva som du fick uppdraget på Ringhals?**
2.4	Nej det gjorde jag inte, jag [jag har jobbat] på [nuvarande konsultföretag] i [runt tio] år, så de sista [runt fem] åren jag har varit på Ringhals.
2.5	**Du skulle säga att du har [runt tio] år i branschen då?**
2.6	Nej, jag började [i slutet av 1990-talet].
2.7	**Hur började du karriären?**
2.8	Jag började som konsult på [telekomföretag]. Jag läste civilingenjör på [universitet].
2.9	**Var det som utvecklare då eller [på telekomföretaget]?**
2.10	Jag började som utvecklare men [i slutet av 1990-talet] började webben bli väldigt stort, mycket webbprojekt, och det var ju rätt nytt. Så det fanns inte riktigt samma struktur i vad man skulle jobba med i projekt som det kanske hade gjort i tidigare projekt, man tog många olika roller i projekten och inte bara som utvecklare utan jobbade i flera olika roller. Och tanken var att det skulle gå lättare och smidigare jämfört med de systemen som systemutvecklingsprojekten man hade tidigare
2.11	**Vilka projekt är du delaktig i just nu på Ringhals?**
2.12	Nu är jag delaktig i tre projekt: [1] Dels så är det ett fortsättningsprojekt ett större projekt som började under förra året då vi gjorde en lista i december och gör en uppföljning, därefter hanteringssystem på Ringhals. [2] Sen jobbar jag i slutfasen på Ringhals, där i ett business intelligence-projekt, där vi ska sätta upp en QlikView-applikation för att analysera data från det här systemet. [3] Sen jobbar jag med… det är mer en utredning/förstudie där vi undersöker olika projektverktyg som man använder på Ringhals, och försöker göra en beskrivning av nuläget. Även med strategier framåt.
2.13	**Okej, var det flera projekt, som jag fattade det?**
2.14	Ja precis.
2.15	**Jobbar ni likadant i alla projekt, eller använder ni mycket olika metoder?**

2.16	Det är lite olika typer av projekt, den ena är en ren förstudie och då är den egentligen inte den intressant för er, utan ni är mer intresserade av ute i projekt där det ingår någon form av utveckling?
2.17	**Ja, precis.**
2.18	Men vi använder ungefär samma kan man väl säga.
2.19	**Hur skulle du beskriva det som?**
2.20	Nämen det… Vi jobbar ju… Det är inte Scrum rakt av, men det är agilt tänk, där vi delar upp själva utvecklingen som skall göras i ett antal iterationer, inför varje iteration görs en prioritering om vad som skall ingå i denna iteration, och det görs kontinuerliga avstämningar i verksamheten och med demonstrationer och är väldigt delaktiga i framtagningen av systemen.
2.21	**Okej, har den något speciellt namn?**
2.22	Nej det har den inte, det är få som kör Scrum rakt av. Men det är en form av ”Scrum-ish”: man tar de delar man tycker fungerar bra, väljer det som passar bra, man gör en egen uppsättning och anpassningen efter de förutsättningar man har.
2.23	**Har ni använt den metoden ända sen du började där?**
2.24	Ja, jag har ju jobbat med en hel del förstudier och där använder man kanske inte den metoden på det sättet, men inom utvecklingsprojekten jag drivit inom Ringhals då har jag använt mig av den metoden.
2.25	**Vet du hur länge den funnits på Ringhals?**
2.26	Nej, det vet jag inte. Ringhals är rätt styrt i vilka dokument som måste tas fram i vilka faser, det gäller att hitta en nivå som fungerar med att man inte dokumenterar för mycket, det är där agila metoderna går ut på, att man gör allt eftersom man jobbar i projektet; att man inte lägger för mycket tid i början att dokumentera. Det gäller att hitta sätt att göra det på, när man har en så pass styrd projektmodell man har på Ringhals. Och det är lite också upp till de olika beställarna, och den beställaren jag jobbat med nu vill att vi jobbar på det här sättet.
2.27	**Okej, skulle du kunna guida oss igenom ett projekt? Det skulle kunna vara ett pågående eller tidigare projekt. Just planeringen, genomförandet, leveransen, uppföljningen, hur det ser ut. Om man börjar då med planeringsfasen, hur ser den ut med den metoden ni använder?**
2.28	Ja, jag funderar på vilket projekt jag skall ta, det beror ju lite på vilken typ av projekt… Om man tar projektet jag drev förra året, där vi skulle införa ett nytt system som skulle ersätta ett befintligt, så var det speciellt på grund av den anledningen att vi skulle ut på marknaden och välja ett system, och då får man jobba med en kravdel för att sätta upp kraven som man gjort i sin förfrågan.

	Det gjorde vi, och när vi hade valt system och leverantör så kommer en ny kravdel där man jobbar mycket ihop med leverantören, för att gå igenom det som ska ställas på projektet – inte på systemet, utan på projektet. Och där försökte vi hålla på en rätt hög nivå för att vi ville jobba… Vi delade upp projektet i fem-sex olika iterationer, där vi inför varje iteration förstod vi att kommer lära oss under projektets gång; vi vill inte göra för mycket jobb i början, utan vill ha möjlighet att göra förbättringar när vi kan gå genom produkten och vet mer om vilka möjligheter som finns. Men vi var ändå tvungna att sätta grundkraven, och en grundarkitektur för hur vi tänkt bygga upp systemet. Sen kommer vi in i själva utvecklingsfasen eller implementeringsfasen och där hade vi delat upp leveranserna i sex delleveranser där vi jobbade med… Dels hade vi en roll som kallas för funktionsansvarig [FA] på Ringhals och det är en person från verksamheten som är ansvarig för att hålla ihop kraven från verksamheten och det kan man jämföra med product owner som det kallas i Scrum, det är typ product owner men inte riktigt samma mandat att fatta beslut. Högst upp inte som en product owner. Sen jobbade vi med fem referensgrupper beroende på vilken iteration det var, olika delar av systemet med olika intressenter, där jobbade vi mycket med att träffade referensgruppen och gjorde demonstrationer allteftersom kom fler system, titta på hur det såg ut. Sen gjorde vi inte… om man tänker Scrum, är det att varje iteration ska leverera en körbar produkt i slutändan men innehåller många brister, vi testade också men vi involverade inte, vi gjorde inte acceptanstesterna utan det samlade vi upp till slutet av projektet, så gjorde vi acceptanstest utav hela systemen.
2.29	**Vilka är delarna som accepteras då?**
2.30	Det var FA:n som har väldigt stor del i acceptanstesterna, så håller vi med folk från olika referensgrupper som har olika ansvarsområden som är med och testar.
2.31	**Hur kommer det sig att ni inte tog in mer folk under resans gång, såsom Scrum förespråkar väldigt mycket?**
2.32	Det handlade helt enkelt om tidsbrist, vi hade så många intressenter till det här systemet så det var svårt att hinna med att göra, de hade möjlighet att gå in och testa i systemet själva och se hur det funkade, men vi hade inte möjlighet att göra några mer organiserade tester. Och det man märker här är har man inte organiserade tester är det väldigt svårt att få folk att testa, det vi fick göra i slutet var att vi fick boka upp dem och att ”nu sitter vi tillsammans och testar”.

2.33	**Men hur involverad är FA:n under resans gång innan man kommer till sista steget då alla kommer in och testar?**
2.34	Väldigt involverad i alla stegen, [FA:n] var inne och testade allteftersom, det var [FA:n], det var test av verksamheten som man hade kunnat dra nytta av. Det är svårt att man sitter med sina roller och svårt att få tiden att räcka till; ett sådant fall var jag involverad i; så gott som 100 % i projektet det är oftast det som gör dom här agila metoderna kan vara svåra, att någon som kan ha tid att delta så aktivt i ett projekt som det krävs.
2.35	**Skulle du säga att metoden ser likadan ut i just samma typ av projekt, eller varierar de olika delarna? Vad ska man säga… Det ni följer i metoden, varierar det mycket efter storleken på projektet, till exempel**?
2.36	Det är vissa grunder, grunddelar som vi kör oavsett storlek på projekt. Nu har vi fortsättningsprojekt, och det är betydligt mindre men vi kör ju samma tänk, vi har ju ett antal iterationer, vi gör prioriteringar, vi har FA är med men inte på heltid. Och olika intressenter. Skillnader nu är att vi har ett system som är driftat och körs ute i verksamheten vilket gör att varje iteration avslutar vi med test och gör faktiskt en produktionstestning också. Det blir mer begränsat, de förbättringar som finns, det är lättare att göra driftsättningar och tester. Det gick inte i vårt förra projekt som skulle ersätta ett system, då fick det komma en viss bit på vägen innan systemet var så färdigt att det kunde ersätta det befintliga.
2.37	**Men om man går tillbaka på det projektet, i planeringsfasen; hur ser den ut under genomförandefasen med iterationer osv?**
2.38	Hmm, du menar den som jag drev förra året?
2.39	**Jag tänkte på det när du började och beskrev planeringsfasen för oss, om man fortsätter titta vidare i processen där, hur väl täcker metoden in fasen och vilka olika delar finns det där? [Med] ”genomförande” tänker jag mer på utvecklingsfasen, innan ni tagit in slutanvändare i slutfasen och testar.**
2.40	Första iterationen hade vi en iteration [som gick ut på] att göra en grunduppsättning; vi har ett system som grund att stå på, sen hade vi involvering utav användaren i referensgruppen för iteration två kan man säga. Då hade vi i förväg gått igenom med dem, grundkraven för hur vi skulle sätta upp deras delar. Det är olika typer av ärenden, de olika referensgrupperna sätter man upp olika typer av processtöd för, jag hade gått igenom innan iterationen startade med referensgruppen, vi såg deras behov av kring systemservice process. Därefter gör man prioritering av ”vad är det vi ska prioritera och sätta upp och hur många timmar har vi tillgängliga och vad får vi med i iterationen?”. Och gör demonstrationer för

	referensgrupperna, ”det var så här ni sa att ni ville ha, då blir det såhär i systemet”. Och så får dom en chans att tycka till, det var så dom menade.
2.41	**Hur lång var den första iterationen?**
2.42	Fyra veckor.
2.43	**Gäller det alltid?**
2.44	Ja, fyra veckors iteration.
2.45	**Gäller det alla projekt?**
2.46	Nej, det beror lite på. Jag kör det lite som en tumregel, jag tycker fyra veckor är okej. I det projektet vi har nu får det anpassas lite efter när det är möjligt att göra driftsättningar så det blir lite längre iterationer där, på fem veckor ungefär.
2.47	**Hur jobbade ni med exempelvis kommunikation och uppföljning under varje iteration, vem pratade med vem så att säga rent officiellt om man bortser från att folk pratar med varandra under projektet, vilka rapportvägar finns det?**
2.48	I mitt projekt kör vi checkmöten, 4 gånger i veckan. Under det stora projektet jobbar vi på halvtid, 2 gånger i veckan. Sen har vi lilla projektet där vi har lite längre projektmöten varannan vecka och så har vi prioriteringsmöten och annat. Kommunikationsvägar så, projektledare kommunicerar med sponsor och beställaren, rapporter varannan vecka, sen finns det en styrgrupp där man har [möten] en gång i månaden. Och på [mötena] så fattar sponsorn beslut om till exempel placering utav tollgates med stöd av styrgruppen, detta är även ett beslut som inte kan den funktionsansvariga kan fatta. Styrgruppen behöver fatta vissa typer av beslut.
2.49	**Kan du ge något ett exempel på vilken typ av beslut dom tar?**
2.50	Ja till exempel då Ringhals har som krav att rapportera vissa händelser till katastrofmyndigheten där vi har interna system att stödja denna rapportering, då hade man, det finns fyra olika block på Ringhals blocken har olika sätt för hur dom satte dom här rapporterar myndigheterna och då var vi tvungna att ena det till att alla fyra blocken körde gemensamt om vi skulle kunna tillåta olika varianter i systemet, då tog vi fram förslag för hur man skulle kunna göra och det gjorde vi inom referensgruppen, men för att fatta beslut för att de är så här vi gör, har man förberett betygsunderlag, sen fattar dom beslutet, så här blir det. Styrgruppen som har fyra gånger per vecka, sen har ju referensgrupperna de som jag har kommunikation med och referensgruppernas uppgift är att vara projektets förlängda arm ut i verksamheten också. Och förenkla och utreda vissa frågor som tas upp på dessa referensgruppsmöten. Sen fanns det ett parallellt projekt för förändringsledning, som jobbade mycket med själva förändringsleden, att skapa acceptans ute i verksamheten, det

	kan vara utbildningar, informationsmöten där man går ut med information om att det är nytt system. Prata på olika ledningsgrupper. Mer vad som händer.
2.51	**Hur många iterationer blev det i det projektet, hur länge höll ni på?**
2.52	Det blev sex iterationer. Nej sju blev det, vi fick en extra.
2.53	**Och sen leveransen, hur såg det ut?**
2.54	Eee, ja, hur menar du?
2.55	**När det väl skulle implementeras, exempelvis med utbildning för användarna, vem hade hand om det och hur såg det ut?**
2.56	Ja, utbildning av användarna var för [projektnamn]-projektet som hade ansvar för det och vi körde ett antal eller dem körde ett antal informationsmöten som vem helst som projektverksamheten kunde anmäla sig. Och det gjorde dom tillsammans med FA. Sen var det ett antal mer riktade utbildningar, mot vissa delar till exempel dom som jobbade med [---] dem jag pratade om innan rapporterar dem [---] var det riktade utbildningar eftersom deras processer va speciellt, och där la vi upp det tillsammans med dom som är ansvariga för verksamheten, själva processen i en gemensam utbildning när det gäller IT-stöd och processer. Sen va det en stor fördel med referensgrupperna, där ingick ju dom människorna som jobbade mycket med dom olika delarna, de hade redan fått en utbildning i och med att dom varit där under iterationerna och även acceptanstesterna. Och det fungerade också som en form av nyckelanvändare, eller expertanvändare i verksamheten. Och där hade vi också med referensgrupperna. Det finns vissa stödfunktioner på Ringhals och dom representerade referensgrupperna för att dom visste väl hur verktygen fungerade. Sen hade vi även med funktionsansvarig som sagt, han var väldigt väl insatt i hur det fungerar och hade en viktig roll i systemförvaltning. Sen har vi… På yttersidan finns en systemförvaltning också, där var projektledaren med under hela projektet. Jag känner att dom delarna gick väldigt bra, just att dom var med under projektet, det är en utmaning när man sitter i ett projekt och det sen ska över till förvaltning och så vet inte förvaltningen alls vad dom får.
2.57	**Men hur kommer det sig att det ofta inte fungerar så bra, tror du?**
2.58	Ofta vet jag inte, men…
2.59	**De gånger det inte lyckas så bra?**

2.60	Ja men det är att dom som jobbar i förvaltningen har ju ett dagligt jobb, i sin förvaltarroll och det är inte säkert det är lätt att kombinera med att ingå i ett projekt. De fallen då detta inte har fungerar så är det nog just det det handlar om när man tar in förvaltningsresurser så är det väldigt svårt att, det kan vara svårt att planera i ett projekt, för att förvaltningsärenden går alltid före projekteffektiviteten. Blir det för mycket förvaltningsärenden drar projekten ut på tiden. I det fallet fick jag en förvaltningsresurs som hade 50 % över som kunde lägga i projektet.
2.61	**Upplever du det som ett problem allmänt, att det är svårt att – så att säga – ta den tiden från även referensgrupper, folk som har sitt dagliga jobb och behöver vara med i projektet ändå.**
2.62	Ja, ja, det gör det absolut, ”så här mycket kommer jag behöva personerna”. Men det är ju så att de har sina arbeten; är det något som blir kris där [så] är det det som går först, det kan man ju förstå. Är det något problem i blocken [kärnreaktorområdena] så går ju det före projektet. Sen samtidigt är det svårt med planering och få dem resurserna som vi behöver, och det fick jag lyfta vid ett något tillfälle där jag hade problem med att få dem resurserna som vi hade efterfrågat. Men det blir ju en utmaning att få resurser. Även om man får dem man vetat innan i veckan, så gäller det också att resurserna måste vara självgående till viss del. Kan inte styra dom exakt vad dem ska göra, de har ett ansvarsområde, och där upplevde jag att det kan vara svårt att få dem att ta ansvar för sitt eget arbete. Där man innan i princip var tvungen att… Men nu har vi bokat upp och alla sitter tillsammans och testar. Och det var det som krävdes för att dem faktiskt skulle gå in och göra det.
2.63	**Upplever du samma problem, när det gäller utbildning gentemot användarna, just med att det kan vara svårt att - föreställer jag mig - att få dom att gå igenom utbildningen som dom inte måste, eller som det inte satts någon tid för?**
2.64	Nej, vi hade dom här informationsmötena som sagt, det var stor vikt att delta på dom och då märker man lagt upp en plan då det låg informationsmöten i god tid innan driftsättningen och de var inte så välbesökta. Men efter driftsättningen var de väldigt välbesökta. Det är vanligt att man skjuter allting tills det blir nödvändigt. Sen finns det utbildningar i systemet, och det finns sen tidigare, vi körde inga speciella inför driftsättningen.
2.65	**Var detta ett system som de flesta användarna använde väldigt mycket eller ganska lite, eller hur stor spridning, hur mycket dom använder i dagligt arbete?**
2.66	Det var väldigt stor spridning, det är ungefär 800 unika användare i systemet. Det gäller in och rapportera avvikelser, händelser, myndighetsärenden. En del går in och rapporterar någon rapportering per år, en del är aldrig inne, det finns dom som jobbar dagligen i

	systemet, så det är väldigt stor spridning på användargruppen. Vilket styrker utmaningen eftersom man har olika behov av att använda.
2.67	**Hur ser uppföljningen ut efteråt? Har ni hand om den också till viss del; inledningsvis och sen förvaltningen, eller hur såg det ut? Just att när systemet väl börjat användas?**
2.68	Vi startade ett postprojekt [efterprojekt]; det tycker jag är jättebra, att ha ett postprojekt, det är först när systemet så att säga testas och användas en period som man börjar se vad som funkar och vad som inte funkar. Och det är en väldig fördel att kunna fortsätta att göra den typen av uppföljning. Den enkla förändringen, så att det var postprojektet som vi tog hand om som gör uppföljning i referensgruppen, dels följde vi upp att användaren använder systemet som tänkt. Det gjorde business case för projektet. Och det ingick att vi skulle avveckla ett antal subsystem, som dom använde tidigare. Det tog mycket tid och planering. Uppföljning, men förändra hur man jobbar, beter sig är en av de stora utmaningarna.
2.69	**Hur hanterar man det bäst ser du?**
2.70	Det vi har gjort, det sa innan, att de är involverade i själva hopsättningen så att de ser att det här är målet och hur vi gör för att komma dit, business caset, sen behövs det uppföljning även efteråt. Utbildning och information, som vi kom överens om. Att där var det bra för där märktes det att det behövdes.
2.71	**Vi måste ändå fråga, hur kommer det sig att ni inte kör exempelvis ren Scrum eller rent något annat utan att ni kör er egen metod?**
2.72	Vi kör det som jag tycker funkar bäst. Jag försöker… Första gången jag körde Scrum var när jag jobbade som konsult på [telekomföretag] [i slutet av 00-talet], där körde vi Scrum. Och det var rätt nytt då skulle jag säga, och jag har egentligen tagit med mig dom delarna jag tycker funkar, med mig till Ringhals, utefter de förutsättningarna som finns på Ringhals.
2.73	**Om du kategoriserar det lite, vad skulle du säga att du tycker… Vi kan börja med: vad tycker du inte funkar så bra med Scrum som du valt att exkludera i din egen metod, så att säga?**
2.74	Det behöver inte vara att det inte funkar bra, det kan också vara det är svårt att hinna med och att prioritera bort det, det här som vi tyvärr prioriterade bort som vi inte gjorde i den omfattning som man ska göra i Scrum, det är… Vi gjorde det innan, varje iteration ska ju avslutas med en erfarenhet, en tillbakablick, och i det gjorde vi samband med att vi prioriterade nästa iteration. Men man kanske skulle gjort den ännu mer noggrant. Men det var ren tidsbrist, inte att vi inte tyckte den fungerade bra. På [telekomföretag] så gjorde vi en sådan breakdown, vad heter det… Den kurvan man ser tiden går neråt. Men du vet vad jag menar?
2.75	**Ja.**

2.76	Vid varje tillfälle, vid varje möte så, så genom att, när aktivitet startas uppskattar man den till 20 timmar, men nu anser vi att det är bara 14 timmar kvar så gör man en graf för att visa hur tiden går ner. Den körde vi inte heller.
2.77	**Om man tittar på rollerna; Scrum har ganska tydligt vilka roller som skall finnas och FA:n har till exempel är en verksamhetsbegränsning; det betyder att den rollen behöver finnas. Generellt, är det motsvarande roller i den metoden ni använt mot Scrum, eller är det stora skillnader?**
2.78	Vi har inte haft en ren Scrum master. Vår FA i så fall tycker jag påminner om product owner, skillnaden då [att FA] inte har samma mandat att fatta beslut i alla frågor, inte stort mandat men inom alla områden.
2.79	**I Scrum har man ofta flera kanske små team, har ni jobbat på det viset eller haft ett stort team?**
2.80	Det har inte varit så stora projekt att vi haft flera små team, utan vi har kört ett team.
2.81	**Hur många var ni involverade i det projektet?**
2.82	Det var tre utvecklare: en person som jobbade mer med användarvänlighetsdelarna, sen va det två personer som jobbade med förändringsdelen kan man säga, sen va det FA:n, sen va det jag.
2.83	**I Scrum spelar man för poäng och sådana grejer exempelvis, gjorde ni något liknande där?**
2.84	Nej, det gjorde vi inte, det gjorde vi… När jag var på [telekomföretag] gjorde vi det.
2.85	**Om man vänder på kakan då, och säger dom delar du tagit med dig som du tycker är väldigt bra med Scrum, vad säger du då?**
2.86	Det är ju prioriteringen, att man kan jobba med prioritering inför varje iteration, stand up-möten. Uppdelning är egentligen prioritering och uppdelning, involveringen från verksamheten som är delaktig.
2.87	**[Med] ”uppdelning”; menar du roller eller tid?**
2.88	Tid, menar jag. Det kan man säga är en skillnad, att Scrum också… Vi har ju våra roller i projektet, Scrum är ju mer att man ska ta den rollen, att alla ska kunna ta rollerna.
2.89	**Skulle du säga att det har med att göra att ni var så pass få. Om du föreställer dig att ni varit tre gånger så många, hade ni genomfört det likadant då tror du?**
2.90	Du menar att vi hade… Ja, då hade vi fått jobba i olika team. Jag tycker den storleken på team vi hade, är ju max tycker jag.
2.91	**Scrum i sig förespråkar, man brukar säga sju personer plus/minus två-ett, upplever du att det är för många att vara 7, 8, 9 stycken i ett team alltså?**

2.92	Ja, det tror jag nog att jag hade tyckt det varit. Så många vi var var hyfsat lagom. 5, 6, 7, 8 var vi. Någon jobbade deltid hela tiden. Tyckte det var lagom storlek hela tiden.
2.93	**Ser du några projekt-former/typer där du ser att den här metoden inte skulle funka? Vi tänker oss exempelvis stora/små projekt, utveckling/utrullning, nyutveckling/förbättring, många/få personer inblandade; vem tror du metoden faktiskt inte skulle fungera för, så som ni använt den nu?**
2.94	Jag vet inte, jag vet sådana som kört Scrum… Vi hade ett gemensamt projektrum vilket har fungerat väldigt bra, vi satt tillsammans; inte alla men de som var inom utveckling satt i samma rum som mig och övriga. Jag vet att andra har sånt projekt när man inte sitter tillsammans, och fått det till att fungera, så det går ju uppenbarligen. Även om jag tycker det är en väldig fördelat att man kan sitta tillsammans. Man ser varandra, att man har den här kommunikationen, likt i Scrum face-to-face, att man ser varandra. Jag vet att det funkat i projekt där man inte haft det, jag tycker det är väldigt skönt att se människan. Det tycker jag är en mycket bra del. Det är alltid en utmaning när man jobbar med virtuella som inte är vana att jobba med den metoden, man får ju, att kunna ticka av att jag fått det jag beställde, där man känner osäkerhet att inte veta exakt hur leveransen blir. Det bygger på ett förtroende mellan beställaren och leverantören i det här fallet.
2.95	**Storleksmässigt, hur ser du på Scrum och den egna metoden, vilken som egentligen, just att… Om vi börjar med Scrum till exempel, ser du det som svårt att om det är väldigt små eller stora projekt?**
2.96	Nu har vi gjort mindre projekt, och det har fungerat väldigt bra med Scrum så att i mindre projekt tycker jag definitivt att det fungerar väldigt bra. Större projekt där man har flera olika team det vet jag faktiskt inte, har för dålig erfarenhet för att uttala mig om.
2.97	**Vi tänkte avsluta lite här, med breda frågor i allmänhet.** **Har du något mer du gärna skulle vilja ha sagt om hur ni jobbade; någon fråga du förväntade dig att få som du inte har fått lov att svara på?**
2.98	Nej, inte vad jag kan komma på, det är väl det här med fördelar tycker jag det är väldigt svårt att jobba med krav, och att det är svårt att förstå om man pratar om slutresultatet, där tycker jag Scrum ger en väldigt bra bild, att man får lära sig under resans gång, och att man får nya kunskaper allt eftersom, även att omvärlden kan förändras och att man har olika metoder att jobba med visualisering utav krav. Det kan ju vara att man jobbar med, i varje projekt olika prototyper, och även att man gör konFigurationer av systemet, att man gör typisk stödhjälp och går in och tittar på det. För jag har ju jobbat i projekt där man inte har gjort så och där jag suttit med på verksamhetssidan och man fått någonting som utvecklarna

	har utvecklat. Så ser man det att "Oj, det var inte det här vi hade tänkt oss!" då, det tycker jag är den stora fördelen, att man faktiskt kan se och förstå. Vad slutresultatet blir.
2.99	**Japp, men då går vi vidare till sista steget, en fråga vi undra över är varför tror du att så pass många IT-projekt generellt misslyckas antingen med att hålla budget eller tiden, det finns olika siffror på det men generellt snackas det om mellan 70 % och 90 % lyckas antingen inte vara i tid eller hålla budgeten som är satt. Vad tror du är största anledningen?**
2.100	Jag är ju… Ett exempel – där jag satt på verksamhetssidan, och inte känner igen kraven man får – att man missuppfattat kraven. Det är lätt hänt om man inte kommer tillbaka och frågar, och sånt kan man undvika med Scrum, får man något i systemet färdigt och går det inte att använda så går det ju iväg då får man börja lägg till saker och ändra, så att det faktiskt blir något man kan använda. Och sen är det ju beror ju på vad man menar med att 70 % av alla projekt går över budget, är det initiala tiden och budgeten man pratar om. Det är vanligt att man jobbar med change request under hela projektet men dyker upp nya behov
2.101	**Det är det första som satt inledningsvis vi menar.**
2.102	Det är ju, man vet ju sällan i början, det är alltid en uppskattning hur lång tid saker kommer ta att göra, det är många okända parametrar. Sen tillkommer det behov, saker och ting ändars som måste fixas, där är Scrums fördelar, att om man har en tidsplan och budget handlar det om att prioritera inom den typen av budgeten, då har man större chans att lyckas, men då kan sånt som man tänkte skulle vara med från början inte komma med och det kan ju en del tycka är jobbigt.

Bilaga 9: Transkribering #3 – Fortnox

3.1	**Vi vill gärna börja lite kort om dig bara, med vilken titel och roll du har i företaget.**
3.2	[Brett IT-ansvar]
3.3	**Hur länge har du varit hos Fortnox?**
3.4	I [1-5] år.
3.5	**Vad gjorde du innan dess?**
3.6	Innan dess var jag […] utvecklingskonsult.
3.7	**Hur länge har du jobbat med utveckling, allt som allt?**
3.8	I [15-20] år.
3.9	**Men du har haft samma roll hos Fortnox sedan du började, eller har du bytt roll?**
3.10	Nej jag har haft samma roll sedan jag började.
3.11	**Vi kör på med metoderna och hur ni jobbar på Fortnox, och även lite om hur du har jobbat tidigare. Kan du beskriva metoden ni använder nu?**
3.12	Ja, när man pratar om metoden så är det ju intressant för… Ska vi prata om Scrum, eller ska vi prata om alternativet till Scrum?
3.13	**Vi vill gärna prata om den metod som ni använder primärt.**
3.14	Ja just det. För egentligen har vi ju en process eller en metod som är en produktutvecklingsprocessmetod, och den spänner över allting från produktinitiativen till release och förvaltning av en leverans. Inne i den finns det en implementationsfas där vi har monterat stora delar av Scrum. Men ska man ha en strikt… Hela idén – eller en av de stora delarna – med Scrum är ju att den ska vara iterativ och inkrementell, och att man i slutet av varje iteration ska ha ett fungerande inkrement. Och det funkar ju – och så gör vi ju – men vi har lagt till en integrations… Eftersom vi driver flera parallella projekt i Scrum, i parallella iterationer, så har vi ett gemensamt integrations- och systemtest i slutet, och det är ju ganska ”o-scrummigt” [ej enligt Scrum]. Vi har också en förberedelsefas som är lite mer omfattande än man brukar ha i Scrum, eftersom vi har en beslutsprocess som innefattar ledningsgrupp och produktråd, om vi överhuvudtaget ska köra igång i ett projekt eller inte. Där måste vi ha ganska omfattande underlag redan innan beslutet tas. Så vi har en förfas och en efterfas som är väldigt o-scrummiga, men under själva implementationen så jobbar vi väldigt iterativt och inkrementellt.
3.15	**Okej. Hur mycket erfarenhet har du av Scrum i sig? Och i er egen metod som ni har nu kan jag anta att det är två år?**
3.16	På mitt förrförra jobb, på [en mediekoncern] var jag med och implementerade Scrum, [i slutet av 00-talet]. Där gick vi från en ganska rörig Vattenfallsmetodik, där vi kände att vi hade

	ganska dålig fart i fabriken, om man säger så. Det gick ganska långsamt och var ganska omständligt, och vi missade våra leveransdatum hela tiden. Och vår dåvarande [IT-chef] föreslog att vi skulle implementera Scrum, och då gjorde vi det. Vi var ett stort utvecklarteam – kanske 100 personer – och då gjorde vi det, och började köra den ganska bokstavstroget. Vi körde den verkligen iterativt och vi släppte på systemtestfaser, och vi märkte att vi fick en betydande försämring i kvaliteten. Inflödet av buggar från produktionen ökade markant. Då backade vi litegrann och lade till ett verifieringssteg – precis som jag gjort här – så det är den erfarenheten jag har tagit med mig därifrån och hit. Så har du en (1) produkt, som inte har några…
3.17	**…dependencies [beroenden]?**
3.18	Ja, egentligen, utanför… Har du externa parter som du kan integrera mot dina API:er så kan du alltid stubba dem. Men om du har andra delar i din… Där ni delar kodbas – där det inte finns några separata API:er emellan – så är det väldigt svårt att köra parallella team i Scrum, utan att ha en verifikationsfas efteråt; det är min erfarenhet. Sen så finns det också en sak i Scrum… Nu när jag nämner det så är jag osäker på om det är uttalat eller outtalat, men att man ska ha ett… Jag har sett många [fall] där man jobbar med Scrum och har avskaffat rollen utvecklare och testare – där alla ska kunna göra allting – och den har jag också backat lite ifrån. Jag tror det är väldigt bra att man odlar sin specialitet, och man har ju ofta olika infallsvinklar som testare och utvecklare, på hur man ser problemet.
3.19	**Jamen det är uttalat – absolut – [i Scrum] att man ska kunna byta eller skifta mellan olika roller, och göra det som behövs för stunden.**
3.20	… Och det är en vacker tanke. Rent ideologiskt eller tankemässigt så kan jag inte ha någonting emot det, men där märker jag att en utvecklare som testar, testar på ett annat sätt än en testare gör. Ibland får man för sig att det finns en uppfattning att genom att man är utvecklare så anses det vara den mest komplicerade rollen i ett utvecklingsteam, så har man genom arv – automatiskt – förskansat sig att de andra… Att man kan allting annat också. Men så är det verkligen inte, tycker jag. En verkligt duktig utvecklare har väldigt lite gemensamt med en bra testare.
3.21	**Upplever du att det som – inom flera aspekter av Scrum – att tanken är god, men att det inte funkar?**
3.22	Jag tror att… Det är en intressant fråga! Jag tror att om det funkar eller inte har inte så mycket att göra med metodiken, utan det har med folket att göra. Det blir aldrig bättre än individerna som ingår i teamet; så spelar det ingen roll vilken metodik du har. Scrum – på något sätt – förutsätter att du har ganska seniort och ansvarstagande team. Och har du det så fungerar Scrum väldigt bra. Men har du 5-6 team och implementerar samma process överallt, så

	kommer du få olika resultat, och då är det individerna som avgör. Men det finns ingen silver-bullet i detta, det är hårt arbete allting! Oavsett vilken process du har så måste du jobba hårt och du måste ha duktiga medarbetare.
3.23	**Men låt säga att man har ett erfaret team som är jätteduktigt; tror du då att Vattenfallsmetoden egentligen skulle funka lika bra som Scrum?**
3.24	Nej, det tror jag inte, för i Vattenfallsmetoden har du en stor… Det är inte det sekventiella i Vattenfallsmetoden som är dåligt – tycker jag – utan det är att… Alltså, om du bryter ner en Scrum-iteration så är det egentligen ett litet Vattenfall i sig – beroende på hur korta iterationer du gör. Problemet med Vattenfallsmetodik är att du inte tar höjd för förändring; för allting förändras alltid. Plus att du tror att någon på något sätt har föreställningen att någon kan sitta i förväg och tänka igenom att möjliga eventualiteter. Och det är ju helt orimligt. Så nej, i ett stort vattenfallsprojekt så spelar det faktiskt ingen roll vilka människor du har, det skulle aldrig gå ändå! Även om man skulle göra det bokstavstroget till någon slags Vattenfall. Jag tror att iterativ utvecklingsmetodik är det bästa sättet att jobba på. Men vi har inte hållit på med mjukvaruutvecklingsprojekt så länge som annan tillverkning, så vi har nog många roliga metodiker framför oss också.
3.25	**Så, du tänker generellt att man inte har utvecklat program så länge?**
3.26	Jag tänker alltså ”inom världen”, att vi har inte utvecklat mjukvaruprogram så länge. Och sen så kan jag tycka att den här IT-boomen förstörde litegrann.
3.27	**Hur menar du då?**
3.28	Jo, plötsligt fick folk för sig att det var väldigt enkelt att göra programvara. Och det är det inte; det är ganska besvärligt att göra programvara. Det är hårt arbete. Man tror att ”man skriver lite kod och sen checkar man in den och sen kan man fakturera”. Och jag märker jag, eller det tycker jag att jag har märkt under mina år, att de som kommer in i arbetslivet har en annan syn på mjukvaruutveckling.
3.29	**Vad tycker du är den största problematiken med utvecklingen av program – som folk underskattar, uppenbarligen? Varför är det svårare att utveckla ett program än att driva ett annat projekt?**
3.30	Därför att… Jag tror att man underskattar att det är ett hantverk att skriva kod. Det är en jätteskillnad mellan om du har jobbat i ett år eller i 15 år. Sen är det inte säkert att den som har jobbat i 15 år är bättre, men det handlar hela tiden om att det ju inte finns någonting som är rätt eller fel; det finns ett oändligt antal lösningar på varje problem, och det är du och din kunskap och dina värderingar som sätter ribban för varje litet beslut du fattar under resans gång; och varje beslut du fattar får ju långtgående konsekvenser – de kommer sannolikt leva kvar så länge applikationen lever. Och det är det som gör det så komplext – att det påverkar,

	och… Och ibland måste alla de här principerna med Dry, till exempel, att man säger att man inte ska upprepa sig; men ibland är det mycket bättre att upprepa sig än att göra en taskig abstraktion av någonting – att det är lätt att man bygger in komplexitet. Och sen är det också väldigt besvärligt att man alltid har tidspress. Det är lätt att man börjar delegera saker till sitt framtida jag; att man gör något fult nu som man ska rätta till sen, fast det är väldigt sällan man har tid sen. Och det blir lätt att man sitter med en svår-maintainbar kodmassa. Så det är svårt att utveckla system, men det är väldigt roligt!
3.31	**Där kommer man egentligen in på vår avslutande fråga, nämligen att det är så många IT-projekt som drar över på tid eller budget; man brukar tala om uppemot 70-90 %. Men tror du att de sakerna du beskrev nu, att det är därför som det sällan går vägen som man har tänkt?**
3.32	Ja, och sen så tror jag också att det finns en… Att man ibland fattar… Jag tror att egentligen så borde man inte starta de allra flesta projekten, och det hade man inte gjort om man hade tänkt till lite till. Och det är kanske något som har hänt när man börjar tänka mer agilt. Att man gör ju inget riktigt förarbete. Man lägger rätt så mycket krut på business-case, där man skannar marknaden och kollar på hur mycket kunder man kan ha och hur mycket intäkter man kan få, och så gör man ett väldigt grovt estimat ”mellan tummen och Linköping” på hur långt det tar att utveckla det. Och det är ändå väldigt tidigt – innan man ens har tagit fram en design eller någonting – och så börjar man utveckla; och så någonstans finns det en förväntan att det är det här estimatet man ska koppla någonting till. Men om du frågar utvecklarna så är det väldigt sällan någon tror på något sådant estimat. Man kan alltid vara säker på att allting tar mycket längre tid än man tror. När du börjar att… Man räknar oftast med att man estimerar med helt effektiv tid; men det är ju väldigt sällan man sitter och kodar och allting bara flyter på, utan det är ju problem hela tiden, som dyker upp. Jag tror att problemet är att man har för dåliga beslutsunderlag – att man startar projekt som man aldrig borde ha startat.
3.33	**Nu kommer vi ju raka vägen in på hur ni genomför planeringsfasen och så vidare. Vi skulle gärna vilja höra dig beskriva ett projekt; alltså från planering, till genomförandet, leveransen, och uppföljningen. Vi kan väl börja där vid planeringen? Ni har ju gjort lite annorlunda där, mot just Scrum. Du får jättegärna beskriva hur.**
3.34	Ja, absolut, det kan jag göra. Vi jobbar ju… Vi har ju en produktägarorganisation, vars uppdrag är att ta fram produktinitiativ. Och de består av en slags överblick av caset och ett business-case. Och då träffas produktrådet – jag tror det är en gång varannan vecka – och går igenom de här sakerna, och beslutar om man ska investera utvecklingsresurser i de här

	initiativen eller inte. Och det är egentligen när de besluten är fattade… Det är det som är vår ingående balans till ett utvecklingsprojekt. Och förberedelsefasen består av en del aktiviteter. Vi har brutit ner det så att vi har en förberedelsefas, en implementationsfas, och en verifikationsfas. Och det låter ju väldigt Vattenfalligt, men det är inte så farligt som det verkar. I förberedelsefasen… Vi börjar med att tillsätta team – det är viktigt – för det är teamet som tillsammans med produktägaren tar fram en specifikation. Och det låter också Vattenfalligt – att man tar fram en specifikation – men det handlar om att vi tar fram en övergripande lösningsbeskrivning; vilka funktioner vi ska ha in – inte exakt hur vi ska göra – men vilka funktioner som ska finnas med. Man kan se det som en övergripande definition-of-done, alltså rent feature-mässigt, att ”när de här funktionerna är på plats, då är projektet helt klart”. Sen tar teamet tillsammans fram en implementationsdesign, där man egentligen vill diskutera igenom och definiera lösningen efter olika aspekter; utifrån ett driftsperspektiv, ett prestandaperspektiv, ett faktureringsperspektiv – det kanske är något som måste faktureras – och utifrån en massa andra aspekter som är specifika för våra program. Och detta ska – innan projektet börjar – godkännas av systemarkitekterna och driftsavdelningen; en officiell sign-off innan vi börjar implementationen. Och där har vi också… Just det – det ska jag säga – i implementationsdesignen sätter teamen själva ambitionsnivån när det gäller automatiserade tester; att de sätter att ”vi ska ha ett test-coverage – både frontend och backend – på 90 %”, till exempel; ”de här delarna behöver vi ha ett Selenium-test för, som testar GUI:t” och ”de här API-förändringarna behöver vi också skriva tester för”, så man sätter dem redan här. Detta blir också en del av definition-of-done, så det är teamet själv som definierar detta, och då ska de se till att det blir gjort sen också. Och just definition-of-done tror jag är den viktigaste biten av Scrum, förutom att det är iterativt och inkrementellt, men det kan vi ta sen… Och då har vi vad vi behöver för att starta implementationsfasen – kan man säga – och den är väldigt Scrummig då, och den börjar ju med en sprint-planering. Och sen måste utvecklarna testa iterativt hela tiden; varje sprint avslutas med en demo, och ett retrospektiv. Eftersom vi kör parallella team har vi Scrums en gång i veckan, där en annan som också har min roll – istället för att ha ett processansvar, har projektportföljsansvar – samlar alla Scrum-masters en gång i vecka, och går igenom status.
3.35	**När du säger alla Scrum-masters, menar du för olika projekt eller inom projektet?**
3.36	För olika projekt; vi har bara ett Scrum-team i varje projekt.

3.37	**Hur stora är Scrum-teamen?**
3.38	Det är lite olika, men max fem personer.
3.39	**Scrum förespråkar ju sju personer, plus/minus två. Skulle du säga att det är för mycket att vara nio?**
3.40	Ja. Men återigen, med rätt individer kanske det är alldeles lagom. Men Scrum handlar ju – eller utveckling överlag – om att ha koll på läget, och ju större grupper du får desto svårare blir det att ha koll på läget. Så fem-sex personer tycker jag är rätt lagom. Vi har några som är ännu mindre faktiskt, men då är det ju svårt att få… Då blir det delade roller och det är svårt att få effekt i de teamen.
3.41	**Det kanske bara var jag som missuppfattade, men du sa att ni bara har ett Scrum-team per projekt.**
3.42	Ja.
3.43	**Om ni gör ett stort projekt och behöver vara fler än fem då, hur löser ni det?**
3.44	Vi har inte ställts inför det ännu, men då skulle vi ju skapa flera Scrum-team – det är ju lösningen, såklart. Men vi har inte ställts inför det ännu.
3.45	**Okej. Men då är vi ju redan inne på genomförandet, utvecklingen och så vidare. Jag förstår att i era projekt har många varit lite liknande, där ni behövt ha lika stora team, men hur stor bredd har ni tidsmässigt på de här projekten? Hur många Scrum-cykler blir det till exempel, hos er, oftast?**
3.46	Det där beror väldigt på. Det är ju egentligen det som är variabeln där, eftersom vi jobbar med ungefär lika stora team, så är det antal iterationer eller sprintar som avgör. Så det är väldigt olika, det går inte att säga. Men generellt kan man säga att vi har haft några ganska så långa projekt, som har haft väldigt många sprintar, och det försöker vi undvika. Vi försöker hålla det kort, koncist och tydligt. För då är det också lättare att förutse var vi kommer landa, tidsmässigt.
3.47	**Då kan inte jag undgå att fundera på om… Om ni alltid kör – låt säga fem personer då – och så blir ett projekt väldigt långt; hur kommer det sig då att ni väljer att köra ett väldigt långt projekt, istället för att ta in ett team till?**
3.48	Därför att det inte finns ”hammare för alla snickarna” ibland. Så är det när man jobbar i samma kodbas, som vi gör. Det finns ingen skalfördel i vissa projekt eller på vissa plattformar, att skala upp med folk. Säger vi däremot att vi bygger något helt fristående – utan de här dependencies – så skulle man absolut kunna skala upp med mer team, men jag har ännu inte vaknat en morgon och känt att ”fasen, idag vill jag göra det lite mer komplext”, och därför så är det mycket bättre att inte paralysera ett sådant arbete.
3.49	**Uppfattar jag rätt att ni mest bygger nya funktioner till ert befintliga system?**

3.50	Ja, vi har en plattform som vi bygger nya produkter [till], och förbättrar gamla produkter.
3.51	**Japp. När ni sen ska leverera… Du var inne på testningen och hur det planeras, men när ni väl har gjort era iterationer och känner att ”nu känns det här rätt bra, och vi känner oss färdiga”, hur ser det ut då?**
3.52	Jo, då går vi ju in i den absolut sista fasen, som är ganska kort, där vi verifierar innan vi går live. Och den består egentligen av att vi gör en release-branch, och då har vi…. Och det här är ju inte heller så Scrummigt, men ibland kan vi ha mer än ett projekt som ska gå ut i produktion. Vi försöker jobba oss utifrån att de ska kunna släppa varje projekt fristående, men just idag kan det vara så att projekt 1 och 2 gör en gemensam systemtest och integrationstest. Så det vi först gör är att vi gör en release-branch, och då har vi lite kriterier; då ska definitions-of-done vara klara, och vi ska inte ha några öppna buggar, alla testfallen ska vara utförda, och allting ska vara incheckat och klart, och alla byggen och tester ska vara gröna. Och så har vi också ett dokument som ska finnas, någonting som vi kallar för software-design-dokument; när man i ett projekt implementerar någonting nytt – en ny funktion eller en ny modul, så måste man skriva av den också, så att det finns dokumenterat i en systemdokumentation - det är just för att undvika en-persons-beroende; men när allt det där gjort så går vi in i verifieringen, och då börjar vi med att brancha – att göra en release-branch – och vi har en testledare som utses, som går igenom eller som utvecklar eller tar fram ett systemtest-scope som man kör igenom som alltså – förutom de automatiserade grejerna – så gör vi tester på delar som vi vill verifiera, så vi gör en regression på delar som vi vill verifiera att de fortfarande funkar. Så då gör vi det, och när vi har gjort den så gör vi en provuppdatering i produktion; kör uppgraderingsscripten och ser så att de funkar som de ska; och sen har vi en formell punkt där testledaren skriver en test-rapport som checkas av utav ledningsgruppen, och så tas beslutet om det ska tas i produktion.
3.53	**Okej. Hur fungerar uppföljningen efteråt?**
3.54	Vi har en approach där utvecklande team tar över förvaltningen, så att… Vi har en release faktiskt idag, där det är två team som suttit med varsitt projekt som rullar ut idag. Och då går de över och sätter sig i hyper-care; det är alltså maintenance-teamet. Så alla eventuella buggar som hittas rättar de själva, för att det är det effektivaste sättet att lösa det på.
3.55	**Hur länge brukar de sitta där?**
3.56	Det beror helt på hur stort teamet… Eller jag menar hur stort projektet är. Ändrar vi mycket kod så exponeras vi ju för större risk. Vi har nästan 90 000 kunder, så en (1) bugg får en väldig effekt! Men maximalt någon månad, och efter det… Eller under tiden så har vi vad vi kallar för Topp-10, med funktioner som är för små för att göra till projekt, som man utför

	kanske. Det är ofta sånt som kund eller kundtjänst har begärt, så kan vi göra de grejerna också.
3.57	**Men när ni tar beslutet att utveckla något nytt från början; kan det räcka med att en (1) kund jättegärna vill ha detta, eller är det något som ni helt avgör själva – att de ser ett behov och då går in och skapar den produkten?**
3.58	Det är en ganska tung process egentligen, eftersom det är ganska… Eftersom man alltid har ett ändligt antal resurser i utvecklingstemat, så handlar det alltid om väldigt hårda prioriteringar. Det är ju alltid det som ger mest värde till våra kunder, som ger mest värde åt oss. Så vi kan ju inte gå all-in på ett utvecklingsprojekt som en (1) kund vill ha, utan vi måste ju utifrån den inputen vi får från alla våra kunder… Det är ju det produktägarens jobb är egentligen, att göra den bedömningen [huruvida] det är rimligt att vi investerar i vissa funktioner för att underlätta kundansvar. Så nej, det är inte en (1) kund som styr, utan det är mycket input som ska knådas och behandlas innan vi kan ta de besluten.
3.59	**Jag kan inte allt för mycket om er produkt, men har ni KPI:er till exempel, så att kunderna själva kan sitta och skapa sina egna små kompletterande funktioner, om det är något de saknar?**
3.60	Nej, det har vi inte, utan det måste gå igenom oss.
3.61	**Vilka skulle du säga är de största skillnaderna mellan Scrum och er metod, just i uppföljningen och efteråt? För du sa ju i början att det är där som er metod skiljer sig mest markant?**
3.62	Ja, innan jag svarar på den frågan kom jag på en sak till som man tror… Eller som är problem med alla utvecklingsprojekt, som många trodde att Scrum skulle lösa – som det inte har gjort – och det är att mäta progress. För det vill ju alla alltid veta, ”När är vi klara?”, och där… Jag hittar inget bra stöd varken i Scrum eller i någon annan metodik heller för att mäta den faktiska progressen – för att kunna förutse en leverans. Och det är svårt, för att har du bara en (1) sprint med få points, så är du ju rätt nära; men har du många sprintar och många, många stories så är det ju svårt att pricka det här. Så det tycker jag är en begränsning i alla metodiker. Jag vet att det finns lite nya initiativ – för vi har en ex-jobbare som håller på och jobbar med oss om det just nu – där vi diskuterar hur vi ska hitta mätetal; det finns ju lite olika tankar om det, med milestones och annat. Men det är svårt, för det blir alltid subjektiva bedömningar som ligger till grund för det, snarare än hård fakta. Men din ursprungsfråga var… Eller vad var det du sa?
3.63	**Vad är det ni har kompletterat med – gentemot Scrum – efter projektet är klart?**
3.64	Innan projektet har vi kompletterat projektet med; vi har… Vi värderar specar högre än vad Scrum gör. Det är egentligen den. Det är den som är den stora grejen i förstudien.

	I verifieringsfasen på slutet så är det väl att framförallt att vi inte har ett fungerande increment efter varje sprint, utan att vi faktiskt gör en integrations- och systemtest efter sista sprinten är slut. Så definition-of-done – ur ett projektperspektiv – är egentligen klar, men sen vill vi inte releasa utan att verifiera en gång till. Så det är väl de stora skillnaderna, tycker jag.
3.65	**Okej. För att gå tillbaks – bara lite kort – till genomförandet; använder ni exempelvis samma roller som Scrum? Just med Scrum-master och…**
3.66	Produktägare tänker du på då?
3.67	**Ja, precis.**
3.68	Ja, det gör vi.
3.69	**Exempelvis User Stories och att man spelar om poäng och liknande, är det något ni använder också?**
3.70	Just user stories använder vi; planning poker använder vi inte överallt, det är inte alla som använder det. Där kan man ju ifrågasätta hur mycket man då ska gå in och styra i en process – enskilda individer – så länge out-puten är bra. Men just med planning poker får man välja hur man vill göra, i varje team.
3.71	**Ser du några svagheter i metoden ni använder, eller i Scrum i helhet, som är väldigt svårt att lösa, och som ni inte riktigt har kommit underfund med?**
3.72	Det är bara förutsägbarheten, som jag ser det. Det är det absolut största. Och det har inte så mycket med metodiken att göra, utan det har att göra med att ”Du kan vända på en sten, och där ligger det något riktigt läskigt, och det finns ingen process i världen som kan skydda dig från det”. Det är det stora problemet som vi brottas med; det är förutsägbarheten. För den skapar ju förväntningar, och det är förväntningarna som skapar besvikelse om man blir försenad. Och vi vill ju inte göra människor besvikna.
3.73	**Då låter det nästan som att man ska sluta budgetera och spika de förberedande eller uppskattande delarna, innan det har börjat? Men jag föreställer mig att det är svårt, för då får man ju aldrig lov att börja någonting.**
3.74	Nej men så är det ju, men det handlar ju om att… Optimalt är ju att du kan prototypa; att du kan bygga saker eller alltså prova olika implementationer och göra det i några veckor, och sedan så slänger du dem – och då vet du hur du ska bygga dem – och sen bygger du den ordentligt. Lite av ett lätt sidospår; det är oftast det som står i kontrast till att du vill bygga testdrivet; att du vill göra en testdriven utveckling, men om du ska prototypa så vill du ju inte skriva tester först, utan då vill du ju först göra en prototype, så du vet hur det ska gå.

	Sen slänger du det och börjar skriva det testdrivet. Men oftast gör man inte det… Utan börjar du prototypa själva utvecklingen. Är man senior nog så backar man när man kör fel, och gör man inte det så implementerar man något väldigt knöligt. Och då kommer testerna alltid för sent sen. I bästa fall skrivs de när allt är utvecklat, och det är inte lika bra. Men din föregående fråga, som jag tänkte kommentera på också… Jo! Det här med förutsägbarheten. Sättet vi attackerar den på; gör vi faktiskt inte alls genom arbetssätt, utan där jobbar vi stenhårt med kompetens hos utvecklarna; att se till att de har så bra domänkunskap som möjligt om de produkterna vi utvecklar, och [att] vi hela tiden snackar om craftmanship och kodkvalitet, så att man tar så många bra beslut som möjligt – löpande. Så det är väldigt mjuka grejer vi jobbar med, men det är de jag tror ger bäst effekt; så är det.
3.75	**Att jobba mer med personerna som är i projekten, än att jobba med varenda detalj kring hur man styr projekten?**
3.76	Ja, det är där man får störst effekt hela tiden.
3.77	**När tror du att det inte funkar att köra agilt; eller funkar det alltid? Eller situationer då – exempelvis – låt säga stora jämfört med små projekt? Vi har fått en del input på att det går inte att köra Scrum när man kör jättestora projekt, för att det inte är värt den kostnad det blir att ha så mycket planering och att alla ska spela user stories och så vidare. Håller du med i den kritiken mot Scrum, eller tror du att de har gjort det på fel sätt, så att säga? Det blir ju lite mer om dina uppskattningar, eftersom ni inte har drivit så jättestora projekt, men vad tror du själv?**
3.78	På mitt… På tidigare jobb har jag varit med på stora Scrum-projekt, och jag har inte upplevt det som ett jättestort problem; det kan jag inte säga. Och vi hade team på flera [platser] där också; några i Indien och några i Sverige, och jag upplevde inte det som några problem. Problemen hade varit mycket större om jag hade kört det i ett Vattenfall – det tror jag. Så jag tror mer att det har att göra med förväntningarna på feedback – alltså på kontroll och förutsägbarhet. För om du drar igång ett stort projekt med många Scrum-team, då har du jävligt dålig koll på läget, som någon slags beställare eller som någon som ska ha koll på hög nivå; för då kör du igång ett maskineri, och du har ingen aning om var det kommer landa någonstans. Och då kanske man får – vad jag tror – en slags falsk trygghet, om man sätter någon som sitter och skriver en spec i ett år först, och så sätter man någon och estimerar ett år, och sen börjar man utveckla. Då kanske man kommer närmre – jag vet inte – men jag tror att känslan av kontroll är större i det senare fallet.
3.79	**Och hur löser man det bäst då, för att man inte ska hamna i den här sitsen om och om igen?**

3.80	Ha rätt folk. Det är faktiskt det; det handlar nästan bara om det. Och ha… Det man också har märkt i Scrum under de sista åren är att produktägarens roll är – om man ska prioritera – så tror jag att produktägarens roll är viktigare än Scrum-masterns. Det märker man, att har man en bra produktägare så har det en enorm effekt på utkomsten av hur det går med projektet. För det är oftast där man kan kapa tid, i att få snabba beslut, och slippa spekulera och gissa; och skriva om, framförallt. Så det står faktiskt i vår utsträckningsprocess att om demon är första gången produktägaren ser vad det är du har byggt, så är det någonting som har gått fel. Oftast här så är de med i alla stand-ups på morgnarna, vilket är fantastiskt bra.
3.81	**I och med att vi har varit och touchat på de sista frågorna vi hade så kan jag inte säga att vi har så många fler frågor nu.** **Har du något som du skulle vilja säga, eller frågor som du hade förväntat dig att få som vi inte har ställt?**
3.82	Det är nog i så fall just kring… Jag upplever att det finns en… Eller jag tycker det känns som att opinionen har svängt kring Scrum – alltså snacket som går – att folk inte är lika övertygade om att Scrum är ”the silver bullet”; men jag vet inte riktigt vad man går mot istället. Jag tror att vad man har problem med hos Scrum är att man har svårt att förutse. Så just den problemställningen tycker jag personligen är väldigt intressant, just i att förutse var man kommer landa. Och den går ju väldigt tvärs emot allt som är agilt, iterativt, och inkrementellt; att vi vet ju inte det egentligen – ”vi börjar här borta, och så kör vi”. Så just den… De som fattar beslut om projekt har oftast förväntningar – och då menar jag inte bara på Fortnox, utan i hela världen – har förväntningar på att veta när det ska vara klart och hur mycket det kommer kosta; och det tycker jag inte att Scrum hjälper oss med.
3.83	**Sen finns det ju de som – enligt de själva i alla fall – kombinerar Scrum och XP, till exempel.**
3.84	Vilka aspekter av XP använder de då?
3.85	**Jag är inte säker i detaljnivå på det, men den uppfattningen vi har fått är att vissa tycker att Scrum är för ”bara projektformsbaserat” – det säger ju väldigt lite om hur du går in och kodar, till exempel – och att det är de aspekterna man tar från XP. Ser du ett behov av det, eller tycker du att ni har klarat er bra ändå; utan att följa en sådan metod för just den rena utvecklingsbiten av projekten?**
3.86	Men den styr vi ju också; det är ytterligare en dimension, där vi har våra kodstandarder och hur vi utvecklar. Att vi inte har uttalat par-programmering, utan det gör vi när vi känner att det behövs. Och vi uppmuntrar det absolut… Men det är litegrann det som är faran med att trycka in en sån grej som att man ska köra eXtreme Programming över hela linjen; att man säger att ”nu ska alla göra det”. Det är inte alltid det passar och inte alltid det funkar. Det

	viktigaste är att varje individ förstår när man ska använda det – när man ska använda vilket verktyg ur sin verktygslåda – mer än att man ska på… Vissa saker, som att ”vi ska skriva enhetstester” och ”vi ska fakturera där det behövs” och ”vi ska inte skriva ful kod”, de grejerna ska vi ha som must-haves – det är en del av definition-of-done hela tiden. Men exakt hur man ska välja att ha saker, det är mer personliga val, och det handlar mer om att alla utvecklare ska förstå vilka verktyg de har i sin verktygslåda och när de ska använda vilket; och då kommer man ju tillbaka till folk igen, och vilka vi har i teamet.
3.87	**Kan det vara så att – nu spekulerar jag ju bara, men – att företag som inte har så seniora eller så duktigt folk, helt enkelt, känner att det blir för spretigt – det de skriver – och att därför behöver de en utvecklingsmetod på det sättet? Tror du att det kan vara på det viset? För jag börjar ju nu genast fundera på varför folk väljer att knö in XP ändå.**
3.88	Jag tror det är svårt att svara någonting på den frågan egentligen, utan att riskera att vara helt fel på. Jag kan ju bara utgå från mina egna…
3.89	**Absolut!**
3.90	Min ytterst personliga uppfattning är att… Jag tror som chef och ledare för ett utvecklingsteam, att… Mitt uppdrag är att de har alla förutsättningar de behöver för att lyckas med ett projekt; och det innebär att de ska ha rätt verktyg, både fysiskt och digitalt. Har man ett litet team så ”fine”, då bestämmer man i det här teamet att ”vi kör XP här, och så utvärderar vi det, och så tar vi det som var bra där och fortsätter göra det, och så tar vi det som var dåligt och gör det aldrig mer igen”. Och så är det med allting; det finns ingenting som är komplett och som har allt du behöver i varje given situation, utan det handlar om att ha ett set av verktyg och veta när du ska använda dem. Därför tror jag det är väldigt… Scrum är på en sådan hög nivå och det kan du implementera över hela organisationen, men XP är hur du löser dina problem, och det är individuellt. Det blir inte effektivt om du ska tvinga alla att göra på samma sätt hela tiden; tror jag inte. Men det är ju min personliga syn.

Bilaga 10: Transkribering #4 – Acando

4.1	**Vi kan börja med att få höra vad din nuvarande titel och din roll i företaget är?**
4.2	Ja, jag [har brett IT-ansvar] [för en av Acandos svenska regioner]. Vi gör lite olika saker inom olika delar av konsultbranschen kan man säga. Pluggade industriell ekonomi [på universitetet] och gick ut [i mitten av 1990-talet]. Har jobbat i konsultbranschen sedan dess och har varit på rätt många olika typer av företag, ofta ganska stora företag, såsom [biltillverkare], [telekombolag], [energibolag], [vitvarutillverkare] och så vidare, och hjälpt dem att införa mycket IT-system; har drivit väldigt mycket verksamhetsförändring. Så det är min bakgrund. Jag har också jobbat som projektledare och programledare i olika typer av projekt.
4.3	**Hur lång erfarenhet har du av branschen?**
4.4	Det blir [15-20] år.
4.5	**Du berättade att du jobbar lite Vattenfallsmetodsaktivt i mailet, men du har också erfarenhet av Scrum och Kanban. Hur länge har du arbetat med just Scrum och Kanban?**
4.6	Får jag göra en recap på det? Jag pluggade kvalitetsstyrning som en av inriktningarna [på universitetet] där på slutet, och där läste man väldigt mycket om kvalitetsstyrning utifrån tillverkande – alltså alla de här Lean, total quality management… Lean kommer ju ifrån Asien, Kina, Japan; framförallt Japan utifrån fordonsindustrin. Kaizen – ni vet alla de här begreppen – ständiga förbättringar och komma ner till kärnan till problem. Någonstans kanske runt… När systemutveckling och IT med Internet… Jag fick mail [i mitten av 1990-talet] så ni fattar hur gammal jag är i formen till IT-branschen – som tar rätt mycket förgivet som inte fanns då. Men någonstans när systemutveckling och systemproduktion började bli en allt viktigare del i företag och industri, så började man anamma mer av de här tillverkningsmetodikerna; typ Lean, Scrum – som man kan säga är en del av Lean-metodiken. Kanske, mitten på 2000-talet skulle jag välja säga, början på 2000. Det fanns saker som eXtreme Programming [XP] och massa andra saker, men det har mer trattat ner till att bli de här metodikerna. De har ju en väldigt kort horisont om vi pratar systemutvecklings-, utvecklingsmetodik, medan Vattenfallsmetoder – alltså traditionella Vattenfallsmetoder… Det är bara till att titta på byggindustrin, vad kör de? Du ska ha ett upphandlingsunderlag, du ska ha en kravspecifikation på ”det här ska byggas”, så det har ju funnits utifrån ett projektperspektiv i 100 år eller mer. Så det är ju därifrån de flesta företag… Jag har ju jobbat väldigt mycket med väldigt stora företag, så Vattenfallsmetodik har jag ju jobbat med i [15-20] år kan man säga. Scrum och de andra delarna då kom jag i kontakt med [i början av 2000-talet] och framåt, så där har jag betydligt kortare praktisk

	erfarenhet själv, men jag har en ganska god insikt i vad som funkar, vad som inte funkar och varför det inte funkar eller varför det funkar.
4.7	**Intressant! Vi kan ju fråga, använder ni dessa metoder ofta när ni arbetar eller skiljer det sig från projekt till projekt? Eller försöker ni använda er av metodiken till alla projekten?**
4.8	Jobbar man i stora företag… Nu blir detta lite komplext då, men ni känner till begreppet ERP-system?
4.9	**Ja.**
4.10	De är liksom hjärtat, lungorna eller ryggraden i de flesta företag, tillverkande företag och så vidare: de behöver inköpsprocesser, produktionsprocesser, lagerstyrningsprocesser, säljprocesser, något sånt där, det ser väldigt lika ut på alla företag. När man ska tillgodose vad dem där behöver är de ganska lika, om du ska ha ett ekonomistyrningssystem och ha en huvudbok, reskonto och så vidare. Där vet man för att lagen säger att ”såhär det ska fungera”, det finns inte någon anledning till att ha ett agilt arbetssätt för där är kravspecifikationen som den är. Sen finns det något som kallas kodsträng, ”Hur eller när vill det här företaget följa upp sin ekonomi?”, ungefär så, och där är det inte heller någon anledning att köra ett agilt arbetssätt för där är det så väldigt tydligt och det ser väldigt lika ut, iallafall på alla publika bolag och aktiebolagslagen styr liksom allt det där. En agil metodik bygger ju någonstans på att du börjar väldigt litet och bygger på det, men om du har en produktionsprocess där du vill byta ut det nuvarande systemet kan man inte byta ut 1/10 av produktionslinan och ersätta det med ett system som du utvecklar agilt för då måste du förhålla dig till det gamla, för du kan inte ersätta allting samtidigt. Så beroende på vad det är man ska göra, verksamhetsmässigt och processmässigt så lämpar det sig mer eller mindre bra. Men om du tar att ett företag ska han en ny webbsida, då kan du leva kvar med din gamla webbsida medan du utvecklar den nya hemsidan agilt och den får växa fram eftersom att man inte vill ha den här stora kravspecifikationen och bestämma att man gör olika provskott i form av något proof-of-concept efter ett pilotangreppssätt och komma fram till att ”såhär blev det”. Sen drar man pluggen ur den gamla hemsidan och lanserar den nya. Där fungerar agil utvecklingsmetodik jätte-, jättebra, eller om du vill ha en mindre applikation där du kan gå in och boka biljetter; den kan utvecklas agilt för det fanns ingenting innan. Det finns så extremt stora fördelar att utveckla agilt. Om du tar ett Vattenfallsprojekt – ett stort affärssystemsprojekt – där du ska göra ett ERP-

	system som kostar mellan 500 miljoner och 1,5 miljard, då vill man någonstans ta ett beslut på, för att applikationen är till förhållandevis enkel, det är en promille i komplexitet om ens det av vad det är att utveckla ett stort affärssystem, men du kan inte sitta på företaget och upphandla ett nytt affärssystem och säga ”Vi kör lite agilt, vi börjar med inköpsprocessen och så får vi se”, det funkar inte. För du måste någonstans ta ett beslut; ”Det här systemet tar tre år att implementera” medan applikationen tar kanske 6 veckor som involverar kanske kommunikationschefen och två till. Medan det stora projektet involverar 300 personer. Hur ska du få till en metodik, ett arbetssätt som de här personerna aldrig har hört talas om innan? De är vana vid att [arbeta] med en stor kravspecifikation, så det handlar om de delarna också. Problemen när man kör ett sådär stor affärssystemsprojekt som är tre år, då hinner företaget omorganiseras sex gånger, de hinner köpa två bolag någonstans i Asien eller någonstans i Europa. Som också ska in i den här kravspecifikationen, som då blir out-of-date efter ett halvår. Vilket gör att affärssystemsprojekt alltid blir för dyra, tar för lång tid och man når aldrig budget samt att alla blir mycket missnöjda. Men du kan inte göra det på något annat sätt och det funkar inte, det finns ingen bra metodik, men det fungerar inte att göra det på ett agilt sätt, för att…
4.11	**”… fånga delar som ska involveras och så”?**
4.12	Exakt. Ta SAP, som jag har jobbat väldigt mycket med; det är gigantiskt. Det jobbar cirka 17 000 systemutvecklare som bara sitter och knackar kod, 365 dagar per år. De hinner producera lite kod. Så att bara att följa med hur systemet växer och köpa komponenter är enormt. Men det man kan säga då om hur vi angriper detta; vi tycker ju inte att det här med Vattenfall är det bästa angreppsättet. Så man kan komma fram tillsammans med kunden att ”det här ramverket på vad ni vill att vi ska göra”. Och någonstans ”Vad är det övergripande inom varje område?”. Låt säga att vi ska införa ett nytt intranät, [med] SharePoint, på en stor verksamhet som intranätet ska stötta. Det är ändå ganska avgränsat; ”Vad ska vi ha på intranätet? Jo, vi ska ha informationssiter för olika delar av verksamheten, vi ska ha samarbetsytor, verksamhetsdokument, policys, guider fån HR och ekonomi”; ändå ganska greppbart, och då kan man säga ”Bra, det här har vi uppfattat det som: vi tror att det kommer att ta ungefär ’så lång tid och kosta ungefär så och såhär mycket pengar’ men vi kommer att jobba tillsammans med HR-avdelningen om hur deras site ska se ut, vilka dokument som de vill ha där och vilka samarbetsytor som dem vill ha. Vi kommer jobba tillsammans med företagsledningen eller olika delar i strömmar där vi har ett agilt angreppssätt där vi kör sprintar på 2-3 veckor där vi tittar på vilka user stories är det som är viktiga för er kära HR avdelning? ’Ja det är dessa user stories’. Bra, då prioriterar vi det och

	så kör vi ett fokus på det i 3 veckor, kör en sprint demo, 'ja det var inte riktigt sådär eller bra det är sådär', då vill vi lägga till de här user storiena. 'Där fungerar det väldigt bra, och där kan vi någonstans kombinera Vattenfall och agilt' och vi kommer överens om kravspecifikationen på någon nivå, alltså inte nere på skruvmutter och detalj men ungefär förväntningar. Men så använder vi oss av agil metodik när vi just utvecklar ett system." Den stora fördelen med att jobba agilt är change management, förändringsledning, för alla dessa projekt vi gör oavsett storlek innebär förändring för våra kunder, de måste ändra sitt arbetssätt. Det kanske ändrar roller och ansvar, den här rollen finns inte längre för den är för den är automatiserad i systemet. Förändringsledning är en jätteviktig del för att det här företaget ska få sitt business case som det handlar om att gå ihop och har du en agil metodik har du en mycket större möjlighet att få den här förändringen och bli tydlig allt eftersom. Eftersom att de här personerna kommer se att så här kommer det nya systemet se ut, de här kommer att få sån påverkan på mig eller på vår organisation, du får en förankring hela tiden under projektets gång. Medan om du kör ett traditionellt Vattenfallsprojekt säger man "bra, nu har vi den här kravspecifikationen, det kommer att kosta er 400 miljoner och vi kommer att vara klara om 2 år". Då skjuter man nästan alltid det här med förändringsledning och förankring hos de enskilda individerna till slutet. Då blir det, "Va, skämtar du? Ska jag vara med om det här? Det här fungerar inte för mig; jag kommer ju inte ha något jobb" eller vad det nu är för något. Men om man har en agil metodik då får du den där förankringen och det blir inte en sån stor chock i slutet av projektet – vilken påverkan det fick på verksamheten – än om man har den agila metodiken, där man får det liksom inbakat någonstans i projektet.
4.13	**Så om vi förestod det rätt använder ni er lite av Vattenfall och lite av agilt. Där ni jobbar med hela organisationen men plockar ut olika delar och arbetar med – som du sa – exempelvis HR-avdelningen. Arbetar ni samtidigt med dessa men i olika team då?**
4.14	Ja, så är det. En annan sak som är viktig och en stor fördel med att jobba i agilt metodik, då tänker jag på Kanban, det är ju visualisering. Att man förstår, för det är också en sån där del, "Hur man får kunderna att fatta vad gör vi på dagarna? Vad fokuserar vi på? Vad jobbar vi med?". Då tycker jag Kanban är väldigt bra, om du har någon form av backlogg, något vi jobbar med kravmässigt, utvecklingsmässigt, test, produktion och driftsättning. Då har man ett antal olika user stories som vi har i scope att vi ska göra, bruttolistan, men vi kan inte jobba med, man pratar om WIP-limits; Work In Progress. Vi kan inte jobba med mer än fyra stycken krav och då brukar man dela upp delarna i olika storlekar, t-shirt size; Small, Medium, Large [S, M, L]. Vi har ett jättestort krav [L] och två stycken små krav [S] sen har vi ett krav alltså komplexitetsmässigt medium [M].

	De här personerna som vi har som jobbar med krav kan inte jobba med mer än ett stort krav, ett medium och tre små krav parallellt. För vi har inte mer folk eller resurser från vår sida eller från kundens sida. Då kommer man överens om det som ska prioriteras, sen kommer kunden och skriker att ”Det här är skitviktig, vi måste göra detta nu”, “Ja men då kära kund, om ni vill ha in det där kravet och ni vill att vi ska göra det nu, då måste vi flytta ut det här kravet för vi kan inte köra två stora krav [L] samtidigt”. Samma sak gäller utveckling, test och sådana saker. Så Kanban, utifrån när jag pluggade [på universitetet] var något helt annat än vad det är idag.
4.15	**Att ”Kanban var något helt annat innan”; hur menar du då?**
4.16	”Kanban” betyder ju ”kort”, så på mitt första jobb, då jobbade jag på ett företag som tillverkade mobiltelefonantenner och basstationer. Då hade vi lagerpunkter, där man beställde komponenter för att bygga dessa, så när vi tog den sista lådan på hyllan av plasthylsorna, då satt ett Kanban-kort i den sista lådan. Då tog vi det Kanban-kortet, la det på faxen – som faxades till underleverantören, som visste att ”då ska vi skeppa 50 tusen nya komponenter till den här kunden”. Så de var liksom en pull-flöde. Och det är lite samma sak här, att du vill ha ett pullflöde, du vet att du ska göra det här och du vill ha en pull genom och få en jämn ström. För att du vet att du inte kan ha mer än så många komponenter i det ledet och så många komponenter där. Det är Kanban då. Så det här är lite annorlunda men bygger på samma princip, det här är ett jättebra sätt, för att återkoppla till det jag sa från början med visualisering; här blir det väldigt tydligt för en kund och för oss vad vi prioriterar, vad vi gör, man har en sådan här Kanban-board i projektrummet, där både vi och kunden ser, ”Okej, det är det här vi jobbar på den här sprinten”, sprintarna blir ju rätt mycket, en sprint kan ju vara över hela, men det är liksom här när vi kör utvecklingen, vi har kommit fram till att det här och det här kravet, som user storien innebär, det är här vi kanske kör de tre-veckors cyklerna, sprintarna. Men det här är något som är jättesvårt att få till ifall man kör Vattenfallsmetodiken, för oftast är det ett stort krav och att man inte kommit överens om att stycka de här kraven i user stories i greppbara komponenter för vad det är vi ska göra för den här kunden, brukar landa i ett IT-system för kunden. Så det finns ju flera olika… men det här tycker jag är nästan lättare att få företag – om man tar traditionella Vattenfallsföretag – det här är lättare för dem att fatta och ta till sig för det är visuellt dem ser och förstår hur det här fungerar. Vi kör det här med postit-lappar liksom, eller så har man Trello… Än om vi säger Scrum, där du har en produktägare som är en person i kundens verksamhet. Det är ju inte vi som ska

	prioritera kraven, det är ju kunden som ska göra det och berätta det för oss. Då måste det vara en person från verksamhetssidan som gör det. Vi måste ha någon form av Scrum-master, alltså en person från vår sida, som kan hålla ihop det här och kan säga till kunden att ”Nej det där funkar inte”, men också en Scrum-master som håller i alla de här daily standups och så vidare som driver teamet utifrån det här utvecklingsangreppssättet och som också håller ihop sprintdemos, när man demar det man har gjort under den här sprinten för kunden osv. Och som också tillsammans med kunden kopplar tillbaka och säger ”Okej, vi behöver göra den här ändringen eller justeringarna i vår backlogg, lägga till flera, eller säga att nu är vi klara”. Då tycker jag att det här är ett väldigt bra och ett lättare steg det blir… det är ju ett agilt angreppssätt och du kan ha dem här rollerna som man stipulerar i Scrum också, men just visualiseringsdelen med Kanban tycker jag är väldigt effektiv.
4.17	**När du förklarar det på det här viset kan vi se varför du tycker det är mycket bra.**
4.18	Det är därför Scrumish; ”Vi kör inte Scrum, vi kör något ungefär”… Och det tror jag för att man har svårt att ta till sig och att de är vana vid att ”Det här är kraven, vi kan inte starta igång någonting och ge konsulterna pengar om vi inte vet exakt vad vi får”. De har svårt för det. Och då tror jag att det här, “Okej, vi prioriterar de där sakerna, det är det vi jobbar på nu och det kommer ta 3 veckor att göra”; då blir det mycket tydligare, vad det är vi kommer att leverera.
4.19	**Men vad skulle du säga… Vilka skulle du säga det är som har svårt för detta? Är det de mer traditionella, lite större företagen som har jobbat mycket med de tidigare metoderna?**
4.20	Ja, dels det, för att det är historiskt liksom. De har inte den inställningen eller insikten. Men det bygger också på… De här kunderna har ju sitt… oftast så gör ju vi någonting som är vid sidan av deras ordinarie [uppgifter]. Låt oss säga ekonomichef; ”Vi ska införa ett nytt ekonomisystem”. Han eller hon rätt mycket att göra som det är i sin dagliga verksamhet. Då säger vi att ”Vi ska köra det här projektet, det kommer att ta 50 % av din tid”, “Vadå? Jag har ju inte tid!”, ”Då får ni anställa en ekonomiassistent som fyller upp dig”. Då är det jätteskönt när man vet, ”Det här tar upp 50 % av min tid under en månad medan vi ska ta fram den här kravspecifikationen”. ”Men sen går ju ni iväg och gör ni det ni ska och så levererar ni systemet till mig.” Det bygger ju på att den här personen är involverad hela tiden och är med på sprint-demos och allt sånt där. Och i vissa företag gillar de inte det av någon anledning. De tycket det är skönt att fixa kravspecifikationen och sen släppa det och ”fokusera på det jag ska göra så löser ni mitt problem”. Det låter kanske konstigt men det är lite mindset-skillnad, de kunder som är med

	på det här och vi får det till att fungera, tycker att det är skitbra, för de har koll hela tiden och vet precis vad som händer, vad de kan vänta sig. Prata med sina medarbetare och kunna informera dem hela tiden, få sina medarbetare involverade när man kör de här sprint-demosarna, [att] få den här förändringsledningsförankringen kontinuerligt. Så det är en del att det kan finnas ett motstånd, men mycket är för att man har alltid gjort så. Det är så vi gör nuförtiden när vi driver ett projekt på ett stort företag. Ja, och dem går ju åt helvete alla projekten och kostar alldeles för mycket men dem tjänar så mycket pengar så dem behöver inte bry sig. För mindre företag som är mindre, eller som är mer känsliga för kostnad… För det stora företaget spelar det ingen roll ifall det här systemet kostar 500 miljoner eller 600 miljoner, spelar ingen roll, för många andra företag är den där diffen på 100 miljoner rätt tung. Där är de mer lyhörda till när vi förklarar fördelarna med vad det här är och innebär för dem i form av kostnadskontroll och massa andra saker. "Ja men fine, vi testar det", där fungerar det bättre när vi jobbar med lite mindre kunder eller om man jobbar med ett litet team hos en kund som ska ta fram den här appen, då fungerar det även i en sån ganska trög verksamhet, så det är lite case-by-case. Men om vi pratar om då stora ERP-prylarna eller sånt; du kan inte utveckla ett system och ersätta delar av ett ERP-system, då ska du hålla på med en massa integrationer och det kostar jättemycket pengar, det blir väldigt komplext.
4.21	**Skulle du säga att det är lättare att skapa acceptans på mindre företag än vad det är på större?**
4.22	Ja, rent generell skulle jag säga. Sen kan det finnas vissa delar av företag eller vissa team eller organisatoriska enheter där man kan. I vissa fall måste man börja i ganska liten skala och visa på att "vi kör det här agilt, när vi utvecklar den här applikationen, det här systemet, då använder vi den här metodiken, titta vad bra det är". Man pratar om success stories; "se vad bra det blev". "Okej", säger kunden, "om det gick så bra där skulle vi inte kunna skala upp det här och göra det lite större så att de skulle kunna ta lite större komplexet i projekt med samma metodik osv?" Och så börjar man bygga på mognaden hos kunden och bygga upp deras förmåga, för det handlar väldigt mycket om deras egen förmåga, som de måste jobba med. Därefter kan man skala upp det. Jag menar Google; hur många anställda är dem? 100,000? Kör de Vattenfall? Nej, glöm det, de kör agilt. Amazon; kör de Vattenfall? Nä, de kör agilt. Men det kommer ju därifrån, de har ju inte... Alfa Laval startade ju 1876; de har ju en annan historik. Google, när startade de? Typ 2000, liksom. De har inte det där legacyt, det arvet av att det sitter i väggarna, av att "såhär vi jobbar och har alltid jobbat och ska så vi göra". Så många av de nya företagen också, de jobbar ju såhär för att Vattenfall finns ju inte

	liksom och det kan va rätt stora företag också, så det är rätt olika, den där historiken i ett företag; mognadsgrad och vad man gör. Amen Google har ju inga fabriker, dem tillverkar ju inte plattvärmeväxlare, så det har mycket att göra med business att göra också, om det är mycket IT-drivet… Alltså hela bank, finans och försäkring; de har ju inga produkter som är fysiska, hela deras produktionslina är IT, telekom lika så, energisektorn till stor del också. Även om dessa också tänker mycket Vattenfall, där är IT-nerven eller DNA:t deras produktionslina för deras verksamhet. Alfa Laval ser ju fortfarande att ”vi tillverkar produkter; värmeväxlare och separatorer”. IT är ju bara ett stödjande support-funktion. De [kanske inte förstår vad som händer om de inte har IT]… Eller de fattar ju det, men det blir inte lika påtagligt. Det är jätteviktigt! Vi skulle kunna göra mycket mer med den här teknologin om vi jobbade på ett helt annat sätt och förstod att IT och verksamhet är integrerade med varandra, än företag där IT är verksamheten, som i en bank till exempel.
4.23	**Om vi fokuserade mer på er; är det standard då att ni mixar era olika metoder som du har förklarat här? Om man skulle fråga er vilka metoder som ni använder; skulle svaret bli att ni blandar väldigt mycket?**
4.24	Ja, det gör vi.
4.25	**Så det är en standard i sig, att ni blandar olika metoder?**
4.26	Om vi pratar om ett team som håller på med affärssystem är det inte standard. Där kör dem Vattenfall punkt slut.
4.27	**Och det är då SAP:s eget?**
4.28	Ja, det är SAP:s eget. Där är de beroende till… Det är därför jag säger att det är så komplext. Där är vi beroende av en produktleverantör; SAP. ”Hur är deras best practice av hur man implementerar deras system? Vattenfall.” Då förhåller man sig till det, men om vi pratar om vårt SharePoint-team där vi utvecklar kollaborationssidor och intranät åt våra kunder; där har ju inte Microsoft i det fallet då en åsikt om hur man ska göra det där. Där kör vi agilt hela tiden, så det beror också på typen av projekt, typen av teknologi, plattform och allt sådant. Men vi som företag, vi som företag vill ju så långt det bara går, där förutsättningarna finns jobba i agilt.
4.29	**När ni jobbar agilt, kör ni det 100 % då?**
4.30	Nej.
4.31	**Vad är det som gjort att ni inte gjort de?**
4.32	Kunderna skulle jag säga, att de inte riktigt har flugit där. Och kanske också för att vi inte… Nämen ta Sharepoint, där kör vi 100 % skulle jag säga, men där är väldigt mycket Kanban med Scrum.

4.33	**Kunderna spelar roll på hur mycket ni använder metodiken, är det olika delar beroende på vilken kund det är, eller väljer ni ut det själva?**
4.34	Nej det är också olika, vi tar ett företag och säger att dem som kör Props som sin projektmetodik. Oavsett om det är vi eller vilken annan leverantör som helst som ska leverera till dem. "Då ska ni använda er av vår metodik och den heter Props. Den har milestones och tollgates, den har kravspecifikation, realisering, test, go live och support." Ska ni leverera ett litet system en applikation eller ett jättestort SAP system ni ska köra Props. Andra kunder kanske inte är lika rediga och kanske frågar "Okej, ni gör det här dagligen, ni kan detta bättre än oss. Vad skulle ni föreslå att vi använder för metodik, så att vi får ut de effekter vi vill nå?". "Då skulle vi föreslå att vi använder oss av en mer agil metodik och jobbar på det här sättet ". Så det beror också på vad det är för typ av kund och vad dem har för mognad och vilken typ projekt, typ av system, typ av plattform. Så att de är lite det jag menar, det är inte svart och vit utan mer en gråzon liksom. Det beror på. Men vi argumenterar alltid – om förutsättningarna finns eller tror att det finns förutsättningar – för att driva projekt på det här [agila] sättet.
4.35	**Vad är det då inom de här agila metoderna… Är det någon metod du förespråkar mer? Scrum, XP, Kanban etcetera? Eller du kombinerar hellre Scrum och Kanban?**
4.36	Ja, det skulle jag säga.
4.37	**För vi har läst på lite innan och det är tydligen vanligt med att kombinera Scrum med XP.**
4.38	Ja, det tror jag beror på oss som företag, XP – alltså par-programmering och alla dem delarna är ju… När du driver väldigt mycket systemutveckling från grunden; du kodar. Vi gör ju ganska lite sånt. Ta ett företag som bara kodar; företag där de är riktigt duktiga systemutvecklare, riktigt grymma och de hjälper företag att utveckla custom-applikationer från grunden, "blankt papper" liksom. Där använder de säkert XP – nu gissar jag bara men – och har software craftmanship och alla dem här komponenterna när de utvecklar system för att de ska kunna leverera kvalitet, leverera robusta IT-system. Vi gör ju det ganska sällan från grunden, eller aldrig från grunden skulle jag säga för att vi använder oss av dem här plattformarna. När vi utvecklar e-handelslösningar till kunder, då använder vi en produkt som heter Sitecore och utgår från den och konFigurerar och anpassar den. Så det blir väldigt lite att vi sitter och hackar .NET liksom eller sitter och hackar Java eller vad det nu är för någonting. Så jag tror det beror lite på vad vårt företag är för typ av konsultbolag och vad det är för typ av system, produkt och så vidare som vi har med i våra projekt och våra leveranser än andra företag, när dem driver sina projekt, exempelvis deras DNA är att de är väldigt duktiga tekniska konsulter, kodar,

	informationsarkitekter. Vi är inte riktigt den typen av konsultbolag. Ni skulle nästan haft en mot-pool till oss, för jag kan bara berätta efter vårt perspektiv och det är inte hela sanningen.
4.39	**Nu har du väldigt stor erfarenhet och vi har bett våra andra intervjupersoner guida oss igenom ett projekt. Genom de olika faserna planering, genomförande, leverans och uppföljning. Skulle du kunna genom hur ett projekt på Acando kan gå till?**
4.40	Ja, absolut. När vi kör… hmm, hur ska jag förklara det… När en kund kommer till oss och har ett problem, en utmaning, de vill göra någonting. Då jobbar man genom att titta på: kunden har någon form av nuläge (as-is) ”här står vi idag och det kan va processer, IT-system, verksamhet allt sådant. Så här ser det ut. Här vill vi vara (to-be), vi vill dit. Vi vill förändra vår verksamhet, ha ett nytt…” Ta ett företag som håller på att köra en galen massa projekt, till exempel e-handel. De ska dubbla sin omsättning 2020, de hinner inte etablera så många varuhus så att dem fixar det. “Vi behöver gå e-handelsspåret”. [Då] får det en påverkan på hela deras logistik. Jag pratar om att du ska kunna sitta på Sri Lanka där det inte finns några varuhus och ska kunna beställa bokhyllan ”Billy”. Det är inte bara till att ha en webbsida där man kan beställa bokhyllan Billy, utan “Var fasen ska jag ha centrallagerpunkten om jag ska leverera på tre dagar i Sri lanka? Hur ska vi lösa det?” Mm, då börjar vi komma överens om först, att det här to-be inte är ett fluffigt moln, utan vi behöver boxa in det. “Det är det här ni menar med to-be”, kravspecifikationen. Ramverket kring kravspecifikationen iallafall. Och så tar vi fram ett avtal och det kostar en summa pengar. Sen startar vi ett projekt. Och då är det egentligen, om jag tar Vattenfall nu, som är mest relevant utifrån det här perspektivet. Då börjar man någonstans i en kravfas, att man börjar detaljera vad det är kunden vill ha. ”Vad är det för process eller för processer som det här systemet ska stötta? Vad ska du ha för funktionella krav och icke funktionella krav?” Någonstans måste vi ta fram det så att det är tydligt. Även vad ett ”här” innebär, för ”om vi ska göras det här projektet utifrån dem här kraven; vad innebär det för påverkan på kundens organisation? Vad innebär det för påverkan på vad kunden behöver?”. Det ska redan ha tagits fram i avtalsdelen, men ”vad innebär det att kunden måste möta upp med det här projektet i form av resurser, i form av hårdvara, licenser allt sånt där?” När man går in därefter, då finns det tollgates och milestones som finns i projekt metodiker. Vi kan ju komma fram till att här, “Shit, det där fungerar ju inte, vi har kommit fram till att det här är mycket mer komplext än vad vi trodde.”, speciellt om det är stora projekt, en av anledningarna till att just Vattenfallsprojekt

	är lite jobbiga att jobba med. Men om allt går bra och vi kommer fram till att vi är överens om att det här funkar, då går man in i någon form av designfas. Där vi börjar ta upp, och de man ska göra, det vi vill göra, man gör någon from av proof of concept. Att man tar en del mängd av dem här kraven och designar upp det här systemet, hur det skulle kunna se ut för en avgränsad del av verksamheten eller av systemet. "Så här kommer det att se ut kära kund", vi ska implementera ett nytt CRM system på ett företag mm, då tar vi ut en mindre målgrupp för vi kan inte ha för stor påverkan, så vi börjar med säljkontoret i England, så kör vi ett prof of concept och förankrar det här med dem, "aa, men den här designen, så här kommer gränssnittet se ut, så här ser de funktionella och icke funktionella kraven, det här kan ni göra, så här ser det här systemet ut". Ja, bra. Det verkar funka liksom, check på den. Därefter går man in i någon form av bygg- eller realiseringsfas, där vi bygger det här systemet baserat på det där, men vi bygger på med flera av dem här processerna för de funktionella och icke funktionella kraven. Någon form av test liksom. Och därefter – beroende på hur stort det här projektet, IT systemet och allt sådant är – så har man någon form av ”go live”, men det handlar om hur vi rullar ut det här i en organisation. ”Börjar vi med England först? Eller Tyskland först? Eller kör vi allt på en gång? Eller hur gör vi liksom?” Parallellt med det här så har vi från vår sida någon form av support på det här. Det kanske visar sig att när vi rullar ut det här första gången och lägger på alla transaktionsvolymer, då, då smäller det. Det klarar inte det här, nej då behöver vi gå tillbaka och kanske göra justeringar på det där innan vi går igång med nästa land eller region.
4.41	**Brukar ni kunna gå tillbaka i avtalet och ändra då? Säg att det är inte som ni trodde det skulle vara från början, ”det kommer kosta såhär mycket istället för det var mycket mer som…”**
4.42	Mm, då skulle många kunder säga såhär: "Mm, tack. Det var roligt att jobba med er."
4.43	**Krävs det att man är väldigt noggrann, har genomgående kontroll vid avtalet?**
4.44	Ja, vi sätter upp… ”Det här kan kosta 500 miljoner, fast pris”; vissa kunder vill ha fast pris. Vilket är en utmaning när man kör Scrum, man kan inte sätta fast pris på Scrum.
4.45	**Hur gör ni då?**
4.46	Då sätter vi dit målpris; ”gör vi det på kortare tid, då kan [vi] få en uppsida i det. Tar det längre tid så tar vi en del av risken”, så det är olika affärsmodeller, olika typer av avtal, det finns många olika typer av avtal. Men det man jobbar i när man pratar avtal då, det de här sakerna, kraven och sådana saker. Men också, det blir ju då assumptions och pre-requisites;

	”För att det här ska gälla, för att det här kommer kosta 500 miljoner, då har vi gjort dessa antaganden och dom här förutsättningarna måste vara uppfyllda. Ni – kära kund – har som ansvar att leverera det här till oss; ni måste beskriva det för oss, på detaljnivå, det har vi antagit, och ni ska leverera det vid det datumet, och till den här kvalitén – annars håller inte avtalet. Vi har också förutsatt att ni har hårdvara, mjukvara, ni har fixat licenser med Microsoft eller SAP eller någonting. Och det skall finnas, där vid den tidpunkten enligt det här formatet, annars gäller inte avtalet.”
4.47	**I vilken utsträckning hamnar ni där ni planerat att hamna, angående pris?**
4.48	Ju större och komplexare ett projekt… SAP-projekt är ett jättebra exempel; det här avtalet är inte på tio sidor, det är kanske på 800 sidor. ”Har kunden läst 800 sidor? Nej, det kanske de inte har”, och så kan de säga, “Med vad fasen, det där har vi inte kommit överens om!”. Då säger vi “Jo, det står här på sida 789, kära kund, font 8, kursivt, här står det.” Någonstans när man går in i de här stora projekten, tar man varandra i hand och så är det partnerskap de kommande 3-5 åren. Det är inget roligt att sitta i de där diskussionerna men det är klart det måste finnas någon form av flexibilitet; men det här är väldigt komplext, det innebär att en avtalsprocess kan ta 12 månader, det kan också ta 3 möten, avtalet är på 10 sidor. Så kör man med sprintar, och bommar vi så är det inte 200 miljoner vi bommar, utan kanske 40 000 och det löser man. Så det är grader av…
4.49	**Jag tänkte om vi kan fortsätta lite på om vi nu ska placera ut roller.**
4.50	Utifrån vårt perspektiv?
4.51	**Ja.**
4.52	Hur ser en projektorganisation ut, i ett sådant här projekt? En projektledning, beroende på… Pratar vi om SAP-projekt sätter man upp det som ett program, då har vi programledare och olika delprojekt av olika huvudprojekt som i sin tur kan vara nedbrutna i projekt. När vi körde det här på ett annat företag var vi 600 man i projektorganisationen liksom. Men man har en projektledare. Vad är det mer för saker som är viktiga? Om vi tittar på våra olika faser; Business Analysis, och det är jag, en person som någonstans översätter verksamhetens krav, kundens krav till, vad innebär det här då vi ska bygga systemet?” Personer som måste förstå både verksamhet och IT. Vi har alltid en form av arkitekt, en person som förstår hur SharePoint är uppbyggt, hur kraven skall omsättas på bästa sätt när vi bygger funktioner till SharePoint. Under arkitekten så finns det olika typer av roller, det kan vara systemutvecklare som oftast sitter där och beroende på hur stora projekten är, tar du ett SAP projekt då har du ett delprojekt för integration. Det här SAP-systemet ska kanske integreras med 40 system, då

	måste vi ha personer som fattar vad är det för krav som gör det här med integration. Du har en integrationsarkitekt, som förstår plattformen. Du har personer som jobbar med systemutveckling i integrationsplattformen. Någonting som är viktigt att… Informationsmotivering. ”Hur ska man… det här systemet… vad är det för objekt? Vi har order, kund, avtal, en massa. ”Vad kan det vara för relation mellan order och kund, produkt och avtal?” liksom. Hur hela den här är sammankopplad, hur den ser ut. Projektledare, business analysis, arkitekt själva teknikgänget, dom jobbar med mycket PLDA arkitekt, kanske jobbar i krav och designenfasen, beroende på hur mycket den här kan och hur komplext det är. Går man vidare in i byggfasen kanske business analysis-rollen inte är lika relevant, då har man redan kommit fram till dessa delar. Då kommer vi in i bygg, så vill vi också börjartesta det här systemet, kanske redan såklart i proof-of-concept. Då gör vi liksom en en miniversion av det här slutgiltiga projektet; då bygger vi proof-of-concept också. Det som är viktigt också, som vi vill få in är change management, förändringsleden, för att börja öka medveten hos kunden, för det här en påverkan på den enskilda individen och organisationen och delar att börja jobba med change management redan där, men även här. Hur vi förankrar det här, det är två viktiga delar, det är kommunikation och utbildning.
4.53	**Är det en ensam roll som har hand om det?**

4.54	Det är ett team, av personer. Är det fyra personer som behöver utbildas på kommunikationsavdelningen, för att vi ska installera nytt intranät eller är det en hel säljorganisation med 700 personer: ”Ja, hur lägger man ut utbildningen på det – olika språk, olika tidzoner, olika roll out plan och så vidare?” Har ni hört talas om Stakeholders management? Det finns ofta någon form av sponsor av det här projektet; den här personen… Om du ska rulla ut ett nytt CRM-system, då är det säljchefen som är ytterst sponsor för det här. Men han har säljchefer, i olika delar i regionen, eller olika delar i världen, dem måste vi förankra här, jobba med, de är som stakeholders, De som har stake i de här projekten, om det skiter sig kommer de få smisk på fingrarna liksom. De personerna måste projektledarna och change management prata med kontinuerligt att de är med på tåget. Sen också, utbildningen kan vara i flera dimensioner, oftast försöker man få ambassadörer som man ibland kallar för champions i företaget, så det inte är konsultfirmor som står för det. Att också de är med på tåget, dom tror på det här, dom vill det här ska göras. Dem i sin tur utbildar vi, och dem i sin tur utbildar sina kollegor. Eller processer eller IT systemet, Onlineträning, E-learning eller vad det är för saker vi jobbar med här och det kan vara olika beroende på vad vi jobbar med för projekt. Sen när vi kommer bort hit, då har man avvecklat den här organisationen, och kliver in i någon form av support. Sen kan det såklart vara olika personer som varit med i projektet, [Business Analyst]-person, systemutvecklare eller vad det nu är för någonting som ingår i den supportorganisationen som har bra koll på vad det här projektet har levererat, vad systemet innefattar och så vidare. Ska du göra förändringar, tillägg, kompletteringar, men då gör vi det nu enligt vårt supportavtal; efter att vi driftsatt. Det är den typ av projekt som jag har jobbat i, och det är det vi gör otroligt mycket, det är för att de vi jobbar med.
4.55	**Kanske ska börja avrunda lite här. Om vi kollar tillbaks på Scrum och Kanban, vad ser du för styrkor kontra svagheter med den metodiken eller arbetssättet, du var lite inne på acceptans, och tydligheten.**
4.56	Fördelarna är: behöver inte sitta och ägna jättemycket tid åt krav på mutternivå, man kommer igång ganska snabbt och får man en acceptans hos kunden, ”så här ska vi göra”, kommer man igång ganska snabbt. Man får den här förankringen, man får ett ägarskap från kunden, om man tar den här produktägarrollen, är prioriteringar, är med och ser det här

	sprint efter sprint, växa fram och de personerna som är med i sprintarna, får man den här förankringen, mycket enklare att justera ett projekt, på tre veckor hinner du inte göra så jäkla mycket, när man hållit på i ett år. Du kan slänga tre veckors utveckling ifall man inte förstod varandra eller du gjorde något helt 180 grader åt det hållet. Tre veckors jobb var ju jobbigt men det var inte ett års jobb liksom. Sen så Kanban; jag gillar ju visualisering, få upp den där boarden, att få upp typ ”Det här gör vi nu; vad håller du på med?”, “Jag håller på med det här Large [L]-kravet, som innefattar dem där delarna. Vi kommer vara klara om 3 veckor”. “Ja okej, vad håller du där på med då?”, “Jag håller på med dem här två små [S] kraven”. När kundens sponsor sen kommer in i projektet och börjar “Vad fan har ni gjort, ni sitter 40 personer, får ni någonting gjort?”. Dem kan man visa tavlan, ”Ja det här håller vi på med”. Istället för att gå till ett stort avtalsdokument och visa att ”det här håller vi på med”. ”Om 4 dagar kommer vi har sprint-demo och då kommer du få se vad vi gjort – istället för 4 månader”. Det är fördelarna. Sen är det mycket beroende på projekt, kundkomplexitet, typ av system, det går inte att ersätta ett visst system med den metodiken, därför att det rent kostnadsmässigt inte funkar att ha två system parallellt levande, du ska svälta ut det gamla samt ersätta det med nya komponenter i ett nytt system, det måste samexistera i 18 månader, då måste mycket pengar läggas på integration för att hantera det här parallellt. Den här siffran blir väldigt hög, så för olika typer av projekt, olika typer av kunder, historik och arv allt det där vi pratade om. Kanske kommer finnas tekniska möjligheter som gör att man kan få till stora projekt att bli mycket lättare. Men som det ser ut idag är vi inte där än.
4.57	**Svagheter?**
4.58	Acceptansenen hos företagen, skulle jag säga kopplat till mognad och komplexiteten i stora och snåriga projekt.
4.59	**Är det svårt att kunna övertala kunderna, i och med att ni har enorm kunskap, det är inte så att dom vågar lita på er?**
4.60	I vissa fall, ibland måste, ibland är det styrt av plattformen.
4.61	**Är det något moment du tycker är mindre starka som andra metoder gör bättre, än Scrum och Kanban?**
4.62	Jag har inte jobbat med så mycket andra. Men någonstans skulle man kunna säga ”Kära kund, ni ska strunta i vad vi kör här (design - test), ’såhär ni stället ett krav och sen mäter ni oss när vi visar systemet’, lyckades vi eller inte på ’de här KPI:erna, kostnad, tid och

	kvalitet’”. Okej, vi kör agilt internt i vårt utvecklingsteam, vi kan ta det här stora klumpiga kravdokumentet och bryta ner det till sprintar, och köra det internt där kunden nödvändigtvis inte behöver vara involverad. Där vi inte har en produktägare som är extern hos kunden utan det är Scrum mastern som agerar produkt ägare också. Då kan vi köra det liksom. Men det beror på vad vi har för avtal med kunden och hur förhållningarna och förutsättningarna ser ut. Så det går att mixa lite.

Bilaga 11: Transkribering #5 – EG

Ohörbara ord/begrepp/meningsdelar är markerade med: [---]
Förtydliganden och ändringar som informanten själv gjort i efterhand är markerade med: { }

5.1	**To start out, can you please tell us a little about your current role or title within the company?**
5.2	My role or title in the company is [broad IT-responsibility]. So I [*redacted for anonymity*].
5.3	**Could you please tell us a bit more about your career; when you started out in IT and how you've come to where you are right now?**
5.4	Yes. I started out in IT about [25-30] years ago, as a programmer, and worked for [a few] years as {an assembler programmer}. And my education: I actually started out to be an electrician. And after that {I went to university to study Electrical Engineer, Datamatiker (Academy Profession degree in Computer Science) and BSc Computer Science & Engineering}. So that's my background. So for [25-30] years I've been working in IT, and I started for [a few years] at [a large automation and robotics company], doing all sorts of {automation and interface} programming, and sorts like that, which was very near to the CPU and stuff. At that point, we were using… I can't even remember the names any more… But we were using some very special methodology to assembly programming. So we were using stuff, but it wasn't very highly implemented; it was more like, you have a few boundaries of how you did it. And further into my career, I started programming in C. And, again, at {an electricity and plumbing company}, we didn't really do anything. It was very functional-oriented, and we were not using any full methodology. If we did, we didn't know what we were doing, actually, so we didn't know if we were using any methodology. The basic thing was that we had chatted and interviewed with the customer, and [---] then we just started doing it. And then – a little bit further out in time – I got very interested in methodologies; especially all the methodologies which were catching on about object-oriented programming and methodologies in how to do and how to use that. So I read up on a lot about that and started using OOAD [Object-Oriented Analysis and Design]. And I was actually also educating or teaching at [a technology institute]. So for us – I would say 20 years ago – I had a key interest in methodologies and started to study it very closely and read a lot about it, and use it and implement it into companies. So, I got really interested in the culture at companies and am now beginning to be an architect – enterprise architect, manager, and {quality manager} in these companies, and also to help implement {development methodologies like

	UP (Unified Process)}. But also development methodologies and, even further up than that, at maturity level-methodologies; I was part of implementing CMMI (Capability Maturity Model Integration) into [a large consulting company]. They wanted to endure, to be a CMMI level 3, so I was part of that implementation, and at that time I was [*redacted role*] at [an American IT consulting company]. So I know a lot about methodologies and I have used a lot myself, in big projects. What I am going to be talking about now is system development methodologies, meaning using Java, C++, C#, and for the last four years I've been working at EG, which is mostly an ERP company, working with Navision [Microsoft Dynamics NAV] and [Microsoft] Dynamics AX, and they {are a little bit different in} methodologies, a different approach to that. So, did I answer your question?
5.5	**Yes, definitely. Great! So, let's move on into the methodology part. Which… I think you mentioned it but I didn't really catch it; which development methods are you using – currently – in EG?**
5.6	Currently in EG, we're using a methodology called {One Delivery Method (ODM)} which follows Sure Step for Dynamics and our own EG Experience, which means we are using Sure Step as the backbone but also our own Experience, which means we are not really rewinding the methodology, but we are improving it; we are improving the {artifacts, templates and such}, the artifacts inside the methodology; the tools are being improved. So that's really what we are doing, when we're using that methodology. I don't know if you have any knowledge of Sure Step?
5.7	**Not that much; we've looked at it just a little bit, but we can dig deeper into that.**
5.8	Basically Sure Step is – when I'm talking dynamics – is meant for CIM systems, SharePoint systems, ERP-systems, like Navision and Dynamics. What is common for those systems are that they are frameworks, where you already have a lot of modules inside it. I'm not sure if you can say the same thing about Visual Studio with all the compiled libraries and that, but here you really have the framework, and if you take it you should actually be able to go out and install this framework at the customer, and it should work out-of-the-box. This is of course very [---] and [---]. So the method is basically building on the same steps as you know from other traditionally methodologies, where you have an analysis, a design, development, and some sort of deployment and operation phase. So we have the BA, the business analysis [phase], where we send out the management consultants to find out "Why do we need this?" They look at value chains, a very high perspective of the companies, [---], they look at processes as-is and processes to-be, and also the realization of if it will be possible. And in the next phase, which is the solution

	analysis, is where we go in and look at the systems from the outside; we try to build the conceptual model, really. In the design phase we look at the system from the inside-out and designing how it should be {implemented}. And upcoming – in development and so-forth – you already know what's going on. The difference from this and to system development methodology is there are not a lot of tools. If we just go high-level processes, which if compared to anything else would be use-cases, and then we discovered that sort of… In levels of maturity, at the high level, at level 1 for example – or level 5, as we call it – which would be the business analysis, very high-level processes, and we call it [---] where we describe it a bit further, and then we describe very component-oriented into the AX system. [---] and class descriptions, that stuff like that. You will see requirements, and you will also see technical descriptions in this database table, "how I should do that and that"; not a lot of artifacts or tools, compared to if you look at a unified process or tool class. What we mean to use that time on is fit-gap, "what is the fit to our system?" and "what is the gap for the system to this business?", and it's mainly the gap they describe. I can see that you've written something about strengths and weaknesses, and that would be that you tend to think that you understand – that you have the same thoughts about the fit as the customer has. And that could be a pitfall.
5.9	**So how do you complement with EG Experience? What parts of Microsoft Sure Step are not enough?**
5.10	We have tried to do an EG Experience; we say "Okay, we know that if you take that pitfall or go down that road and you think that you understand that fit exactly the same way as the customer, that will cost you money; really!". So we do a lot about that phase too, and describe the fit. So all you do is you make that fit-gap from the workshops that have been part of the business analysis, and then you have a new workshop, talking with the customer about business and how you understand the system. So it's a lot about intimacy, being very close to the customer, talking to the customer. "Do you have the same view of the fits and the gaps as we have?".
5.11	**Alright! So, for how long has EG used Sure Step combined with EG Experience?**
5.12	Well, about 80 to 90 % of our business is ERP systems. So… And it is Microsoft, but we also have SAP systems, but SAP has its own methodologies, called ASAP, so we use… Of course we have to be very close to Microsoft, because we use their products, and Sure Step is their methodology, which is also put into our framework in the future, into something called LCS (Life Cycle Service) where you will put all this Dynamics framework into, like AX or NAV, and inside that you will have this framework which will be using Sure Step as

	your own. And if you use that methodology together with all sorts of stuff, process frameworks like APQC, you will be able to actually let the system solve the code or get the code for you. So that will be a 4th generation tool. But you will still use all the methodology and still describe all the processes and stuff, but only describe it once, and then you can reuse it and press a button, and it will actually do all the code or all the conFiguration for you. Ultimately, hopefully we don't have to code at all – we can just conFigure. But we do code, we code in… In AX, it's something called X++. In the future, it will be Visual Studio, because Visual Studio will be built into AX.
5.13	**How much do you know about Scrum, personally? How much experience…**
5.14	I know a lot about Scrum!
5.15	**What would you say are the main differences between the way you work [at EG] and Scrum? Or what are the main similarities? We can start in either end, depending on what you prefer.**
5.16	There are similarities in the way we use… Or we could use… Not the whole package of Scrum, but we will do product backlogs, we will do sprints, we will plan the types of roles and responsibilities and stuff like that. So we use iterative or agile or whatever you would call it.
5.17	**But there's no official use of Scrum within the company?**
5.18	No, there's no official use of Scrum within the company as such, no.
5.19	**If you were to change or switch to Scrum, what would you miss? What are the main benefits of the methods that you use, that are not covered by Scrum?**
5.20	It depends on how you define Scrum, really. Because there are a lot of definitions of Scrum. So if your definition of Scrum… Do you have a contract with a fixed-price or what… How do you work with Scrum?
5.21	**I can't say I'm sure in specific details like that, to be honest. So no, I'm not sure about the contract and details like that. But I mean, there is kind of a fixed framework nowadays. Scrum has developed a lot in the last years, since it came out, and it basically covers every aspect of project management nowadays. But even though it actually does that, not very many companies seem to use it fully; in most companies it's used just like you do it, where you pick the best parts of it, and what we're trying to Figure out is "what parts do people pick from it, and why do they not pick the other parts; why do they keep doing things in their own way?".**
5.22	That would be that we use the agile part of Scrum; we actually do that a lot. We use product backlogs, we do price planning, we do sprint planning and have a sprint backlog, we do [---], sometimes we do daily Scrum meeting – but not all the time; we do test and stuff like

	that. We will still have {EG PMM} project management methodologies. When I talk about "EG development methodology", you must also understand that we have an EG project management methodology, which is the Project Management Toolbox, with change management, issue management, project planning, [contract *or* construct] management, resource management, communication management, and all that. So we have formal project teams and formal project management, and all that, inside of it. So what we use is… We have stopped calling it "Scrum" and we're calling it "more agile". And we use it inside. You can either do a Waterfall-model, or you can do the agile or the iterative model. The iterative model – for us – is the agile/Scrum/sprint/something, where we have picked the best stuff out for different methods. I think one of the reasons for why we are doing this is… It could be our own lack of knowledge or whatever, but one of the reasons we try… Or we would love to make projects by the hour, but that's not how the customers function! The customers want a fixed price. They want an as low price as possible, and we had {few projects} where we tried to be very Scrum or very agile, in six{cases}, and that really gave us some bad numbers [laugh]. Some <u>bad</u> numbers. So we decided to improve our {approach}, and I think that's why now… We now have made the approach where we will follow a Sure Step methodology. And of course there are different ways to get into it. So we would first have this business analysis; this is where we would ask the customers to tell us about his business as-is, we will look at the to-be thing and all that, and then we will have a {workshop} where we ask the customer "okay, is this how you want to go with this; do we understand your problems or your pains the right way?", and then we will go into sort of a pre-analysis, where we will do workshops with the customer and do a high-level fit-gap, and again we will go into a milestone and say our case to the customer, or again go over our case and say to the customer "is this… have we understood your business?". And therefrom we will then, if they say "yes", begin to do a callback-log, and now begin to investigate the… Do we have a lot of space left? I'm not sure if I'm answering your questions?
5.23	**Yeah, you are – definitely! And more to it as well. Would you say that estimating – that the estimation part of IT projects – is tough?**
5.24	I think this is the part of… If you look at IT; we talk about IT being the most difficult thing in the world to do. But actually the most difficult thing in the world to do… It's even more difficult than brain surgery. And we say that because it is intangible, what we are doing. You cannot really see it in your hands. At the same time, look at the IT industry – which I think is a paradox; a lot of us would still not use methodologies; there are still a lot of IT people or IT companies that do not even have a formal methodology for project

	management, development and stuff like that, which is implemented all within the company – which I think is a paradox, when we say that it is the most difficult thing in the world, to me. So we definitely need it – that's for sure. Look at innovation. I mean, we need to have the pre-analysis or the business analysis or something like that to actually give a reliable estimate. One of our critical success factors is to have a reliable estimate. If you look at the Chaos Report – I don't know if you know that, but – The Standish Group have something called a Chaos Report.
5.25	**No, I haven't heard of that, no.**
5.26	Okay, but you can look it up!
5.27	**Absolutely!**
5.28	They talk about "How many IT systems are failing or are outright stuck?". According to them, about 2/3 of all IT projects are either failing or not running as they should. One of the key things is reliable estimate; and resources available, amount of knowledge, experienced project managers, buy-in from the management, and all that. Estimation is something that is very important to us, and also to the customer – of course. We tend to think about the world [of IT systems] in three different ways: [1] We think about it in something we call "tailored solutions", which are solutions – even in ERP systems – where you have to build it for one customer, which means we have to customize a lot. There, we definitely have to be very sure on what we are actually going to make. So we need all this analysis stuff in the beginning, so we have a reliable estimate. [2] Then we have the same one, which we call {packaged}. And that is where we talk about business best-practice; like "Okay, I understand your business and I know how it is to be a sports shop – or whatever – I know exactly what you have in a sports shop or in a bakery, or whatever. Here you have it, I have made a box – a very closed off box – a package for you, which you can use in your bakery. It's a very fair price – a low price – because we have made it for 100 [customers], not only for one. We only have to make very small corrections in conFiguration or something." [3] And then we have the third part, which would be called "business-in-a-box". That is components we have built, which you can download from the Internet. Like a {service, cloud solution}. Of course, getting here, you have to have a product, which has a {low} price. But there's no risk in that. The highest risk is tailored. The only important thing is "Have you understood the customer correctly? Have you understood his pain areas? Have

	you remembered everything, to actually conFigure or code everything for this pain area – the system for this pain area?". So, this aspect is very important for us, because we like to get the customer making money!
5.29	**Yeah, of course! How would you say that you actually tackle the problem with estimation?**
5.30	We try to tackle it… Because we are back to realizing that you need to have those three, sort of like… Views of the world; tailored, packaged and business-in-a-box. Most of what we are doing is actually tailored. You can't just pick out a package and say "This is your bakery!" and you say "Yes, thank you very much! I don't want to pay too much for it, but I also want that leverage that is very special to me", and then it tends to {end} up as a tailored. And what we do is, we do this pre-analysis; and if we are {lucky}, we can sell the pre-analysis; and if we're not so {lucky}, we cannot sell it. Which means that sometimes, we will do it without getting paid. But the pre-analysis is typically something we would spend maximum a week on. Basically, a 3-5 days' workshop with the customer. And, you have to understand that we have been in the industry for many, many years, so we have already pre-built components or business best-practices, for, like, finance, or parts of finance. We have these boxes or components built, so we can put them together and say to the customer "Okay, is this what you want?". We built this high-level process description together with the customer during the workshop, and we will then describe it afterwards so we can develop a report, and estimate. And that estimate… Will typically be very close to what is the truth. So that's how we try to do it. It's all customer involvement, understanding the customer's business and what they want, really.
5.31	**Yeah. How well does it usually work out for you? I mean, how well do you estimate – generally – and actually match how long it takes and how much it costs?**
5.32	[This answer was modified by the interviewee (for clarification)] {I think we're like any other IT company. We have our bad projects, also, but mostly, we have good projects. If you look on the Internet, you can see that EG is doing quite well for itself. But of course, we also pay our price, for where we have estimated wrong. We do have estimation methods and automatic calculations of functions - but there's no silver bullet in this area. An important experience we have made is that it's very important to understand and design the system accordingly to the customers' demands, and it is very important that the customer and us have the same picture of fits (functions out of the box), but it is equally important to understand how the change management of the company in installing a new (or

	upgraded) system. Who does that? I mean, who is responsible for that? How we come out with a brand new system for "this company", which could end up in employees having a new job, or another job, or getting laid off, or… And stuff like that. So how is all this implemented into the company? If we do not take all of this into consideration from the beginning, we have a pitfall where the customer might have the view that it was included in the offer – but we have a different view, this is a possible estimate breaker.} And Scrum doesn't really say anything about that.
5.33	**No, not at all.**
5.34	And one of the benefits is the realization of this. Because, the manager at the company; the MD [Managing Director] at the company is interested in one thing; he's interested in the bottom-line. So "How does this benefit… What is the return-on-investment? How much will I earn in five weeks or years or whatever?". And Scrum doesn't really take that into consideration.
5.35	**What would you say are the key takeaways from the projects that have either failed completely or haven't worked out as you expected it to? What were the main things that went wrong? Do you see a pattern, for example?**
5.36	[This answer was modified by the interviewee (for clarification)] {Well, the other main thing is that we didn't get close enough to the customer. And, of course, that we have not been good enough to cover the pain arears at the beginning. Okay, sometimes we make mistakes, but sometimes we also get customers which are very, very difficult to handle. And you have to put the right team together to work with the customer. But I think a lot of the time, it has been because we have not taken the time to do the analysis thoroughly enough, and we haven't gotten close enough to the customer, and we haven't gotten the formal methodologies implemented the correct way, and stuff like that. And you use the resource which was on hand, instead of using the right resource – management-wise. Yes. I would say that are some of the big errors.}
5.37	**And I'm pretty confident that you're not alone in that aspect.**
5.38	I'm not alone – I know I'm not alone [laugh]!
5.39	**That's what we've found, generally, that estimation and analysis are probably the biggest problems that IT companies face; especially in the planning part and planning projects. And Scrum, for example, does not cover that part of the projects. There's not really any… There might be good methods for approaching it, but they're not very well known, at least. We're trying to find them, but we haven't been very successful in that aspect. And yeah, that might be one of the key problems; that if you cannot estimate, you cannot get the customer; or maybe you will get the customer on an**

	estimate that's wrong, and then it's not going to work out in the end anyways, because if you run out of time or run out of money… Then you're trapped, and you've spent all that time on something that you could have spent on something that actually would work.
5.40	Yeah. I don't know what the customers… Or how the customers are really in Sweden, but in Denmark they are talking a lot about price. They are very, very price-oriented. And…. Yeah. That makes IT companies sometimes go into a compromise of the estimate of the time you would use on test, and for organizational implementation, and stuff like that. And that will get you all sorts of trouble there.
5.41	**So whose responsibility is it to adapt the organization, would you say?**
5.42	Come again?
5.43	**Whose responsibility is it to adapt the organization? You mentioned that that's a problem, that no one really knows "Okay, who should educate the users and 'create' the success factors?"**
5.44	You have to make an agreement with the customer about who is actually having that responsibility. And if the customer tells you that "I'll pay for that", or [instead] "I'll do that myself" I think it's okay, you can do that. I think, if… If you want to [take care of it by yourself], I think we then should give him maybe five pages, telling him "Please remember this and this and this; be aware of those and those stops or pitfalls, or whatever". Because we know some things, which can really mess up the whole thing. And… If you want to have this system, it is important for the customer to get this information in the right way. And employees that are happy with this system. [---] knowledge about [---], where we have been taking this into consideration and where we have employees – the users of the system – isn't what you use it or very quickly gave an impression of "This system didn't work". Even if the system actually works good enough, it was all about us not being… or the company not being good enough to get the user involvement into this, and get them to understand how this was working or why; basically – sometimes – the whole success story in this. So it's very important!
5.45	**So how do you get that acceptance? Either way if it's your responsibility or the customer's responsibility, how do you get the users on board?**
5.46	We have a system… As I told you, we have this business analysis system, where we have management consulting, which… All the plans for that and take care of all the organizational change, by implementing this resource system. So, we have a whole procedure of documents and training and all sorts of stuff. If the customer wants us to take care of it, which is not all of the time. A lot of the time, we sell systems without all this,

	because first of all, the customer is very price-oriented, so they don't want us to send out management consulting, so they just want go directly to programming, really. So, I think it's a bit funny because, as I said before – in the beginning – when I started programming, we didn't use anything. Maybe we did, but it wasn't organized. Anyway, at that point, we chose to do it in a structured way, so that we could prove and document what we had done. Then we started to do this methodology, like Scrum and OOAD and all the other system methodologies. And [a recent customer] were really screaming at us to just go ahead. I mean, we can… Believe, in the ERP business, it is a bit like that. But AX and NAV systems are out-of-the-box systems, really. If I look at the package, it will tell you that it has got finance, it's got stock, it's got logistics, it's got BI and all that. So "Why can you just not record it into my computer and press play?". So that's also a thing in the ERP business, which we have to deal with. They don't really want us to use our time. And we're trying to educate the customer and say "We have to look at all this, we have to understand your business" and stuff like that. So that's… It's a bit turning around, as I said from the beginning. And NAV is very [---] actually, that you just call for a NAV consultant, which will go out to your company do stuff on your computer, and go away again. Even if you need, really… With no documentation. And in my world that is danger, danger, danger.
5.47	**Skipping out on the documentation?**
5.48	Mm [agrees].
5.49	**Let me just check my notes quickly.**
5.50	How is your thesis… I think I've read it, but something about Scrum?
5.51	**Yeah. The title is not completely fixed yet, but we're thinking about…. Well it's going to be in Swedish, but translated it will be something in line of "Why is Scrum not used…" or "When is it not used and why?". That's the main theme of the thesis. Because we feel that Scrum can be considered – when you talk or read about it – to be some universal solution, but that very, very few companies use if fully, or use at all, and we find that gap interesting. That "it's considered to be so good, but some don't use it, and "how come?".**
5.52	That's a very interesting question! I have thought a lot about it myself, because I know Scrum and I've been involved in it at [a global consulting firm] where we used Scrum a lot, but we didn't use the full Scrum, or we divided it into steps or made the method what we wanted it to be. And I think one of the things, first, was that [---] buzzword.
5.53	**Sorry?**

5.54	For a long time for me, Scrum was just a buzzword; one of the IT buzzwords. "Now it's Scrum, now it's eXtreme Programming, and now it's something else." And then Scrum really began to kick in and we began to look at it and try it. One of the findings we also had was that Scrum didn't have – as mentioned before – the first part; the analysis and estimation part. And we don't have all about this organizational change, as "How [do] we organize, how do we implement this into your organization?". Maybe the other methodologies don't have that either, so… And the next thing is, I think, at a point, when we looked at this, that the overview part… The possibilities to have a full overview over what you were going to do, was limited. Then of course it could be because we didn't use it the right way or weren't clever enough, and stuff like that. So really tended to… We would use Scrum for things like project backlog or sprint backlog, doing sprints and stuff, but we were doing a lot of stuff besides that, to make it hold. That was our path. Because… It looks very good in the theory, this is all backed, but when you came to the implementation, we had to do all sorts of stuff and all sorts of tricks. And we had certified Scrum masters and all that, which were in the project, so it's not that we didn't have the education on how to use it.
5.55	**Yeah, and that's so far what we've found as well. We've had a few interviews… And, well, we've had one interview with a company where they use Scrum, almost to 100 %, and they still add to it in the planning phase and in the delivery and kind-of "follow-up" phases; so before & after, basically. I think that summarizes that we've found so far, that maybe that's the key take-away or the key "problem" with Scrum; that it's lacking in the planning and the after-phase or the delivery phase, and also that it might be difficult to use it in ERP environments, because you might… I mean, if you for example would develop an app for a phone, there's no dependencies and stuff like that; you can create one new function, you can test it, and you can show it to people. But in big ERP environments, it can be very hard to do that, because everything is dependent on something else. SAP or NAV or any ERP system is already out there, but it's hard to develop a new function that's not dependent on something else in the system, and that has to be there already. So it can be difficult to use Scrum in very big, big programs. Or that's what we've found, anyway. You can still use agile quite well in different ways, in different constellations.**
5.56	And that's – as I've told you – we use it as… We don't choose between Waterfall or agile, we use both. We have described our model of methodologies, doing both.
5.57	**When you do both, what do you take away from the Waterfall that you find useable, that's not really…?**

5.58	We remove the split of design and development, so we design and develop in the same phase, really. But we have a very, very firm solution analysis before, where we have a very closed or conceptual description of these processes. We have a "start goal" where we actually get the confirmation from the customer. So when we go into the design phase, we're quite sure [about] what we are actually designing and developing, at the same time. A lot of the gaps could only be "conFigure that table with that data" or something like that. It's not always that we have to develop something. You can say that with ERP systems, we are mainly a little bit different, or we are a bit different to ordinary development, because then you have to develop from the bottom-up; you have to develop everything. Whereas, we can have a lot of gaps in our, which we [solve with] conFiguration; to put in data or… Yeah.
5.59	**Would you say it's more deployment than development?**
5.60	Well, we still call it development in the ERP world [laugh]. You will not do this in a live system. You will do all this conFiguration in a sandbox, and you will migrate data and stuff like that, to try out and test all that. You will not… In a live environment, that would not be very smart – you would be shot [laugh]!

Bilaga 12: Transkribering #6 – Qlik

6.1	**Can you please start out with your current title and your rolls?**
6.2	I'm [working with agile concepts] here at Qlik. I've been with Qlik for [a few] years now; as a consultant for [a few] years, and [within the last] year I decided to join Qlik. But in that roll I've been [working mainly with the use/implementation of *agile*]. Just background prior to that. I think I've been in the industry for [15-20] years, a few in the UK and a lot them in Sweden. I started as a developer, and then moving towards Scrum master or [agile]. It's sort of within [agile] I've been for about 10 years, on various different assignments in various different countries, mainly in Sweden, but again picking up different experiences from those companies. And then finding a home in Qlik, which I think is a beautiful place to be [role within agile]. So that's the background!
6.3	**So, let's move on to the methodology. What software development methods are you currently using at Qlik?**
6.4	We've got Scrum as a framework, and there are a number of teams that are practicing Kanban or elements of, like, "smaller-piece-flow". I'm not sure how disciplined they are in terms of the limits they put. And without a limit, it's not really Kanban. So it could well just be more of a "less boxed Scrum variant". In general we work in two-week iterations. In fact… Well we can come back to that. But it's like Scrum would be kind of the essence or the main thing that we work with. We might not – as I'll describe later – kind of use the rules, as in Scrum master or product owner, or we might have our way of doing that, that are suited to the scale that we work in. But the essence of it – the spirit – of what we're doing is these small iterations, and work in an agile fashion with it. So that would be the, kind of… I would say Scrum more so than Kanban, but there are elements of Kanban.
6.5	**Alright. So what elements do you take from Kanban then?**
6.6	I think… When it comes to prioritizing as a whole for the organization, there are elements of Kanban there, which we use to decide on "Alright, so what's the… How do we prioritize what we are doing?". Obviously, there's a sort of capacity that the organization has, so we've got to meet that capacity. There's only "so much" work that we can put into the organization. So you can look at it as an organization as a whole, and you can say that "There's an element of Kanban coming in; 'what's in progress and what's not in progress?' What are you choosing to do and what not to do?". Then there are teams on a lower level which will tend to use Kanban because of their work being a little more varied and unpredictable. So they are getting more order´, rather than kind of "Here's a set piece of work". So they have chosen that [Kanban] as a way of, kind of, addressing that variance in their in-flow. Whereas other teams have a little more stable in-flow, a little more

	predictable; then they can choose, you know, "This is the chunk of work we're going to do in the next two weeks", and try to work on that.
6.7	**So, just for the detail; how long have you been working with Scrum [at Qlik]? I think you mentioned it in the e-mail [prior to the interview]…**
6.8	Since about 2012; that's roughly when we started. One of the things is that we don't really… I'm not sure if it was a choice, I think perhaps it was, not to go and say "This is Scrum, and let's lock ourselves into that and not anything else". We refer to it, or approach to it, being agile. We're open to news – whatever methods, approaches or elements, which we feel can support or help us. So, there's a… If you look at it under the hood, there's a lot of influence from Scrum. I mean, when you look at it – Scrum as a whole… I think one of your questions originally was "Scrum's leading or perceived to be the best, why do so many people revere or steer away from it?"? And I think that if you look at the framework as a whole, the framework's kind of…. It goes back – it's ages old! It's "Plan, Do, Check, Act". The framework itself is pretty robust. I think it's in between those pieces that people choose to pull in different aspects. For example, "How do we do our planning? We've got our backlog or to-do list, we've got our product owner… But who is this holy product owner? How do we cope with that? Like, is there multiple people doing that product owner stuff? There's a to-do list of things to be done, but how do we prioritize that as a group, if we're sprint planning as a team…. Set up a kind of framework; that we need to pull in a certain amount of work, and we need to plan and design that work, and understand how we're going to approach the next iteration, the next two weeks." That leaves a lot up to the imagination, in terms of "How do we do that? How do we get a team together? How do we form as a team? How do we do our estimations together?". So there's a lot… The same goes through the cycle from the implementation phase and "How do we hold our modaily meetings?", to the review phase, to the retrospective phase. So the framework is there and I think that the framework is kind of… It's really hard to pick on the framework. I mean, I think the kind of framework really does stay… "Here we choose something to do. We plan it together as a team. We implement it, over a period of a couple of weeks. We check how that went, from a product perspective, and then we adapt, in terms of the team and the process". I mean, it's very hard to say that "Naah, that's some, not…". It's the little bits in between, that's like "How do we do these things?", that can be a sticking point for teams, and kind of be an issue for teams.
6.9	**Do you thinks it's too high-level, Scrum?**
6.10	No, I think it's a framework. So it sets up the structure, and then enables you to explore, experiment, and find what works for your organization, what works for you team – within

	that framework, which is very much "Plan, Do, Check, Act – and repeat". So I like it, that it's that kind of sense of "No, we get to ask the questions, on what suits this organization, this group of people, this team?". In comparison to where we were before the likes of Scrum and other agile methodologies, we used to have very, very high… RUP was an example; a fairly prescribed way of doing things, which became a distraction in itself – a focus in itself. There were so many things you could do and should do, and it got in the way of actually solving complex problems, as a team of smart people. I like the fact – personally – that "Here's your framework; find a way of working within the framework". The framework still gives you a sense of "Here's your fast points; we're planning a little bit, we're implementing a little bit, we're reviewing how we went on and we change how we did it, and now we're repeating again – having learned something!". So I think that's the essence of it. And some people might forget, the essence. And here's an example of it; what the problems that are perceived with Scrum. And that's when some people… They don't really get the framework and try to chop out parts of the framework. So if you take any of those moments away – the plan bit, the do bit, the check bit, and the act bit – if you take one of them away, then you're really missing a fundamental piece of the whole jigsaw [puzzle]. So you'll find a lot of teams that will remove the retrospective bit. Because – maybe – they've run it a number of times and it really hasn't given them the output or results that they wanted, or it's a ceremony that's not really being fulfilled, and rather than correcting it, they say "Let's skip it this time; maybe next time", so you end up doing plan, do, review, with no retrospective. So you've lost an attempt to continuously improve the way we do stuff. Or they don't review, so they just keep… It's almost like they become a factory or a machine, generating stuff; but we're not reviewing if what we set out to do made the mark. And "heaven forbid" [laugh] if they don't do something, because that's like the essence, and missing out a very key part. The do part also – even within that – has elements of that "Do we need to do daily meetings on a daily basis, or should we have it every other day?" and aspects which we have here at Qlik as well, which is "globalness"; how do you handle that? You look in the Scrum framework and the Scrum framework is not going to say "When you've got people around the world, in Boston or Iceland and different time zones, then 'this how you hold daily meetings'". So you've got to experiment again and be disciplined enough to keep it going. And planning is the same; again, you've got – this is the beauty – you've got a real different group of people all the time in your teams, that you work with; with different personalities that you want to tap into, and maximize the value that they bring to the table and their

	ability to work as a team. You may think that university provides you with all the groundwork and the experience and the practice through that, but you really kind of "get it" when you're in a team, trying to work as a team to get something done. And that takes practice. It takes discipline from the teams, and themselves, to contribute, to be here and to feel recognized. So there are aspects to this that are – I would say – the framework itself; but are essential for the framework to function.
6.11	**But are there certain parts of Scrum that you just don't use? That you don't see… Planning Poker is one example of something that we've heard some interviewees say that "No, we don't use that – we think it's too complicated" or "we don't see the benefit with it".**
6.12	There are some teams that use it, but again there's freedom to choose. If it serves the team, "go for it". Some teams use swim-lanes, just to… On a whiteboard, to do some stuff. Again, this is where it is really, really important that people don't get caught up on the tools, because then we will head into this very prescriptive "Oh, you did an estimation and you did it this way". It's like, you have to understand the fundamentals of why you're doing it in the first place. You're doing it to try to get a relative estimate, for example – in the terms of estimation. So when you know that you're looking for a relative estimate and there's various different techniques out there, then take one that works for you and your team. And what works in this team might not work in another team, and you've got to be prepared for that as well. So planning poker might very well work in this team; you might find that it's a giggle and that they're having the conversation that they need to have, and they have this swap of knowledge that they needed to have. But others might find that [standing] in front of a whiteboard, on a swim-lane with little post-it notes to represent the cards, they generate the same kind of conversation and it feels more fun for them to it that way, and more fun and effective to get it done. So always understand why you're doing it, and that gives you the freedom to explore different techniques to pull in, rather than being descriptive and say "Alright guys, across the organization, we're doing estimation and it's done this way!". We like to say more like, "In order for us to understand what we can tick off, and in order for us to plan our work as a team, we need to understand what our capacity is, and this is our method of doing it". So it's like having freedom and flexibility to let the team explore and find a way that works for them, knowing that in two weeks time, they'll get a change to go "Hey, you know what? Planning was a bit shaky this time. Maybe we should try a different way, a different approach? How about… You know, what this other team is down the corridor; why don't

	we bring one of those guys in and get some insight into that?". So, I think that's… With the freedom of the framework and also understanding the essence of why you're doing these different moments, I think is a key to success in a Scrum initiative, or the agile initiative.
6.13	**What would you say are the key benefits of working agile, compared to the Waterfall or traditional way?**
6.14	I think it's two aspects, two fundamentals: **[1]** First is the fact that we're living in a complex world; things are complicated; and we've got to find approaches that work together, to solve problems in that complex world. And so, it's quite unpredictable. And I think, in the old days… Waterfall is really designed for that sense of predictability; "We know what the problem is, we know how we're about to solve it, we can break it down into predictable chunks, and we can deliver on that". If it's a six month project, you can close the door and work for six months and come out the other side, saying "There you go". But we don't have the luxury of living in that world! And even if we had, I think I would still run it in a more agile manner. We live in complex world where we have a hypothesis, which needs to be tested. We need an approach that enables us to do that, in a very much more flexible, fast-feedback way. We've got to learn faster. Think of it as an organism – it's evolution. The faster you learn, the more chance you have of surviving; the more chance you have of absorbing and responding to different changes, which were either in your control or outside your control. The ability to change. You want to start learning that flexibility, to make that a part of how we work and how we live as a team, as a product, as an organization. So that sense of, "Alright, here's a task or a hypothesis that we want an answer to in the next couple of weeks." And after two weeks, having done something, we get to stand up and review and check "Did we get the information we were looking for? Did we… Are we heading in the right direction? Are the signals giving us the right, kind of, 'vibe'? Let's continue, or let's change our course!". That's one of the fundamentals that's so important to the fact that our organization – and I think that most organizations do – live in a very complex world. So I think we need responsiveness. **[2]** And another aspect is this essence of team, and people. Because that's what it's built on and that's what we pride ourselves on, this kind of culture where you power and enable the people. And Scrum – or agile – does that, in the scope of things. It really puts the focus on smart together. Give them a simple framework, a simple kind of "guide", that doesn't get in their way, that just gives enough structure to get things done together. And then get out of their way, and let them solve these complex problems. So it's given that "light touch"; put

	the reliance on "You guys have the skill and the ability to solve these complex problems". Whereas I think, in the older days, there was more… There was almost a sense of "I can do the thinking and I can give you guys some stuff to do, and I can manage you guys, and I can direct you guys to produce different stuff; and I'll know when you're about to be finished and when someone else can finish or tile the pieces together". And it just became… It seemed so… You're not tapping into the collective wisdom of that team and not enabling what could be even more powerful, than the way we're doing it at the moment; which is like saying "We're not perfect, but we're trying to create this environment where we really enable teams to take on problems; learn how to solve problems, and getting better at learning and solving problems". And that's where the mechanism gives you – if you think of the review and the retrospective – that change to breathe, stand up and look back, reflect, change anything that needs to be changed. And it's important, not to just reflect, but to actually do a committed effort to change something. And then to go and hold each other accountable to actually implement that change, going forward. I love the fact… I think it was Arthur Ashe, a famous tennis player, who had a very simple mantra. He said "Start where you are. Use what you have. Do what you can." And I love the essence of it. I think it appeals to me, and the agile way to work as well. You don't need a perfect scenario to… You don't need the perfect team or the perfect organization in order to get started. You've got two-week iterations, or maybe three of four weeks – or maybe even one week, depending on what the iteration events are – but we've got that opportunity, to go "Let's understand the problem, let's plan for it, let's execute something, and review and look back on how that went; then make sure the next iteration is a little bit better than this one". And if you think about it, that's a beautiful cycle to get into, which means you can start anytime and anywhere. Do what you can and start evolving towards that. There's another reality, which I think ties into the complexity, which is the fact that – especially in large organizations, but I'm sure it happens in small organizations too – that people come and go. So whatever effort you make as an organization has to be sustainable. You're investing in your people. So it has to evolve and it has to sustain. And I've found over the years that the agile way, with small iterations, these fast feedback-loops, these fast learning-cycles, don't just work for the product, in terms of us trying to solve complex problems and deliver products that are great for the market, but they are also brilliant in the sense that teams change in dynamic, and people will come and people will go. We have new consultants, we have new hires. People leave and go on to other work, and you want that essence that we're working on to survive and to sustain, and not be just like "back to square

	one again" every time someone leaves or someone comes. So you want to be absorbent of those... Absorbent of change. Change is not just in the product or in the customer demands, but also in the organization as a whole. And I've found over the years that this setup accelerates someone's ability to come in and be productive quicker, as well. Which is of course great for the survival of the organization – the survival of the organism. So I think that's two fundamentals: [1] The unpredictable and complex world we live in, and [2] the fact that it is very much based on people – skilled people – and gives them the framework, in order for them to grow and feel that they are making a difference and always be learning and have that cycle of learning. I think that's the things… For me to make standard, from where it was before, since they are the important parts that people should take with them.
6.15	**So what are the limitations of agile?**
6.16	The limitations of agile – as opposed to… I means it's… Again – if you…
6.17	**…compared to the traditional way of doing things.**
6.18	This… This is difficult. Because I think that things that are stumbling blocks for agile or for us, are not related to the method itself; they are not related to the process. They are related to skill, they are related to learning, they are related to the product. <u>They</u> are kind of blockers or the impediments. It's not so much the spirit or the principles. It's hard to knock them when you're reading them. You go "Yeah, yeah, I get that, I get that" and if you look at the framework, say for example the Scrum framework. The thing that is complex is that "Okay, I get it for a small team. How do we make this work when we've got 13 teams? How do we handle that situation?". And I think that's kind of diagnostic of… Whether it's an agile setup or a non-agile setup, it's something we've got to deal with. Skill, global, changes in personnel, product architecture – which grows in time… How do we keep that agile also? A process that doesn't become stuck in its way and that doesn't become stuck in its way and become rock solid, has to be agile and responsive to the organization. These are the kind of aspects that I think are the tricky bits. You can look at agile and – in fairness to your question – you can look at agile and say, you know, "I get it!" [positive]. You have this guy or gal called an agile coach, there's a product owner who can prioritize the list, you can get some help from the team, this Scrum master role is in there to guide the team to the goal. You can read it and go "Okay, I get it". And you can still play those roles very differently. Like, if you don't get the spirit of what you're trying to do … I have seen a lot of Scrum masters and agile coaches who perhaps get the agile bit, but who don't get the coaching bit – or vice versa. And product owners who can run the role of being dictator, like "I decide guys! This is 'my way or the highway'", but in a way that is not growing the team or

	enabling the team. So there are ways that you can still play the game and use the framework, but you can read it completely wrong, or in a way that is detrimental and a way that doesn't grow; it kind of either holds it at the same level or actually goes against itself. You can kill a team, in terms of not enabling them – "clip their wings" – and sort of box them. You can really… There's nothing in the framework that says that "This is how you grow a team. This is how to get them in the greatest and best environment possible". That's… Not a "dark art" but definitely something that has to be learnt, and you have to be willing to step in and "Alright, this may not be the role I was in before – this is a different role, and I've got to learn some stuff in here. I better be vulnerable in this area over here, and I've got to create this environment that enables this type of learning to happen – both in the product and process sides, and in the people"; these are the key elements within an organization. Given a purpose, it's the product, the process, and the people. And we've got to be able and be man or woman enough to learn on all fronts, and to step into those roles and say "How can I be the best in this role? How can I help this unit or team to grow, learn, and be more productive?". And I don't mean productive… Productive often has a very negative connotation and it shouldn't have to be [that way]. It's sort of like, "that it's easier to get things done". So I think that's an important aspect as well, that you're prepared to come in and learn and to lead by example; be prepared to be vulnerable with the team, and grow and learn together. That should just exude from you – that this is what it's all about. That people feel… That you create a safe environment for the teams to grow. Another aspect, or a weakness perhaps… Again, it's not against agile itself, but it's against the larger organizations that… You've got to remember that a team is a very important unit in agile. It's a unit that you like to see grow and become more productive, and really sustain and enjoy building that. But as an organization, there's multiple teams. And you can very quickly create "silo'd" organizations where the team unit was defined by that small group of people, and it protects itself. There's almost some stuff in there, in the agile literature, about Scrum masters protecting themselves from the outside… But you have to understand that there's – in a large organization – a larger team, kind of like… Not really a "Scrum of Scrums" but there's – without even mentioning the Scrum names – you're more than just that team that you're working for; you're part of the organization as a whole. So you've got to remember that as well, as a whole. That "While I'm building a real solid team-feeling here, I'm also part of something bigger". And that's important, that people don't get locked-up in that; that this team is a silo from the rest of the organization. That's part of learning too.

6.19	**Would you say that you <u>could</u> work traditionally in any case, that it could be beneficial? Is that something that you could see?**
6.20	Would there be an opportunity to work traditionally, or would there be…?
6.21	**…when it [working traditionally] could actually work better than working agile?**
6.22	Well, yeah… I mean, clearly, if the setup is right, if what you're working on is totally predictable, not subject to change, and it's got… Even complicated works in that predictable environment. So, yeah, go ahead and work Waterfall – absolutely. You can do that. But I still think that my tendency – my bias – would be to perhaps play a more… To challenge those kinds of constraints that we felt that we had at the beginning, and to be a little more prepared – just in case – by running a more responsive approach, and getting that demo'ing and that kind of insight. So I think that even if someone was to try to convince me that "Well, [interviewee's name], the basis for starting our… Well it's predictable, it's a no-brainer!", I would still be very tempted to say that "You know what? I think we can still get further faster with more value by taking a more agile approach". Because I think one of the fundamentals that is important is that backlog of stuff, the to-do list of your product or your project that you're working on. There's still a lot of hypothesizes in there and a lot of assumptions in there, which often aren't perceived as hypothesizes or assumptions; they are perceived as the truth. It's a problem that really exists out there and we need to solve it – we need to solve it completely. If you take that mindset, then there might be overkill to go into small iterations and just small increments of the product. You might say that "This is the problem". But if you take a more… "Well, what's in that backlog is just a list of the prioritized questions, prioritized hypothesizes. And of all these, this is the top one – the one that I want to get feedback on." And if you treat it as a hypothesis – an assumption which needs an answer – then you can create something, an increment of the product, which serves as an experiment for getting that information you need. Suddenly you start to again – and that ties into that whole "we're learning" – we're learning what the product or customer needs are; right there trying to fulfil that cycle, and get as much out of it as possible. And I just find it very hard, to in this complex environment that we live in… Not to see that as an advantage – to go in there, almost humbly, and go "We <u>think</u> this might be what's needed" and then get that real feedback that says "Yay" or "Ney". There's something I think is a real power in doing that. Again, as a product, as a process, and as a team – just to get into that mantra. Everything is a bit agile, a little bit more responsive, a little bit more, to get that information that we need. So, yeah, I think there are times that – when the basis is there – to run it a bit more

<table>
<tr><td></td><td>“Waterfally”, sure. But my bias would definitely go and say that “I’ll take an agile approach to it”.</td></tr>
<tr><td>6.23</td><td>Just to summarize that; what would you say are the key benefits of working as a Waterfall or whatever you want to call it? I mean, there has to be some strengths in it, because some people still choose that above agile, in some projects.</td></tr>
<tr><td>6.24</td><td>Well, I… [silence]</td></tr>
<tr><td>6.25</td><td>Is it simplicity? We have touched on that; that if you have a very fixed problem, and you might even know how to solve it.</td></tr>
<tr><td>6.26</td><td>Mmm… [silence]</td></tr>
<tr><td>6.27</td><td>For example, something we’ve come across is that in ERP implementations – where you already have the system and everything – where you can’t implement one part at a time, and you can’t… “pick <u>one</u> department”, because you have to have data from other departments to make it work, and in that case, it might be beneficial to work Waterfallish.</td></tr>
<tr><td>6.28</td><td>No, I would still challenge the ability to get that red thread through even something like that. I would see it as a challenge to bring life into the… Not just into the little building blocks, but also to get that talking end-to-end as quickly as possible. I’m not sure! I’m not sure if I could go further and kind of… Defending – or not defending – but kind of finding that “Oh, this would be perfect for a more Waterfall approach”. As I say it, I’m almost sold in the… I mean, if you look at Mother Nature – who has been kicking around for longer than we have, and who always seems to be surviving, despite what our best efforts are to do anything or the other. That’s how she gets by. It’s kind of like, probing, learning, responding, and she’s at the mercy of a lot of things that we do, and has no control over everything we do, but seems to be able to respond, and… I think that and the essence of an organism – finding out what works in a certain context – for me is the paradigm or model that we will follow – always. Yeah, and I’ll stick by that! It’s a bit strong, but I will stick by that and always try to follow that approach. Well… Unless it’s <u>really</u> simple [laugh]… If it’s really simple and predictable, well… Okay! There’s a very useful model called Canefyn [pronounced /kʌnɨvɪn/]. The essence of it is that there are the simple, the complicated, the complex, and the chaotic. [Drawn on a whiteboard as follows:]
<table>
<tr><td>The Complex</td><td>The Complicated</td></tr>
<tr><td>The Chaotic</td><td>The Simple</td></tr>
</table>
[The Canefyn model]
In <u>The Simple</u> is where you put your best-practice. This is where “I can just give you the recipe and you can follow that – check-list by check-list”. So there is a best-practice in here.</td></tr>
</table>

	The Complicated might seem like the Complex, but it's really just… If you really took a little more time, there's experts in here who can get it, who can unravel it and see that "A follows B". It's complicated and hard to understand, but it's still quite predictable. In here, it's The Complex. There are so many variables in there and so much knock-on effects, that… You know, "you touch this and it's systemic" and… In here is where I feel that, this is my world – in The Complex. And it's in here that you probe, to try to understand. Things that were once complex can become complicated, once you've really probed it and think that you have the model of how it works. So, in The Complicated you've got that sense that there's an expert. There may not be a "best way" of doing things, but there's maybe "good ways" of doing things. But over here [in The Complex] it's like you're probing to understand. And I think that a lot of our product development, that we do, is in this [The Complex] domain. Which is why the agile, the – almost kind of – "testing the hypothesis", experimental, iterations, learning with faster feedback, is suitable to this environment. And that's why I kind of… This is my bias. So I've got to be very careful, because this is my bias. So "Are there moments when following a check-list is better? Yeah.". If it's a recipe you're following, then sure, "shoot". Even if it's complicated, there still might be "X" number of steps to take and make happen, or maybe an expert to rely on, like "How would he do it in this situation?". So yes, I might not be so naïve to say "I will always stand by it [working agile]!" but I certainly am biased [laugh]. But I think that might be useful for your research as well, Canefyn; it's a Welsh word. You should look it up, because it might give an element of structure to your thinking as well. It might be a useful framework to hang some of your thoughts over.
6.29	**So I would also like to just touch on estimation, because it's something we've come across to be difficult. Some say that up to 70-90 % of IT projects either don't make it on time or on budget, and we assume that estimation might be the problem. We've come across that in some cases, people can believe that working traditionally/as a Waterfall helps with the estimation, because you know when you start what you're going to make; while working agile… They might be doing it the wrong way – but anyways – that if you work agile, you never really know what it's going to cost, and that you might add so much during your travel that it's difficult to estimate. What's your take on that?**

6.30	I would wholly disagree [laugh]! First of all, I think it's a false argument that everything is known beforehand. It's also a false argument that… One of the Waterfall or… If we take a Waterfall approach to it has been that we're following the plan and that success is defined as "Did we reach the plan? We planned to do 'this', did we reach that? Yeah, we did – 100 %, or 90 %, or 80 %, or 110 %. But we kept to the plan and we stuck to the plan – hallelujah! Your guys are great! That's fantastic". Whereas the agile says "Let's take another approach to that, and focus on value. We'll learn as we go along, what's more valuable than something else". So you run the risk that – let's say it's a six-month project – when you reach the end of those six months, that you are not where you predicted that you would be at the beginning of those six months. But the hope is that you're in a more valuable position than you were. So if you're measuring the success of your project on ticking boxes against the plan; "Did you stick to the plan?" then "Agile? Ouch! No, it's nothing good!". "You said you were going to be there but you're over here. What's up with that? Clearly, you guys have got some learning to do, in terms of planning". But "being over on that other mountain was the most valuable place to be, so that's the place we should be at. We should be honored for being or doing the most valuable thing for the company, for the organization, for the customer, for the user – that's what we 'vote' after". People can get very addicted to – and frozen by – following the plan. And it's about being… Having to have that – it's not even fingertip feeling – it's about reacting, responding, and focusing on value. That's the key, that's the measure! If you're going to measure me on my ability to stick to the plan, then you'll compare, kind of, "Well [interviewee's name], that's not where you're supposed to be; you're supposed to be over here". And that's a bad measure. And that's one of the fundamentals of agility, whereas the Waterfall is about being predictable, and the agility is about being responsive and being value-focused. Success in the predictable world was that "We followed the plan, to 100 % – woho!". Being in the agile world is "We responded based on value. We responded based on feedback. And in that same period of time (let's say it was six months), we've gotten to a more valuable position, or we've increased the odd of getting to a more valuable position than you have with that other project [which followed the Waterfall]". I challenge what's defined as success. In terms of timing, budget and estimates; if I have a group of six people dedicated for six months – let's keep it simple and say that there's just one team – working two-week iterations… I mean that cost is quite predictable. The thing that is less predictable is "Where will they end up?". But I know that where

	they'll end up is of more value to the customer – because we've got him/her/them in the loop, in every two-week iteration, or whatever it might be – to get us that feedback, and I'm always testing that "Is this what you wanted? Are we heading in the right direction? Are we getting that feedback? Are we ending up where we need to be?". And you kind of hope that it's in a different place, than where I first set out to think or believe that we would be. And then you might go "Oh, but that's crazy. It seems so chaotic! You guys are going to end up in a different place?" but it's a more valuable place. Every two weeks you get to see the pivots that we're making or the changes we're making, which you are part of. So it's not that we "lock the door" or "You told us where to go, so we'll lock the door and not let you in in another six months, when you expected us to be on this hill but we're over on that hill". It's not like that! Although that's how some people perceive it. It's like every two weeks. They're playing with both models, they're going like they play with their old paradigms and their old predicted Waterfall, looking at how they do it, and they're taking parts of that and arguing against the agile way of doing it. But here [in the agile way] we're always searching for feedback, or that learning and that responding. And I think there are other ways of measuring the team – not just based on value, but also on… If we choose for that team to go longer; say that "We loved where they got after those six months" – in both those concepts; both the predictable "Waterfally" and the more responsive, agile… I've [personally] done well and I believe that the team that has been growing and learning from each other – not just in terms of the product, but also in terms of the process and how they work as people; and they have been given that opportunity to every two weeks to learn and tweak and change and stuff – I think that that team has got a bigger chance of being sustainable six months down the line as well, than this team [the team following the Waterfall] that has been more, kind of, managed to stick to plans and "hold on to plans – regardless of what's happening outside – stick to the plan!". I would say that my gut feeling and experience tells me that this method – the agile or more responsive method – creates an environment where I can grow and sustain teams for a longer period of time, in a more healthy state than I've seen them be a part of [the team following the Waterfall]. I think that still… Estimates… You can still estimate and you can still give cost estimates to how much, and you can burn "X" number of… You can pull the plug in this project – the agile and responsive – as you get feedback and insights every couple of weeks. I've seen few where you cannot… Where you haven't been happy to play in that world and have a sense of "Alright, I get insight into this and I get control over this

	and I'm part of this" as opposed to "here", where… So, well, I don't know if I buy the whole [Waterfall]. There's an element of the unknown over here [in the agile] – from the predictable; there's changes to be made. Of course it feels fantastic to know that you've got this solid set of requirements and that that's what the team is going to work on. And you look at that solid set of requirements and you go "Wow, that's a six months project – we can do it in six months" but you'll learn so much when you're working through that and there will be so many things that are… Like changes that you've got to make and new learnings that come in, as you try to implement this – and then you've almost got to reinvent, if you want this to proceed and be predictable. You've almost got to reinvent these "checkpoint moments" where you check that everything is okay and… But it seems so still and stiff here. Whereas in the other world – the agile and responsive world – it's designed for that! It's designed to say that "Alright, okay, with your best understanding at the moment, that's your fixed bunch of requirements. And let's see – let's see! It's cool that it changes. We're heading with the same vision of where we want to get to, right from the start, but we're prepared to respond and change as new information comes to life." I think that fundamentally, that's a very sane way to approach complex situations; to be open to change – not fight it – and to respond to it when it comes. To side with things, based on the information available. Knowing that as the weeks go on, we'll have more information and we'll be able to make smarter and more informed decisions. We're open to it and we have a way of doing it. It's a mechanism, it's a framework. You know that when "In two weeks' time, we're going to have a demo". You don't think it's like "Oh no, when are we having our next meeting?" that it's kind of humbling. You know that in two weeks' time, you're going to have your eyeballs on it again and see how things are going. So, yeah! [laugh].
6.31	**But how do you put that in a contract? I mean, the value-based goal, instead of "We must deliver this and this and this".**
6.32	I'll give you an example – and it's not related, really, to Qlik. When I was consulting to Qlik – originally – my first contract with them was an agile contract. And I, as a consultant, said "I'll work in seven two-week iterations and at the end of each iteration… I'll work each iteration almost like it's agile; we'll plan together and we'll sit and work together, we'll review how things are going and we'll do a little retrospective on the things that need to change. We'll do that seven times; so 14 weeks. And while we're doing that, we'll be making results and I'll be feeding back. The demo is a perfect opportunity to feedback – not just to people who are involved in the initiative that you're working on, but to everyone

	who's around it." And so we created a contract based on that. You've still got a fixed cost – effectively – but you're getting paid per iteration, a certain amount of money. And in any stage they can say "You know what, [interviewee's name]? Not in this environment; it doesn't really work; your chemistry is not right" or "whatever". The opportunity to say "Let's terminate this thing, and let's do it early – for the value of both parties, vice versa!" I could have said this, that "You know what guys? You're not really ready for this and there's lots of things that have got to land, before we can go into this". So what it created was – for me and for them – a kind of partnership. We were in this together! We were working "this" and every two weeks they got an insight into what was going on. And we tweaked it and went on to the next two weeks, and tweaked it. So that was just an example, that's not necessarily related to product development, but it's related to how you can – even as a consultant, who believes in agile and fast-feedback learning – come into an assignment and go "Let's set it up like this!". Se we can totally build… You've got a team and they've got a fixed cost. There has to be something that varies. You've got the cost scope and time, someone of these has to give in – and often it's the scope. We've got "times our release points" and "times to market" and other areas around us, that need to work as well. We've got not just a development organization – we're not just R&D; we've got marketing, sales, delivery, finance, and so on – they're all around us and we can't work within that bubble. So you've got to be as transparent as you possibly can to the rest of the organization, so that we – as a whole – can work together.
6.33	**Well, isn't it the case that, usually, when a company wants – let's say an ERP system – implemented, they go out and look for someone and the price and the time for that, to get that implemented. Is that the problem, that they know exactly what they want?**
6.34	Well, again, that's a different scenario. "Here are the LEGO pieces, let's just integrate them; that's the project delivery." Whereas what we're doing is product development, which is a complex and creative, innovative adventure. Well, if you've got a shelf of tools – let's say you're putting a car together – then there's various car parts; just pick them off the shelf, put them together and… You've got a little line going and that's not the world we're working in. I would hazard to guess that even in an ERP-system you've still got customizations to make and there are still different ways to link these blocks together, which might be more problematic, and there is still data to get out that might be problematic. There are still elements of that, that can be complicated. And for them, maybe an agile approach around those complicated areas – getting that incremental feedback – isn't a bad approach. But if you're putting together something that's quite "off-the-shelf, plug-and-play", then it would

	make sense to do a more "plug-and-play approach" to it. But that's not where we [Qlik] are in. The world we're in is product development, which is more complicated than that. We've got multiple teams – even teams that are distributed across the globe – and we've got complex problems to solve. And not just so solve, but to explore and understand. That sets us in another paradigm, having to play a different game. And we have to find a system and a game that works for that. And we're in it for the long term, so you're in it also, like… I think this is an important play; that you're in it to sustain. So you're in it to evolve and – as I say – you're not just evolving the product – which is a great thing to evolve – but you're also evolving its people! You want this organization to… You're investing in all these people and you want these people to be around you – this is the family. You want them to grow. And then there's also the process; and the process is how you do stuff. Of course you want it to be as easy as possible, and for you to have the freedom to be yourself in this organization – at the same time as for me to be able to be myself in this organization, and for us as a team to be ourselves in this organization – and still feel connected to this. So you want this, kind of like, freedom – but with enough constraint; be it the framework or be it the walls or the organization as a whole – to give us that sense of "Yeah, we're part of something that's responsive, sustainable, evolving, seeking to be the most valuable thing it can possibly be – to the people that it's trying to serve. For me, that sets us up for this kind of paradigm where we've got to be agile, and we've got to be more responsive – we've got to be striving to be more responsive. I think that's another aspect that may be a good thing to take with you as well, that "Where we are today is…" I wouldn't say "a night-and-day difference to where we were three years ago" but it's certainly a transition and an evolution – an evolution across teams and across learning – and you've got to have this mindset that you're in it for the long term. You've got to survive – of course – while you're going through this evolution, but you've got to always be learning, absorbing this learning, spreading this learning; trying to get things done easier, trying to make sure that the things you are doing are providing value, and maximizing that amount of value that you do; and then you've got to be playing up for setting up for the next year, which is not just content of where you are, but also "playing the next chess pieces and the next moves", to make sure that you're here still in a or two or three year's time – thriving and evolving as you were previously. So I think that aspect – it's another dimension, but I think it's a very important dimension,

	that you've got to be responsive top-to-bottom; as a team, as a product, as an organization. You've got to have that mindset. And you'll trip up more than enough; you'll make mistakes and have to be responsive, and you'll fight change – even as [working with agile] there can be some times when you can go "Oh no!", and then you remember that "Wait a second, this is okay – we're learning. That was difficult and there's a better way of doing this, but at least we tried that way. And what did we learn from that?" and keep going. Just as a natural human being, there will be days when you will just go "Don't change anything!". The essence as a whole is that "Wow, it's so refreshing that we're not where we were yesterday or a year ago" and the hope that where we are today will not be where we are within a year's time or three years' time – that we will have evolved into something that suits the environment that we're part of. And I think that's the bit that I crave and that I want to create in my role as [within agile]. I want to help create an environment – together with my colleagues – that is responsive and creates that environment that never goes to sleep and just "Tick box – we've done it, we're agile and we're doing whatever the next thing is". It's almost like "Do we have the spirit for responsiveness? Are we agile in the way we're doing things? Are we still learning? Are we still having fun doing this? Are we still attracting, collecting and keeping the best minds? And are those minds sharper than they were a year ago, because they are challenged to learn. It should be an addictive environment – not like a cult – but an addictive environment where… "I'm still learning here, and I'm still curious, and I'm still getting my needs fulfilled within this environment. And we're still successful!" An important part as well is that – within these paradigms – what you do gets results. You can't live in isolation, you're part of a system. So your business and your organization has to get results and it has to maximize the value it provides. And I truly believe that – given this complex environment that we're in – the agile approach is the approach for us. There may be other organizations that are predictable and that convince themselves that they're "plug-and-play" and that they're just building… That they're in a factory and just putting things together. And for them you can say "Well, we'll go approach a more predictive approach for things", and that may work for them.
6.35	**I think that's basically everything we have time for. Thank you so much for your time – it's been great!**
6.36	***[Talk about our findings etcetera]***

6.37	[What kind of company you ask; for example whether they have their own product or are 3rd-party consultants] also has a slant on how people perceive things. I've noticed – even myself, as previously having been a consultant – that the space which you play in changes. When I was a consultant, you were in for maybe three months or six months or a year, and that's a pretty short time-frame for doing an organizational change, or to implant a more agile way of working. Whereas, when you're in the company, there's a longer game that you have to play. You don't have to break the team or "shoe-horn" or force something in; you can play a longer strategy and you can win small victories over "this" iteration. There's more of a "humane" approach to it, when you're in the company itself. It's a different approach, altogether.

Bilaga 13: Transkribering #7 – Tretton37

7.1	**Vad är det för metod ni använder er av just nu?**
7.2	Vi har ingen metod som ska funka på allt, det beror extremt mycket på, framförallt på vad kunden är mogen för. Man får skilja på när vi levererar grejer här inifrån, från kontoret där vi har total kontroll. När vi är hos kund varierar det väldigt mycket beroende på tid och hur mogen kunden är. Och det man kan sammanfatta, ute hos kunder är det väl ingen som kör ”Scrum by the book”, utan det har som ett ramverk och kikar man här internt så är agile vårt DNA, det är mer ett förhållningssätt för oss. Man kollar liksom på Dreyfus-skalan, vilken nivå inom ett område man är, så brukar man… När man kommer upp på de högra nivåerna brukar det bli en del av en själv; man reflekterar inte utan man rör sig så, det är lite så jag tror vi fungerar som bolag. Vi vet anledningen med agile, vi har det i oss, i vårt DNA och därför kan vi röra oss utan att behöva ha ett skyddsnät i form av en viss process. Utan det är mer hur vi attackerar vår dagliga interaktion här inne. Man kan säga att det är ett hopkok av lättrörliga metoder där egentligen huvudsaken är att vi jobbar mot externa kunden och kollar man på agila manifestet och customer collaboration. Vi tror att om inte kunden är med oss i det och vi blir inmålade i kontrakt och införhandlingsproblematik, så kommer vi aldrig kunna leverera bra mjukvara. Och framförallt inte inkrementellt leverera det, vi försöker få med kunden i inkrementella investeringar, att dom är med på det tänket, att dom verkligen ser totalt transparent ändå från införsäljning till leverans, vi sitter i samma båt, eller har samma mål. Alla kunder kanske vi inte kan ha den här relationen med, dem är kanske inte mogna, utan försöker ändå bedriva det. Vi gör egentligen redan från införsäljningen om vi ska leverera inifrån vårt kontor, om det total kontroll över leveransen. Så börjar vi redan i införsäljningen med att paketera det, ”Okej, ni vet inte riktigt vad ni ska ha”, vi har lite olika startup service, idégeneration, ta fram en backlog, eller två veckors workshop där vi tar fram, vad är viktigt för er, kvalitet, tid eller vad är det som driver just det här projektet, för det varierar så enormt. När kanske dom två veckorna är avslutade, har dom en artefakt dom kan välja att gå vidare med oss, till liksom nästa steg, utveckling osv eller annan leverantör. Så total transparens är med där, vi tar betalt för det här arbetet. När vi jobbar, vi är helt öppna med, ”går ni med oss så är det det här vi förväntar oss”, vi kommer kräva det, den kundrelationen. Det är inte alla kunder som förstår det, att man måste själv vara delaktig i ett mjukvaruutvecklingsprojekt för att det ska bli lyckosamt.

	Många tror fortfarande idag att om man skriver att på ett papper, en stor backlog så kommer det vara exakt samma förutsättningar om sex månader. Och det är ju bara att få dem att inse att så funkar det inte. Man kan inte veta allting. Mjukvaruutveckling är inte som att bygga ett hus; det är en hemsk metafor folk använder – att det är som att bygga ett hus – men det är långt ifrån det. Det som kan kännetecknа det är att i utveckling… Kunden är delaktig, är med och ska vara med och driva kraven, vi kan komma med idéer, men de ska de som driver från affärssidan. Sen är vi låsta i den iterationslängden, säg att den är två veckor, så är vi låsta för förändringar om det inte är något totalt akut, de har tänkt helt fel. Det handlar mycket om att försöka skapa självgående team, sen om dom väljer att implementera sin del av processen är upp till teamen. Det man kan säga är att agil utveckling utan XP-tekniken, eller andra olika sätt att hantera utvecklingen, är ganska värdelösa enligt mitt tycke. Det första man har är det här skyddsnätet i form av en testsvit eller en automatiserad deploy. Det är först när man har den biten och kedjan som man kan bli ett produktivt team, det är ändå det man är ute efter. Hur kan man leverera så mycket värde så snabbt som möjligt. Och, hur teamet väljer att implementera det, det är lite upp till dom, vi kan påverka givetvis och säga, okej, men vi kan väl ta kvalitetsarbetet till exempel, då tar vi kanske in några kodade testar i början av ett projekt, sen några veckor i iterationer, sen kan de trappa ner och låta utvecklarna ta ännu mer. Just att vi ska kunna röra oss så snabbt, vi pratar också mycket om att hela tiden stödja kunden, ifrågasätta vad dom gör, vissa kunder kan tycka att det är en grön knapp detta, varför är det grön den ska vara blå, de fokuserar på helt fel saker och dem kanske missar affärsnyttan för att dom inte är vana vid att jobba i mjukvaruutveckling, det är vår roll och skyldighet som leverantör att pusha dom i rätt riktning och få dom att inse att ”det är inget jätteviktigt just nu, det kan vi ta senare” flöden, affärsnyttan i det. Här inne har vi som sagt ingen tydlig process, mer än att vi vet vad som driver oss och vad som ska driva projektet. Och det är lite mer svårt att formalisera, vi har vissa steg som hur starta projekt och hur avsluta det, vad som krävs för att sätta igång ett projekt, vi kräver att alla projekt har en “definition av done”, den typen av enkla grejer. Vi andas och lever det så det är inget som behöver stå på en vägg, det är vad som drivs av projekt i sig och våra agila coacher som driver projektet; de kan ses som Scrum-master, fast har lite mer kundkontakt än en Scrum-master.

7.3	**Ni säger att ni inte har någon standardmetod, men det är fortfarande vissa delar ni använder, till exempel vissa delar av Scrum ni tycker ”den här ska vi använda”, ni bestämmer att ni ska ha möten, roller och sådant?**
7.4	Vi har morgonmöten, det är en klassiker. Är det små team, kanske två utvecklare är ett morgonmöte en diskussion, en transaktionskostnad. Är det team på fem-sex personer då behövs ett morgonmöte; det är lite projektstyrt, mycket med ceremonierna som kommer med Scrum är, kostar mycket av transaktionskostnader och distributionskostnad, och man får inte nyttan om man är ett väldigt litet team, då är det egentligen bara en ceremoni. För att man ska ha den, och mycket av den kommunikationen sköts ändå i dagliga arbetet, så helt onödigt. Men visst, planeringsmöten, sprint-planning eller vad man nu gör inför varje iteration. Backlog, och en produktägare, från kunden eller om det är vår agila coach som driver fram den med kunden, det varierar. Det vi vet är att acceptansen måste finnas inom vi påbörjar ett arbete om det inte är ren prototyping, får de är det helt fritt, då är det mer målet med vår Lite var ni vill vrida den här diskussionen, vill ni kika hur vi jobbar internt, eller ut mot kunden?
7.5	**Internt, så ni arbetar internt. Ni säger att ni kan variera ert arbete, blir det aldrig problem inom era team? Att ”hur är det vi arbetar nu igen?”. Du säger att ni alla lever på det här, och på det sättet kan förstå varandra väldigt bra, men är det aldrig så att det kan bli problem i och med att ni inte har strikta linjer på att ”så här gör vi, så här ska vi göra”?**

7.6	Jag tror att vi försöker vara ett bolag där vi har ganska mycket… Den stora nyttan är, agiliteten, självorganiserade team. ”Ni är kunskapsarbetare, ni vet bäst hur man styr upp ett arbete”. Sen är vissa konstellationer av team är mer mogna eller mer omogna än andra, då måste vi rikta det stödet mot de mer omogna, vi kan ha team med tre utvecklare helt självgående, allt dom behöver är att dom träffar kunden inför den här iterationen, ”nu har vi det”, sen så sköter de det mesta själva, sen ett annat team med två tre stycken som inte riktigt är vana att ha den synoriteten. Och det finns kanske inte någon som naturligt tar ledarrollen i teamen, då blir det att vi måste ge dom större processtöd, och då måste vi coacha dem mer och vara lite mer, alltså skedmata dem mer, nu ska ni tänka såhär, ”hur går det?”. ”Hur går det, innan ett team är välorganiserat är det en fin balansgång utan command and control och låta dom få chansen att bli självorganiserade, situationsanpassat, hur moget är teamet?”, vi har inga team som sitter inaktiva, det är oftast att det byts ut efter ett projekt. Det jag vet är att det kan vara viss förvirring i avsaknaden av… Jag har varit i ett projekt med den här kunden, där hade vi den här typen av “definition of done” det var vad som var viktigt är den här kunden. Kalendertid var extremt viktigt för den här kunden, då kanske vi medvetet valde bort vissa kvalitetsaspekter för att hinna klart, för att de kanske hade en stenhård deadline som gjorde att de kanske missat något. Säg att det var till någon påskkampanj eller något. Allt hade blivit slöseri. Det kanske är det som varit jätteviktigt och drivet och måste vara klart, då får vi scoopa ut visa saker för att det är MVP:n som ska igenom, sen efter det kanske man hamnar i ett projekt där kvalitén är jätteviktig, det som driver projektet är det ett långsiktigt projekt, ett system som ska drivas i 10-15 år. Det ska vara ”byggt korrekt”. Det ska vara ingen teknisk skuld, aktivt arbeta för att öka kvaliteten. Då kan man bli förvirrad, vi har kanske varit för dåliga att förklara vad som driver det här projektet ur kundens perspektiv. Det är nog snarare så att det är förvirringen, ”vad är viktigt för den här kunden?”. Och jag tror att skulle vi införa Scrum som process eller liknande, totalt, i bolaget, våra kunder jobbar kanske lite mer med icke klassiska produkter, mer löpande bara, de har system som ska byggas det kommer hela tiden nya grejer, det är klurigt, kunden vet inte riktigt var dom kommer vara om ett halvår, hade det varit produkter, hade kunnat bygga upp en stor portfolioplanering med allt från version 2, 3 och om fyra år med en agil roadmap, den möjligheten har inte de som jobbar med produktutveckling. En grej som återkommer är att vi använder en roll-and-wave planning, i alla projekt; vad som händer om tre månader har vi ganska bra koll på, det som händer om sex månader

	mindre koll på, och det som händer om ett år, ingen aning. Det kan på hög nivå och vara väldigt luddigt, det är något som återkommer hela tiden, man brukar prata om att ”agile is all about planning”, och det är något som verkligen stämmer, det är det många bolag missar, oavsett vilken, om man säger Scrum eller Kanban eller vad det nu är. Så är det liksom att hela tiden omprioritera, omplanera och att ha disciplinen att kunna göra det, och en grej vi märker… Ni frågade innan om folk blir förvirrade som inte har något speciell process, så kan det väl vara att, brukar säga att agile är extremt mycket om disciplin och man vet inte om man har den disciplinen förrän det går åt skogen. Om du har vilken process som helst har du ett normalflöde, det Kanban och Lean gör till exempel eller ja, det gör Scrum också och hela agila rörelsen; man försöker hantera risk och variation, förstör variation i en process visar sig som en försening, det är den risken man vill hantera. Allt som kan gå fel i en systemutvecklingsprocess kan skapa förseningar. När det blir för mycket variation till exempel, att när det inte går enligt protokoll, det är då man vet om man har disciplin, då vet man hur man ska attackera problemet eller om man bara ska springa, inte veta vart man ska, och det upplever jag, vilket inte händer jätteofta i detta bolaget, vilket jag är glad över. När något skiter sig, när något ändras totalt, så är vi ganska snabba på att attackera det, istället för att vi kan få panik och inte kunna hantera det. Det är också svårt att sätta en definition om varför det är så. Är det en mognadsfråga? Många tycker ja. Är det för att vi har bra agilister inom bolaget? Ja, kanske. För oss har det blivit en naturlig del för att hantera en risk, och hantera variationer, tid förhåller sig normalt när det skiter sig.

7.7	**När skulle du säga att agila metoder är bra, att hantera risker?**
7.8	Det enda sättet, att kunna hantera den typen av risk, utan att det ska bli en jätteformell process, [är att] hela tiden planera om, ifrågasätta, planera om, ifrågasätta, exekvera. För att kunna hantera risker, då saker förändrar sig. Även om det handlar om i en iteration, eller årssikt, eller halvårsikt så otroligt central. Det är så mycket okända grejer, vi kan bara anpassa oss till. Vi kommer komma på, vi vet inte innan vi sätter igång vart vi kommer hamna, vi vet vad som driver projektet utifrån kundperspektivet, så länge vi vet det och anpassar oss efter det för att leverera värde. Pratar vi värde, värde är olika från punkt till punkt. Ibland är det kalendertid, ibland kvaliteten eller någonting helt annat, det är alltid en risk. Så länge vi har en riktning i ett projekt så kommer vi alltid navigera kring problem som uppstår. Det hade också kunnat vara på samma sätt om vi hade kört RUP eller någonting annat, sen kan man ändra på det; Kanban-sidan, det finns en service-organisation som kör mer Kanban-orienterat, och pratar med förvaltningsavtal och liknande, när det kommer in en bugg i ett existerande system, vi har lite av det i vår verksamhet, det är nästan lite ”incident, nu händer det någonting, nu behöver vi agera snabbt”. Och då kan det ändras för dagen, och då hade till exempel Scrum eller något varit helt katastrofalt, för att det är för lång tid mellan, verkligheten ser annorlunda ut för hantering, vissa saker kan inte vänta i två veckor. Det går ju inte. Så sen är det också det med spekulanter, i vissa fall i vår organisation är kraven att det kan hända så snabbt så vi kan inte ha en låst iteration, utan det kan vara löpande i form av ett Kanban-system, och det har sina för- och nackdelar.
7.9	**Kanban och Scrum, hur skulle du säga att de är som visualisering av arbete?**
7.10	Ett moget Scrum-team kör automatiskt Kanban ungefär, kan man se det. Det som sker är egentligen, i Kanban bygger du egentligen på att du har tydligare, du visualiserar någonting i din process. Någon form av överlämning, även om det inte är fysisk överlämning så kanske man vill visualisera något bara för att visa på att vi är på rätt status. Scrum i sig säger inte att ”din tavla måste se ut såhär”, utan ett moget team anpassar tavlan efter vad dom vill visualisera. Så du kommer få retrospekt att du har problem med att utvecklarna slarvar med sina tester innan de checkar in det. Släng in en ny kolumn på din Scrum-tavla som visualiserar det statet, så ni kan verkligen se att det går igenom det att man måste fysiskt flytta ett kort för att man ska ta det dit. Hela tiden bli en påminnelse för allt vi gör, eller vad vi ska göra.

	Lika Kanban, visuell och levande, en kö är en levande indikator, är det fullt ser jag det, då kan jag inte göra så mycket mer. Ett moget Scrum-team arbetar på samma sätt, säg att du har en Scrum-klassisk ”ready for test, testing done” och enligt Scrum ska du köra dom högst prioriterade först, det ska vara klara innan du påbörjar nästa arbete. Ett moget Scrum-team ser till att detta sker, blir det en flaskhals av att det är mycket som sitter på ready for test, då hjälper utvecklarna till att testa det, så dom kan dra igenom det, så dom kan vara klara med storyn. Det Kanban kan göra är att, man tvingas… Ett moget Scrum-team gör det, det är en naturlig del. Men Kanban kan vara bra för att man får en hård WIP [Work-In-Progress]; det får inte plats med mer på tavlan, och Kanban är liksom kö-teori, det är fortfarande inte överdrivet, att ju mer saker och pågående desto längre tid tar alla att avklara. Så Kanban kan vara bra på det sättet. Kanban i sig behöver inte vara agilt, Kanban är egentligen bara en processkontroll, det finns inget i Kanban som i sig är agilt, det kan vara det men det kan lika väl vara ett vattenfallsprojekt som tar delar av Kanban. Problemet jag också ser i ”Kanban-implementationer” är att det blir alldeles för ad-hoc. Folk slarvar med planering och nedbrytning, man använder Kanban som en anledning till att inte planera, att ifrågasätta grejer, ”Varför kan vi inte planera det här, varför kan vi inte ge någonting mer?”.
7.11	**Varför gör man det?**
7.12	Varför man slarvar?
7.13	**Ja, varför man ser det som en anledning till att inte planera?**
7.14	För jag tror att, lite där, som en WIP, ”Jag har som max tre och nu är det tre WIP-limits på tavlan – nu kan jag inte göra mer”, likväl har du inget i Kanban som säger att du måste planera. I Kanban måste man hitta rytmen, vi kör planering när vi får tillräckligt många kort igenom, vi måste sätta upp det som en process, vad som sätter takten, och i vissa fall, i vissa team, din Kanban-tavla, korten du ska dra in, någon kolumn du har, på femte kortet står det en notering med ”kallas till planeringsmöte”, för dom vet att om fem kort så är det slut på det vi gör och då måste vi kalla till planeringsmöte. Så det blir mer, så du sätter rytmen baserat på arbetet som flyter igenom och jag tror att många, Scrum är svårt att implementera för en organisation, det är enkelt ramverk med få artefakter och ceremonier men det krävs ganska mycket disciplin, det var ganska svårt att bolagen de var inte riktigt redo eller mogna för det, och försökt jobba med Scrum-implementation två-tre år, inte riktigt fått fart, fått med produktsidan, delar av organisationen som behövs, mycket hinder i organisationen i andra kontaktytor till teamet som var får sen dragna så dem inte fick igenom arbetet för att dem kunde inte lösa upp dem här flaskhalsarna.

	Och då såg man Kanban, det är någonting, där behöver vi inte planera eller ha så mycket struktur, det är ganska enkelt, det är inga ceremonier, inte någon som behöver ta en kurs. Därför såg man det som det nya häftiga, många Kanban-varianter har skrotas, blivit ad-hoc och kaos. precis som jag har sätt många Scrum-implementationer gå åt helvete för att man inte haft stöd från organisationen. Och jag tror att, tittar man slaviskt på en process eller en processkontroll, så, om man inte förstått andemeningen så kommer det skita sig oavsett vilken metod du väljer. Har du inte förstått tankarna bakom ett push-pull system ska du inte hålla på med Kanban. Har du inte förstått syftet med WIP-limits ska du inte hålla på med Kanban. Inte förstått syftet med interaktionen eller den typen av sprintplanering, så ska du inte köra Scrum. Kanban behöver inte vara agilt, exempel behöver inte ha några som helst agila tecken. Men med Scrum så måste du ha agila förutsättningar.
7.15	**Du var lite inne på här… om vi fortsätter på Scrum-spåret; ser du några svagheter med Scrum, något som ni kanske har valt bort och kört något annat istället?**
7.16	Alltså Scrum är jättebra i vissa organisationer, det är jag helt övertygad om. Nackdelarna jag ser i Scrum – för en organisation som kanske inte riktigt är mogen – är att man tar väldigt lätt på förändringsarbete. Alltså Kaizen handlar ju liksom om continuous improvement. Och efter ett retrospektiv är det kanske 10 saker som ska åtgärdas och ibland är det inte alla som blir åtgärdade, utan en kanske blir det och det håller i två iterationer sen är man tillbaka på ruta ett igen. Så man tar väldigt lätt på förändringsarbetet och det tror jag att Scrum i sig som ramverk har ganska stora brister i är att man pratar alldeles för lite om… Om liksom hur bedriver man förändringsarbetet i iterationerna. Säg att man lagt in en sak, att man alltid bara en sak, alltså man kollar alltid på lean-teori så är det alltid en flaskhals i ett system som definierar genomströmningen. Flyttar du den flaskhalsen eller löser upp den så kommer det bli en ny på ett annat ställe. Den flyttar sig alltid. Så det är alltid en flaskhals som sätter genomströmningen. Och då har du en iteration också; hade Scrum-ramverket sagt “ja men okej, det är en grej efter varje retrospekt som ska åtgärdas” så att man hade fokuserat på förändringsarbetet också, hade det varit lättare. Ett moget Scrum-team gör det här; de tar 1-2 grejer och ser till att det fungerar innan de tar in nya saker. Sen tror jag att man historiskt sett la för lite vikt i produktägarrollen, klassiska “skit in, skit ut”. Är det luddiga krav, då utvecklar man det fel. Rework; vad är rework? Jo det är waste och det skapar inte bara waste i form av att man utvecklar fel grejer som man kanske måste göra om sen utan skapar även frustration hos teamet över att utveckla någonting som sedan var fel. Det blir som ett slag i ansiktet, “Varför ska jag bry mig om att göra den här grejen korrekt, när ni ändå kommer komma på

	att det inte är så här ni vill göra". Och med det menar jag att "agile is all about planning"; har man inte tiden och mognaden att planera vad jag egentligen vill ha ut av de här iterationerna så ska man inte hålla på med agil utveckling. Då kanske man inte ska hålla på med systemutveckling överhuvudtaget.
7.17	**Om vi går in på överlag för agil utveckling; vilka styrkor ser du där?**
7.18	Riskhanteringen och man får chans att göra inkrementella investeringar. Vi kan vrida på ett projekt ganska snabbt ifall kunden ändrar sig, utan att ha planerat för två år framöver; skapa extremt produktiva team genom själv organisation, få commit och ansvar; få en tillit och en transparens gentemot kunder, istället för att diskutera kontrakt och "blablabla"; ha en agil kontraktsmodell. Andra fördelar… Producerar bra mjukvara så det är det som man pratar om, hela tiden hitta bättre sätt att utveckla mjukvara och det är många saker som ändras sen det agila manifestet skrevs men andemeningen är fortfarande kvar och "det är det". Man kan kalla det vad man vill men vi vill producera bra mjukvara och vi vill att våra kunder ska göra det med oss och teamet ska vara drivande. Vi som bolag och kunden kan han en riskkontroll.
7.19	**Jag tänkte ställa en fråga om… ni säger att ni jobbar mycket agilt, kan estimering och att tidsplanen hålls vara ett problem?**
7.20	Alltså, "Hur långt är ett snöre?". Eftersom att kunderna inte vet vad de vill ha, de tror de vet det, men kommer inse om någon vecka att det gör de inte. Åter igen tilliten; vi gör inga fastprisprojekt, det är inte så vi arbetar som bolag. Vi har olika typer av agila kontrakt som handlar om kanske rättighet att avbryta någonting, äganderätt till det vi skapat och likande. Vissa kunder kan vi inte jobba agilt med, vi har vissa vi jobbat väldigt, väldigt vattenfall med. Det händer då och det kommer in "Jag vill ha det här gjort"; "Okej hur kan vi göra detta?". Då kan vi hitta någon här inne som kan vara produktägare för honom. Så vi kan ändå arbeta internt. Tilliten är extremt viktig, och den tilliten måste vi få, när det väl sker vilket det gör ofta med agil, kunder som inser att mjukvara är komplex, få in ett mänskligt beteende i en dator är inte helt lätt. Efter ett tag inser det flesta att alla vinner på det här, det blir billigare och dem får rätt sak.
7.21	**Men det kan vara så i början att ni först måste visa dem alla fördelar för att kunderna ska förstå?**
7.22	Framförallt… Eftersom att vi pratar mycket om inkrementella investeringar, vi är transparenta och vi vill inte låsa upp kunden, "Det här kommer ta ett år". Utan det är snarare såhär; om vi får två månader och sen får vi ser vad vi hamnar, då har vi en vision och hela tiden att leverera någonting med bra value så att kunden blir "Wow!". Kan man få det här

	på två-fyra veckor då får vi den här tilliten. Alla kunder är inte heller mogna för att hantera det på andra sätt, det… Alla projekt som vi har total leveranskontroll på har vi den approachen, att vi styr det så. Sen kommer det beställningar som säger sig själv, ”Vi vill ha det här gjort, vi tror det tar en månad; äsch vi tar det löpande”. Så vi tar det inte fastpris, det gillar ingen; blir lätt misstro.
7.23	**Men ni tar även kunder som ber er att köra den här metoden typ vattenfall?**
7.24	Inte som vi levererar från vårt kontor, då har vi leveransansvar och då driver vi med projektledning. Annars skulle inte vi kunna gå i garanti att vi ska hinna klart och så vidare. Vi tror inte på upplägget.
7.25	**Hur skulle du säga om acceptansen för era arbetsmetoder? När ni är hos kund, använder ni user stories för att visa att det verkligen fungerar?**
7.26	Ute hos kund?
7.27	**Ja.**
7.28	Vi har ju i rätt många fall… Många av våra utvecklare är duktiga på det agila och kan agera som change agents, och i många fall blir vi intagna; “Vi vill ha några grymma utvecklare, och jag vill att de ska hjälpa oss att driva agile-satsningen också. Jag hade ofta den typen av roller tidigare. Det är rätt ofta som stora kunder kollar på oss och har börjat göra affärer med oss för att de ser att outsourcingen fungerar inte. Då kommer de hit och vi hjälper dem förstå värdet av detta. ”Rom byggdes ju inte på en dag” som man säger. Att få en organisation att förändra sig tar extrem tid; jag skulle säga att kanske efter två år har man ett bra svar på om det gått bra eller inte. Det är så långsamt och komplext. Sen positiva grejer med Kanban; man började med Kanban för att man ville ha något enkelt, man ska inte förändra allt på en gång, den approachen kan man även ha om man implementerar en agil process, sen om den slutar upp och blir som Scrum eller vad det nu blir. Men man kan alltid börja med en liten grej för att bli mer agil. Och vi jobbar med… När vi är ute hos kund har vi en team-coach som är ute och hjälper dem med just sådana frågor en gång i månaden. Hur ska kunden skriva bättre acceptanskriterier, har du provat det här?
7.29	**När ni arbetar ute hos kund; kräver ni mycket tid av dessa då? [Jag] tänker på involvering då, att det kan vara ett problem i det agila?**
7.30	När vi levererar här inifrån eller?
7.31	**Ja.**
7.32	Ja vissa kunder är mer problematiska än andra och då får man istället ha en produktägare här. Som sköter all kommunikation och så vidare. Tyvärr blir det så ibland.
7.33	**Hur löste ni det sa du?**

7.34	Att vi sätter en intern produktägare som har all kund kommunikation och kanske träffar honom en gång i månaden, lägger upp road-mapen och planen tillsammans och tar fram vad är det som driver integrationen och så vidare; blir liksom domän-experten. För att kunden inte kan vara tillgänglig eller har tid, eller att kunden inte har det intresset.
7.35	**Är det ett vanligt problem?**
7.36	Nej inte jättevanligt, jag tror det är vanligare på… I organisationer som försöker implementera Scrum tycker det är vanligare; för att här inne… Vi arbetar med kunder som vill få ut någonting och de inser oftast själva att de måste vara delaktiga. Annars hade de kunnat lägga ut det här på Indien. Så de inser att det är viktigt och det är något vi kräver i kontraktet också, att de ska vara delaktiga. De behöver inte sitta här men de ska vara tillgängliga för planeringsmöten och så vidare.
7.37	**Hur mycket tid skulle du säga att produktägaren måste avsätta?**
7.38	Det beror helt på, scope, road-maps etcetera. Det är svår att generalisera det, det jag tycker man ska ha och nästan kräver det är commit men framförallt en syn framåt. Lite ”roll-wave planning” liksom, för att återigen vet de inte vart de är på väg. Okej, vi pratar med teamet och säger ”Vi vet inte riktigt vart vi ska, nu protypar vi lite, vi vill testa oss fram”. Då är det fine, det är ett strategiskt beslut som teamet är med på. Och när produktägaren säger ”Jag vet exakt vad ni ska göra ni i tre iterationer”, men sen inte har koll på vad som händer om ett halvår och inte förmedlar det till teamet kan det hindra att teamet gör vissa beslut i arkitekturen eller i lösningen, som blir ganska dyrt, för det är en teknisk skuld sen när det ändrar sig efter sex månader. Ett stort tema i appen eller något, teamet vet inte det här, gör antaganden och sen om sex månader måste allt byggas om. Det är sen waste, det commitet måste man ha som produktägare; man måste vara extremt tydlig och många missar det.
7.39	**Arbetssättet som ni har nu har ju växt fram, men vad hade ni från början? Hade ni en modell ni arbetade efter innan?**
7.40	Alltså det här bolaget skapades för att vi var trötta på hur man levererade och sålde in och utvecklade mjukvaruprojekt i större organisationer. I det här bolaget har det alltid varit agile mindset och sen bara, ”wing it”. Vi lär oss medan vi driver projekt; vad vi kan göra bättre och så vidare. Det är viljan att vilja förändra något och inte sätta allt i sten som jag tycker är viktig.
7.41	**Att ni inte sätter ett pris, förlorar ni potentiella kunder för det?**
7.42	Oftast är det kunder som kommer till oss som prioriterar kvalitet. Men ja vi har säkert missat någon kund på det. Kan vi lära oss av det? Har vi varit dåliga på att förklara varför? Ja, kanske.
7.43	**Det är nästan så man måste arbeta; det är svårt att sätta ett pris?**

7.44	Det är helt omöjligt, extrem risk för oss och kunden.
7.45	**Ni sätter inte ett målpris?**
7.46	Det kan man göra, de är ett agilt kontrakt liksom, vi kommer hamna någonstans mellan ”där och där”. Och sen kan vi ändra det under resans gång. Men visst att dem får en uppfattning om det. Men det är återigen att det är ett kontrakt som inte går ut på att leverera exakt det här till det här datumet.
7.47	**Vi har fått svar på det vi ville, om du inte har något som du trodde vi skulle ta upp eller som du vill tillägga?**
7.48	En viktig grej som jag kan säga är att oavsett om det är Kanban eller Scrum eller vad du ni kör. Tror många att ”Om vi skickar Kent och Anna här på en kurs kommer det att lösa sig och det kommer bli jättebra”? Man är inte medveten om förändringsarbetet som det kommer krävas, vilken process och vilket jävla jobb det är. Och det är lite det som är Scrums nackdel kanske, att man gick och ”produktifierade” och den typen av grejer. Många bolag tycker att ”Vi har gjort om här så då skickar vi dessa på kurs; sen är det bra med det”. Så är det tyvärr inte, det finns inga genvägar på det sättet. På det viset kan… Om man har ett par drivna individer i en organisation som vill arbeta på ett mer lättrörligt sätt, låt dem börja smått, ändra en liten grej. Man behöver inte ta in ett jätteramverk, om de är drivna och har en bra mognad. Är det däremot… Om man är väldigt ny till det och vill ha stödhjul, då är Scrum och den typen av processkontroll bra, för då har man tydliga riktlinjer, ”Vi ska göra det här och det här”. Så jag tror att om det är bra eller dåligt i en given situation beror lite på organisationens förutsättningar och på teamet. Och det missar många; de försöker applicera ett (1) tänkande på alla typer av team och organisationer och tar väldigt enkelt på förändringsarbetet. Och det är inte bra.

8 Referenser

Acando (2015): *Organisation*. http://www.acando.se/om-acando/organisation/ (besökt 2015-05-08)

Armour, P. G. (2014): The Business of Software, Estimation Is Not Evil, Reconciling agile approaches and project estimates. *Communications of the ACM*. Vol 57, No 1, Sid 42-43

Awad, M. A. (2005): A Comparison between Agile and Traditional Software Development Methodologies.

Azizyan, G., Magarian, M. K. & Kajko-Mattson, M. (2011): Survey of Agile Tool Usage and Needs. *Agile Conference*. Sid 29-38

Beck K. & Andres, C. (2004): *Extreme Programming Explained: Embrace Change*, 2:a uppl. New Jersey, Pearson Education Inc.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001): *The agile manifesto*. http://agilemanifesto.org/ (besökt 2015-03-26)

Boehm, B. & Turner, R. (2003): Observations on balancing discipline and agility. *Agile Development Conference*. Sid 32-39

Boehm, B. (2002): Get Ready for Agile Methods, with Care. *Computer*. Vol 35, No 1, Sid 64-69

Cho, J. (2008): Issues and challenges of agile software development with Scrum. *Issues in Information Systems*. Vol 9, No 2, Sid 188-195

Cockburn, A. & Highsmith J. (2001): Agile Software Development: The People Factor. *Computer*. Vol. 34, No. 11, Sid 131-133

Cohn, M. (2005): *Agile Estimating and Planning*. New Jersey, Prentice Hall

EG (2015): *EG i siffror*. http://eg.se/om-eg/vilka-aer-eg/eg-i-siffror (besökt 2015-05-08)

Fortnox (2015): *Om Fortnox*. http://www.fortnox.se/om-fortnox/ (besökt 2015-05-08)

Frey, J.H & Oishi, S. M. (1995): *How to Conduct Interviews by Telephone and in Person*. Thousand Oaks, Sage

Ikonen, M., Kettunen, P., Oza, N. & Abrahamsson, P. (2010): Exploring the Sources of Waste in Kanban Software Development Projects. Software Engineering and Advanced Applications. Sid 376-381

Infor (2014): *About Infor*. http://www.infor.com/company/ (besökt 2015-05-08)

Jacobsen, D. I. (2002): *Vad, hur och varför?* Studentlitteratur, Lund

Kniberg, H. & Skarin, M. (2009): *Kanban och Scrum - Få det bästa av två världar*. C4Media

KPMG (2013): *Project Management Servey Report 2013*. http://www.kpmg.com/NZ/en/IssuesAndInsights/ArticlesPublications/Documents/KPMG-Project-Management-Survey-2013.pdf (besökt 2015-05-14)

Mckinsey & Company (2012): *Delivering large-scale IT projects on time, on budget, and on value*. http://www.mckinsey.com/insights/business_technology/delivering_large-scale_it_projects_on_time_on_budget_and_on_value (besökt 2015-05-18)

Munassar, N. & Govardhan A. (2010): A Comparison Between Five Models Of Software Engineering. *International Journal of Computer Science Issues*. Vol 7, No 5, Sid 94

Parnas, D. (2006): Agile methods and GSD: The wrong solution to an old but real problem. *Communication of the ACM*. Vol 49, No 10, Sid 29

Qlik (2015): *Who is Qlik?* http://www.qlik.com/se/company (besökt 2015-05-08)

Rikspolisstyrelsen (2013): *Utvärdering av ärendehanteringssystemet SiebelPUST (Rapport 2013:10*), Rikspolisstyrelsen

Ringhals (2015): *Fakta om Ringhals AB*. http://corporate.vattenfall.se/om-oss/var-verksamhet/var-elproduktion/ringhals/ (besökt 2015-05-08)

Rubin, K. S. (2013): *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. New Jersey: Addison-Wesley

SCB (2014): *Företagens användning av it*. http://www.scb.se/Statistik/_Publikationer/NV0116_2014A01_BR_IT02BR1402.pdf (besökt 2015-05-18)

Schwaber, K. & Sutherland, J. (2013): *The Scrum-Guide*.

Sharp, H., Rogers, Y., & Preece, J. (2007): *Interaction Design: Beyond Human-Computer Interaction*, 2:a uppl. New Jersey, Wiley

Shiohama, R., Washizaki, H., Kuboaki, S., Sakamoto, K. & Fukazawa, Y. (2012): *Estimate of the appropriate iteration length in agile development by conducting simulation*. Agile Conference, 1/1/2012

Solinski, A. & Petersen, K. (2014): In press: Prioritizing agile benefits and limitations in relation to practice usage. *Software Quality Journal*

Statskontoret (2010): *Granskning av Försvarsmaktens införande av ett integrerat resurs- och ekonomiledningssystem (PRIO) (Rapport 2010:6),* Stockholm, Statskontoret

Stoica, M., Mircea, M. & Ghilic-Micu, B. (2013): *Software Development: Agile vs. Traditional*. Informatica Economică. Vol 17, No 4/2013, Sid 64-76

Sutherland, J., Viktorov, A., Blount, J., & Puntikov, N. (2007): Distributed Scrum: Agile Project Management with Outsources Development Teams. *System Sciences*. Sid 274

Sverrisdottir, H. S., Ingason, H. T. & Jonasson, H. I. (2014): The role of the product owner in scrum- comparison between theory and practices. *Procedia - Social and Behavioral Sciences*, Vol 119, Sid 257-267

Tanner, M. & Mackinnon, A. (2013): Sources of Disturbances Experienced During a Scrum Sprint. *Proceedings of the European Conference on Information Management & Evaluation*. Sid 255-262

The Standish Group (2013): *Chaos Manifesto*. http://www.versionone.com/assets/img/files/ChaosManifesto2013.pdf (besökt 2015-05-14)

Toyota (2015): *Just-in-time - Philosophy of complete elimination of waste*. http://www.toyota-global.com/company/vision_philosophy/toyota_production_system/just-in-time.html (besökt 2015-04-28)

Tretton37 (2015): *Lundabolag vill växa men fortsätta som IT-ninjor*. http://tretton37.com/press (besökt 2015-05-08)

Turk, D., France, R. & Rumpe, B. (2002): *Limitations of Agile Software Process*.

Wake, W. C. (2000): *Extreme Programming Explored*.

VersionOne (2015) *9th Annual State of Agile Survey*. VersionOne, Inc.