GÖTEBORGS UNIVERSITET

Hur överförs kunskap i agila systemutvecklingsprojekt?

- En fallstudie i hur Scrum används för att säkerställa kunskapsöverföring.

How is knowledge being transferred in agile development projects?

- A case study on how Scrum is used to ensure knowledge being transferred.

Marcus Wittenstam
Michael Jonsson

Kandidatuppsats i Informatik

Rapport nr. 2015:018

Göteborgs universitet
Institutionen för tillämpad IT
Systemvetenskap: IT, Människa och Organisation
Lindholmen Campus, Göteborg, Sverige, 22 Maj 2015

Abstrakt

I alla år som mjukvara har existerat så har systemutvecklingsprojekt skådats som komplext och svårhanterligt, utan någon helhetslösning för komplexiteten. Det är känt att systemutvecklingsprojekt har en låg kundnöjdhetsgrad med endast 10-40% nöjda kunder. Systemutvecklingsprojekt är beroende av att kommunikationen mellan människor fungerar, och utformningen av precisa krav vilar på hur effektiv kommunikationen är. Studiens teoretiska ramverk förklarar hur krav genereras från kunskap, och hur kommunikation är medlet för att överföra kunskap. Agila systemtvecklingsmetoder har skjutit i popularitet det senaste decenniet. Den här studien syftar därför till att undersöka och analysera hur agila systemutvecklingsmetoder används för att säkerställa kunskapsöverföring. Frågeställningen har formulerats enligt följande:

"Hur används Scrum för att säkerställa kunskapsöverföring i systemutvecklingsprojekt?"

Studien har utfört en kvalitativ fallstudie på ett IT-konsultbolag och ett Scrum-forum. Slutsatsen visar att Scrum som agil systemutvecklingsmetod har en serie moment som leder till kunskapsöverföring. Studien visar även på att Scrum tenderar att ha en viss vaghet i hur dessa moment ska utföras.

Nyckelord: krav, kunskap, kommunikation, agil systemutveckling, Scrum

Abstract

Software development projects has throughout the history of software been viewed as complex and difficult to manage, without any comprehensive solution for this complexity. It's known that software development projects has a general low degree of satisfaction with only 10-40% pleased customers. Software development project depends on communication between various parties, where the accuracy of the requirements depends on the effectiveness of this communication. The studies theoretical framework shows that requirements are created through knowledge and communication is the means to transfer knowledge. Agile software development has become one of the most popular software development methods in the last decade. This study aims therefore to research how agile software development methods is used to ensure knowledge transferring. The issue of the thesis was formulated as following:

"How is Scrum used to ensure knowledge being transferred in system development projects?"

The thesis conducted a qualitative case study at a IT-consultancy and a Scrum-forum. The conclusion shows that Scrum as a agile software development method has as series of tasks that leads to knowledge being transferred. The thesis also shows that there tends to be a vagueness in how these tasks should be fulfilled.

This thesis is written in Swedish

Keywords: requirements, knowledge, communication, agile software development, Scrum

Tack

Studiens författare vill ägna ett tack till Alten och dess informanter som ställde upp och bidrog med värdefull information till studien.

Ett stort tack utlyses även till Urban Nuldén som stöttat och väglett författarna genom hela studien.

Innehållsförteckning

1. Inledning

Av all folktro som skrämmer oss mest är varulvar en av de värsta då de oväntat förvandlas från våra kära till våra mardrömmar. För att skydda oss mot dessa varelser söker människor silverkulor som magiskt kan ta kol på dem. Systemutvecklingsprojekt har liknande karakterister. De ser oskyldiga och enkla ut, men är kapabla att förvandlas till ett monster av komplexitet, missade deadlines, sprängda budgetar och misslyckade produkter. Men mot detta finns ingen silverkula. Ingen enskild metod som kan skapa ordning i produktivitet, tillförlitlighet eller komplexitet (Brooks, 1986).

Systemutvecklare har länge plågats av denna problematik, tron om att nya verktyg och tekniker kan lösa alla systemutvecklingsproblem utan att ägna den tid och energi som krävs för att förstå sig på orsaken och relationerna relaterade till problemen (Fenton et al, 1994). Utan den förståelsen adopteras varje ny innovation som den nya silverkulan, för att därefter ratas då orginalproblemen kvarstått och med risk att få sällskap av nya problem (Banker et al, 1998).

Brooks (1986) och Leveson (1997) menar att ingenjörer ständigt stöter på komplexitet i all deras arbete, men att de arbetar utifrån en struktur. Att det finns enhetliga lagar som fysiken rättar sig efter, och matematiska formler som på ett enkelt sätt beskriver komplexa tillstånd. Utveckling av system har däremot inga enhetliga lagar eller formler som går att förhålla sig till. Systemutveckling förhåller sig istället till människor och mänskliga interaktioner. Människan, och att förstå en annan människas budskap är komplext (Brooks, 1986).

Fler studier stärker teorin om att systemutvecklingsprojekt är komplexa och svårhanterliga. Så mycket som 60-90% av alla systemutvecklingsprojekt uppnår inte den lönsamhet som kunden räknat med (Escalle et al, 1999; Robey et al, 2002). Dessa siffror är indikatorer på att den generella nöjdhetsgraden är väldigt låg inom systemutvecklingsprojekt. Forskning visar även på att en mängd olika faktorer ligger bakom detta, varav en stor anledning är att samtliga parter inblandade i ett systemutvecklingsprojekt ofta har olika bilder på problemsituationen och hur problemet eventuellt ska lösas.

De flesta systemutvecklingsprojekt hanterar krav, och krav har som uppgift att förmedla en gemensam bild för kunden och utvecklarna om vad för problem som ska lösas (Sommerville & Sawyer, 1997). Brooks (1986) menar att den svåraste delen i att skapa ett system är att ta reda på exakt vad som behövs utvecklas. Kravhantering är den viktigaste delen utvecklaren gör för användaren, och ingen annan del av projektet kan skada resultatet så mycket som kravhanteringen kan. Ingen annan del är så svår att fixa till i slutet av systemutvecklingen som kraven (Appan & Browne, 2012; Brooks, 1986). Umble och Umble (2002) menar vidare att förändringar av krav är en av de största anledningarna till att systemutvecklingsprojekt fallerar. En av de största faktorerna som bidrar till dessa förändringar är att det sker misskommunikation i projektets kravhantering då kunden har svårigheter med att förmedla vad de verkligen behöver (Umble & Umble, 2002). Appan och Browne (2012) menar också att misskommunikation mellan de involverande parterna i systemutvecklingsprojekt generellt sätt hämmar utformningen av exakta krav.

Oavsett systemutvecklingsmetodologi behandlas krav, och olika antagande görs i varje metodologi gällande hur krav hanteras. Kunskapsöverföring via kommunikation framkommer mellan involverade parter i kravhanteringsprocessen i systemutvecklingsprojekt för att identifiera problemområdet och den eventuella lösningen. Oavsett val av metodologi och antagande, är krav högt utsatt för att bli felaktigt utformade på grund av misskommunikation (Appan & Browne, 2012).

Nonaka (1994) menar att kommunikation syftar till att skapa en gemensam förståelse mellan individer genom kunskapsöverföring. Kunskapsöverföring kan förekomma i formerna: externalization, socialization, internalization och combination. I systemutvecklingsprojekt kan det också användas en rad olika tekniker för att uppnå en kunskapsöverföring mellan de involverade parterna. Oavsett teknik kan kunskap dock inte överföras såvida inte kommunikationen sker på rätt sätt. Misskommunikation uppkommer ifall krav och behov inte tolkats lika av de involverade parterna, vilket kan leda till att produkten riskeras att misslyckas (Sommerville & Sawyer, 1997). Några stora anledningar till att misskommunikation leder till felaktiga krav är bristen på kommunikation mellan involverande parter, svårigheter för användare att artikulera krav samt oviljan från användare att förmedla krav (Appan & Browne, 2012). Den största anledning är dock att de involverade parterna ofta har olika mentala bilder och uttrycker sig genom olika terminologier, vilket ökar risker för misskommunikation. Misskommunikationer bör försökas att minimeras då problemet hämmar utformningen av precisa krav (Appan & Browne, 2012).

1.1 Syfte/Frågeställning

Med problemområdet i baktanke kommer studien att undersöka och analysera hur kunskap överförs i systemutvecklingsprojekt. Både vattenfallsmodellen och agila metoder används mycket i nutida systemutvecklingsprojekt, men agila metoder har däremot blivit alltmer populärt de senaste åren, och fortsätter än idag att växa (Appan & Browne, 2012). Scrum är idag en av de snabbast växande agila systemutvecklingsmetoderna, och är framtagen för att traditionella metoder visat sig vara ineffektiva i praktiken (Softhouse, 2006; Light, 2009; Norton, 2008). Scrum är även känt för att vara effektiv på att hantera förändringar av krav (Softhouse, 2006). Studiens författare har därför valt att använda Scrum som granskningsobjekt för agila systemutvecklingsmetoder. Studien kommer därför att undersöka och analysera hur Scrum används för att säkerställa kunskapsöverföring, vilket mynnar ut i frågeställningen;

“Hur används Scrum för att säkerställa kunskapsöverföring i systemutvecklingsprojekt?”

För att besvara denna frågeställning presenterar teorin relaterad forskning för att behandla problemområdet. I teoriavsnittet beskrivs krav, kunskap, kunskapsöverföring och kommunikation, samt relationen mellan dessa, vilket mynnar ut ett teoretiskt ramverk som beskriver att kunskapsöverföring sker via kommunikation. Detta ramverk används för insamling och strukturering av det empiriska materialet. Vidare definierar teoriavsnittet Scrum och hur Scrum används för att säkerställa kunskapsöverföring. Studien innefattar en kvalitativ fallstudie med två

delar, en del på ett IT-konsultbolag samt en del på ett digitalt Scrum-forum[1]. Fallstudiens resultat syftar till att skapa en fördjupning i hur Scrum som systemutvecklingsmetod tacklar problematiken med kunskapsöverföring. Studiens empiri baseras främst på transkriberade kvalitativa semistrukturerade intervjuer från ett IT-konsultbolag samt deltagande observationer på Scrum-forumet.

Studien har valt att fokusera på kunskapsöverföring vid kravhanteringen inom systemutvecklingsprojekt. Detta motiveras genom att en av de största anledningarna till att systemutvecklingsprojekt fallerar är på grund av förändringar av krav och att en av de största anledningarna till dessa förändringar beror på misskommunikation mellan involverade parter (Umble & Umble, 2002; Appan & Browne, 2012). Vid kravhanteringen sker det kunskapsöverföringar där kommunikation är medlet för att skapa en gemensam förståelse mellan individer.

Studien kommer att bidra till en ökad förståelse i hur agila systemutvecklingsmetoder används för att säkerställa kunskapsöverföring i systemutvecklingsprojekt.

1.2 Disposition

Uppsatsens disposition är uppbyggd genom att först presentera en inledning av studien. Därefter följer en presentation av relaterad forskning kring krav, kunskap, kunskapsöverföring och kommunikation, samt hur Scrum förhåller sig till de begreppen. Kopplingen mellan de beskrivna begreppen har vidare genererat ett teoretiskt ramverk. Efter teorin presenteras metodavsnittet där tillvägagångssättet för resultatet beskrivs. Följt efter metodavsnittet presenteras resultatet i form av bilder, citat och reflektioner. I diskussionsavsnittet poängterar och diskuterar sedan studien det viktigaste av den empiriska data som samlats in samt hur materialet förhåller sig till teorin. Studien avslutas med en slutsats som kopplar resultatet till frågeställningen.

2. Teori

Studien presenterar i detta avsnitt relaterad forskning för att ge en bakgrund till studiens teoretiska ramverk som studien bygger vidare på. Det teoretiska ramverket bygger på kopplingen mellan krav, kunskap, kunskapsöverföring och kommunikation. Dessa begrepp beskrivs i teoriavsnittet för att ge en förståelse för deras innebörd och sammanhang till det teoretiska ramverket. Vikten av kommunikation för att uppnå kunskapsöverföring, belyses i det teoretiska ramverket. Ramverket fungerar som ett vägledande verktyg inför fallstudien och upplägget av det empiriska materialet. Avslutningsvis i teoriavsnittet presenteras Scrum, och dess användning för att säkerställa kunskapsöverföring via kommunikation. Värt att notera är att engelska begrepp inom Scrum har böjts enligt svenska språket.

2.1 Krav

Utformade krav i ett systemutvecklingsprojekt är ett resultat på en skräddarsydd analys av en användares verksamhet och problemområde (Sommerville & Sawyer, 1997). Analysen handlar

[1] https://www.scrum.org/Forums/aff/1

om att samla kunskap om användarens problem, hitta funktioner som ska lösa problemen och hur den framtida lösningen ska kunna kommunicera med andra befintliga system. Detta görs för att användaren sällan vet vad den egentligen behöver, de vet sällan vilka frågor som behöver besvaras, och de har nästan aldrig tänkt igenom sitt problem i detalj (Brooks, 1986). Brooks (1986) påstår också att det nästan är omöjligt för användare, även de användare som arbetar med systemutveckling, att exakt specificera kraven som behövs för systemet innan prototyper av systemet har utvecklats och testats.

Det vanligaste sättet att ta fram krav är genom kommunikation mellan utvecklare och användare, samt genom observation av användarens verksamhet (Sommerville & Sawyer, 1997). Krav har som uppgift att förmedla en gemensam bild för samtliga parter inom systemutvecklingsprojekt om vad för problem som ska lösas. Hantering av krav är det viktigaste momentet i systemutvecklingsprojekt. Kraven kommer att förbli viktig genom hela projektet eftersom de ligger till grund för hela systemutvecklingen (Sommerville & Sawyer, 1997). Brooks (1986), menar även att kravhantering är den viktigaste delen som utvecklare gör för användaren, att iterativt extrahera och förfina krav för systemet. I systemutvecklingsprojekt är det också vanligt att krav ges olika prioriteringar. Ifall krav med hög prioritet inte utvecklats har utvecklingsteamet misslyckats att leverera värde till användaren, vilket kan leda till att kunden blir missnöjd och att systemet blir oanvändbart (Sommerville & Sawyer ,1997).

Enligt Oakland (2004) har krav en stor betydelse då kvalitet och kundnöjdhet är en mätning på hur väl den efterfrågade lösningen uppfyller kundens krav. Enligt Oakland (2004) räcker det inte att endast ta reda på och förstå användarens krav, utvecklarna måste även ha en förståelse över sin egen förmåga om att uppfylla kraven. Ställer en användare till exempel ett krav på att en utvecklare ska utveckla en liknande sökmotor som Googles, under en veckas tid, så kommer utvecklaren förstå vad kunden vill ha samtidigt som hen förstår att kravet är bortom hens förmåga. Förståelsen om sina egna förmågor att uppfylla kraven är lika viktigt som att förstå vad kraven innebär (Oakland, 2004).

När en utvecklare designar en produkt eller ett system räcker det inte med att veta vad det är för något som ska designas, utvecklaren måste även ha en förståelse för i vilka sammanhang och situationer som systemet ska användas i. Vid ett samtal om kvalitetskrav för exempelvis en stol är det viktigt att diskutera dess syfte och i vilka sammanhang den kommer att användas. En kontorsstol kan skapas i högsta kvalitet, men ifall syftet var att använda stolen som en TV-stol, kommer kontorsstolen inte uppnå kravet. Skillnaden mellan kvaliteten på kontorsstolen och TV-stolen är inte på sättet de tillverkades på, utan för vad de designades för (Oakland, 2004).

Kvaliteten på design mäts genom hur väl produkten är designad för att uppfylla de överenskomna kraven. Det viktigaste att utgå ifrån när utvecklare designar en produkt är att uppnå kraven från användaren. Alla involverade bör ha förstått sig på de olika kraven innan designprocessen tas vid för att misskommunikation inte ska inträffa (Oakland, 2004).

2.2 Kunskap

Nonaka (1994) utgår från att kunskap är relationen mellan sann och rättfärdigad tro. Denna förklaring av kunskap härstammar från den traditionella epistemologiska synen på kunskap. Det epistemologiska synsättet ser kunskap som något absolut, statiskt, och en form av formell logik. Däremot finns det ett modernare perspektiv som belyser vikten av att ta hänsyn till att kunskap är en slags personlig tro. Denna personliga tro trycker på vikten att kunna rättfärdiga kunskapen. Synsättet ser kunskap som en dynamisk mänsklig process av rättfärdigad personlig tro som en del av strävan att nå sanning (Nonaka, 1994).

Information och kunskap är två närbesläktade begrepp, men har en distinkt skillnad. Information är ett flöde av meddelanden. Detta flöde genererar kunskap, utifrån mottagarens redan befintliga kunskap, först när mottagaren tolkat informationen. Information kan till exempel vara, en lapp med ett recept på. Informationen på lappen överförs till kunskap först när en individ tar till sig och tolkar informationen (Nonaka, 1994).

"*In short, information is a flow of messages, while knowledge is created and organized by the very flow of information, anchored on the commitment and beliefs of its holder*" (Nonaka, 1994. s.15).

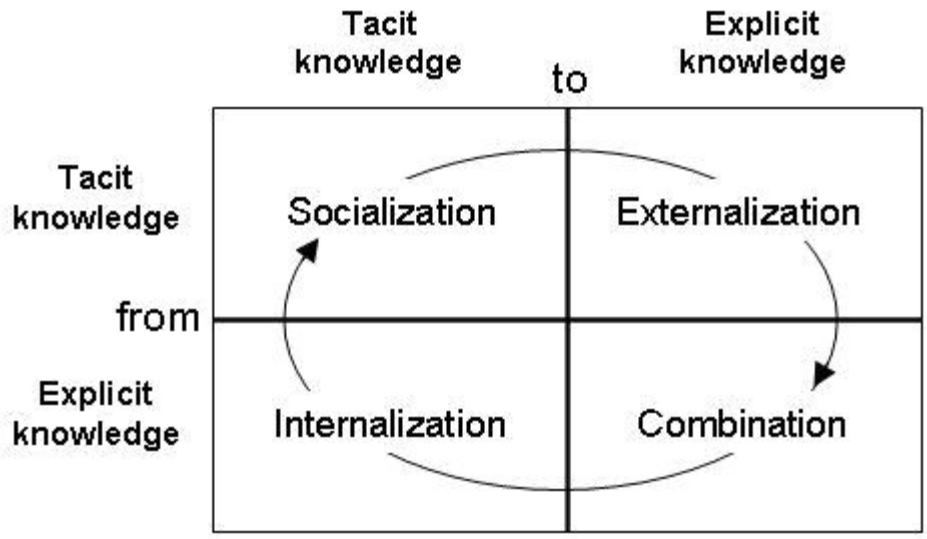
Information är ett nödvändigt material för att initiera och forma kunskap. Kunskap är information tolkad av en individ. Med kunskap har individen förmågan att förstå, återge och tillämpa informationen.

Information kan ses utifrån ett syntaktiskt eller ett semantiskt perspektiv. Det syntaktiska perspektivet fokuserar på volymen av informationen utan att ta hänsyn till värdet. En telefonräkning är till exempel baserad på längden och distansen av samtalet och inte på vikten av det som kommunicerats. Det semantiska perspektivet på information fokuserar istället på värdet av informationen (Nonaka, 1994).

2.3 Kunskapsöverföring

Enligt Nonaka (1994) finns två olika typer av kunskap: tacit- och explicit kunskap.
Tacit kunskap är kunskap som är så självklar att den är svår att definiera. Det är till exempel svårt att beskriva hur det går till att cykla även fast det är så självklart att kunna cykla.
Explicit kunskap är däremot något som snabbt kan förklaras och tolkas.

De två typerna är inte oberoende av varandra utan kan ses som ömsesidigt kompletterande entiteter. Tacit och explicit kunskap interagerar med varandra genom mänskliga aktiviteter på individ eller gruppnivå. Ny organisationskunskap kan skapas genom mänskliga interaktioner med de två typer av kunskap samt olika innehåll av kunskap. Bilden nedan representerar de sociala processer som skapar kunskapskonvertering (Nonaka, 1994).



Figur 1. Bild över kunskapsöverföringsprocesser

De två typerna av kunskap kan omvandlas till varandra i olika sammanhang.
Internalization kallas processen där explicit kunskap absorberas och leder till en djupare förståelse vilket leder till utvecklingen av tacit kunskap genom repeterande utförande av den gällande kunskapen.
Socialization beskriver hur tacit kunskap överförs till tacit kunskap mellan individer, en process som oftast sker inom ett team. Detta görs främst via utbyte av erfarenheter. Socialisation sker genom att en individ till exempel beskådar, tolkar och sedan härmar en annan individ. För att processen ska tas vid krävs ett forum där individerna kan kommunicera i samma tid och rum (Nonaka, 1994).
Externalization beskriver hur tacit kunskap blir ny explicit kunskap. Som tidigare nämnt är tacit kunskap väldigt svårt att definiera. Genom att använda sig av metaforer som ett hjälpmedel blir det däremot lättare att genomföra kunskapsöverföringen. Det är lättare för en individ att förstå, om individen kan sätta den nya kunskapen i perspektiv med något individen kan relatera till (Nonaka, 1994). Att använda sig av User Stories för att definiera krav är en teknik utvecklare använder sig av för att förstå och förmedla vad som ska utvecklas.
Combination är den process från explicit till explicit kunskap, och den sker ofta via koordinationer mellan avdelningar.

Persson (1997) beskriver generell kunskapsöverföring med hjälp av en enkel metafor. Han förklarade att ifall man har två spolar som det löper en ström igenom. Där den ena spolen introducerar ström till den andra spolen. Vet den ena spolen inte vilken ström som når den andra spolen, eftersom de två spolarna aldrig har någon fysisk kontakt. Ifall strömmen är svag och avståndet är långt mellan spolarna är det inte säkert att det uppstår någon ström överhuvudtaget i den andra spolen. Men en stor del av strömmen kan överföras även ifall verkningsgraden aldrig kan uppnås till hundra procent. Och eftersom olika spolar är lindade olika så kommer den starkaste strömmen att uppstå på olika sätt för olika spolar (Persson, 1997). Med detta menar Persson (1997) att två personer som kommunicerar med varandra aldrig riktigt kan veta vad den

andra har förstått när den ena pratar, eftersom de är två skilda personer. Är avstånden mellan personerna stora på grund av fysisk distans, eller psykisk distans så som kulturella eller politiska skillnader, så är det inte säkert att budskapet nås fram på rätt sätt. En kommunikation kan aldrig bli perfekt, men mycket av budskapet kan ändå nå fram. Alla människor är olika och därför är olika tillvägagångssätt för att kommunicera olika bra för olika personer.

Stein (1996) påstår att det finns en rad med faktorer som påverkar ifall kunskapsöverföringen mellan en sändare och en mottagare kommer att tas vid. Dessa faktorer är bland annat:

- Har kunskapen någon nytta för mottagaren?
- Är mottagaren lämplig för att ta emot kunskapen?
- Tjänar eller förlorar sändaren på att överföra kunskapen till mottagaren?
- Har mottagaren absorberat denna kunskap vid ett tidigare skede?
- Tjänar organisationen på det?
- Har sändaren tid?

Kunskapsöverföring inom ett systemutvecklingsprojekt kan exempelvis ske via tekniker av olika slag. Det finns olika sådana tekniker, med syfte att samla in och förmedla krav. Det krävs ofta en blandning av tekniker för att skapa en fullständig kravanalys. Det finns inte heller någon insamlingsteknik som passar varje enskild situation. Därför är det viktigt att överväga vilken teknik som är mest lämplig vid varje specifik situation (Eriksson, 2007). De vanligaste kravinsamlingsteknikerna är listad nedan:

Workshop är en effektiv teknik för att på kort sikt identifiera, strukturera och prioritera krav. Workshop påminner mycket om ett vanligt möte, skillnaden är att man med workshop kommer fram till en gemensam överenskommelse samt att man använder sig av tekniker för att fysiskt aktivera deltagarna. Alla deltagare är tillsammans ansvariga för resultatet de kommer fram till (Eriksson, 2007).

Intervjuer går ut på att samla in relevant information genom direkt kommunikation med systemets användare. Det finns olika typer av intervjuer som lämpar sig för olika situationer. Det finns personliga intervju-former av strukturerade, semi-strukturerade och ostrukturerade intervjuer. Det finns även intervju-former som sker på distans så som telefonintervjuer eller epost-intervjuer (Eriksson, 2007).

Enkäter kan användas för att dels identifiera krav som tidigare inte har upptäckts. Tekniken kan även användas för att utvärdera ett redan existerande system. Enkäter når på ett enkelt sätt ut till en större publik med hjälp av skriftliga frågor (Eriksson, 2007).

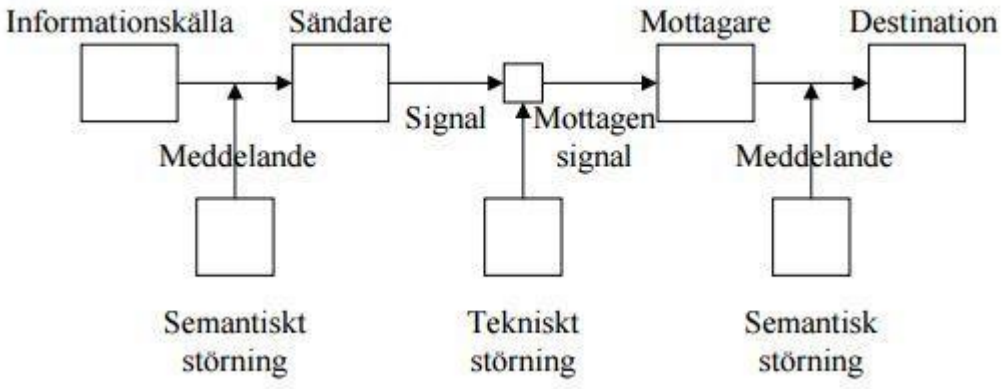
Observationer går ut på att kravhanterare iakttar användare i den dagliga verksamheten. Tekniken gör det lättare för kravhanteraren att förstå användarens arbetsprocesser och behov, vilket kan vara lätt att missa under intervjuer. Observationer är ett bra verktyg för att identifiera nya, smidiga lösningar på användarens arbetsuppgifter (Eriksson, 2007).

Användningstest går ut på användarna testar en rad definierade uppgifter. Detta görs för att systemets användarbarhet ska kunna utvärderas för att identifiera krav som kan förbättra systemets användarbarhet. Ett sätt att utföra denna teknik på är att be användaren tänka högt när de utför de definierade uppgifterna. På så sätt kan utvecklarna lättare upptäcka vad användaren upplever är svårt och förväntar sig av systemet (Eriksson, 2007).

Prototyper används för att kvalitetssäkra krav samt identifiera nya krav genom att observera hur användare interagerar med prototypen. Prototyper är anmärkningsvärda på det sättet att alla kan förstå dem, till skillnad från ett kravdokument.
User Stories är en teknik som används för att beskriva hur användare interagerar med specifika delar av systemet och av vilken anledning. Det är en teknik som används för att sammanlänka olika kommunikationsproblem och för att skapa en gemensam bild på problemet mellan användare och utvecklare (Eriksson, 2007).
Personas går ut på att skapa en rad fiktiva användare, som sedan används i en diskussion för relevanta krav som ska uppfyllas i systemet. De fiktiva användarna beskrivs tydligt genom att till exempel lista mål, yrke, personlighet, ålder, kön och eventuell inställning till teknik. Tekniken är lämplig att använda då det inte finns en tydlig målgrupp för systemet (Eriksson, 2007).

2.4 Kommunikation

Kommunikation innefattar lika mycket konst, musik och film som det gör samtal och text. Mängden information och kunskap som kan överföras i kommunikationer ökar i takt med minskning av brus och störningar. Mängden brus som uppstår i en kommunikation påverkas bland annat av valet av överföringskanal (Shannon & Weaver, 1949).



Figur 2. Bild över hur kommunikationer sker.

Meddelandet som sändaren skickar till mottagaren passerar ett antal störningsmoment på vägen, dessa moment delas upp i semantiska och tekniska moment. Det första störningsmomentet som meddelandet påverkas av är ett semantiskt störningsmoment som sker när kunskapen ska omvandlas från tanke till tal, text eller aktion. Efter att meddelandet har omvandlats till den fysiska världen påverkas den även av den fysiska världen i form av tekniska störningsmoment. Det kan vara oljud i bakgrunden, men det kan även vara brist på visuella redskap vid kommunikation i telefon. Det andra semantiska störningsmomentet uppstår när mottagaren ska avkoda och tolka meddelandet genom mottagarens kapacitet att omvandla tal, text och aktion till tanke och förståelse (Shannon & Weaver, 1949). Semantiska störningsmoment kan vara kulturella skillnader, politiska ställningstagande, intelligensskillnader, språkbruk och dyslexi.

Men olika människor tolkar data på olika sätt när den presenteras för dem. Detta beror på att olika människor har olika förkunskaper som de använder sig utav när de tolkar ny data. Olika

människor har också olika kunskapsabsorberingsförmågor, vilket gör att olika människor kräver olika mycket tid för att tolka samma mängd data. Motivation är en annan bidragande faktor som påverkar människors förmåga att absorbera och tolka kunskap, är mottagaren inte intresserad av meddelandet kommer mottagaren inte vara tillgänglig för att ta emot meddelandet. Det andra semantiska störningsmomentet överröstar då meddelandet (Langefors, 1980).

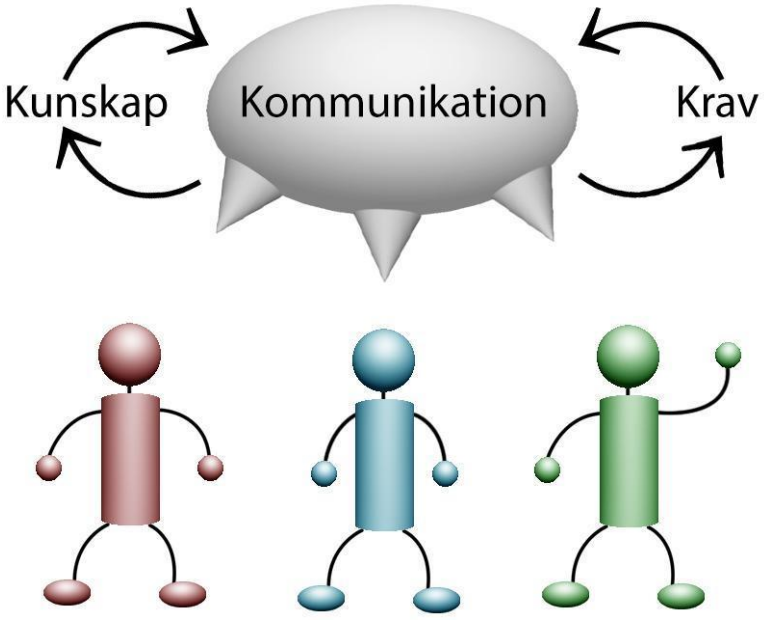
Missinformationseffekten är ett begrepp känt inom kommunikation. Missinformation har vanligtvis skådats i kontexten av ögonvittnens förmåga att minnas detaljer från en händelse de bevittnat. I den kontexten handlar missinformationseffekten om att rapportera felaktig information på grund av exponering av missledande information direkt efter händelsen. Som exempel kan en person som bevittnat en bilkrasch, bli intervjuad direkt efter händelsen där intervjuaren ställer en ledande fråga angående att bilisten passerat en stoppskylt. Ögonvittnet kan då minnas att bilisten passerat en röd stoppskylt, trots att ingen skylt var närvarande. I framtida konversationer finns det då en stor risk att ögonvittnet kommer minnas att en röd stoppskylt, trots att händelsen inte involverade någon skylt (Loftus et al. 1989).

2.5 Teoretiskt ramverk

Studiens problemområde belyser problematiken med krav och dess hantering, samt att misskommunikation vid kravhantering ökar andelen fallerade systemutvecklingsprojekt. Därför är syftet med studien att ta reda på hur agila systemutvecklingsmetoder används för att säkerställa kunskapsöverföring i systemutvecklingsprojekt, med Scrum som granskningsobjekt.

Studien har identifierat att kommunikation är den sammankopplade nyckeln för att generera krav från kunskap. På samma sätt förmedlas krav till kunskap genom kommunikation. Med rätt sorts kommunikation mellan utvecklare och användare kan kunskapsöverföringen öka och andelen fallerade systemutvecklingsprojekt minskas.

Figuren nedan visar hur krav genereras av kunskap från kunskapsöverföring via kommunikation samt hur krav kan förmedla kunskap från kunskapsöverföring via kommunikation. Kommunikationen är nyckeln för kunskapsöverföring. Individer har möjligheten att absorbera kunskap eller utveckla ny kunskap via kommunikationer i form av externalization, socialization, internalization och combination. Problematiken ligger dock i att det finns en komplexitet med mänsklig kommunikation och hanteringen av de semantiska och tekniska störningsmoment som kan uppstå. Därför krävs det att kommunikation sker på rätt sätt för att mottagaren ska kunna förstå sändarens budskap så som sändaren menar. Som tidigare nämnts finns det olika tekniker för att samla in och förmedla krav. Dessa tekniker används för att försöka säkerställa kommunikationen mellan användare och utvecklare. Alla dessa tekniker är lämpade för olika situationer och Eriksson (2007) menar att tekniker bör kombineras för att uppnå största möjliga kunskapsöverföring.



Figur 3. Figuren illustrerar relationerna mellan kunskap och krav och belyser vikten av kommunikation för att uppfylla kunskapsöverföring.

Detta teoretiska ramverk förmedlar att kommunikation är nyckeln vid kunskapsöverföring. Kunskapsöverföring är väsentligt för att kunna generera och förmedla krav. Ramverkets logik kommer att användas vid studiens empiriska datainsamling. Studiens resultatavsnitt kommer därmed också att fokusera på hur kommunikation skapar och förmedlar krav samt kunskap.

2.6 Agil systemutveckling

På senare tid har agila metoder tagits fram, eftersom att systemutvecklingsprojekt generellt har en väldigt hög misslyckandegrad (Beck, 1999). Ett centralt begrepp för agila metoder är arbeta nära användaren, att vara adaptiv och snabb på att anpassa sig efter nya krav, samt förbättra kommunikationen mellan användare och utvecklingsteam (Softhouse, 2006). De agila metoderna är anpassade för förändringar genom att innehålla iterativa processer för att på bästa sätt hantera förändringarna. Vid en första ögonkast kan iterationerna efterlikna faserna i vattenfallsmodellen, men skillnaden är att iterationerna inte fyller någon specifik funktion utan syftet med en iteration bestäms när iterationen inleds (Highsmith, 2004).

Agila metoder förespråkar att leverera kvalitativa produkter på ett snabbt sätt genom att lära sig att göra rätt och att göra det effektivt. Ett exempel på det är att ta bort allt som inte ger något värde för kunden, och att prioritera det med mest värde, för att utesluta utvecklingen av onödiga funktioner (Highsmith, 2004; Beck, 1999). Det positiva med agila metoder och dess iterationer är

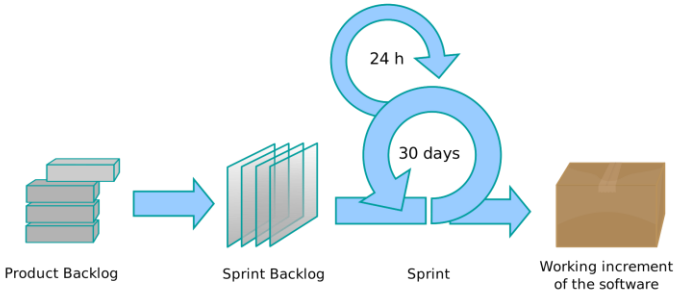
att de tillåter projekt att inledas utan vetskapen om slutresultatet, utan att förutse alla problem och utan att ta tidiga beslut som inte går att ändras i en iteration (Highsmith, 2004).

Agila metoder är bra på att hantera förändringar, och allt fler utvecklare väljer att använda sig utav agila metoder i systemutvecklingsprojekt för att kunna tackla förändringar från kundens behov på ett snabbt och enkelt sätt. Missförstånd reds ut tidigt och designfel upptäcks tidigare. Agila utvecklare får tidigt feedback från användare och testpersonal får en jämnare arbetsbelastning eftersom att test utförs kontinuerligt i dessa metoder. Enligt Beck(1999) är det även viktigt att utvecklare är ärliga med vad dem klarar av i agila projekt. Det är till exempel ett tecken på mod när en utvecklare berättar att hen inte klarar av en viss uppgift och detta hjälper projekt framåt. Det är även viktigt att utvecklare visar respekt gentemot varandra för att lättare kunna samarbeta och förstå varandra (Beck, 1999).

Eftersom att agila metoder generellt arbetar nära kunden med hjälp av ständiga tester och delprototyper så sker det en ständig kunskapsöverföring i systemutvecklingen. Det sker kunskapsöverföring i alla Nonakas (1994) beskrivna former i agil utveckling. Kunskapsöverföring i form av combination då kravlistor förmedlas, förändras och förfinas. Internalizationprocessen tas vid då användare repeterande gånger testar systemet. Socialization sker då utvecklare inom teamet samarbetar och utbyter kunskap genom att observera varandra. Externalizationprocessen genomförs mellan användare och utvecklare när användarens krav och behov ska dokumenteras samt när prototyper presenteras och utvärderas.

2.6.1 Scrum

Scrum är en globalt känd och accepterad agil systemutvecklingsmetod som används för att bedriva komplexa projekt. Kraven från användaren struktureras i en Product Backlog som sedan bryts ner i så kallade Sprintar. En Sprint pågår oftast mellan två till fyra veckor och Scrum Team:et har under denna period ett speciellt utsatt mål som teamet jobbar efter (Deemer et al, 2012; Schwaber, 1997; Softhouse, 2006).



Figur 4. Bild över Scrums:s utvecklingsprocess

Ett Scrum Team består oftast av Product Owner, Scrum Master samt ett Development Team, och brukar i regel bestå av fem till nio personer. Teamet bestämmer hur arbetet ska utföras och hur arbetsfördelningen ska se ut. Alla i medlemmar i gruppen ska kunna byta uppgift med varandra om så behövs (Deemer et al, 2012; Schwaber, 1997; Softhouse, 2006). Detta gör att medlemmarna i teamet behöver lära sig av varandra vilket i störst mån sker genom observationer och diskussioner via socialization och externalization.

Softhouse (2006) förklarar att det finns ett par specifika titlar baserat på personens ansvarsområde inom Scrum Team:et. Product Owner kallas personen som är kundens länk och röst till Development Team:et och kan oftast själv vara kunden. Denne ser till att Development Team:et jobbar med rätt saker utifrån ett affärsperspektiv. Product Owner är också ansvarig för projektets Product Backlog. En lista med krav på funktioner som ska utvecklas och finnas med i den framtida produkten. Vidare menar författarna att Product Owner också ansvarar för att prioritera kraven i listan för att specificera ordningen som kraven ska utvecklas och lanseras i. Denna lista är synlig för alla som jobbar inom organisationen för att alla ska kunna bilda sig en uppfattning om slutprodukten, en kunskapsöverföring via combinationprocessen. Product Owner bör ha goda kunskaper inom teknik, marknad- och affärsprocesser. Product Owner är ansvarig för att prioritera kraven i listan för att specificera ordningen som kraven ska utvecklas och lanseras i. Product Owner skriver även estimat för kraven i Product Backlog tillsammans med Development Team (Deemer et al, 2012; Schwaber, 1997; Softhouse, 2006).

Scrum Master kallas personen som fungerar som en hjälpreda för Scrum Team:et. Denne ser till att teamet har bästa möjliga förutsättningar för att kunna utföra sina uppgifter under en Sprint. Scrum Master har kontakt med teamet varje dag genom ett kortare uppföljningsmöte. Är det någon utanför projektet som har åsikter att ta upp så görs detta med Scrum Master och inte med teamet. Detta för att inte störa arbetet som pågår under en Sprint. Efter varje Sprint har Scrum Master ett utvärderingsmöte med Development Team:et där Sprint:en utvärderas för att bland annat se vad som kan förbättras tills nästa Sprint och om kraven har uppfyllts (Deemer et al, 2012; Schwaber, 1997; Softhouse, 2006).

Enligt Deemer et al (2012); Schwaber (1997); Softhouse (2006) inleds arbetsprocessen med att Product Owner sammanställer alla begäran och krav i en så kallad Product Backlog. Kraven kan vara funktioner som ska finnas med i den nya produkten eller vilken typ av användare som ska använda produkten. Efter att målen har blivit definierade och prioriterade så styckas Product Backlog:en in i mindre delar tillsammans med Development Teamet. Product Owner rangordnar som tidigare nämnts vilka delar som ska prioriteras och utvecklas först. De högst rangordnade kraven är mycket mer detaljerad i sin beskrivning än de lågt rangordnade eftersom de kraven kommer börjas utvecklas först. Alla krav som Product Owner har kommit fram till ges sedan en estimerad tidsram för utveckling. Scrum beskriver inte hur kraven kan samlas in eller förmedlas för Development Team;et mer än i en rangordnad lista. Däremot tillåter Scrum som metod att olika tekniker används för detta syfte. En allmän accepterad teknik för att förmedla krav är att skapa User Stories (Deemer et al, 2012).

Enligt (Deemer et al, 2012) är estimat en teknik för att ge Scrum Team:et och kunden en uppfattning om hur lång tid det tar för Development Team att utveckla de olika kraven. Estimat tas fram på olika sätt och ett tillvägagångsätt är att Product Owner:n förklarar och presenterar kraven i följd av en prioritetsordning där Development Team individuellt röstar på en summa points för varje enskilt krav. En summa väljs fram när hela Development Team är överens. Ifall en medlem av Development Team har satt ett högt eller lågt värde till skillnad från resterande medlemmar kan detta vara en indikator på att vissa av teamet inte riktigt förstått sig på kravet i fråga. Det kan lika gärna vara den som skilt sig från majoriteten, som majoriteten som har missförstått kravet i

fråga. För att hantera detta inleds därefter en diskussion där medlemmarna får motivera sitt val, tills dess att alla har fått en gemensam förståelse över vad kravet innebär och hur kravet ska utvecklas. Därefter görs en ny röstning för att ta fram ett gemensamt point på kravet. Hur många points ett team klarar av att utveckla i en Sprint utgör teamets Velocity (Schwaber, 1997). Estimatet blir då en teknik för att avgöra hur många krav som Sprint Backlog:en kan innehålla (Deemer et al, 2012; Schwaber, 1997).

Deemer et al (2012); Schwaber (1997); Softhouse (2006) menar att varje krav i Product Backlog:en utvecklas i en Sprint där kraven delas upp i mindre hanterbara delar av hela Scrum Team:et. Detta gör att Scrum Team:et tillsammans får en gemensam bild över arbetsuppgifterna. Sprint:en pågår oftast under två till fyra veckor. I Sprint Backlog:en fördelas arbetsuppgifterna och tiden som krävs för att utföra dem beslutas. När detta är bestämt kan Product Owner inte göra ändringar i Sprint Backlog:en förrän Sprint:en är över. Skulle Product Owner i extrema fall vilja ändra något i en pågående Sprint så avslutas den pågående Sprint:en helt och en ny Sprint Backlog skapas (Deemer et al, 2012; Schwaber, 1997; Softhouse, 2006). Development Team:et går vanligtvis in i varje Sprint med vetskapen att kraven som Sprint:en ska uppfylla inte kommer att ändras under Sprint:ens gång. Detta medför att teamet kan helhjärtat fokusera på att uppfylla målen i Backlog:en (Schwaber, 1997).

Enligt Deemer et al (2012); Schwaber (1997); Softhouse (2006) utför Scrum Master varje dag, vid samma tidpunkt ett kortare uppföljningsmöte med Development Team:et under Sprintens gång. Mötet sker ofta stående för att hålla det kort. Detta möte är till för att teamet ska kunna klargöra vad som gjorts från föregående möte, vad som ska göras till nästa möte och om det är något som förhindrar att en medlem i teamet inte kan utföra sin uppgift. Syftet med detta möte är att teamets medlemmar ska kunna få en insyn i hur Sprint:en utvecklats och om det är något någon behöver hjälp med. Denna kunskapsöverföring i mötena sker i processerna externalization och combination där tacit kunskap hos en medlem i teamet uttrycks och överförs till de andra medlemmarna i form av både tacit och explicit kunskap. Detta möte kallas för Daily Scrum.

Enligt Deemer et al (2012) sker det en process som kallas för Product Backlog Refinement i slutet av varje Sprint där hela Scrum Team:et engagerar sig för att förfina och uppdatera Product Backlog:en. Denna förfining kommer ligga till grund för framtida Sprint:ar, alltså inte för den nuvarande Sprint:en. De krav som förfinas och uppdateras är oftast de högst prioriterade kraven som finns kvar i Product Backlog:en. Product Backlog:en är alltså ett levande dokument som ständigt uppdateras. Det är i denna process som Product Owner har möjlighet att lägga till, ändra eller ta bort krav. Scrum uttrycker inte hur denna process ska utföras, men en vanlig teknik att använda sig utav är workshops (Deemer et al, 2012). I denna förfiningsprocess sker en kunskapsöverföring där både tacita och explicita kunskaper utbyts och definieras.

Deemer et al, (2012) förklarar att en Sprint avslutas genom att den utvecklade mjukvaran visas upp och tillåts testas av alla som är involverade i projektet och är intresserade av resultatet. De är oftast Product Owner, Development Team, Scrum Master, kunderna, användarna, investerare, experter och representanter för ledningen som deltar på detta möte. Mötet, som kallas Sprint Review, är oftast två timmar långt och har som tidigare nämnts syftet att visa upp delprodukten

och låta de närvarande testa delprodukten i form av en prototyp. Mötet tillåter Scrum Team:et att fånga upp feedback för att kunna förbättra produkten. Mötet är alltså inte en presentation där Scrum Team:et själva presenterar hur prototypen fungerar, utan fungerar mer som en “sandlåde-miljö” där de närvarande själva kan testa delprodukten. Detta genererar en informationsrik feedback och en mer naturlig respons från testarna. Mötet ligger också till grund för ett utvärderingsmöte där Scrum Master, Product Owner och Development Team försöker komma fram till vad som kan göras bättre tills nästa Sprint utifrån feedbacken från Sprint Review (Deemer et al, 2012; Schwaber, 1997; Softhouse, 2006).

3. Metod

För att skapa ett djup i hur Scrum används för att säkerställa kunskapsöverföring i systemutvecklingsprojekt har studien valt att utföra en kvalitativ fallstudie som är uppdelad i två delar. Ena delen på ett större IT-konsultbolag i Sverige, och den andra delen på ett digitalt forum. Användandet av kvalitativa metoder som insamlingsmetod motiveras med att studien vill skapa ett djup snarare än bredd i studien (Klein & Myers 1999; Patel & Davidson 2011). För att skapa en grund till fallstudien har studiens författare presenterat relaterad forskning och tagit fram ett teoretiskt ramverk som studien utgått ifrån vid insamlingen och strukturen av det empiriska materialet. Studien har även använt sig utav semistrukturerade intervjuer och observationer vid insamlingen av det empiriska materialet (Nandhakumar & Jones 1997; Patel & Davidson 2011; Walsham 1995).

Enligt Patel och Davidsson (2011) finns det varken en bättre eller sämre datainsamlingsmetod. Det är situationen som styr vilken form av metod som är mest lämplig för att samla in mest användbar information om sitt problemområde utifrån de resurser som finns tillgängliga. Löwgren och Stolterman (2004) håller med om att det är upp till studiens författare med sitt kritiska tänkande att ta ställning till metodens tillämpbarhet och bedöma vilken metod som är mest lämplig för det specifika fallet. Hove och Anda (2005) har en liknande åsikt där de säger att kvalitativa intervjuer är en bra och vanlig datainsamlingsmetod som ger en insyn på användarens värld, men att kvaliteten på datainsamlingen beror på hur intervjun genomförts.

Kvalitativa metoder har även sina nackdelar eftersom det kan vara svårt att kvalitetsäkra det insamlade materialet samt att forskare som utför insamlingen kan komma att sätta prägel på resultatet (Klein & Myers, 1999; Walsham,1995).

3.1 Fallstudieobjekt

Den ena delen av fallstudien har utförts på teknik- och IT-konsultbolaget Alten, som hädan efter kommer benämnas “Konsultbolaget”. Konsultbolaget ingår i en större koncern med över 16.000 medarbetare och bedriver verksamhet i 16 länder. Studien finner företagets erkänt etablerade position på marknaden som en positiv aspekt för fallstudiens trovärdighet. Studiens författare utförde fallstudiens på ett av Konsultbolagets egna kontor.

Den andra delen av fallstudien har utförts på Scrums officiella webbsida som skapades 2009 av en av grundarna till Scrum. Sidans uttalade mål är att ge Scrum-utövare de rätta verktygen för att kunna använda och utvinna värde av Scrum. Webbsidan har flera certifierade utbildningar i Scrum samt även forum där Scrum-experter är aktiva. Detta gör webbsidan och dess forum till en tillförlitlig källa för studien att utföra en fallstudie på. Hädanefter kommer detta forum på webbsidan benämnas “Forumet”.

3.2 Datainsamling

Studien valde att utföra tre styckena kvalitativa semistrukturerade intervjuer hos Konsultbolaget samt deltagande observationer på Forumet med experter och nya användare av Scrum. Ett större antal informanter hos Konsultbolaget hade bidragit till mer tyngd och större djup i studien, men tidsramen satte begränsningar på studien med tanke på tiden som semistrukturerade intervjuer samt observationer kräver i form av utförande och analys (Patel & Davidsson, 2011). Därför kompletterade studien detta med att samla in data från experter och nya användare på Forumet. En problematik med fallstudien på Forumet var dock att den inte kompletterade bredden i hur Konsultbolaget hanterar kunskapsöverföring med Scrum som systemutvecklingsmetod. Däremot gav Forumet en bredare insyn på hur Scrum-användare generellt sätt hanterar kunskapsöverföringen med Scrum som systemutvecklingsmetod.

Eftersom att studien utfört en fallstudie på både Konsultbolaget och Forumet så presenteras tillvägagångsätten i två avsnitt.

3.2.1 Konsultbolaget

I ett tidigt stadium undersökte studiens författare vilka möjliga bolag det fanns i Sverige att utföra en fallstudie på genom att ta kontakt med olika företag via mail. Efter att fått en bra och tidig respons från Konsultbolaget med en försäkran om att företaget aktivt jobbar med Scrum i utvecklingsmiljö valde författarna företaget som fallstudieobjekt. Kontakten med Konsultbolaget inför fallstudien skedde via mail och telefon.

Den 11-12:e maj 2015 utfördes fallstudien hos Konsultbolaget. I fallstudien användes semi-strukturerade intervjuer med konsulter som aktivt jobbat med Scrum som utvecklingsmetod. Studien genomförde tre intervjuer med informanter från tre olika utvecklingsteam. Den första informanten jobbade i ett agilt-team som hade influenser till Scrum. Denna informant kommer vidare benämnas “Informant 1”. Den andra informanten jobbade i ett team som helt adopterat Scrum som systemutvecklingsmetod. Denna informant kommer vidare benämnas “Informant 2”. Den tredje informanten hade en likadan roll som Informant 2 och kommer vidare benämnas som “Informant 3”. Alla informanterna har liknande roller som kravhanterare i sina respektive team.

Innan intervjuerna ägdes rum informerades informanterna om studiens syfte. Vid semi-strukturerade intervjuer till skillnad från enkäter, är varje individs bidrag mycket viktigt då relativt få informanter kommer att intervjuas. Det gäller då att motivera informanten att vilja befinna sig i samma rum som intervjuarna. Detta kan göras genom att klargöra syftet med intervjun och

försöka relatera syftet till individens egna mål, samt betona vikten av individens svar för studien och vilka resultat det kan mynna ut i (Patel & Davidson, 2011).

Intervjuerna har utförts på plats hos Konsultbolaget och en mobiltelefon har använts för att spela in samtalet. Fördelen med att spela in samtalet är att informantens alla åsikter och tolkningar enkelt fångas upp utan att intervjun störs (Patel & Davidson, 2011). Denna metod gav en mer fullständig bild över vad som diskuterats än vad anteckningar skulle ha gjort. Det finns dock en risk att informanten kan känna sig pressad och obekväm av att bli inspelad, vilket kan påverka intervjun negativt (Walsham, 1995). Den ena författaren har fungerat som intervjuledare medan den andra har spelat in och antecknat, under intervjun. I en semi-strukturerad intervju måste intervjuaren ibland improvisera och ställa uppföljningsfrågor för att få fram relevant information, då kan det vara bra att det finns två intervjuare som hjälper varandra att ställa fler och rätt frågor (Hove & Anda, 2005).

Studiens författare har även utfört intervjuerna med hjälp av en utformad intervjuguide. Intervjuguiden innehöll ett antal diskussionsteman som väglett författarna genom intervjuerna med Konsultbolaget. Intervjuguiden skickades även till informanterna innan intervjuerna för att dessa skulle kunna reflektera och förbereda sig inför intervjun. När en forskare använder sig av kvalitativa metoder så som semi-strukturerade löper denne risk att informanten inte genererar några direkta svar då intervjuerna inte är speciellt strukturerade (Patel & Davidson, 2011). Det gäller då att vara välplanerad genom att ha tagit fram ett tema och frågor för intervjun. Det kan även vara bra att skicka med frågeformuläret innan intervjun så att informanten kan förbereda mer kvalitativa svar (Patel & Davidsson, 2011).

De semi-strukturerade intervjuerna innehöll öppna frågor och varade i ungefär 30 minuter. Frågorna behandlade hur Konsultbolagets verksamhet hanterade krav och kunskap genom kommunikation samt vilken roll informanten hade i verksamheten. Informanternas svar ledde till nya öppna följdfrågor. Efter ena intervjun fick studiens författare betrakta och navigera i ett virtuellt verktyg som Konsultbolaget använde sig av i sin kravhantering. Intervjun avslutades med att informanten fick uttrycka om det var något övrigt som var intressant att ta upp i ämnet.

3.2.2 Forumet

I ett tidigt stadium säkerställde studiens författare att det valda forumet var det mest aktiva forumet på Internet efter att aktivt ha sökt efter forum och jämfört dessa. En ytterligare aspekt som gjorde att studiens författare valde Forumet var att erfarna Scrum-utövare var aktiva på forumet. Anledningen till valet att utföra datainsamling på ett digitalt forum var för att fånga in en bred mängd experters åsikter från en koncentrerad miljö.

Enligt Patel och Davidson (2011) ska en observation vara systematiskt planerad samt att informationen systematiskt ska dokumenteras. Studiens författare har utfört observationen på Forumet genom att ha bearbetat majoriteten av trådarna i Forumet via en systematiskt planerad logik. Logiken gick ut på att använda sig av Forumets sökfunktion där författarna matade in begrepp som var relevanta för studien. Resultatet från sökningarna dokumenterades och bearbetades tills att det sökta resultatet var slut. På så sätt kunde studiens författare få ut det

material som var relevant för studien. Det utvunna materialet bearbetades sedan ytterligare en gång för att få ut det mest väsentliga.

Studiens författare har även varit aktiva i Forumet genom att ha gjort inlägg med öppna frågor. Dessa frågor har genererat en diskussion på Forumet som vidare har skapat ett mer berikat datainsamlingsmaterial som studien kunnat nyttja.

3.3 Urval

Studien valde att utföra tre stycken kvalitativa semistrukturerade intervjuer hos Konsultbolaget samt observationer och aktivt deltagande på Forumet.

Studien valde informanter hos Konsultbolaget som hade en liknande roll av kravhanterare i sina respektive team, vilket var relevant för studien. Fler antal informanter hos Konsultbolaget hade däremot bidragit till större tyngd och djup i studien. Tidsramen satte däremot begränsningar på fallstudien med tanke på tiden som semistrukturerade intervjuer samt observationer kräver i form av utförande och analys (Patel & Davidsson, 2011).

För att tackla detta kompletterade studien insamlingen av det empiriska materialet genom att samla in data från experter och nya användare av Scrum på Forumet. En problematik med Forumet var dock att den inte kompletterade bredden i hur Konsultbolaget hanterar kunskapsöverföringen med Scrum som systemutvecklingsmetod. Däremot gav insamlingen en vidare bredd på hur Scrum-användare världen över hanterar kunskapsöverföringen med Scrum som systemutvecklingsmetod.

Att använda sig av ett Forum innebär att det inte garanterat går att veta vem som skrivit inläggen. Detta innebär att studiens trovärdighet kan ifrågasättas. Forumet tillåter dock endast medlemmar som uppgett personuppgifter och blivit godkända att uttrycka sig. Alla inlägg på Forumet måste även accepteras av en admin på Forumet för att kunna publiceras, vilket gör att inläggen håller en seriös standard.

3.4 Etik

Informanterna från Konsultbolaget fick reda på att all information som genererats av intervjun kommer att anonymiserats i studien. Walsham (2006) menar dock att det är väldigt svårt att säkerställa fullständig anonymitet. Det finns alltid en risk att en läsare av en rapport kan göra en kvalificerad tolkning om vem informanten är eller vilket bolag som undersökts. Studien har ansträngt sig för att på bästa sätt anonymisera data utan att kvaliteten på studiens påverkats.

Studien har utfört observationer samt citerat argument på Forumet utan medlemmarnas vetskap. Forumets medlemmar har där av inte godkänt denna fallstudie, vilket gör att det går att diskutera studiens etiska förhållningssätt. Studien har därför valt att anonymisera allt resultat som genererats från fallstudien på Forumet. Risken som Walsham (2006) belyser om anonymitet kan även uppstå från datainsamlingen av forumet. Läsare kan göra en jämförelse med studiens

presenterade resultat, med trådar från Forumet och därmed lokalisera vem som skrivit de olika inläggen.

3.5 Dataanalys

Det finns inget rätt eller fel när det gäller hur en kvalitativ studie analyseras, utan bör anpassas till situationen. Forskaren väljer själv en metod att analysera data från empirin då kvalitativa metoder inte har några färdiga mallar på hur data ska analyseras (Patel & Davidson 2011). Studien har valt att använda sig av kvalitativa metoder för att behandla och analysera det material som samlats in från datainsamlingen.

Kort efter intervjuerna genomfördes transkriberingar av det inspelade materialet. Transkriberingarna skapade ett överskådligt material som senare låg till grund för identifiering av den mest relevanta data för studiens syfte. Citaten från den mest relevanta data kategoriserades sedan utifrån det teoretiska ramverket som studien tagit fram.

Bearbetningen av data från forumet utfördes genom att studiens författare kopierade ut inlägg från tio olika trådar med diskussioner som var relevanta för studien. Dessa inlägg kopierades sedan in ett gemensamt dokument där författarna bearbetade och kategoriserade materialet utifrån det teoretiska ramverket för att sedan paras ihop med materialet från Konsultbolaget.

Den gemensamma data har presenterats och kategoriserats i resultatavsnitten utifrån studiens teoretiska ramverk ihop med reflektioner.

4. Resultat

I detta avsnitt följer resultatet från den empiriska datainsamlingen. Studiens huvudområde har varit att granska och analysera hur Scrum används för att säkerställa kunskapsöverföring. I teorin har studien identifierat att kommunikation är nyckeln för att generera krav från kunskap samt att förmedla kunskap från krav. Dessa två kunskapsöverföringar kommer ligga till grund för resultatets struktur. Scrum har fler kommunikationsmoment som berör förmedling av krav än generering av krav, vilket även vår datainsamling speglar. Värt att ha i åtanke är att förmedling av krav även leder till en större kunskap, vilket i sin tur kan generera nya krav. Studiens resultat presenteras i form av slutsatser och citat.

4.1 Kommunikation som genererar krav

Eftersom att Scrum Guide inte uttrycker hur krav ska samlas in så upplever informanterna att det är svårt att veta vilka kommunikationstekniker som bör och kan användas.

> *“Scrum i sig säger ju ingenting om hur man tar fram eller uppdaterar krav. “*
> - Informant 2

Det är alltså upp till användarna av Scrum att själva avgöra vilka kommunikationstekniker som lämpar sig bäst. Informant 2 tar upp att workshops är ett forum för parter att uttrycka sina åsikter och behov igenom.

> *"Man ju ha workshops med stakeholders och ibland poppar det upp en användare som skrivit user stories om något. Vi tar emot all feedback och slänger in det i icebox. Sen tar jag en titt på dem ihop med product ägarna och prioriterar om de osv"* - Informant 2

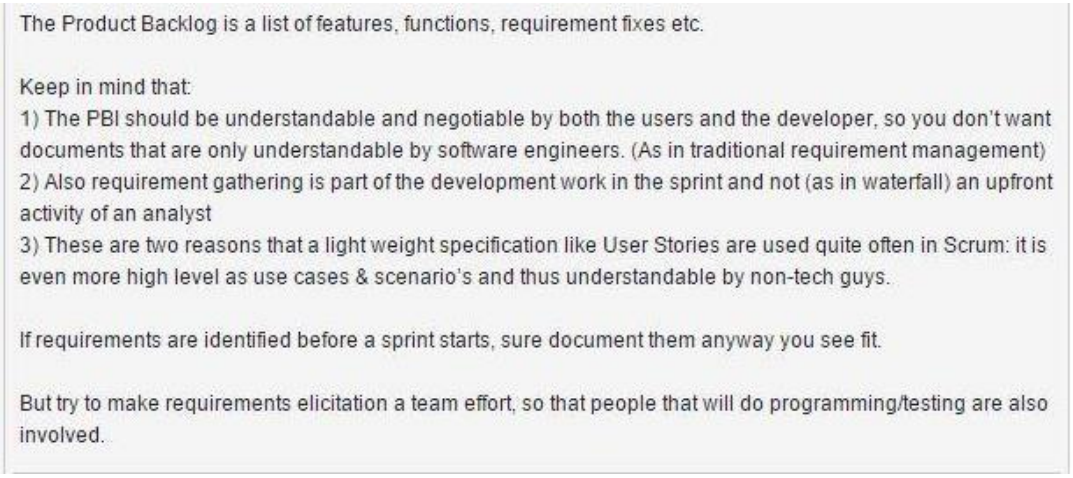
Ofta så presenteras krav som kunden vill ha från dem själva. Då kan Development Team fokusera på att formulera kundens krav och behov på ett sätt som Development Team:et kan arbeta ifrån.

> *"Ofta har de en hyggligt bra bild om funktionaliteten de vill ha på plats. De kan skicka bilder över hur de vill att det ska se ut och vilka funktioner de vill ha. Då är det lätt för oss att gå igenom funktionalitetsområde för funktionalitetsområde och skriva user stories för dem."* - Informant 3

Det kan dock uppkomma en problematik eftersom att kunden inte alltid vet vad dem vill ha samt att dem har svårt att uttrycka sina behov.

> *"Kraven är väldigt ofta utformade som lösningsförslag från deras sida. De glömmer väldigt ofta att belysa saker som behov, och varför det ska göras och av vem, speciellt varför. Väldigt ofta får man prata med kunden om detta. Så det vi får börja med är att döna ner massa user stories."* - Informant 1

Processen att hantera krav bör vara ett samarbete inom hela Scrum Team:et för att ge teamet en gemensam bild över vad som ska göras och hur det ska lösas.



The Product Backlog is a list of features, functions, requirement fixes etc.

Keep in mind that:
1) The PBI should be understandable and negotiable by both the users and the developer, so you don't want documents that are only understandable by software engineers. (As in traditional requirement management)
2) Also requirement gathering is part of the development work in the sprint and not (as in waterfall) an upfront activity of an analyst
3) These are two reasons that a light weight specification like User Stories are used quite often in Scrum: it is even more high level as use cases & scenario's and thus understandable by non-tech guys.

If requirements are identified before a sprint starts, sure document them anyway you see fit.

But try to make requirements elicitation a team effort, so that people that will do programming/testing are also involved.

Med transparent arbete får hela Scrum Team:et en gemensam bild över problemområdet och hur det ska lösas. Kunden kan dock vara motvilliga att vara transparenta då känslig information kan spridas.

> *"En av de bästa tillvägagångsätten för att förstå sig på kunden och vilka behov dem har är att kunden ger total transparens och insyn i deras verksamhet till samtliga i utvecklingsteamet. Det gör att alla utvecklare kan bilda en egen bild och komma med åsikter. Det är dock något som kunden är restriktiva med då man är rädd för dela känslig data. "* - Informant 3

Genom att arbeta transparent kan alla involverade parter också se Product Backlog:en och på sätt enkelt se vilka krav som ska uppfyllas, vilket gör att användarna kan komma med proaktiv feedback. Men trots detta är den största delen av feedbacken reaktiv.

> *"Vi har gått igenom alla user stories och de vet vad som kommer i kommande sprintar, det vet ju även slutanvändarna, så dem kan ju ha åsikter där. Men de flesta är ändå reaktiva och kommer med åsikter efter att sprinten är gjord. Men då tar vi in de som förändrings alltså det blir nya stories för nya sprintar."*
> - Informant 3

Den reaktiva feedbacken kommer oftast från Sprint Review mötena. Där sker en dialog mellan Scrum Team:et och alla de andra involverade parterna som kan ha åsikter om resultatet Development Team utvecklat.

> *"Vi har testomgångar hela tiden med superusers. Vi får feedback därifrån och den dialogen, mötet har vi med dem varannan vecka i alla fall. Det är inte bara jag och superanvändarna utan är det är även ledningen från kunden som är inblandade där och sånt. Det är en ganska bra dialog, vi får mycket feedback. Men vi får ibland väldigt kortfattad feedback och ibland skrivet på fel sätt. Det är ju inga vana testare som sitter där, det är ju vanliga affärsmänniskor. "* - Informant 2

Sprint Review:s är också ett positivt kommunikationsmoment då den frekventa dialogen genererar en kvittens på ifall Development Team förstått sig på kundens krav, eller ifall någonting behövs förändras.

> *"Att ha två veckors Sprintar med avslutande feedback om vad som behövs ändras osv. Det är nog det bästa sättet tycker jag att säkerhetsställa att man har förstått uppgiften korrekt och att dem känner sig nöjda med lösningen och sådär"*
> - Informant 1

Att ha Sprint Review med korta mellanrum möjliggör för Development Team:et att förändra kraven ifall det framkommer att de missförstått kunden eller ifall kunden ändrat sig.

“Att ha närmare kontakt med kunden och ha Sprint Reviews efter varje sprint, låta kunden se progress, gör att vi kan och hinner ändra sig och sådär. Gör att vi kan vara tillmötesgående, flexibla och vara förändringsbenägna”
- Informant 2

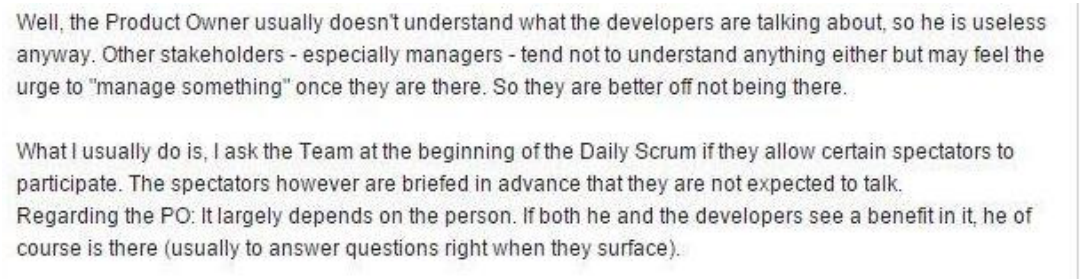
4.2 Kommunikation som förmedlar krav

Informanterna menar att Daily Scrum kan hjälpa Scrum Team:et att hantera problem de stöter på, samt att alla inom teamet får en gemensam bild över hur projektet fortlöper. Mötet ställer också krav på att medlemmarna producerar, då mötet kräver att deltagarna ska förmedla vad som gjorts.

“Det som är bra med det mötet är att man kan hantera problem som andra uttrycker, samtidigt som man själv kan få hjälp med sina problem. Man får även en inblick i hur projektet går framåt. Även ifall vi sitter i en samlad miljö och har insikt i hur alla jobbar och hur projektet utvecklas, så får man en ännu större inblick tack vare mötena. Det ställer också lite krav på en själv att på ett sätt visa vad man gjort.” - Informant 3

“Jag tycker det är superviktigt att det ska vara en daily scrum varje dag. Alla vågar uttrycka sina problem och så.” - Informant 2

En Product Owner bör inte delta på Daily Scrum:et, säger en informant på Forumet som svar på en fråga. Detta för att Product Owner:n sällan besitter de tekniska kunskaperna för att förstå sig på den terminologin som används i mötena. I vissa fall kan Product Owner:n få medverka, med Development Team:ets godkännande, samt vetskapen att Product Owner:n endast är där för att observera och svara på frågor.


Well, the Product Owner usually doesn't understand what the developers are talking about, so he is useless anyway. Other stakeholders - especially managers - tend not to understand anything either but may feel the urge to "manage something" once they are there. So they are better off not being there.

What I usually do is, I ask the Team at the beginning of the Daily Scrum if they allow certain spectators to participate. The spectators however are briefed in advance that they are not expected to talk.
Regarding the PO: It largely depends on the person. If both he and the developers see a benefit in it, he of course is there (usually to answer questions right when they surface).

“Som Product Owner är jag inte delaktig i daily scrum såvida dem inte bjuder in mig. Dem kör det varje dag men inte på fasta tider.“ - Informant 1

En informant på Forumet poängterar en problematik som kan uppstå ifall Product Owner:n medverkar i mötet, vilket är att medlemmar i Development Team kan uppleva att mötet handlar mer om att rapportera projektets framgång än att fokusera på syftet.


I had a poor experience in the past when the PO was also the part of the Development Team thus he attended the meeting. The Team felt as they were reporting to the PO about the status.

En annan problematik som kan uppstå med Scrum är att Product Owner:n inte alltid besitter den kunskap eller den tid som krävs för att agera som Product Owner.

> "Kunden har sällan *varken tiden eller kunskaperna för att agera som Product Owner." - informant 2*

Några informanter på Forumet tar även upp att det kan uppstå problematik ifall Product Owner inte finns tillgänglig för kommunikation. Att Development Team måste göra antaganden för att se till så att projektet fortlöper enligt tidsplanen. Dessa antaganden kan i ett senare skede visa sig vara felaktiga.

> A Product Owner should be committed to the delivery of product value, and should spend enough time with the Development Team to meet that commitment.
>
> The other option is to share what impact the PO not being available is having on the team and its productivity. Such things can be - time spent waiting for a decision, time spent going back and changing something as an assumption was made and then 3 days later the PO was available and made a decision in a different direction.

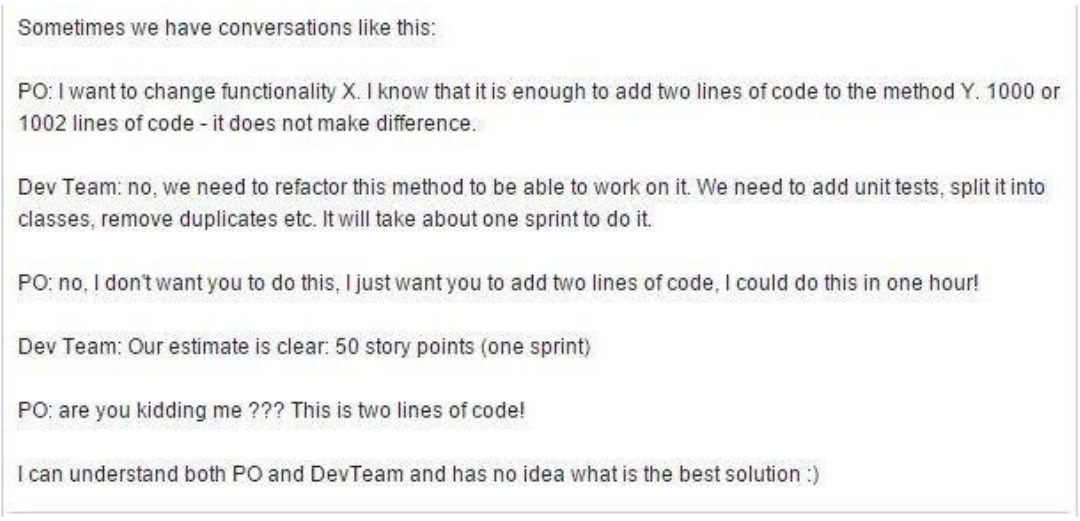
Informant 1 går vidare in på att kunden ofta har svårt att uttrycka behovet, då de glömmer att nämna vem som ska använda funktionen samt av vilken anledning.

> *"Kraven är väldigt ofta utformade som lösningsförslag från deras sida. De glömmer väldigt ofta att belysa saker som behov, och varför det ska göras och av vem, speciellt varför. Väldigt ofta får man prata med kunden om detta."-* Informant 1

Svårigheterna kunderna upplever kan bero på att det ofta är personer som är ovana att agera som Product Owner:s, som ska förmedla kraven om systemet till Development Team. Att ta fram bra formulerade User Stories är även ett av de svåraste momenten. Scrum Guide är även tyst om hur detta moment ska genomföras.

> *"Det är ofta nya personer som ska skriva krav till systemet, och det är inte alltid att det funkar. Att ta fram bra krav, eller asså user stories som har en affärsnytta. Det är ju det som är det svåraste. Har man det hemma så...Och där säger ju Scrum inte hur man ska gå tillväga"* - Informant 3

Eftersom det oftast är kunden som har rollen som Product Owner, så har Product Owner:n inte alltid förståelse för arbetet som krävs för att uppfylla ett krav. Detta medför att det kan uppstå kontroverser mellan Product Owner:n och Development Team:et. I exemplet nedan som en informant på Forumet uttryckt, har Product Owner:n en bild över vad han vill få gjort, men förstår inte arbetet som krävs för att uppfylla kravet.


Sometimes we have conversations like this:

PO: I want to change functionality X. I know that it is enough to add two lines of code to the method Y. 1000 or 1002 lines of code - it does not make difference.

Dev Team: no, we need to refactor this method to be able to work on it. We need to add unit tests, split it into classes, remove duplicates etc. It will take about one sprint to do it.

PO: no, I don't want you to do this, I just want you to add two lines of code, I could do this in one hour!

Dev Team: Our estimate is clear: 50 story points (one sprint)

PO: are you kidding me ??? This is two lines of code!

I can understand both PO and DevTeam and has no idea what is the best solution :)

Informant 2 berättar att Konsultbolaget har utvecklat en egen roll för tackla problematiken som kan uppstå när kunden har rollen som Product Owner. Kundernas Product Owner har sällan den kunskap, energi eller tid som krävs för att agera som Product Owner, vilket kan leda till att Product Backlog:en blir dåligt utformad. Rollen de utvecklat har döpts till Product Owner Proxy och fungerar som en länk mellan kundens Product Backlog och Development Team.

> *"Min roll är då, jag sitter som Product ägar proxy, kallar vi det för. Det är en roll vi utvecklat själva. Rollen har kommit till för att kundens Product Owner sällan har kunskapen om kravhantering eller tiden för det. Då finns det risk för att Product backlog:en blir dåligt utformad, man inte riktigt vet vad man vill ha. User Stories är väldigt dåligt beskrivna så utvecklarna mer får anta vad som ska göras. Det finns också risk att Product Ownern inte har tid till att intressera sig för projektet och svara på frågor. Då fungerar Product Owner Proxyn som stöd för allt detta."*
> *- Informant 2*

Product Owner Proxy är anställd hos Konsultbolaget och är den enda personen som äger Product Backlog:en. Men Product Owner Proxy har inte rättigheter att uppdatera Product Backlog:en utan kundens Product Owner:s medgivande.

> *"Proxyn är anställd hos oss, och är den enda som äger product backlog. Självklart så får en ändring inte göras utan Product Owner:s medgivande. Det förs alltid en kontinuerlig diskussion mellan Proxy:n och Product Owner:n innan nått förändras i Back Log:en." Informant 2*

Vidare tar Informant 3 upp att Product Owner Proxy:n sitter tillsammans med Scrum Team:et eftersom att den riktiga Product Owner:n inte har tid med detta eftersom att den sitter hos kunden. Detta medför att Proxy:n kan iaktta och fånga upp saker på ett lättare sätt.

”Egentligen kan man säg att jag är Product Owner:n inom vårat Scrum Team och fördelen med att jag sitter med teamet hela tiden är att jag fångar upp mycket mer saker. Men den riktiga product ägaren sitter hos kund då va.” - Informant 3

Informant 2 belyser dock att det kan uppstå en problematik med tanke på att Product Owner Proxy är anställt av Konsultbolaget. Detta kan medföra att Product Owner Proxy:n främst jobbar i Konsultbolagets intresse.

“Men det finns lite problematik med min roll eftersom jag sitter på utvecklarsidan när jag egentligen borde sitta på kundsidan. Skulle en utvecklare vilja ta en genväg, så vet jag ju om det till exempel.” - Informant 2

Product Owner Proxy:n medverkar även i moment som handlar om att ta fram estimat för varje krav i Product Backlog:en. Estimaten görs inledningsvis i form av points.

“När man gör estimat för att estimera sin backlog innan man kommit så långt att man estimerar i timmar så ger man dem point genom att titta på komplexiteten och att titta på relationerna mellan varandra” - Informant 2

Estimat görs för att de involverade parterna ska få en överblick över hur lång tid de olika kraven tar och när systemet estimeras att kunna lanseras. Konsultbolaget berättar om sin processen att ta fram estimat för kraven och beskriver att Development Team gemensamt får rösta efter att Product Owner Proxy:n har förklarat kraven för dem. Ifall det uppstår stor differens mellan rösterna inleds en diskussion där medlemmarna får motivera sitt val. Ett estimat beslutas när Development Team är eniga.

“Vi gör för att se hur lång tid kraven tar och för att se när vi kan lansera. För att räkna ut ett estimat för ett krav så röstar varje medlem i utvecklarteamet individuellt på en siffra. När alla medlemmar bestämt sig för en siffra så visar alla upp dem samtidigt. Har alla medlemmar satt ungefär samma siffra så går det fort att bestämma estimatet. Om någon/några däremot skiljer sig radikalt får alla medlemmar argumentera för varför dem satt sin siffra. Då kanske det visar sig att vissa inte tänkt på vissa aspekter. Efter diskussionen får medlemmarna sätta en siffra igen och samma process utförs igen. Då kanske resultatet blir annorlunda. Men ofta så har faktiskt medlemmarna samma bild över hur lång tid något tar så det är sällan dem skiljer sig radikalt från varandra.” - Informant 2

Många användare av Scrum finner svårigheter om hur kommunikationstekniker så som User Stories används för att förmedla krav. En informant på Forumet understryker problematiken med Product Owner och dess bristfälliga kunskap om hur User Stories används.

I try to use Scrum for the first time. The Product Owner was ordered to write his User Stories but what I got was something like:

1. The Tool must not have any problems with special characters
a. The filename for each company has the following structure: companyname_id.xlsx
b. Name of the first Worksheet is "Code"
- Name of the first column is: xxxx

Scrum Guide är inte alltid tydlig i hur olika moment inom Scrum ska utföras. Denna vaghet medför ibland problem för Scrum-utövare eftersom User Stories inte är en del av Scrum. User Stories är en kompletterande kommunikationsteknik för att förmedla krav och inte en del av Scrum.

Important to note that the User Stories practice is not part of Scrum, but is a complementary practice to Scrum. Many teams use US to represent PBI's.

"User stories är inte en del av Scrum men user stories är ett ganska vedertaget sätt att förmedla krav på. Hur den används står inte i Scrum" - informant 2

En annan vaghet med Scrum som en informant på Forumet belyser är hur Scrum Team:et ska vara placerade. Det råder delade åsikter om ifall Scrum Team bör vara placerade i en lokal miljö eller kan vara spridda över världen. En informant argumenterar att dagens teknologi medför att samlokalisering inte längre är nödvändigt för att kunna samarbeta och kommunicera.

I have a really hard time with one facet of Scrum that I just do not agree with, and that's the collocation concept. I know that collocation is an important concept and that teams may be more productive if located together how does an organization account for remote employees (no nearby office) and offshoring possibilities? I just don't see collocation being a necessary, or even a viable solution anymore in a virtual world. Does anyone else out there think the collocation bias needs to change?
FYI - (cross posted in LinkedIn - Scrum Practitioners Group)

En annan informant understryker att Scrum Guide inte uttrycker förbud mot varken centraliserat eller ett decentraliserat team. Men de flesta informanter verkar eniga om att centraliserat är att föredra.

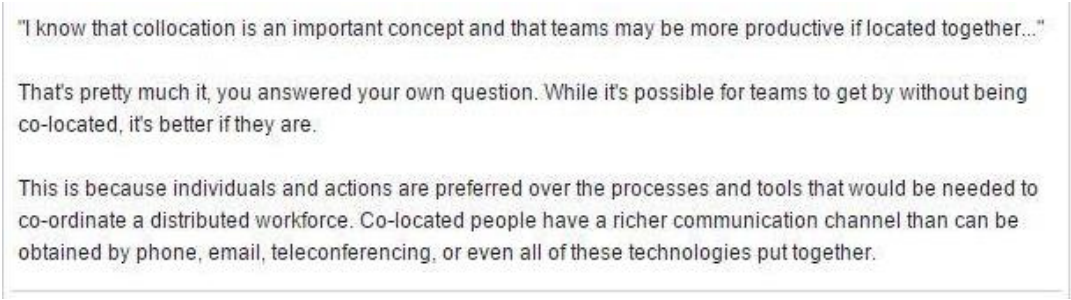
While its always favorable, I didn't find a mandate where Scrum requires a co-located team to to produce an increment.

My curiosity had me checking the Scrum Guide and didn't see anything for this in Page 15 where the Daily Scrum is described.

I've worked with several different team set ups, and I found being co-located or at keadt in the same time zone was the most favorable.

But I agree with Nitin, I can't find any part where scrum "forbiddes" colocating teams.

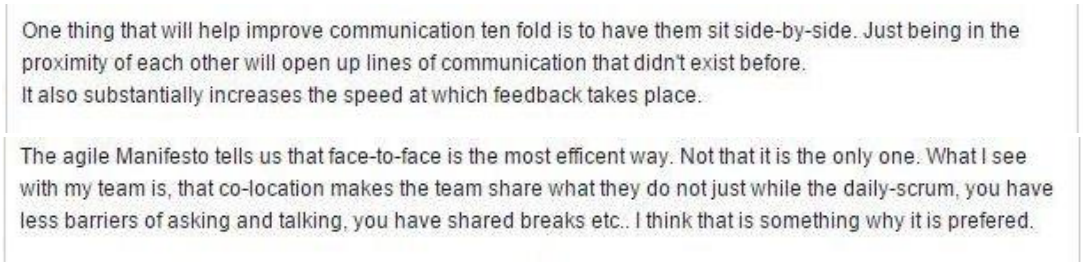
Informanter från Forumet uttrycker även att ett centraliserat team får mycket rikare kommunikation än vad ett decentraliserat team kan få via till exempel email, telefonsamtal och telefonkonferens.


"I know that collocation is an important concept and that teams may be more productive if located together..."

That's pretty much it, you answered your own question. While it's possible for teams to get by without being co-located, it's better if they are.

This is because individuals and actions are preferred over the processes and tools that would be needed to co-ordinate a distributed workforce. Co-located people have a richer communication channel than can be obtained by phone, email, teleconferencing, or even all of these technologies put together.

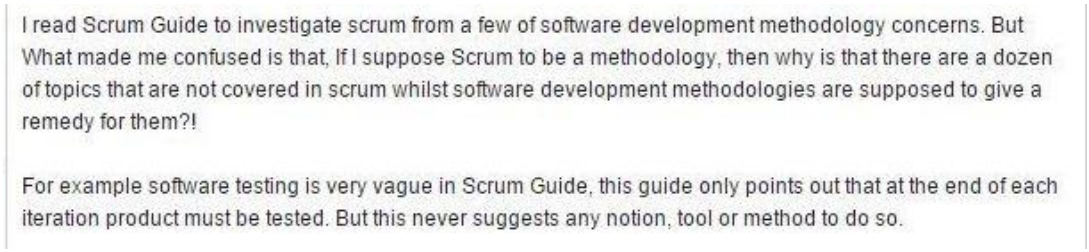
Informanter belyste även en annan fördel med ett centraliserat team. Fördelen de belyste var att teamet får en tendens att ständigt överföra kunskap mellan varandra via daglig och muntlig kommunikation, istället för endast vid Daily Scrum. Enligt informanterna blir kommunikationen i ett samlokaliserat team ofantligt mycket bättre än vid ett decentraliserat där feedbacken kan dröja.


One thing that will help improve communication ten fold is to have them sit side-by-side. Just being in the proximity of each other will open up lines of communication that didn't exist before.
It also substantially increases the speed at which feedback takes place.

The agile Manifesto tells us that face-to-face is the most efficent way. Not that it is the only one. What I see with my team is, that co-location makes the team share what they do not just while the daily-scrum, you have less barriers of asking and talking, you have shared breaks etc.. I think that is something why it is prefered.

"...för vi sitter och kommunicerar med varandra varje dag hela tiden ändå. Är det någonting som dyker upp så tar vi det direkt. Detta är fördelen med att sitta och jobba tillsammans." - Informant 1

Många användare av Scrum har uppmärksammat denna beskrivna vaghet och har därför inlett en diskussion där informanterna argumenterar för att Scrum inte är en metodologi utan ett ramverk. En informant inled diskussionen med en förvirring där han skrev att Scrum är en metod, men vid noggrannare analys finner att Scrum egentligen är ett ramverk.


I read Scrum Guide to investigate scrum from a few of software development methodology concerns. But What made me confused is that, If I suppose Scrum to be a methodology, then why is that there are a dozen of topics that are not covered in scrum whilst software development methodologies are supposed to give a remedy for them?!

For example software testing is very vague in Scrum Guide, this guide only points out that at the end of each iteration product must be tested. But this never suggests any notion, tool or method to do so.

Ett flertal informanter svarar på hans förvirring och skriver att de håller med i hans resultat om att Scrum egentligen är ett ramverk. De motiverar det med att Scrum inte är beskrivande. Användare

av Scrum behöver själva lokalisera de tekniker som krävs för att uppfylla de olika momenten i Scrum.

> +1 to what Fredrik said, and I would only add that Scrum is not a methodology. It is a **framework** within which multiple techniques, practices, and methodologies can be used. The Scrum Guide, the official definition of Scrum, is simply a framework, with intentional gaps or variation points to let the implementer decide what is best for their team/product context.
>
> As Charles has said, Scrum is not a methodology hence it is not prescriptive. You should find the missing pieces and find what is important for your software yourself and plug it in the Scrum framework.

5. Diskussion

Studiens syfte har varit att undersöka och analysera hur agila systemutvecklingsmetoder, med Scrum som granskningsobjekt, används för att att säkerställa kunskapsöverföring i systemutvecklingsprojekt. Teorin har behandlat hur kommunikation är nyckeln för kunskapsöverföring och hur krav är kopplat till kunskap samt hur Scrum används för att hantera denna kommunikation. Med hjälp av fallstudien på Konsultbolaget och Forumet, har studien även analyserat hur Scrum används för att säkerställa kunskapsöverföring. I detta avsnitt diskuterar studiens författare resultatets betydelsefulla delar mot bakgrund av teorin.

Efter att ha analyserat de intervjuer som utförts hos Konsultbolaget och observationer på Forumet går det att konstatera att området är komplext. Det finns ingen silverkula för hur system ska utvecklas eller för hur människor ska kommunicera på bästa sätt. Det finns däremot hjälpande verktyg i form av metoder och tekniker för att underlätta kommunikationen. Komplexiteten beror på att det i grund och botten är människor som interagerar med både teknik som andra människor. Alla människor är olika, därför passar inte alla kommunikativa metoder eller tekniker alla människor.

Scrum Guide, som är den officiella beskrivningen av Scrum, är medvetet vag på vissa delar. Guiden beskriver till exempel inte hur krav ska samlas in, men den beskriver att det ska finnas en Product Backlog med prioriterade krav. Vidare förklarar också Scrum Guide att Scrum Team ska utföra Daily Scrum:s och Sprint Review:s, men ingenting om hur detta ska gå till. Appan och Browne (2012) tar upp att utvecklingsmetoder ofta förespråkar att användare är med och medverkar i kravprocessen. Däremot menar författarna att metoderna ofta inte säger hur denna process ska gå till och hur denna kommunikation ska uppnå ett effektivt resultat. Hela processen att involvera användare riskerar då att bli ineffektiv (Appan & Browne, 2012). Detta stämmer överens med studiens empiri då användare av Scrum argumenterar för att Scrum inte är en metodologi, utan mer fungerar som ett ramverk och planeringsverktyg, eftersom Scrum uttryckligen inte säger hur de olika momenten ska genomföras.

Denna vaghet medför att moment i Scrum riskerar att förlora sin poäng om individer som är främmande med Scrum helt själva får avgöra hur momenten ska utföras. Resultatet visar på ett sådant exempel, där medlemmar i ett Scrum Team blivit utplacerade på olika destinationer eftersom projektets ledare ansåg att det var irrelevant att i dagens virtuella verklighet jobba i en fysisk sammanförd miljö. Argumentet uppdagades på Forumet och ett antal Scrum-experter svarade enat att Scrum Team bör sitta i en gemensam miljö för att på bästa sätt kunna kommunicera och utföra Sprint:ar så effektivt som möjligt. I intervjuerna med Konsultbolaget betonade även informanterna vikten av kommunikation i Scrum Team. Informanterna menade att

de föredrog när kommunikationen skedde i en samlad miljö och där kunskapsöverföringar i form av socialization och externalization dagligen sker på ett naturligt sätt. De finner att den sorts kommunikation är mycket mer informationsrik och uppskattar att den uttrycks i rätt tidpunkt. Sitter de utspridda upplever de en stor försening av kunskapsöverföring, vilket påverkar deras arbete negativt. Nonaka (1994) menar att socialization är en tacit kunskapsöverföringsprocess som oftast sker inom ett samlat team. Den processen sker genom att medlemmarna i team:et utbyter erfarenheter genom att exempelvis beskåda, tolka och härma varandra. Vidare förklarar Nonaka (1994) att denna process kräver ett gemensamt forum där de kan kommunicera med varandra i samma tid och rum. Shannon och Weaver (1949) lyfter även fram att meddelanden påverkas av den fysiska världen så fort den omvandlas från en persons tanke till den fysiska världen. En stor distans mellan sändare och mottagare kan även leda till fler möjliga tekniska störningar, som exempelvis dålig uppkoppling eller brist på visuell kommunikation vid samtal med telefon (Shannon & Weaver, 1949).

Med detta i åtanke bör Scrum Guide rekommendera att Scrum Team vistas i en gemensam miljö. Skulle det vara så att Scrum Team behöver ha utvecklare placerade på olika destinationer så hade det även varit bra om Scrum Guide:en förklarar alternativa sätt för Scrum Team att kommunicera. Det finns exempelvis kommunikationskanaler som email, Skype, telefonsamtal och chatt som kan användas för att kommunicera via distans. En kombination mellan audiella och visuella kanaler är att rekommendera eftersom att en individs kroppsspråk då också inkluderas. Dessa exempel är dock mer tidskrävande och kräver fler moment än samtal i en lokal miljö. Dessutom belyser (Shannon & Weaver, 1949) att det även finns risk för ännu mer tekniska och semantiska störningar vid användandet av dessa kanaler.

Scrum Guide är även tyst om hur processen för att samla in och förfina krav ska gå till. Detta möjliggör för Scrum Team att själva välja vilken kommunikationsteknik som passar bäst för ändamålet. Enligt Eriksson (2007) är User Stories en vedertagen teknik för att förmedla och kommunicera krav på, som beskriver vem som ska göra vad och varför. En annan teknik är att hålla i en workshop där deltagarna får brainstorma idéer (Eriksson, 2007). Studien empiri har däremot visat att det finns användare av Scrum som inte känner till alla de olika tekniker som finns för att samla in, förfina och kommunicera krav på, vilket kan leda till att användare inte väljer den mest lämpade tekniken. Scrum Guide:ns brist på rekommendationer av tekniker kan alltså leda till att dessa moment inte genomförs på bästa möjliga sätt.

Scrum Guide:ns vaghet inom momenten gör också Guiden tidlös, vilket betyder att rekommendationer på tekniker och verktyg inte behöver uppdateras. Guiden kan istället för att vara tidlös, ge uppdaterade rekommendationer på tekniker för att lösa de olika momenten som Scrum Guide uttrycker ska finnas med. Då får användare av Scrum ett bra stöd att luta sig emot ifall de inte känner till någon mer passande teknik.

En Product Owner äger Product Backlog:en och har ansvaret för dess prioriteringar och förändringar. Det är oftast också kunden som är Product Owner i systemutvecklingsprojekt (Deemer et al, 2012; Schwaber, 1997; Softhouse, 2006). I studiens resultat identifierades dock att kunder i många fall varken besatt kunskap, tid eller energi för att agera som Product Owner. Detta har visat sig kunna leda till misskommunikation mellan Development Team och Product Owner. Appan och Browne (2012) menar även att misskommunikation kan påverka kraven negativt eftersom att kommunikation mellan utvecklare och användare har en stor påverkan på hur gedigen kravdokumentationen blir. Studiens empiri visar även att bristfällig kunskapsöverföring mellan Product Owner, Scrum Master och Development Team kan leda till osämja mellan parterna. Appan och Browne (2012) menar vidare att misskommunikation även kan leda till stora implikationer på design, strategi och teknikval. En god kommunikation mellan utvecklare och

användare medför istället en större chans till att parterna får en gemensam bild över problemet och lösningen. För att möjliggöra en god kommunikation mellan utvecklare och användare så bör en Product Owner ägna så mycket tid som möjligt med Development Team. I resultatet framgick det dock att en Product Owner inte alltid har tid att vara närvarande. Resultatet visade även på brister i Product Owner:ns förmåga att skriva och förmedla krav. Detta kan i sin tur resultera i att Development Team:et inte förstår innebörden av dessa krav.

Studiens empiriska resultat presenterar en lösning på denna problematik. En lösning som inte teorin behandlar. För att hantera problematiken med bristfällig tid och kunskap hos Product Owner:n har Konsultbolaget utvecklat en egen roll. Konsultbolaget kallar rollen för Product Owner Proxy och rollen fungerar som en länk mellan kundens Product Owner och Konsultbolagets Development Team. Product Owner Proxy:n äger Product Backlog:en tillsammans med Product Owner:n och förändringar i Product Backlog:en får inte utföras utan bådas medgivande. För att undvika missförstånd och hålla en god struktur i Product Backlog:en så är det endast Product Owner Proxy:n som fysiskt får göra förändringar i Product Backlog:en. Product Owner Proxy:n ansvarar även för att förmedla och skriva Product Owner:ns krav på ett sätt som Development Team förstår. Resultatet visar att kraven ofta förmedlas via User Stories. I resultatet framgick det även att konfrontationer kan uppstå mellan Development Team och Product Owner:n. Då kan Product Owner:n fungera som en medlare mellan parterna. Det finns dock en problematik med rollen, då personen inte är anställd hos kunden utan hos Konsultbolaget. Risken finns att Product Owner Proxy:n inte agerar i kundens intresse på grund av lojalitet mot den egna arbetsgivaren. En informant på Konsultbolaget förklarar till exempel att hen ibland kan upptäcka att Development Team försöker ta genvägar i systemutvecklingen. Då kan Product Owner Proxy:n få svårt att avgöra ifall hen ska rapportera detta till kunden och eventuellt skada sitt egna bolag, eller se genom fingrarna och eventuellt skapa missförtroende hos kunden.

Det är viktigt att individer har en bra kommunikation för att kunna samarbeta (Appan & Browne, 2012). I många fall krävs det verktyg för att det ska vara naturligt för individer att göra detta. Daily Scrum är ett sådant verktyg som tillåter samtliga deltagare att kommunicera (Deemer et al, 2012; Schwaber, 1997; Softhouse, 2006), till skillnad mot ett vanligt möte där deltagare lätt kan bli bisittare och inte har något krav på att aktivt delta. Daily Scrum ska även utföras varje dag (Deemer et al, 2012), vilket ökar kommunikationen och förståelsen för uppgiften. Andra traditionella systemutvecklingsmetoder har inte bestämda dagliga möten med krav på kommunikation på det sättet som Scrum har. Detta kan medföra att kommunikationen inom teamet inte sker lika naturlig, eller inte alls, vilket kan påverka projektet negativt. Studiens resultat belyste också vikten av kommunikation inom teamet, och berömde Daily Scrum som ett verktyg för att uppfylla det.

Resultatet visade att Product Owner:n ibland var delaktig i Daily Scrum, trots att Scrum Guide:n uttrycker att denne inte bör vara med (Softhouse, 2006). Slutsatser kan dras från studiens resultat att Product Owner:ns deltagande i mötet kan ha en negativ påverkan på mötet. Resultatet visade nämligen att det finns en risk att Development Team kan uppleva att de rapporterar status på projektet och inte vågar ta upp eventuella problem de stött på, ifall Product Owner:n deltar på mötet. Detta kan leda till misskommunikation och att mötet riskerar att förlora sin poäng. Det finns också en risk att mötets fokus hamnar på att behöva förmedla kunskap om tekniska detaljer för Product Owner:n. Om däremot Product Owner verkligen behöver delta bör denne inte uttrycka sig, utan endast vara en bisittare. Då kvarstår dock problematiken med att mötets deltagare inte vågar uttrycka sig till fullo.

Schwaber (1997) förklarar att ett effektivt sätt att säkerhetsställa att utvecklare och användare har samma gemensamma bild över problemen och lösningarna är genom att ha regelbundna

uppföljningar där kunden och dess användare får testa och granska delprodukter av systemet. På ett sådant möte kan kunden även få se att projektet går framåt och kunden får även se en visuell representation av kraven som sammanställts (Deemer et al, 2012; Schwaber, 1997; Softhouse, 2006). Detta uppföljningsmöte är även positivt för utvecklarna då de får en bild över ifall de förstått sig på kunden eller inte. Har utvecklarna missförstått något får de en tidig feedback på vad som bör ändras (Softhouse, 2006). Scrum har ett moment som kallas för Sprint Review, som möjliggör detta. Enligt Schwaber (1997) ska detta möte utföras efter varje sprint som varar mellan två-fyra veckor. Studiens empiriska data visade på att en Sprint vanligtvis pågår i två veckor med ett avslutade Sprint Review möte. Det framgick även i resultatet att användare av Scrum ansåg att Sprint Review var ett enastående sätt att se hur nöjd kunden är med delprodukten, se ifall utvecklarna och kunden har samma problem- och lösningsbild, samt ett bra sätt att samla in feedback och förbättringsförslag på. Det gick också att urskilja vissa nackdelar med Sprint Review ur studien empiri, vilket var att feedbacken ibland kunde vara kortfattad eller skriven på sätt som utvecklarna inte föredrog. Detta grundar sig i att personerna som testar systemet inte är professionella testare. Sprint Review:n är å andra sidan inte tänkt för att identifiera buggar (Schwaber, 1997). Mötet är som tidigare nämnt till för att användare av systemet ska få en chans att se hur systemet utvecklas och ge feedback i tidigt skede (Softhouse, 2006). Både studien teori och studien empiri påstår att detta är fördelaktigt då det möjliggör för utvecklarna att ändra riktning och ändra funktioner i tidigt skede istället för att vänta längre perioder eller till hela systemet är klart som andra traditionella systemutvecklingsmetoder förespråkar.

Något som blev tydligt i det empiriska materialitet som även Shannon och Weaver (1949) belyser är att det är viktigt att kommunikationen utförs på ett korrekt sätt med få störningsmoment för att skapa en gemensam bild mellan involverade parter. Vidare menar empirin att både Product Owner och Development Team tillsammans bör vara delaktiga i alla moment som rör Product Backlog:en för att tillsammans skapa en gemensam bild över vad som ska utvecklas. Enligt Schwaber (1997) går dessa parter igenom Product Backlog:en innan varje Sprint avslutas i en så kallad Product Backlog Refinement, för att uppdatera och förfina kraven samt ge dem nya estimat. När en Sprint väl har påbörjats görs inga ändringar, detta för att Development Team:et helhjärtat ska kunna fokusera på uppgiften (Deemer et al, 2012).

Enligt (Deemer et al, 2012) är estimat för krav till för att ge Scrum Team:et och kunden en uppfattning om hur lång tid det tar för Development Team att utveckla de olika kraven. Både i Deemer et al (2012) artikel och studiens resultat går det att urskilja att estimeringsprocessen där Product Owner tillsammans med Development Team sätter ut estimat för varje enskilt krav i form av points är ett bra sätt för utvecklare och kund att få en gemensam bild över problem och lösningar för systemet. Är teamets Velocity okänt blir estimaten inte lika användbart.

I studiens empiri framgick det att transparens kan generera positiva aspekter. Konsultbolaget använder sig av transparens genom att ha Product Backlog:en, Sprint Backlog:en samt användarnas feedback och förslag som berör systemet, synligt för alla inblandade parter. Konsultbolaget menar att en sådan transparens gör att alla parter får en gemensam bild över projektet eftersom de kan se vad som ska göras, vad som håller på att göras och när de olika kraven estimeras att vara klara. Det framgick vid intervjuerna hos Konsultbolaget att det dock finns risker med transparens då viktig information enklare kan spridas till bland annat konkurrerande intressenter. Detta kan medföra att kunden vill strama åt transparensen i projektet. Konsultbolaget menar även att kunden kan ställa sig negativt till transparens då dem får svårare att skylla eventuella misslyckande och förseningar på utvecklare då de hela tiden har access till projektets förlopp.

Avslutningsvis visar även resultatet att det finns en tydlig risk med att inkludera kunden kontinuerligt i utvecklingsprocessen. Olika intressenter hos kunden kan förvänta sig ett resultat som gagnar de själva vilket kan skapa problematik och förvirring för Development Team:et och Product Owner:n, samt skapa en negativ bild om lösningen. Då kan det vara bra att vara tydlig och kommunicera ut varför till exempel Product Backlog:en har prioriterats som den gjorts samt att även ta till sig och utvärdera dessa förväntningar. Det kan vara så att de involverade parterna missat ett behov.

6. Slutsats

Studien stödjer det faktum som teorin och resultatet belyser om att kommunikation bör betraktas som en faktor som har stor påverkan på systemutvecklingsprojektets utfall. Vidare går det att urskilja att kommunikation mellan människor besitter mycket komplexitet och att det inte finns någon silverkula som löser problematiken med misskommunikation.

Studiens frågeställnings löd: *“Hur används Scrum för att säkerställa kunskapsöverföring i systemutvecklingsprojekt?”*

Studiens frågeställning besvaras genom ett konstaterande att Scrum som agil systemutvecklingsmetod innefattar en rad kommunikativa moment som används och anpassas för att säkerhetsställa kunskapsöverföring i systemutvecklingsprojektet.

Dessa kommunikativa moment kan användas i iterativa processer i genom hela systemutvecklingsprojektet. De dagliga kommunikationsmötena samt de löpande utvärderingsmötena, säkerhetsställer kunskapsöverföring. Studien visade däremot på en stor vaghet i hur dessa kommunikativa moment ska utföras. Denna vaghet kan problematisera processer i systemutvecklingsprojekt. Detta har resulterat i vidareutveckling av roller med syftet att hantera problematiken med misskommunikation som vagheten skapar.

Studien syfte var att analysera hur agila systemutvecklingsmetoder används för att säkerställa kunskapsöverföring, och använde Scrum som granskningsobjekt. Syftet med studien går även att applicera på andra systemutvecklingsmetoder.

6.1 Förslag till vidare studier

1. Scrum är medvetet vag i hur momenten som metoden innehåller ska utföras. Denna vaghet kan ses som en positiv aspekt då Scrum kan skräddarsys för olika situationer. Som studien belyser kan denna vaghet även leda till problematik i kunskapsöverföring mellan involverade parter i systemutvecklingsprojekt. Vidare studier kan utföras i vilken grad vaghet i systemutvecklingsmetoder främjar eller hämmar systemutvecklingsprojekt.

2. Rollen Product Owner Proxy, som presenterats i studien, är en roll som skapats för att hantera problematiken med kommunikation. Vidare studier kan utföras för att identifiera nya lösningar som behandlar problematiken med misskommunikation.

7. Referenser

Appan, R. Browne, J. (2012). The Impact of Analyst-Induced Misinformation on the Requirements Elicitation Process. *MIS Quarterly*, 36(1), ss. 85-106.

Banker, R. Davis, G and Slaughter, S. (1998). Software Development Practices, Software Complexity, and Software Maintenance Performance: A Field Study*. Management Science*, 44(4), ss. 433-450

Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. United Kingdom: Addison-Wesley.

Brooks, F. (1986). *No Silver Bullet - Essence and Accident in Software Engineering.* University of North Carolina at Chapel Hill.

Deemer, P. Benefield, G. Larman, C. Vodde, B. (2012). *The Scrum Primer: A Lightweight Guide to the Theory and Practice of Scrum.* Version 2.0.

Eriksson, U. (2007). *Kravhantering för IT-system.* Denmark: Studentlitteratur

Escalle, C.X., Cotteleer, M.J. and Austin, R.D. (1999). *Enterprise Resource Planning ERP: Technology Note.* Boston: Harvard Business School Publishing

Fenton, N., S. L. Pfleeger, and R. L. Glass. (1994). Science and Substance: A Challenge to Software Engineers. *IEEE Software*, 11(4), ss. 86- 95.

Highsmith, James A. (2004). *Agile project management: creating innovative products*. Boston, MA: Addison-Wesley

Hove, S. E. & Anda, B., (2005). Experiences from Conducting Semi-Structured Interviews in Empirical Software Engineering Research*.* Software Metrics, 11th IEEE International Symposium. Como, Italien 1-1 September. 2005, ss. 10-23. DOI: 10.1109/METRICS.2005.24

Klein, H. K., & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS quarterly*, 23(1) ss. 67-94.

Langefors Börje. (1980). Infological models and information user views*. Information systems,* 5(1). ss. 17-32

Leveson, N. (1997). Software engineering: stretching the limits of complexity. *Communications of the ACM,* 40(2). ss. 129-131

Light, M. (2009). *How the Waterfall Methodology Adapted and Whistled Past the Graveyard, Gartner Research.* Report #G00173423, December 18.

Loftus, E. F., Donders, K., Hoffman, H. G., and Schooler, J. W. (1989). Creating New Memories that are Quickly Assessed and Confidently Held. *Memory and Cognition*. 17(5). ss. 607-616

Löwren, J & Stolterman, E. (2004). *Design av informationsteknik*. Lund: Studentlitteratur AB

Nandhakumar, J., & Jones, M. (1997). Too close for comfort? Distance and engagement in interpretive information systems research*. Information Systems Journal,* 7(2). ss. 109-131.
Nonaka, I. (1994). A Dynamic Theory of Organizational Knowledge Creation. *Organization Science*, 5(1). ss 14-37.

Norton, D. (2008). *The Current State of Agile Method Adoption*. Gartner Research. Report #G00163591, December 12.

Oakland, John. (2004). *Oakland on quality* management. England: Elsevier Science

Patel, R. & Davidsson, B. (2011). *Forskningsmetodikens grunder*. 2. uppl., Lund: Studentlitteratur AB.

Persson, B. (1997). *Kunskapsöverföring till yrkesverksamma inom landskapsarkitekternas arbetsfält.* Diss., Alnarp: Sveriges lantbruksuniv. Swedish University of Agricultural Sciences, Sweden. Alnarp

Robey, D., Ross, J. W., & Boudreau, M. C. (2002). Learning to implement information systems: An exploratory study of the dialectics of change. *Journal of Management Information Systems* 19(1). ss. 17– 46.

Schwaber, K. (1997). *Scrum Development Process.* Advanced Development Methods. Burlington

Shannon, C. E. and W. Weaver (1949). *The Mathematical Theory of Communication*. Urbana: University of Illinois Press.

Softhouse. (2006). *Scrum in five minutes.* Malmö: Softhouse Consulting

Sommerville, I & Sawyer, P. (1997). *Requirements engineering: a good practice guide.* New York: John Wiley & Sons Inc

Stein, Johan. (1996). *Lärande inom och mellan organisationer.* Lund: Studentlitteratur.

Umble, E.J & Umble, M.M. (2002). *Avoiding ERP Implementation Failure*. Industrial Management 44(1). ss. 25-34.

Walsham, G. (1995). *Interpretive case studies in IS research: nature and method.* European Journal of information systems, 4(2). ss. 74-81.