LUNDS UNIVERSITET
Ekonomihögskolan

Agil systemdokumentation

– En fallstudie i Scrum

Kandidatuppsats, 15 högskolepoäng, SYSK02 i informatik

Framlagd:	Maj, 2014
Författare:	Martin Lindqvist Oskar Nielsen
Handledare:	Magnus Wärja
Examinator:	Anders Svensson Nicklas Holmberg

Abstrakt

Titel: Agil systemdokumentation – En fallstudie i Scrum

Författare: Martin Lindqvist
Oskar Nielsen

Utgivare: Institutionen för informatik, Ekonomihögskolan vid Lunds Universitet

Handledare: Magnus Wärja

Examinator: Anders Svensson
Nicklas Holmberg

Slutseminarium: 2014-05-30

Uppsatstyp: Kandidatuppsats

Nyckelord: Kandidatuppsats**,** agil, Scrum, systemdokumentation, krav, arkitektur, design, skapande

Abstrakt: Med systemdokumentation som centralt begrepp utreder vi i studien hur agila projekt i Scrum värdesätter och hanterar skapandet av systemdokumentation. Systemdokumentation, per definition beskriver ett system och hur det är uppbyggt. Vi tittar på olika typer av systemdokumentation med ett antal aspekter och begrepp som vi sammanfattar i en undersökningsmodell, vilken varit utgångspunkten för vår empiriska undersökning. Den genomfördes i intervjuer med fyra IT-konsulter som tillsammans representerar alla roller i Scrum. Studien pekar på att Scrum stödjer skapandet av systemdokumentation genom att varje projekt kan implementera och anpassa en egen dokumentationsprocess som stämmer överens med kundens och organisationens krav på systemdokumentation.

Innehållsförteckning

1 Introduktion

Detta kapitel inleder studien med att redogöra för dess bakgrund, syfte och omfång.

1.1 Bakgrund

Sedan det agila manifestet publicerades i februari 2001 har en rad agila metoder presenterats. Agil betyder ungefär ”rörlighet” och är ett samlingsbegrepp för lättviktsmetoder som sätter vikt på flexibilitet och god mjukvara över hårt styrda processer och omfattande dokumentation (Highsmith, 2002). Bland agil metodik är Scrum den vanligaste att använda, där någon variant av Scrum används i mer än 70 % av fallen (VersionOne, 2013). Scrum är en metod som används sedan 1990-talet för att hantera komplex produktutveckling, som är både lättviktigt och lätt att förstå (Schwaber & Sutherland, 2013).

Traditionella Vattenfallsmetoden lägger stor vikt på dokumentation. Den är dokumentationsdriven och dokumentation ligger till grund för varje ny fas som ingår i projektet (MaRSDD, 2009; McConnell, 1996). Inom agil metodik betraktas dokumentation däremot som en aktivitet som fått stor kritik eftersom det inte skapar ett direkt värde (Ambler, 2012b). Författarna Highsmith & Cockburn (2001) var tidiga med publicera en vetenskaplig artikelserie om det agila manifestet. De påstår att agila projekt bör prioritera utveckling av mjukvara över en omfattande dokumentation (Highsmith & Cockburn, 2001).

I en vetenskaplig tidsskrift från IEEE Computer Society finner vi en kommentar till den ovan nämnda artikeln: *“We want to spend all our time coding. Remember, real programmers don’t write documentation.”* (Rakitin, 2001, pp.4). Forskning visar att det finns brister på design och dokumentation inom agil metodik. Vilket tros bero på att det finns en attityd hos utvecklare där de varken bryr sig, eller kan skriva en korrekt dokumentation (Prause & Durdik, 2012).

Dokumentation ses hos vissa som ett slöseri i projekt, eftersom de inte bidrar med ett direkt värde till slutprodukten. Trots detta, är det viktiga för projektets framgång, distribution, underhåll och vidareutveckling. Fastän dokumentation ofta ses ner på av utvecklare, bidrar det faktiskt med ett värde för bättre programvarukvalitét och underhåll (Prause & Durdik, 2012). Detta ger ett indirekt värde genom att underlätta kommunikation och förståelsen hos dem inblandade i utvecklingsteamet (Ambler, 2012b).

Vissa menar att dokumentation kan med enkelhet ersättas med verbal kommunikation. Alistair Cockburn är en av grundarna till den agila rörelsen, och han påpekar att det agila arbetssättet utnyttjar en verbal dialog eftersom människor har en möjlighet att förmedla information och idéer bättre än att skriva och läsa dokument (Highsmith & Cockburn, 2001). Annan forskning har även visat att det effektivaste kommunikationssättet är verbal dialog mellan individer vid en whiteboard (Ambler, 2012b). Men varje projekt behöver till viss del dokumentation och modeller, eftersom det finns avsevärda fördelar att ta till vara på, både sett till projektets deltagare och till dess intressenter (Ambler, 2012b).

Denna studie omfattar begreppet systemdokumentation, vilket omfattar ämnesområden som exempelvis dokumentation av krav, design och arkitektur.

1.2 Problemformulering

Det finns en viss problematik med systemdokumentation inom organisationer. Trots att systemdokumentation har fördelar, så betraktas inte det som en prioritet i agila projekt (Gustavsson, 2007). Scott Ambler (2002) skriver i boken *Agile Modeling* att en av de största missuppfattningarna med agila metoder är man tror sig inte längre behöva dokumentera. Detta kan tänkbart tillskrivas det agila manifestet, i vars värderingar betonar vikten av att prioritera utveckling över omfattande systemdokumentation (Cockburn 2001; Ambler 2002; Highsmith 2002).

Vid 2013 års upplaga av undersökningen "State of Agile" (VersionOne, 2013) angav 24 % av tillfrågade experter att bristen på systemdokumentation är en av de största bekymren vid införandet av agil metodik i deras egna verksamhet.

Prause och Durdik (2012) utförde en enkätundersökning om systemdokumentation i agila metoder på *The European Software Engineering Conference and the ACM SIGSOFT Symposium on Foundations of Software Engineering* (ESEC/FSE) under september 2011. Respondenterna var 37 stycken experter som representerade både det akademiska området och privata näringslivet. De fick frågor om vad de anser anledningen är till de problem som finns med design och systemdokumentation. Statistiken visade att alla respondenter från näringslivet anser att det finns en brist på design och systemdokumentation inom agila metoder. 68 % från akademin delar denna åsikt rörande systemdokumentation, och 92 % rörande design.

Prause och Durdiks (2012) studie ovan belyser denna problematik, att design och systemdokumentation har tydliga brister inom agila metoder, där design är det moment som flest är eniga om (100, respektive 92 %).

Studien visar vidare att anledningen till detta beror på att:

- 49 % anser att vissa utvecklare inte bryr sig om det
- 46 % anser att vissa utvecklare inte vet hur man utför det korrekt
- 46 % anser att vissa utvecklare saknar tid (Prause & Durdik, 2012).

Avslutningsvis tas problematik upp som berör olika aspekter som exempelvis ansvarsfördelning och prioritering i skapandet av systemdokumentation. Med tanke på de fördelar som systemdokumentation kan tillbringa projekt är det av stor betydelse att också kunna stödja detta i agila projekt. Då Scrum är den mest frekvent använda agila metoden (VersionOne, 2013), är det därmed viktigt att undersöka om Scrum faktiskt kan stödja skapandet av systemdokumentation med hänsyn till några av dessa nämnda aspekter. Därför väljer vi den följande forskningsfrågan:

Hur värdesätts och hanteras skapandet av systemdokumentation i Scrum?

1.3 Syfte

Studien syftar till att identifiera Scrums stöd för skapandet av systemdokumentation. Den syftar till att presentera hur agila projekt i Scrum värdesätter och hanterar skapandet av systemdokumentation.

1.4 Avgränsningar

Studien kommer att beröra dokumentation av kategorin systemdokumentation, men undersökningens främsta syfte är att undersöka dokumenteringen av följande begrepp: teknisk dokumentation, design och arkitektur, samt kravdokumentation. Systemdokumentation har även två andra begrepp, användar- och marknadsdokumentation. Denna studie kommer inte att behandla dessa två begrepp. Detta beslut tar vi på grund av att vi inte anser att användar- och marknadsdokumentation faller inom definitionen för systemdokumentation, som finns angiven i delkapitel 2.1.

Undersökningen inriktar sig på systemdokumentation i förhållande till tre aspekter och begrepp som vi identifierar och redogör för i undersökningsmodellen (se delkapitel 2.8).

1.5 Centrala begrepp

Agila och traditionella metoder som Vattenfallsmodellen eller Scrum har olika benämningar. I vår studie väljer vi att använda begreppen *metod* eller *metodik* för att beskriva de olika agila och traditionella utvecklings- eller projektledningsmetoder som hänvisas till. Vad de har gemensamt är att de alla är metoder, och för enkelhetens skull väljer vi därför att kalla de detta. Konsekvensen av detta val är att vi inte är exakta när vi hänvisar till metoden i förhållande till dess definition. Av detta skäl tillgås en tydlig definition av vad exempelvis Scrum är (se sektion 2.3.3).

2 Litteraturgenomgång

Detta kapitel presenterar en teoretisk grund som undersökningen behandlar. Kapitlet identifierar de viktiga begrepp, aspekter och problem som ligger till grund för det ämne som vi har för avsikt att undersöka. Vi redogör för dokumentationsbegreppets definition och syfte, och beskriver hur systemdokumentation har fungerat historiskt i traditionell metodik. Detta gör vi som en kontrast till den agila metodikens dokumentationsförfarande. Teorierna sammanfattas därefter, som står till uppgift att artikulera de viktiga begrepp och aspekter som senare diskuteras.

2.1 Dokumentation - Ett samlingsbegrepp

Dokumentation är ett vanligt förekommande ord som i stora drag har en otydlig definition. Inom kontextet informationsteknologi (IT) kan dokumentation exempelvis innefatta användarmanualer, tidsplaner eller kravspecifikationer. De är alla moment som traditionellt förekommer i utvecklingsprojekt och som faller inom ramen för dokumentation. För att undvika missförstånd har vi valt att dela upp begreppet dokumentation i två kategorier.

De två kategorierna för dokumentation är (1) *processdokumentation* och (2) *produktdokumentation*. Dessa begrepp bygger på Sommervilles (2010) definitioner för dokumentation. Dokumentation som behandlar projektet och processen har namnet processdokumentation, och dokumentation som behandlar produkten är produktdokumentation (Somerville, 2010). Nedan följer några exempel på dokument och olika typer av dokumentation som faller in under respektive kategori:

1. *Processdokumentation*
    - Tidsplaner
    - Projektplaner
2. *Produktdokumentation* (d.v.s. vanligtvis systemdokumentation i mjukvarusammanhang)
    - Teknisk systemdokumentation
    - Kravdokumentation
    - Design- och arkitekturdokumentation
    - Användardokumentation
    - Marknadsdokumentation

Först och främst behöver vi tydliggöra definitionen för produktdokumentation. Vi väljer att kalla kategorin för systemdokumentation eftersom studiens forskningsområde är informatik (mer specifikt mjukvaruutveckling). Vi har även en gällande definition för systemdokumentation, som lyder: *systemdokumentation hjälper att beskriva hur systemet är uppbyggt* (Barker, 2002; Somerville, 2010).

En ytterligare beskrivning av författaren Magnusson (2004) som förklarar att systemdokumentation utgår i en beskrivning av systemet ur ett perspektiv från de som utvecklat och förvaltat systemet. Om vi tar till exempel användardokumentation, syftar den till att hjälpa användare med att använda produkten och är skriven för systemets slutanvändare (Barker, 2002; Somerville, 2010). Andra typer av systemdokumentation är exempelvis systemdokumentation för krav, teknisk systemdokumentation, design/systemarkitektur, samt marknadsanalyser hur systemet ska lanseras på marknaden (Barker, 2002).

2.2 Systemdokumentation

Systemdokumentation är ett hjälpmedel för verksamheter när ett system skall underhållas och vidareutvecklas (Avison & Fitzgerald, 1995; Sommerville, 1992 refererad i Magnusson, 2004). Vidare underlättar systemdokumentation, i måttlig mängd, kommunikation, förbättrar kunskapsöverföring, bevarar historisk information, och uppfyller statliga och rättsliga krav (Rüping, 2003). I uppsatsen *Användningen och nyttan av systemdokumentation* (2004) skriver författaren att systemdokumentation har två övergripande varianter, extern systemdokumentation och inline-dokumentation.

Inline-dokumentation är en typ av systemdokumentation som finns i form av kommentarer i källkoden. Extern systemdokumentation är systemdokumentation som beskriver systemet ur en extern andrahandskälla, till exempel i textdokument och modeller (Magnusson, 2004). För att försäkra oss om att vi använder korrekt och konsekvent terminologi som är i linje med aktuell forskning gällande begreppet inline-dokumentering och de närliggande varianterna för dokumentering i källkod, väljer vi att kalla dessa för koddokumentation (DocForge, 2013; Ambler, 2012b).

Sammanfattningsvis har vi identifierat två dokumentationstyper, som är underordnade kategorin för systemdokumentation. Dessa är *extern systemdokumentation* och *koddokumentation.* Dessa är två begrepp som vi kommer återkomma till senare i studien.

2.3 Hur projekt dokumenteras

Highsmith (2002) poängterar att det viktiga är att inte dokumentera för dokumenteringens skull, utan efterlyser en balans. Ambler (2012b) påpekar att det även är viktigt att skriva dokumentet gentemot den som ska läsa det. Dokumentation ska vara lätt, men tillräckligt detaljerad för den berörde att förstå hur systemet är uppbyggt (Rüping, 2003; Cockburn, 2001; Ambler, 2012b).

Ambler (2012b) diskuterar användbarheten gällande koddokumentation och extern systemdokumentation, och ställer sig frågan hur dessa dokumentationstyper av systemdokumentation ska användas. I systemdokumentation som riktar sig till utvecklare, underlättar det att använda sig mer av koddokumentation, samt med en kort systembeskrivning (Ambler, 2012b). Rüping (2003) tillägger att om ett projekt har för avsikt att dokumentera sin kod eller funktionalitet är det bäst lämpligt genom dokumentering i källkoden. Separata externa dokument ska dock finnas för att beskriva system på högnivå, dessa omfattar dokument för systemöversikt, krav, samt design och arkitektur (Rüping, 2003).

Situationen är komplicerad eftersom systemdokumentationen har flera olika syften under ett systems livstid och följaktligen flera tilltänkta läsare (Magnusson, 2004). Därför måste det finnas systemdokumentation för de som inte förstår kod och vill ta del av informationen (Ambler, 2012b). Den kommer inte att enbart läsas av utvecklare, och där spelar den externa systemdokumentationens roll in (Magnusson, 2004).

I ett kapitel ur boken "*The Myth of No Documentation in Scrum Projects*" (2012) av Mitch Lacey, beskriver han att varje projekt behöver en viss mängd systemdokumentation. Vidare nämner han att det finns tre tillvägagångssätt när systemdokumentationen utformas i ett projekt. De är de följande:

1. Dokumentera i början
2. Dokumentera i slutet
3. Dokumentera medan projektet utförs.

Lacey (2012) förklarar att respektive tillvägagångssätt är en definierande karaktär för olika metoder. Inom den agila metodiken, och i fallet för Scrum, förespråkas det att till största del att dokumentera medan utvecklingen sker i projektet (Lacey, 2012). Lacey (2012) betonar att detta är ett generellt tillvägagångssätt som innefattar Scrum.

När det gäller att dokumentera i början, respektive i slutet, hänvisar Lacey (2012) till den traditionella metoden Vattenfallsmodellen. När man arbetar med Vattenfallsmodellen så dokumenteras det mesta antingen i början eller i slutet av projektet. För att tydligare förstå hur systemdokumentationen hanteras i agil metodik, använder vi därför oss av Vattenfallsmodellen som kontrast.

2.3.1 Den traditionella dokumentationsprocessen - Vattenfallsmodellen

Vattenfallsmodellen har länge varit ett vanligt arbetssätt vad gäller systemutveckling. En av de största fördelarna med Vattenfallsmodellen är dess enkelhet och förmågan att utnyttja sunt förnuft. Först tar du reda på vad som behövs göra, sedan dokumenterar du hur du ska göra och sen utför du det. Därefter kontrollerar du om du verkligen gjorde rätt, och tillslut använder du det (Gugenberger, 2007).

När det kommer till systemdokumentation i Vattenfallsmodellen så skapas nästintill all systemdokumentation i det första två faserna, krav och design (MaRSDD, 2009). För att identifiera kundens krav på systemet så dokumenteras dessa krav på en hög abstraktionsnivå. Det är viktigt att alla krav och förväntade problem hittas och dokumenteras redan i början (Petersen et al., 2009). I slutet av varje fas i Vattenfallsmodellen så görs det en utvärdering. Utvärderingen är till för att kontrollera så att allt är redo för att övergå till nästa fas. Ifall utvärderingen inte blir godkänd så stannar projektet i samma fas tills den är klar (McConnell, 1996).

Vattenfallsmodellen är dokumentdriven, med det menas att den viktigaste komponenten som förs från fas till fas är systemdokumentationen. Faserna i Vattenfall sker sekventiellt och inte parallellt, de utförs steg för steg och överlappas inte (MaRSDD, 2009; McConnell, 1996).

När mjukvara utvecklas tas den inte bara fram av utvecklingsteamet, som består av utvecklare, analytiker och arkitekter. Det krävs även dedikerad personal som testar och dokumenterar processerna och produkten. Tekniska skribenter (“technical writers”) är ett begrepp för en förekommande professionell dokumenterarroll där de arbetar med att skapa och underhålla systemdokumentation för mjukvaran (Ascezen Consulting, 2014b).

Avslutningsvis påpekar McConnell (1996) att Vattenfallsmodellen minimerar överliggande planering eftersom all planeringen görs direkt. För de som är bekanta med den här typen av projekt, kan systemdokumentationen agera som indikation för att se hur väl utvecklingen går (McConnell, 1996). Den största kritiken mot Vattenfallsmodellen är att den inte är tillräckligt flexibel i mjukvaruutveckling. När förutsättningarna i projektet förändras blir det kostamt att gå tillbaka och revidera (Avison & Fitzgerald, 1995; McConnell, 1996).

2.3.2 Agil systemdokumentation

I studien beskriver vi om en rådande problematik inom den agila metodiken för systemdokumentation. Det handlar om ett felaktiga synssätt gällande systemdokumentation, att agila projekt inte behöver dokumenteras (Ambler, 2012b). Det stämmer inte, och för flertalet projekt behövs vissa dokument och modeller (Ambler, 2002).
Istället föreslår forskare att det bör finnas ett minimumkrav gällande dokumentering i projekt, att det ska vara lätt, men tillräckligt (Cockburn, 2001; Ambler, 2002; Rüping, 2003).

Värt att notera är att det dock ofta finns krav på vad som skall dokumenteras från kund, myndigheter eller den egna organisationen (Magnusson, 2004).

Gustavsson (2007) skriver att projekt inte kan skära ner på omfattande systemdokumentation utan att få negativa effekter. Istället talas det för att ersätta den omfattande systemdokumentationen med kommunikation i projekten, för att lyckas kompensera för de fördelar som finns med en tydlig systemdokumentation (Gustavsson, 2007). Dokumentation i sig är en kanal för kommunikation, men forskning pekar tydligt att detaljerad systemdokumentation och mailutbyte är den sämsta formen av kommunikation i projekt (Ambler, 2014b). Bäst är den verbala dialogen mellan projektmedlemmar på plats, och gärna med hjälp av en whiteboard (Ambler, 2014b).

Med detta i beaktning, poängterar ändå forskare att verbal kommunikation har sina brister. Wyssocky (2003) hävdar att det inte är hållbart att förlita sig enbart på verbal kommunikation i projekt. Exempelvis kan svårigheter uppstå när systemutvecklare ersätts i projekt, då tid måste avsättas för att förklara hur systemet är uppbyggt (Uikey et al., 2011). Dessutom riskerar organisationen förlora kunskap då systemutvecklare byts ut i projekt (Wyssocky, 2003).

En tydlig agil strategi i skapandet av systemdokumentation att skapa lätta dokument. De är översiktliga och i tillräcklig detaljmängd för den ämnade målgruppen att förstå (Ambler, 2014b). Agil systemdokumentation beskriver stabila begrepp och inte spekulativa begrepp, och följaktligen dokumenterar man så sent som möjligt i livscykeln (Ambler, 2012b).

Vi har tidigare identifierat tre begrepp som hör till systemdokumentation. Nedan går vi igenom dessa i kontextet av agil systemdokumentation.

För *kravdokumentation* skapar det agila arbetssättet en övergripande vision för kraven i början av projektet. Dessa krav utgör en plan för vad som ska göras, och är enbart översiktlig med anledning av att kraven ska ha möjlighet att förfinas och förändras löpande under projektets gång (Ambler, 2014a).

Den *tekniska systemdokumentationen* är framförallt riktad till utvecklare och förvaltningspersonal (Ambler, 2014b). Ambler (2012b) talar vidare om att agil systemdokumentation finns i dess rätta sammanhang. En vanlig uppfattning bland utvecklare är att se tung systemdokumentation med stor skepsis utanför koden, eftersom den är sannolikt utdaterad (Ambler, 2012b). Därför hävdar forskare att koddokumentering (exempelvis inline-dokumentering) i koden vad gäller teknisk systemdokumentation är mer lämpligt (Uikey at al, 2011; Rüping, 2003; Ambler, 2012b). Därtill bör en kompletterande systembeskrivning finnas, som syftar till att ge läsaren en översikt över systemet och att hjälpa andra att förstå systemet (Ambler, 2012b).

I *design- och arkitekturdokumentation* arbetar agila metoder med modellering och beskrivning precis som andra metoder. Det rådande tillvägagångssättet för agila metoder är att modellera den övergripande arkitekturen relativt tidigt i projekten. Detta för att identifiera sin valda tekniska strategi för systemet (Ambler, 2012a).

Arbetet med design och arkitektur i agila projekt ter sig naturligtvis olika mellan olika projekt och organisationer. Varje system har en arkitektur, och agil arkitektur är driven utifrån krav och behov. Till exempel i små projekt med projektlag som alla arbetar i samma rum kan uppleva att de inte behöver dokumentera modeller bortom en whiteboard-skiss (Ambler, 2012a).

Värt att nämna är att eftersom agila metoder, och i synnerhet fallet för Scrum, har en plattare hierarkisk ledarstruktur är design och arkitektur ett arbete som alla i projektlaget är delaktiga i, och inte bara tillhör arkitektens ansvar (Ambler, 2012a). Strategin för agila metoder och Scrum är att uppmuntra till kommunikation och samarbete, och inte diktatur. På så sätt kan fenomenet med elfenbenstorn undvikas i projekt, där arkitekturen tas fram i isolering från resten av den dagliga utvecklingen och passas "nedåt" till utvecklare vid fulländning (Ambler, 2012a). Det kan även styrkas genom en av det agila manifestets tolv principer: "bäst arkitektur, krav och design växer fram med självorganiserande team" (Beck et al., 2001).

2.3.3 Dokumentering i Scrum

Scrum är ett ramverk som används som en agilt utvecklings- eller projektledningsmetod (Schwaber & Sutherland, 2013). Scrum är inte en given process eller teknik för att utveckla produkter, utan som ett ramverk kan verksamheter införa egna processer och tekniker som en egen implementation av Scrum (Schwaber & Sutherland, 2013). Scrum ligger i linje med andra agila metoders mål, att underlätta arbetssättet genom samarbete, kommunikation, ökad form av påtagliga resultat, och kontuerlig kunddelaktighet (Ascezen Consulting, 2014a).

Agil systemdokumentation i Scrum ignorerar inte helt och hållet betydelsen i systemdokumentation och processer (Ascezen Consulting, 2014a; Lacey, 2012). Scrum fokuserar däremot mer på vad som behöver implementeras, snarare än att försöka dokumentera i detalj det som redan har implementerats (Ascezen Consulting, 2014a).

Före Scrum lanserades fanns en vanlig uppfattning att ineffektiv kommunikation var en av anledningarna till att många fel uppstod i utvecklingen av programvara. Scrum har tagit vara på det och insett vilken betydelse kommunikation har i utvecklingen. Dagliga Scrum möten är ett exempel på förbättring av kommunikation (Cho, 2008).

För att ta reda på hur systemdokumentationen skapas i Scrum, är det också viktigt att förstå de roller som Scrum föreskriver. Roller i Scrum när det gäller dokumentering är produktägaren, ScrumMaster och utvecklingsteamet.
Produktägarens roll är att representera kunden och användarna och för in krav i produktloggen. Produktägaren är även den som är med och prioriterar krav i produktloggen.

ScrumMaster agerar som en medlare mellan samtliga och tar hand om systemdokumentationen på en mer kommunikativ nivå. Denne ser även till vid det dagliga mötet att arbetet är i fas, och att alla har de resurser som krävs. Utvecklarna använder sig av en del av kraven och formar det till en sprint backlog och bestämmer sedan hur kraven ska implementeras (Uikey et al., 2011).

I Scrum finns systemdokumentation i form av krav, egenskaper, funktioner, förbättringar och åtgärder som tillsammans utgör de förändringar som ska appliceras för programvaran i framtiden. En del av kravdokumentationen består av så kallade “user stories” och är bland annat krav som samlas från användare, kunder, ledning och andra intressenter (Uikey et al., 2011). User stories representeras som poster i produktloggen och är ofta ordnade efter risk, värde, prioritet och nödvändighet. Desto högre upp i produktloggen posterna finns, desto tydligare och mer detaljerade är dem. I samband med förändringar i produktloggen granskas och uppdateras dessa poster, på så sätt är kraven alltid aktuella (Schwaber & Sutherland, 2013).

2.3.4 Fördelar och nackdelar med tillvägagångssätten

I delkapitel 2.3 beskriver vi tre övergripande tillvägagångsätt för skapandet av dokumentation. För att djupare gå in på hur olik agil och traditionella projekt dokumenterar är för- och nackdelar en vital aspekt att belysa.

Det första tillvägagångsättet är att dokumentera det mesta i början, och det klassas oftast som ett traditionellt projekt, till exempel Vattenfallsmodellen. I början av ett traditionellt projekt ägnas tiden åt att skapa dokumentering och ingen konkret kodning existerar än. Lacey (2012) menar att det är ett dåligt tillvägagångssätt eftersom chansen att detaljerade dokument ska vara korrekt är så pass liten att det blir en riskfull och onödig process. Det är ett engagemang som är resurskrävande och i slutändan är många av dokumenten oanvändbara (Lacey, 2012). Istället för att skapa en detaljerad dokumentering i början av projektet, bör det skapas en viss mängd dokument i början av projektet men den ska vara mindre detaljerad. Vidare ska det skapas någon form av dokumentering vid varje fas ur projektlivscykeln (Lacey, 2012).

Det andra tillvägagångsättet är att dokumenteringen sker i slutet av projektet. Det betyder att all fokus i början av projektet ligger på att utveckla och under den tiden utförs ingen dokumentering. Det är först i slutet där allt dokumenteras över vad som har pågått i projektet. Det finns fördelar med det här tillvägagångssättet, och det är att programvaran är i fokus och att det blir klart i ett tidigt stadie.

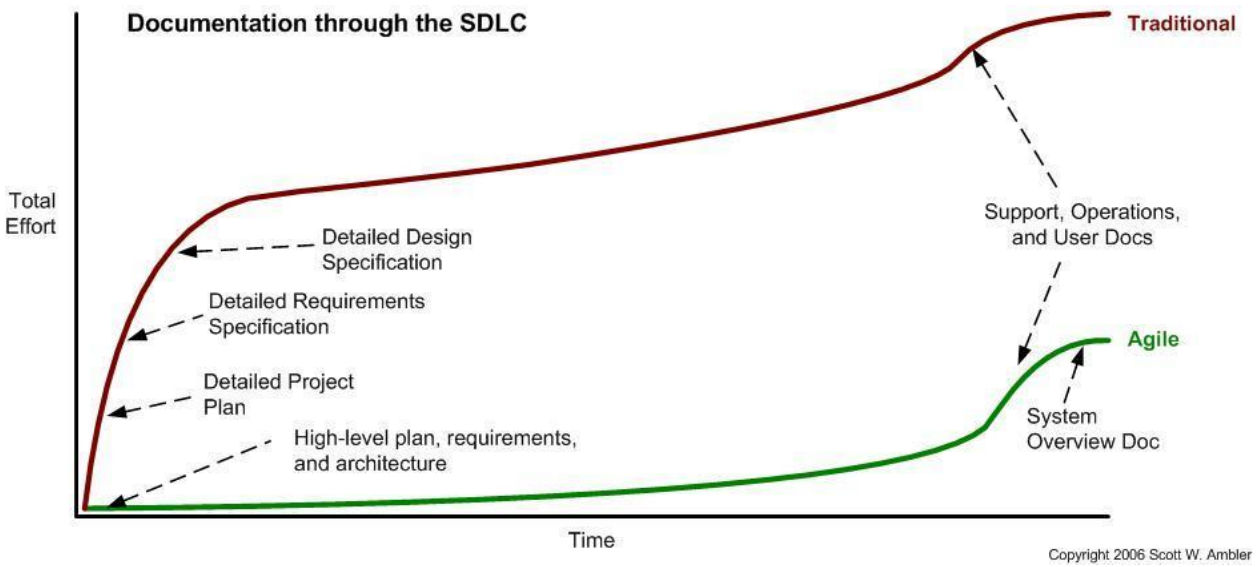
Dessutom i slutet när det är tid att dokumentera, kommer systemdokumentationen att vara en reflektion över systemets göromål och inget annat (Lacey, 2012).

Till följd av att vänta med att dokumentera till slutet av projektet kan problem uppstå. Glömska är ett vanligt problem när det kommer till hanteringen av det valda dokumenteringssättet (Lacey, 2012; Ambler, 2012b). Projektdeltagarna har problem att komma ihåg vad, när och varför besluten togs under utvecklandet. De projektdeltagarna som påbörjar ett projekt är inte alltid dem som avslutar projektet. Det kan därför bli svårt för de projektdeltagare som ska avsluta ett projekt om de inte har tillgång till den väsentliga information som behövs för att färdigställa dokumenteringen (Lacey, 2012).

Det typiska agila tillvägagångssättet är att utföra systemdokumentationen under projektets gång. Genom att arbeta agilt förespråkar man att dokumentera lite, men tillräckligt, och kontuerligt under arbetets gång. Det handlar om att dokumentera det som krävs och inte mer (Lacey, 2012; Ambler, 2012b). Agil systemdokumentation behöver inte vara perfekt, det räcker med att vara tillräckligt begränsad sett till detaljer och korrekthet, såvida dokumentet kan uppfylla dess syfte (Ambler, 2012b).

2.4 Skillnaderna mellan traditionell och agil dokumention

En av de grundläggande tankarna kring agila principer är att inte planera för långt in i framtiden med anledning av att undvika bortkastad arbetstid med preliminära beslut och diskussioner som måste ändras vid ett senare skede (Gustavsson, 2007). Agila projekt dokumenterar stabila begrepp, och inte spekulativa. Därmed förespråkar agil systemdokumentation att utföras så sent som möjligt i sprintarna (Ambler, 2014b). Agil systemdokumentation betonar vikten på löpande dokumentering, jämfört med tidig systemdokumentation av system och krav (Lacey, 2012). Figur 1 nedan (Ambler, 2014b) illustrerar de övergripande skillnaderna mellan det traditionella och agila tillvägagångssättet för systemdokumentation, samt vilka dokument som skapas under olika tidpunkter inom projektlivscykeln.



Figur 1. Skillnaderna i dokumentationssätten mellan de traditionella och agila arbetssätten.

En ytterligare definierande karaktär gällande agil systemdokumentation är betoningen på kommunikation mellan projektlaget och intressenter, samt en annorlunda syn på systemdokumentation i projekt (Uikey et al., 2011). Vi har tidigare skrivit att den traditionella modellen är dokumentdriven och lägger stor vikt på detaljerad systemdokumentation (MaRSDD, 2009; McConnell, 1996).

Skillnaden mellan de traditionella och agila tillvägagångssätten är tydlig då traditionella metoder ställer krav på systemdokumentation, och agila metoder ställer krav på kommunikation (Gustavsson, 2007). Arbete som i traditionella metoder beskrivs och förmedlas i detaljerad systemdokumentation, sker i agila metoder genom verbal kommunikation och sparsam dokumentering (Gustavsson, 2007). Eftersom agil metodik ställer mindre krav på systemdokumentation talar Ambler (2014b) om en fundamental skillnad i dess synsätt. Det handlar om att se systemdokumentation som ett affärskrav och inte som ett tekniskt krav från den valda metoden (Ambler, 2014b). Kravet ska inte finnas för att processen säger så, utan för att projektets kunder och intressenter efterfrågar det (Ambler, 2014b). En annan forskare hävdar liknande vad gäller värderandet av systemdokumentation: "*vikten av kommunikation och kortfattad systemdokumentation som är efterfrågad*" (Gustavsson, 2007, pp. 46).

2.5 Verktyg och hjälpmedel för systemdokumentation.

Hantering av dokument kan göras på flera sätt. Somerville (2010) menar att för stora projekt är ett specialiserat dokumenthanteringssystem av stor användbarhet. De används för att integrera lagringen och underhållet av den stora mängd information som dokumenten innehåller, samt att göra dessa tillgängliga för medlemmar ur projektlaget på olika platser.

Scott Ambler (2014b) påpekar att agila utvecklingsteam har allt mer börjat använda wikisidor för att hantera systemdokumentationen av extern systemdokumentation. Genom att använda wikisidor samlas all information på ett och samma ställe. Varje wikisida utgör ett specifikt ämne.

Det finns även andra verktyg och hjälpmedel för skapandet av systemdokumentation, som forskare påpekar stöds i Scrum och allmänt för agil metodik:

- *Digitala kameror och skanners*. För att bevara bilder och sketcher som är gjorda på exempelvis whiteboards eller papper, används kameror och scanners för att dokumentera dessa (Ambler, 2002).
- *Wikisidor*. Wikisidor ett sätt att samla all information på ett och samma ställe. Wikisidan implementerar projektets dokumentlandskap, med möjligheter till att arkiveringsfunktioner för att hantera versioner och säkerhetskopiering via back-end plugins, något som tidigare enbart tillhört specialiserade dokumenthanteringssystem (Rüping, 2003). Rüping (2003) talar om och för användandet av wikisidor som ett forum som hjälper kollaborativ kommunikation i projekt, både genom interaktiv och asynkron kommunikation.
- *Elektroniskt whiteboard*. Ett enkelt sätt att få tillgång till all information i pappersformat genom att skanna innehållet på whiteboarden (Ambler, 2002).

2.6 Införandet av professionell dokumenterarroll i Scrum

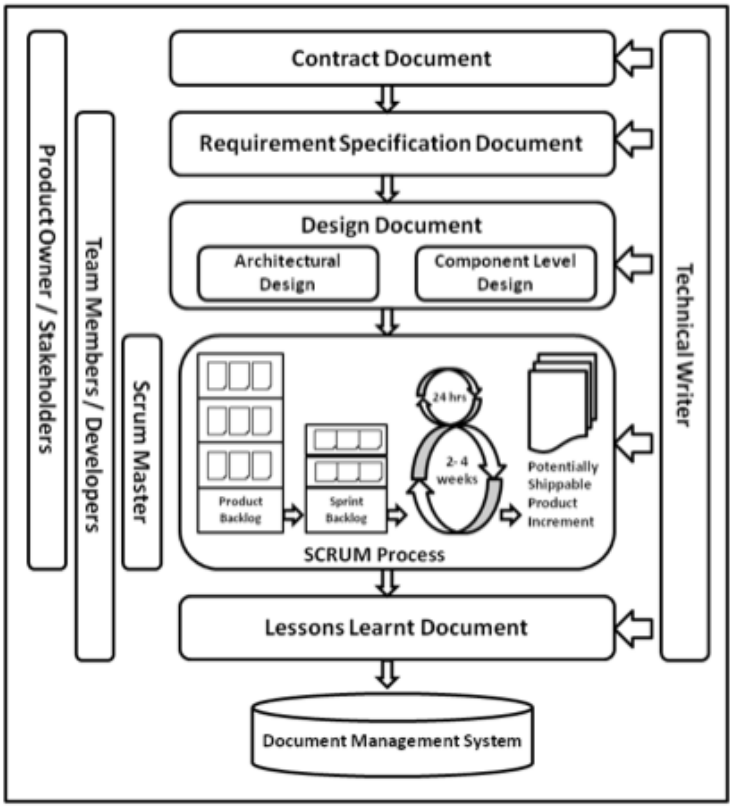
Som tidigare beskrivits i delkapitel 1.2, finns det en viss problematik med systemdokumentation i agil metodik. Författarna Uikey at al. (2011) har med beskrivit denna problematik, och som därför presenterar en egen teori för att lösa denna problematik genom att införa en ny roll i Scrum.

Figur 2 nedan är en modell från artikeln Uikey et al. (2011) publicerat, som ger en översikt för metoden, vilken de valt att kalla “Documented Approach”. Metoden är något av en hybridmodell då den bygger på Scrums tillvägagångssätt som grund för utvecklingen, men inför vissa förändringar som påminns om i det traditionella tillvägagångssättet i Vattenfallsmodellen. Till exempel föreslår de införandet av två formella dokument som skapas före ett projekts utveckling inletts. Efter det att ett avtal skrivits, påbörjas projektet med att skriva en klassisk kravspecifikation. Ett formellt design- och arkitekturdokument efterföljer därefter, och det är inte förrän dessa dokument skapats som utveckling kan startas enligt Scrums principer.

Den betydande skillnaden, i modellen nedan, är införandet av en ny roll i Scrum. Uikey et al. (2011) föreslår en fjärde roll ska införas, som teknisk skribent eller professionell dokumenterare.

Denne roll har för ansvar att utveckla systemdokumentation för krav, design och arkitektur, och med god förmåga. Denne är delaktig under alla faser i projektlivscykeln och kan på så sätt skapa alla nödvändiga dokument med hög kvalité (Uikey et al., 2011).

Rollen, och även modellen i förlängningen, står till syfte att ta vara på de fördelar som god systemdokumentation innehar, samt att även försöka reda ut den problematik som vi redogjort för i delkapitel 1.2 (Uikey et al., 2011).



Figur 2. Documented Approach (mod. efter Uikey et al., 2011).

2.7 Teoretisk sammanfattning

Litteraturgenomgången har lyckats identifierat ett antal begrepp och aspekter, som vilket studien syftar till att undersöka. Genom att dela upp begreppet systemdokumentation till handgripliga mängder, som vi därefter kan undersöka och diskutera forskningsfrågan. Det är framförallt tre aspekter och begrepp som har en viktig roll sett till vår forskningsfråga. Vi har sett att systemdokumentation i sig är en underkategori för systemdokumentation, och består i huvudsak av tre systemdokumentationsbegrepp som presenterat i sektion 2.3.2.

Vi har identifierat ett antal relevanta aspekter i förhållande till vår givna forskningsfråga och problemformulering. Bland annat rör det sig om dokumentering i förhållande till ansvarsfördelning och vem som dokumenterar projekt. Vi har valt att kalla denna aspekt för *roller och ansvarsfördelning*.

Vidare har vi identifierat två varianter för dokumentering, som vi kallar koddokumentering och extern systemdokumentation. Dessa är begrepp som beskrivs i delkapitel 2.2, vars aspekt vi väljer att kalla för *dokumentationstyper*, och är vital för undersökningen på grund av dess närhet till forskningsfrågan.

Till sist har den agila metodiken ett annorlunda strategiskt förhållningssätt till systemdokumentation. Exempelvis, har forskare beskrivit den traditionella Vattenfallsmodellen som dokumentationsdriven (McConnell, 1996), och den agila metodiken som kommunikationsdriven (Gustavsson, 2007). Aspekten har vi valt att kalla *strategi och tillvägagångssätt*.

Nedan går vi igenom alla aspekter var för sig och vad forskare och författare talar om under respektive aspekt.

2.7.1 Roller och ansvarsfördelning

Författarna som diskuterar roller har liknande slutsatser. De påstår att när det kommer till rollerna inom Scrum, är det till största delen utvecklarna som måste ansvara för systemdokumentationen. Till exempel påpekar Uikey at al. (2011) att det främst är utvecklarna som dokumenterar i Scrum. Författarna hävdar att utvecklare besitter en god kunskap för hur system är uppbyggda, men att de saknar en förmåga att skapa en bra systemdokumentation. Därtill hör även brist på tid och motivation för att skapa en systemdokumentation av kvalité. Inom fallet för design och arkitektur är det ofta utvecklarlaget tillsammans som utför detta arbete, istället för ett specialiserat yrkesansvar.

2.7.2 Strategi och tillvägagångssätt

Vad gäller det agila tillvägagångsättet hävdar författarna att det ska skapas en viss mängd av systemdokumentation i början av projektet. Författarna påpekar att krav dokumenteras i början. Exempelvis tar Uikey et al. (2011) upp att Scrum uttrycker sina krav i form av user stories som produceras innan utvecklingen har startat. Lacey (2012) påstår att för krav ska det skapas en översiktlig beskrivning av kraven i början av projektet, som fångar idéer och koncept, med ytterligare systemdokumentation löpande vid varje sprint.

Ambler (2012b) påpekar däremot att skapandet av dokumenteringen bör återhållas så länge som möjligt för att undvika att dokumentera spekulativa begrepp, vilket handlar om att dokumentera det som krävs och inte mer. Gustavsson (2007) pratar om att agil systemdokumentation ska vara kortfattad och efterfrågad.

Författarna delar samma uppfattning om att den agila systemdokumentationen ska vara lätt, kortfattad eller begränsad, och i alla fallen, tillräcklig. Agil systemdokumentation ska vara efterfrågad av beställare eller andra intressenter. Det gäller att inte skapa systemdokumentation för att metodiken eller för att processkrav säger så, utan för att kunden efterfrågar det (Ambler, 2012b; Highsmith 2002; Gustavsson, 2007).

2.7.3 Dokumentationstyp

Gustavsson (2007) påstår att en detaljerad systemdokumentation som anses tillhöra den traditionell metodik, ersätts inom det agila med verbal kommunikation och med en lätt systemdokumentation. Dessutom ska den vara riktad åt rätt läsare. Är det en utvecklare som ska läsa systemdokumentationen bör den bestå av mer koddokumentation i förhållande till den externa systemdokumentationen (Ambler, 2012b).

Koddokumentation är en form av systemdokumentation som med stor frekvens används och bör användas i agila projekt. Rüping (2003) påstår exempelvis att dokumentation för en teknisk systembeskrivning är bäst lämpad i form av koddokumentation. Extern systemdokumentation måste också finnas för att på en hög nivå beskriva systemets översikt (Rüping, 2003; Ambler, 2012b).

2.7.4 Nyckelord

Nedan följer en matris, i vilken författarna och deras respektive bidrag sammanfattas med vissa nyckelord.

Aspekter / Författare	Roller & ansvarsfördelning	Dokumentationstyper	Strategi & tillvägagångssätt
Ambler, 2014	Utvecklare; gemensamt arbete	Lämpligt val för målgrupp; koddokumentation, med översiktliga dokument	Tidig övergripande plan; sker så sent som möjligt; lätt, men tillräckligt; efterfrågad
Gustavsson, 2007			Kortfattad och efterfrågad; Löpande
Lacey, 2012			Lätt; löpande
Uikey et al., 2011	Utvecklare	Koddokumentation; Extern krav-, design och arkitekturöversikt	
Prause & Durdik, 2011	Utvecklare		
Cockburn, 2001			Lätt, men tillräckligt
Rüping, 2003		Koddokumentation; Extern krav-, design och arkitekturöversikt	Lätt, men tillräckligt
Highsmith, 2002			En balans; Efterfrågad

Tabell 1. En sammanslagen tabell för författarnas teoretiska bidrag i förhållande till undersökningens valda aspekter.

Vi har tidigare talat om de tre begreppen (se sektion 2.3.2) och deras betydelse. De kan tyckas vara mystiskt frånvarande i tabell 1, men de är naturligtvis viktiga. Detta eftersom aspekterna genomsyrar alla begreppen, som till exempel kravdokumentation. De medverkar också till att undersöka systemdokumentation som ämne i ett tydligare sammanhang, samt att det ger varje aspekt en mer nyanserad källa för diskussion.

2.8 Undersökningsmodell

Nedan presenterar vi en enkel undersökningsmodell som sammanfattar och återkopplar de centrala aspekter och begrepp som vi redogjort för i litteraturgenomgången och sammanfattat i det teoretiska ramverket.

Tabell 2 nedan presenterar de tre aspekter och begrepp som vi har för avsikt att undersöka. Modellens kolumner är studiens tre övergripande aspekter som återfinns i delkapitel 2.7, och raderna är de tre begrepp som vi identifierat i sektion 2.3.2. Modellen utspelar en viktig roll vid utformandet av undersökningen, samt vid presentering och analys av empirin.

Aspekter / Begrepp	Roller & ansvarsfördelning	Dokumentationstyper	Strategi & tillvägagångssätt
Kravdokumentering			
Teknisk systemdokumentation			
Design & arkitektur			

Tabell 2. En sammanfattning av de tre aspekter och begrepp som vi undersöker.

3 Metod

Detta kapitel redovisar hur vi skapat och genomfört undersökningen, från val av undersökningsmetod till analys av empiriskt data. Den redogör för vilken vetenskaplig ansats vi valt för att genomdriva forskningen och undersökningens studieobjekt. Vidare redogör den för undersökningens upplägg, och hur den empiriska datan analyseras. Sist i kapitlet har vi en del där vi diskuterar hur undersökningen beaktar etiska aspekter och vårt val av metod.

3.1 Kvalitativ undersökningsmetod

Vi har valt en kvalitativ undersökningsmetod, med koppling till dess fördelar. Den kvalitativa undersökningsmetoden insamlar rik och ostrukturerad data som är av en icke kvantifierbar natur, med syftet att undersöka ett problemområde djupgående (Backman, 1998). Studiens syfte måste i huvudsak styra ens val av undersökningsmetod, och inte tvärtom (Andersson, 1985). Eftersom det ligger i studiens uppgift att mer djupgående granska begrepp och aspekter som många är av vars natur svåra att kvantifiera, lämpar sig det därför bättre att använda sig av en kvalitativ undersökningsmetod. Med tanke på forskningsfrågan eftersträvar vi till att undersöka människors uppfattningar och erfarenheter som finns i konsultbranschen för Scrum, vilket också talar för valet av en kvalitativ undersökningsmetod.

Vidare talas det om studiers vetenskapliga inriktningar, vilket rör sig om en deduktiv eller induktiv inriktning. Den deduktiva inriktningen kännetecknas av hypoteser eller teorier som skapas före insamlingen av data inleds, och studien inriktar sig därmed åt att antingen bevisa eller motbevisa den givna hypotesen. En induktiv inriktning är istället då forskaren skapar hypoteser eller teorier vid en analys av insamlat empiriskt material (Backman, 1998). Enkelt kan de vetenskapliga inriktningarna deduktion eller induktion beskrivas som “hypotesbeprövande” respektive “hypotesgenererande” (Backman, 1998, s.48).

Den kvalitativa undersökningen och dess påföljande dataanalys utgår i den undersökningsmodell vi redogjort för i delkapitel 2.8. Vad gäller studiens vetenskapliga inriktning, väljer vi den induktiva inriktningen då den är bäst lämplig givet studiens syfte och kvalitativa undersökningsmetod. Vi har inte för avsikt att inleda undersökningen med någon hypotes, utan låter det empiriska resultatet styra formandet av eventuella teorier. Därmed sagt, måste vi tillägga att vi som forskare naturligtvis har vissa förväntningar av undersökningens utfall.

Kalla det vår vetenskapliga nyfikenhet, men det är något som vi hoppas kan beprövas bortom ramarna för denna studie.

3.2 Utformande av intervjuguide

Vi väljer att använda oss av en semistrukturerad intervjuform. En semistrukturerad intervju är när intervjufrågorna är strukturerade enligt ett manusskript som alla får besvara, men frågorna är av öppen karaktär så intervjupersonen får en chans att utveckla sina svar. Det skapar en kommunikation i intervjun, och tillåter intervjupersonen att uttrycka sig om sina egna synpunkter och idéer (Denscombe, 2000).

Vi eftersträvar att utföra muntliga intervjuer i person, som medverkar till att skapa en mer avslappnad miljö mellan oss och intervjupersonen. Ifall någon komplikation skulle uppstå och det inte går att utföra intervjun på plats, utförs den via telefon för att eftersträva samma resultat. De personer som intervjuas kommer representera alla tre rollerna i Scrum. Därmed är det lämpligt att föra intervjuerna i en avslappnad miljö där vi personligen involverar oss i deras svar med följdfrågor som kan berika diskussionen.

Intervjupersonerna kommer att tillgodoses med frågorna på förhand för att förhoppningsvis få ett bättre och mer genomtänkt svar. Det är även en god gest, och som tillåter intervjupersonerna att förbereda sig och skapa en bättre förståelse rörande ämnet.

3.3 Intervjufrågor

Varje intervjutillfälle inleds med att bland annat redogöra för ämnet och syftet med undersökningen. När intervjun påbörjas ställer vi några bakgrundsfrågor till intervjupersonen. Det sätter en lätt ton på samtalet där intervjupersonen i fråga får introducera sig själv och det företag som denne representerar, för att sedan dyka ner i djupare frågor som berör ämnet för vår undersökning.

Bilaga 1 innehåller undersökningens frågeformulär, som vi använder vid samtliga av våra intervjuer. Strukturen på frågorna är i förhållande till undersökningsmodellen (se delkapitel 2.8). Varje aspekt är en frågedel och utgör en huvudrubrik, under vilken varje begrepp vävs in i delfrågorna. Vi anser att detta intervjuupplägg är lämpligt. Detta eftersom det ger oss en möjlighet att anpassa intervjuns frågor under intervjun i förhållande till intervjupersonens svar och rådande kompetensområde. Med hänsyn till undersökningsmodellen, har vi möjlighet att ställa följdfrågor frågor där en aspekt korsar ett begrepp. Därmed har vi chans att skräddarsy frågor i områden som intervjupersonen är sakkunnig i.

På så sätt skapar vi en levande diskussion, och utgår inte i ett statiskt manus. Givet vår forskningsfråga hoppas vi kunna ta vara den kvalitativa undersökningsmetodens fördel att bedriva en djupgående undersökning (Backman, 1998).

Dessutom följer ett antal punkter nedan som vi använt som underlag när vi träffat intervjupersonerna. Varje punkt har förklarats före vi påbörjat själva intervjun. Detta använde vi för att försäkra oss om ett bättre svarsresultat och så etiska aspekter beaktas.

- Redogör för vårt ämne, vad undersökningen syftar till och vad vi hoppas få ut av den
- Redogör för de tre begrepp och aspekter som vi undersöker
- Vi önskar att lyssna till era uppfattningar och erfarenheter i detta ämne. Det är viktigt att ni beskriver "verkligheten" så nära som möjligt, men ni är givetvis välkomna till att väva in era egna åsikter och tankar
- Vill du så kan vi göra era namn och/eller företagsnamn anonymt när uppsatsen publiceras
- Det är möjligt att du kommer besvara en fråga i ett tidigare svar, men vi kommer ändå undvika egna antaganden och ställa frågan för undersökningens skull. Så får du helt enkelt repetera dig.
- Vill du avbryta intervjun eller undvika att besvara en fråga, får du självklart göra det.

3.4 Val av företag och intervjupersoner

Vid val av företag väljer vi att fokusera på de företag som arbetar inom konsultbranschen i mjukvaruutveckling. Undersökningen är en fallstudie i Scrum på nyutvecklingsprojekt. Studien är baserad på Scrum med anledning av att skapa en homogen undersökningsgrupp.

Vid urvalet av intervjupersoner eftersträvas det en varierande bakgrund, som ger oss en nyanserad bild av systemdokumentation i Scrumprojekt. I undersökningen är även alla företag eller intervjupersoner anonyma, och deras namn återfinns inte i vårt material.

3.5 Analys av data

Jacobsen (2002) presenterar en teori för analys av kvalitativ data som vi använder oss av för att ta fram ett resultat. Teorin består av en iterativ tre-stegs process där varje intervju genomgår tre steg, eller faser. De tre faserna är i sin rätta ordning *beskrivning*, *systematisering och kategorisering*, respektive *kombination.* Jacobsen (2002) hävdar trots det att faserna är mer parallella vid den kvalitativa ansatsen, där nya idéer och rön formar analysen och presentationen av data löpande. Samtliga faser är ytterst viktiga eftersom hela undersökningen kräver reliabilitet och validitet, och denna teori stödjer just detta.

Precis vad ordet beskrivning betyder, anspelar också första fasen på att beskriva samtalen så noggrant som möjligt (Jacobsen, 2002). Tack vare modern teknik är detta inget bekymmer och därför spelas varje intervju in vid sitt tillfälle, efter intervjupersonen beviljat detta. Efteråt, transkriberas intervjun.

Intervjun identifieras via en tillförordnad siffra som tillhör intervjupersonen och det företag som denne representerar Exempelvis får den första intervjupersonen som intervjuas från företag A beteckningen A1, och intervjun identifieras därigenom. Fullständiga transkript finns att tillgå alfabetiskt i bilagorna. En ytterligare sak värd att nämna är att i varje transkript finns svaren numrerade med siffror i vänsterspalten, vilket vi lagt till för att lättare hitta citat när vi hänvisar till dem i empiri och diskussion. Detta förbättrar transparens och spårbarheten i vår undersökning.

Andra fasen är systematisering och kategorisering, som syftar till att sålla ut och förenkla den insamlade informationen. Fasen är viktig för att få en överblick över informationen, som därefter indelas in i olika kategorier efter ämne (Jacobsen, 2002). Vi har tidigare skrivit att struktureringen av intervjun sker enligt undersökningsmodellens tre aspekter och begrepp. Detta gäller även fallet för vår dataanalys där information enkelt kan struktureras enligt undersökningsmodellens aspekter och begrepp som datan berör. Undersökningsmodellen är även ett bra mätdon för oss för att avgöra huruvida en intervjus information är tillräcklig eller inte, och i så fall kräver komplettering på efterhand.

Vid systematisering- och kategoriseringsfasen är undersökningsmodellen vital eftersom den översiktligt presenterar varje begrepp som korsar en aspekt. Detta utgör ett ämne för presentation och vidare diskussion av empirin. Vi har därför valt att strukturera all information som berör exempelvis krav under en rubrik, med de tre aspekterna som vävs in.

Till sist står kombinationsfasen vid dataanalysen till syfte att påbörja en tolkning av datan och på så sätt identifiera samband för det ämne vi väljer att undersöka (Jacobsen, 2002). Varje begrepp som korsar de tre aspekterna utgör en rubrik och kommer således att granskas med relevant insamlat material från intervjuerna, samt med egna observationer från intervjun som inte kan uppdagas i respektive transkript.

Eftersom undersökningsmodellen har tre aspekter och tre begrepp har vi givetvis tagit andra tillvägagångssätt i beaktning för att strukturera, analysera och presentera datan.
Men vi anser att presentera varje begrepp för sig och väva in aspekterna ger bäst flöde i texten, jämfört med att göra motsatsen eller att presentera varje aspekt som korsar.

3.6 Etiska aspekter

Att intervjuerna utförs med frivilliga deltagare är en förutsättning enligt Jacobsen (2002). Intervjudeltagarna i vår undersökning är frivilliga och informerade om att de kan avbryta intervjun eller avstå från att besvara en fråga ifall de önskar det. Dessutom är de informerade om anonymitet ifall intervjupersonen önskar att hålla sitt namn ute ur undersökningen.

För att skapa ett bättre resultat för intervjuerna, har samtliga intervjupersoner blivit informerade om vad undersökningen syftar till, vad vi undersöker och de tre begrepp och aspekter som vi ligger till grund för frågorna. Intervjufrågorna ges ut på förhand för att berika svaren och underlätta intervjuprocessen så till vida att deltagarna har en kunskap om vilka frågor som kommer att ställas. Intervjufrågorna kommer att ges ut i rimlig tid, företrädesvis en dag i förväg, men inte mer. Detta för att begränsa informationsmängden och så att det inte förekommer anpassade svar.

3.7 Kritik av metodval

En kvalitativ undersökning av semistruktrurerat upplägg medför enligt Jacobsen (2002) nackdelar. Jacobsen (2002) diskuterar några nackdelar som vi identifierat, vilka påverkar vår undersökning. Vi resonerar att en negativ effekt som medförs genom att vi utför en kvalitativ undersökning är att det kan vara finnas föremål för subjektivitet Till skillnad från en kvantitativ undersökning som lättare kan bepröva en teori genom en direkt fråga, väljer vi ut ett antal teorier, vars tolkningar påverkar undersökningens utformande. Givet valet av den kvalitativa metoden, blir subjektivitet till viss del svårt att undvika. Men eftersom vi är två författare till studien resulterar det i att den blir mindre subjektiv.

Vårt mål med frågeställningen i undersökningen är att skapa öppna frågor så intervjupersonen kan besvara frågan utan någon vinkel eller att vi påverkar svaret för mycket. Däremot kan det förkomma hänvisningar i frågeprotokollet till de aspekter och begrepp vi vill undersöka, för att se till vi får ett komplett svar per intervju, med hänsyn till undersökningsmodellen. Därmed kan det förekomma att vi ställer följdfrågor (för ett svar gällande exempelvis krav), vilket kan bidra till att frågorna kan verka mindre öppna.

Vidare menar Jacobsen (2002) att generalisering är en nackdel som förekommer vid valet av en kvalitativ undersökning, då det oftast innebär att undersökningen omfattar färre respondenter jämfört med en kvantativ undersökning. En undersökning som omfattar fler respondenter ger en större tillförlitlighet och giltighet, och gör det empiriska resultatet därmed mindre vinklat. Då vi intervjuar alla roller inom Scrum ger det oss en tydligare bild av helheten, givet vår forskningsfråga. Avslutningsvis hävdar vi att valet av en kvalitativ undersökning ger oss empiri som stödjer vår forskningsfråga, och finns grundad i reliabilitet och validitet.

4 Empiri

Detta kapitel presenterar resultatet av den empiriska undersökningen i hur systemdokumentation skapas i projekt med Scrum, i förhållande till undersökningsmodellens tre begrepp och aspekter. Kapitlet presenterar den insamlade empiriska datan från intervjuerna enligt den struktur som vi presenterat i undersökningsmodellen. De tre begreppen i undersökningsmodellen utgör huvudrubriker, under vilka vi presenterar relevant empirisk data från intervjuerna och väver in aspekterna.

4.1 Undersökningens studieobjekt

Vi har intervjuat totalt fyra IT-konsulter, vars bakgrund är mot nyutveckling av mjukvara i Scrum. Nedan följer ett antal beskrivningar av de företag och intervjupersoner som vi pratat med.

4.1.1 Beskrivning av företag A och intervjupersoner

Företag A är ett litet och relativt nyligen uppstartat företag som verkar inom konsultbranschen. De profilerar sig som ett kunskapsbaserat mjukvaruutvecklingsföretag som inriktar sig på .NET-plattformen i Lean och agil metodik, och har en stor kompetens inom Scrum.

Från företag A har vi intervjuat tre personer som representerar samtliga roller i Scrum. Trots att de alla är anställda på företag A, varierar deras yrkeslivserfarenhet från person till person med blandning av helt olika projekt av både intern och extern karaktär. A1 och A2 har de senaste åren arbetat på externa uppdrag hos kund. Vi redogör för kundorganisationen under varje intervjupersons beskrivning.

Intervjuperson A1
Intervjuperson A1 är anställd på företag A som ScrumMaster och har ungefär fyra års erfarenhet som den rollen. För tillfället är A1 på uppdrag hos kund och har varit det de senaste två åren och arbetar just nu som projektledare och agil coach.

A1 arbetar på ett globalt företag med kontor över hela världen som utvecklar mjukvara för att analysera data för att underlätta beslut. Användarna av produkten finns inom alla branscher, allt från finansbranschen till hälsovård.

Intervjuperson A2
Intervjuperson A2 är anställd på företag A som produktägare/funktionsansvarig och har jobbat inom branschen sedan 2005. Han har de senaste två till tre åren arbetat på uppdrag hos kund som kravanalytiker.

Företaget som A2 arbetar på är ett dotterbolag till ett globalt företag som specialiserar sig på energieffektivisering. De har över 150000 anställda i mer än 100 länder. Företaget inriktar sig på intelligent fastighetsstyrning vad gäller exempelvis värme, ventilation och larm. Det är en produkt de utvecklar och den riktar sig både mot den privata sektorn och offentliga sektorn.

Intervjuperson A3
Intervjuperson A3 arbetar som affärsområdesansvarig för applikationsutveckling på företag A. Han är en av grundarna till företaget och har bred erfarenhet i såväl tekniska som mjuka roller, då som exempelvis utvecklare, agil coach och mentor. Är för nuvarande aktiv på företagets kontor med ett internt projekt.

4.1.2 Beskrivning av företag B och intervjuperson

Företag B är ett delat servicecenter för ett antal mediabolag. De är inblandade i både bolagsspecifika projekt och sådana som innefattar hela koncernen eller en delmängd av bolagen. Gruppen B1 jobbar i, arbetar med BI och uppföljning både på en övergripande nivå och hela vägen ner till smådetaljer. Till exempel har de har infört ett koncerngemensamt CRM-system och annonsbokningssystem.

Intervjuperson B1
Intervjuperson B1 arbetar på företag B. Hon har erfarenhet både som ScrumMaster och utvecklare. I nuläget arbetar hon som BI-utvecklare hos en kund, och även som ScrumMaster för hela teamet av utvecklare på företag B. Intervjun med B1 utfördes via telefon på grund av olämpligt geografiskt läge.

4.2 Kravdokumentation

Majoriteten av våra intervjupersoner arbetar aktivt med user stories och produktloggar, som medel för att beskriva krav. Nedan följer en beskrivning av empirin utifrån begreppet kravdokumentation, där vi väver in de tre aspekterna.

4.2.1 Strategi & tillvägagångssätt

När det gäller dokumenteringen av krav så är hanteringen av user stories en form av systemdokumentation som förespråkas i projekten. För att ta fram kraven i form av user stories så förklarade A1 i ett scenario där de började med att dokumentera kraven i så kallade "themes". Där themes är en övergriplig vision där produktägarna och produktledningen beskriver sina visioner, vilket kan vara väldigt otydligt. Därefter utkristalliseras dessa visioner till något som är mer hanterbart.

Alla intervjupersoner påpekar att de använder sig av user stories men hur de skapade dem skedde lite olika. Det varierade från projekt till projekt men alla skapade sina user stories från någon form av vision på en högre abstraktionsnivå. När kunden har framfört sin vision berättar A3 att de skapar en visionworkshop där de tillsammans med kund fångar upp visionen och får en enig förståelse för det kunden vill skapa.

Under workshoppen skapas user stories som underlag utifrån visionen. Visionen och kraven skapas innan ett projekt inleds. För A3 satte de först igång med ett projekt när en visionsworkshop var klar. Utifrån kraven som fångas upp under workshoppen sätts ett team ihop i förhållande till kraven och den expertis som efterfrågas. A3 beskriver: *"Det de handlar om är att du ska ha en teamsammansättning som har all den kompetensen för att klara av den uppgiften de ska klara av. När du har pratat med kunden och har förstått vad kunden vill ha, så får du designa teamet utifrån det*" (A3, 122).

A1 och A2 beskriver deras user stories som ett krav, de båda använder sina user stories på fysiska kort där på ena sidan kravet i form av user story och på andra sidan acceptanskriterier. A2 beskriver: "*Vi använde oss av user story cards eller kort. Där skrev vi user storyn på ena sidan ("I want to, so that…") och på baksidan hade vi acceptanskriterier, på lapparna skrev vi estimat och så vidare"* (A2, 63). A3 anser däremot inte att user stories är i form av ett krav utan att det är ett underlag eller stöd för diskussionen. Vi citerar A3: "*Eftersom en user story är ca tre rader text kan man inte riktigt beskriva ett krav*" (A3, 18). A3 syftar på att user stories bara ska användas som ett underlag för att kunna föra en diskussion med en kund för att sedan kunna ta fram detaljerade krav, i form av accpetanskriterier.

På A2's senaste projekt agerade han som en kravanalytiker, och berättar att de inte skapade sina krav förrän så sent inpå sprinten som möjligt. Detta eftersom det sker förändringar och genom att sätta kraven så sent som möjligt minskar också risken för att kraven ska behövas ändras inför sprinten.

4.2.2 Dokumentationstyper

Både A1 och A2 skapade kravdokumentatation både fysiskt och digitalt, vilket A2 upplevde var redundant. Men detta var ett krav från kundorganisationen eftersom de arbetade enligt Vattenfallsmodellen utanför A2:s projekt. För att tydliggöra den fysiska och digitala kravdokumentationen, citerar vi A1: "*Sen på daglig basis så har vi post-it lappar på fysiska tavlor, men vi har ändå ett centraliserat digitalt system där alla kraven finns*" (A1, 31).

Majoriteten av tillfrågade i undersökningen använde sig av ett kravhanteringssystem där de digitalt lagrade och hade tillgång till de krav som var aktuella. När de dokumenterar krav i A2:s projekt finns de digitalt i form av en produktlogg som finns tillgängligt via deras intranät. Genom att hantera krav digitalt underlättade det för A2 att kunna ändra prioritering av user stories i produktloggen, och lägga fokus på det som var mest aktuellt inför nästa iteration.

4.2.3 Roller & Ansvarsfördelning

I skapandet av kravdokumentation, specifikt user stories, är alla roller delaktiga. Eftersom kraven har olika abstraktionnivåer, med allt mellan en otydlig vision till en detaljerad user story med acceptanskriterier. Därmed innebär det också att alla roller blir involverade vid något tillfälle av kravlivscykeln. Men i huvudsak är det utvecklare som tillsammans tar fram user stories: "*Jag behöver inte dokumentera särskilt mycket, mer än att sköta kommunikationen mellan nivåerna och individerna. Det är ju teamen själva som utvecklar user stories, så jag är mer av ett stöd så att de självorganiserade teamen kan lösa det. Men jag hjälper de skriva user stories bland annat*" (A1, 37).

4.3 Teknisk systemdokumentation

Teknisk systemdokumentation har visat sig majoriteten använder sig av och värderar olika i olika utsträckning. Däremot är källkodskommentarer något som många väljer bort. Nedan följer en beskrivning av empirin utifrån begreppet teknisk systemdokumentation, där vi väver in de tre aspekterna.

4.3.1 Strategi & tillvägagångssätt

Våra intervjupersoner svarade mycket olika vad gäller teknisk systemdokumentation. Exempelvis, kunden som A1 var på uppdrag hos, värdesatte systemdokumentation av funktionalitet högt. Det kan observeras genom det faktum att varje team har en tillförordnad dokumenterare eller skribent från organisationens egna dokumentationsstab, vilken A1 kallar "dokumentation".

Vi citerar A1: *"När en user story är färdig, så har vi ett acceptanskriterie att det ska vara dokumenterat. Och systemdokumentation har sina egna user stories för att inte missa dem här sakerna*" (A1, 49).

Rent strategiskt ser vi i detta exempel att kundens organisation prioriterar systemdokumentation. De väljer att använda user storyns acceptanskriterier som ett verktyg i samband med dedikerade dokumenterare för att säkerställa sig om korrekt och tillräcklig teknisk systemdokumentation.

"*Ingenting är viktigt för mig att dokumentera. Vad som viktigt för mig är att skapa en bra mjukvara som kunden är nöjd med. All systemdokumentation är bara ett medel att nå dit"* (A3, 21). I andra änden av spektrumet finner vi A3 och A2, vars uppfattningar är att kodning måste gå före systemdokumentation. A3 har uppfattningen att systemdokumentation egentligen saknar ett egenvärde och A2 som menar att omfattande systemdokumentation kräver för mycket tid att uppdatera. A3, som en av grundarna till företag A sätter därför inga krav eller mål på projekt i sin verksamhet. Istället att för att skapa systemdokumentation måste den vara efterfrågad av kund, som kan uppge goda skäl. De inte dokumenterar bara för att dokumentera, utan att det finns lagkrav eller ett syfte med varför kunden efterfrågar systemdokumentation. Detta påstående motsägs då han faktiskt använde digitala verktyg för att anteckna inloggningar och beskriva vilka saker som ändrades sen senaste lanseringen av produkten: "*Sådana saker kan vara bra att skriva upp och då är wiki bra*" (A3, 78).

A1 påstår däremot att systemdokumentation skapas både för sig själv och för kunden, som skapar möjlighet till att göra egna vidareutvecklingar för denne: "*Det är både och. Vissa kunder är intresserade för den tekniska systemdokumentationen ifall de ska göra sina egna vidareutvecklingar eller använda produkten på ett annat sätt. Som ett affärskrav är det viktigt att det finns på plats för att det finns ett mervärde för kunden*" (A1, 112).

Vad gäller tillvägagångssätt för skapandet av för teknisk systemdokumentation, svarade majoriteten av våra tillfrågade att den tekniska systemdokumentationen skapas löpande i projektlivscykeln. A1 beskriver den som iterativ, och vidare, att den även anpassas och förändras enligt principerna för Scrums sprintar: "*Det arbetet är en del och sker i varje iteration, så det anpassas hela tiden också utifrån de förändringar som uppstår*" (A1, 73).

A3, som uttryckt att systemdokumentation ska vara efterfrågad som ett affärskrav, ger stöd till A1 vad gäller agil systemdokumentations tillvägagångssätt: "*Poängen med ett agilt projekt är att du ska kunna efter en iteration säga att du är färdig. Och det kan du inte göra om du har bara systemdokumentationen kvar, om kravet finns att systemdokumentationen ska finnas*" (A3, 107). Värt att tillägga är A1 som beskrev att det är viktigt att inte dokumentera saker direkt, utan att vänta tills koden nått en bra grund innan man dokumenterar det. A2 har en liknande uppfattning då han menar att systemdokumentation ska uppdateras så sent inpå som möjligt, i slutet av en sprint.

4.3.2 Roller & ansvarsfördelning

Majoriteten av intervjupersonerna uppgav att ansvaret i första hand ligger på utvecklare för skapandet av teknisk systemdokumentation. A2 svarade att på uppdraget han var anställd på, hade organisationen krav på systemdokumentation för produkten de utvecklade till sina kunder. Där samarbetade utvecklarna för att dokumentera funktionaliteten som de utvecklade, och för varje iteration även för uppgift att komplettera och uppdatera gammal systemdokumentation.
A1 var undantaget, däremot, vars arbetsplats vi beskrivit tidigare. A1 arbetade hos en kundorganisation som värdesatte systemdokumentation högt, och som därför instiftat flera medel för att säkerställa hög dokumentationskvalité. Där hade de en professionell dokumenterare tillordnad per team, där dokumenterare hade acceptanskriterier för att säkerställa att varje user story blev dokumenterad.

4.3.3 Dokumentationstyper

Ingen av intervjupersonerna förespråkar källkodskommentarer. Trots detta, så använde A2 och B1 i sina projekt källkodskommentarer, och i fallet för B1 med en kort övergripande beskrivning i ren text. A3, till exempel menar att "*Jag som normalfuntad utvecklare med den erfarenhetsnivån jag har måste kunna sätta mig och läsa koden och kunna avgöra vad den gör"* (A3, 48). Han tillägger: *"Kod ska vara enkel och självbeskrivande"* (A3, 48). A1 har liknande uppfattning, trots att han visat sig ha en allmänt positiv uppfattning av systemdokumentation. Han anser också att bra skriven kod talar sitt tydliga språk och ska därför inte behöva beskrivas. En tydlig skillnad som vi observerade mellan A1:s åsikt och verkligheten, var att de i hans projekt faktiskt beskrev koden. Detta gjordes visserligen inte i källkoden, utan via verktyg som onlinehjälp och interna wikisidor som fanns tillgängliga för alla.

Avslutningsvis, har undersökningen visat att övergripande systemdokumentation är vanligt med hjälp av digitala webbverktyg, jämfört med detaljerad systemdokumentation i källkod eller pärmar och papper. Alla tillfrågade använder sig av wikisidor, för olika användningsområden. Både A1 och B1 använde sig av en wikisida som kallas Confluence. B1 hävdar att den hjälper nyanställda och att den innehåller övergripande kodbeskrivningar. Hos kunden där B1 arbetar, har de ett processkrav och riktlinje att all systemdokumentation ska finnas på denna wikisida. A1 menar att dokumentera viss funktionalitet har sina fördelar: "*Vissa konsulter kan ju komma in och bygga en hel del funktionalitet, och efter x antal månader försvinner de kanske. Därför är det viktigt att deras tankar och hur funktionaliteten är tänkt ska fungera, dokumenteras så det sparas trots att personen har lämnat projektet"* (A1, 55).

4.4 Design- & arkitekturdokumentation

I förhållande till vår forskningsfråga skiljer sig design och arkitekturdokumentation mycket mellan projekt. Nedan följer en beskrivning av empirin utifrån begreppet design och arkitekturdokumentation, där vi väver in de tre aspekterna.

4.4.1 Dokumentationstyper

Vad gäller design och arkitektur, och systemdokumentation av den, svarade samtliga att den oftast befinner sig på whiteboards, som vars nästa steg är att ta ett fotografi av som sedan läggs upp på interna wikisidor eller intranät. B1 beskriver att den oftast är i fotostadiet och menar "*Meningen är att det ska koka ner till en klassisk systemdokumentation, men det sker sällan*" (B1, 45). Ingen hade heller några riktlinjer för skapandet av design eller arkitektur, utan det skapades på en situationsbasis i förhållande till projektets omfattning och kunden. "*Men vi har inga riktlinjer vad gäller design eller arkitektur. Det kan röra sig om en person som stått och ritat på en whiteboard och tagit en bild av den, och lagt upp på wikin*" (A1, 61).

4.4.2 Strategi & Tillvägagångssätt

A3 beskriver tillvägagångssättet för skapandet av arkitektur och använder ett begrepp som han kallar agil arkitektur. "*Vilket är i princip ingen arkitektur alls. Det finns en grundtanke i att det finns vissa idéer och koncept som finns, men som kommer kanske att förändras över projektlivscykeln och hur saker skapas*" (A3, 33). I hans projekt så modellerade även de på whiteboard, men som gick sällan bortom ett fotografi, som exempelvis en formell dokumentationsbeskrivning, eller modell med digitala verktyg som Microsoft Visio. A3 menar att det är meningslöst att ingå i en stor arkitektonisk diskussion för att planera en arkitektur i förväg, det riskerar att sluta med en pappersprodukt efter sex månader in i ett projekt. A3 har uppfattningen att arkitekturen är evolutionär, och andra har en liknande åsikt. Majoriteten av tillfrågade svarade en övergripande arkitektur skapas i början av ett projekt, som sedan utvecklas i inkrement.

4.4.3 Roller & ansvarsfördelning

Alla tillfrågade svarade även att det är utvecklingsteamet som tillsammans tar fram en arkitektur eller designar processflöden. Det enda undantaget i undersökningen gällande roller och ansvar där, var i fallet för A2. I dennes projekt hade kundorganisationen avsatt en systemarkitekt som i början tog fram en övergripande plan för hur utvecklingen skulle ske. A2 berättade samtidigt att utvecklarna hade fria händer att utveckla och anpassa arkitekturen: "*Vi levde vårt egna liv och kunde sköta och styra arkitekturen själva, och de besluten togs av utvecklarna tillsammans med någon övergripande systemarkitekt, men som litade ganska mycket på utvecklarn*a" (A2, 33).

5 Diskussion

I detta kapitel förs en avslutande diskussion med utgångspunkt för de tre begrepp i det empiriska materialet som presenterades i föregående kapitel.

5.1 Agil systemdokumentation i Scrum

Vi har tidigare fastställt att Scrum är ett agilt ramverk för systemutveckling och projektledning, som varken kan definieras som en given process eller teknik att utveckla mjukvara på. Därmed stödjer Scrum, i teorin, införandet av egna processer och tekniker som passar ens egen verksamhet och önskemål. Vi hävdar att denna beskrivning stämmer överens väl med resultatet av undersökningen.

I de fyra intervjuerna, vars intervjupersoner alla arbetade i olika projekt med olika kunder, arbetade de alla i en unik miljö med en egen implementation av Scrum. Det infattar även de processer de hade infört som stödjer skapandet av systemdokumentation. Intervjun vi hade med A3, vars påståenden reflekterar Scrums flexibilitet: "*Det är väldigt mycket upp till ens egna implementation av Scrum. Såsom vi arbetar anpassar vi modellen till kunden vi arbetar med, och efter hur de vill jobba*" (A3, 18).

Vi har identifierat flera exempel i föregående kapitel på intervjupersoner i vars projekt hanterar, värdesätter, och prioriterar systemdokumentation olika i sin implementation av Scrum. Vi ser inget samband som stödjer teorier att systemdokumentation i agil metodik skulle vara efterfrågad från kund och skulle ses som ett affärskrav. Vi har lyft fram exempel i föregående kapitel som visar att systemdokumentation betraktas både som ett affärskrav och ett processkrav i Scrum. Exempelvis arbetar A1 på uppdrag hos en kund, vars organisation kräver skapandet av systemdokumentation. A1 hävdar att de dokumenterar både för sig själva och för kunden. Kunden i denna kontext är en slutkonsument av en produkt som utvecklas av flera team. Därför resonerar vi att de värdesätter skapandet av systemdokumentation högt, med anledning av riskminimering och geografisk distribuering av deras team.

Vi ser därmed ett samband, med koppling till ovan citat om Scrums flexibilitet, att skapandet av systemdokumentation stöds i verksamheters egna implementationer av Scrum.

Det finns inget samband som stödjer forskares påstående att systemdokumentation skulle vara lätt, men tillräcklig. Scrum har ett antal riktlinjer angivna för hur krav ska tas fram, där backlogs och user stories används. Detta har vi också sett samtliga i undersökningen följt. Men vad gäller övrig systemdokumentation, implementerar varje intervjupersons projekt en egen dokumentationsprocess.

5.2 Dynamiska och gränsöverskridande roller i Scrum

Undersökningen har identifierat att Scrum stödjer dedikerade dokumenterare som ansvarar för dokumenteringen av utvecklingsteamets och andra rollers aktiviteter. Detta är delvis i linje med det som presenterades i den modell (se figur 2) som presenterades, med undantaget av dokumenteraren som ingår i rollen för utvecklingsteamet. Vi resonerar att denna teori inte är praktiskt tilltagbar eftersom en statisk dokumenteringsroll i Scrum tar bort den flexibilitet som undersökningen visat att Scrum faktiskt stödjer. Detta eftersom undersökningen visat att systemdokumentation värdesätts och skapas i olika grad. Därmed hävdar vi att dedikerade dokumenteringsroller blir en tyngd som negativt präglar den så kallade "lättrörliga" agila metodiken.

Majoriteten av intervjupersoner i undersökningen hävdar att det är individerna i utvecklingsteamet som tillsammans ansvarade för dokumenteringen. Vi resonerar därför att rollerna i Scrum anpassar sig dynamiskt i förhållande till tidigare resonemang, hur respektive projekts implementation och hur de värdesätter skapandet av systemdokumentation. Dynamisk teamuppsättning, kopplat med ett gränsöverskridande ansvar för hur själva systemdokumentationen skapas i teamen.

Undersökningen har bevisat att samtliga projekt följer Scrums riktlinjer vad gäller de tre officiella rollerna. En tydlig skillnad vi identifierat är den interna teamuppsättningen i rollen för utvecklingsteamet. För utvecklingsteamets uppsättning ser vi att det finns givna roller, så som utvecklare, med specifika ansvarsområden. Trots detta ser vi att flera av våra intervjupersoner har givna roller, fast med friare ansvarsområden: "*Vi har givna roller, fast många sitter på flera stolar.*" (B1, 69). Vi märker att rollerna i utvecklingsteamet är givna men uppsättningen sker efter erfarenhet och kunskap som passar in på projektets uppgift.

5.3 Självbeskrivande kod

Litteraturgenomgången definierar begreppet för koddokumentation, som ett sätt att dokumentera systemets uppbyggnad internt i källkoden. Exempelvis är källkodskommentarer en form av koddokumentation.

I undersökningen visade det sig att A2 och B1 använder sig av källkodskommentarer. B1 som "*försöker skriva en kort beskrivning i koden*" (B1, 36), och A2 som tror att utvecklarna använder källkodskommentarer. A2, som kravanalytiker, har dock varken haft insyn i eller ansvar att granska källkoden. Vilket vi resonerar gör A2:s påstående ej trovärdigt.

Vi ser ett samband där både A1 och A3 påpekar att källkodskommentarer är onödigt. Att det är en systemdokumentation som inte ska behöva göras. Är det en funktion som är svårkodad så har du alltså gjort fel. Vidare påpekar A3 att källkodskommentarer är onödigt för att kod ska vara enkel och självbeskrivande. A1 motiverar: "*Absolut, för viss bra skriven kod, så talar den sitt tydliga språk, då ska det inte behövas så mycket annat.*" (A1, 106).

I litteraturgenomgången förespråkar flera forskare användningen av källkodskommentarer. Vi kan inte hitta ett sådant samband i undersökningen som stödjer detta, utan i praktiken är det få som ser nyttan i det, och färre som faktiskt använder det. Med begränsad erfarenhet och förståelse för hur utveckling sker i praktiken, kan vi se nyttan i källkodskommentarer för att hjälpa oss som nyanställda i ett projekt. Samtidigt resonerar vi som A1 och A3, att kod ska vara självbeskrivande och bör tala för sig själv. Dessutom så motsäger det teorier att agil systemdokumentation ska vara lätt, något som vi i undersökningen sett i beblandad mening.

5.4 Webbaserad systemdokumentation

I litteraturgenomgången beskrev vi hur forskarna påpekar att agila utvecklingsteam har börjat använda wikisidor allt mer för att hantera dokumentation av extern systemdokumentation. Mer specifikt, påpekar forskare att genom att använda wikisidor så samlas all information på ett och samma ställe. Vi resonerar att detta är till fördel för de som arbetar geografiskt distruberat. I undersökningen har vi ingen direkt insikt i hur projektens team var geografiskt distribuerade, eftersom detta inte var vår uppgift att undersöka. Däremot har vi hittat att samtliga deltagare i undersökningen har i sina projekt använt sig av interna wikisidor. Där har de placerat sin systemdokumentation i olika utsträckning med hänsyn till krav, design och arkitektur, samt teknisk systemdokumentation. Vi resonerar som A1 att systemdokumentation finns på en digital webbplattform: "*Tidigare har varje produkt varit en pärm, här har ni, så här använder ni. Nu används det mer och mer en onlinetjänst*" (A1, 88).

Som tidigare nämnt i litteraturgenomgången beskriver Rüping (2003) att wiki implementerar projektets dokumentlandskap, där det finns möjligheter till arkiveringsfunktioner för att hantera versioner och säkerhetskopiering via back-end plugins, något som tidigare enbart tillhört specialiserade dokumenthanteringssystem. Efter undersökningen kan vi avgöra att dokumenthanteringssystem fortfarande används. Majoriteten av intervjupersonerna använde sig av ett dokumenthanteringssystem när det berör systemdokumentationen av krav.

Däremot använder de också wiki som en kombination med hanteringssystem för att även kunna dokumentera övrig systemdokumentation. Vi kan se ett tydligt samband som pekar på att lagring och hantering av systemdokumentation övergår mer till en digital plattform.

5.5 Agil systemdokumentation - iterativt och evolutionärt

"*Beroende på vilken feedback vi och våra utvecklare får från olika team som utvecklat i iterationer, så sätts det allt på spel. Därför ser vi över och gör all systemdokumentation kontinuerligt*" (A1, 70).

Scrums tillvägagångssätt för systemdokumentation är en viktig aspekt som vi tagit stor hänsyn till vid undersökningen. Litteraturgenomgången beskrev det agila tillvägagångssättet i förhållande till systemdokumentation i detalj. De teorier som vi presenterat från forskare och författare i litteraturgenomgången stämmer bra överens med empirin, med hänsyn till de tre begreppen som vi undersökt. I det teoretiska (se tabell 1) hittar vi oberoende forskning av både Ambler (2014a) och Lacey (2012), som påstår att under krav ska det skapas en översiktlig beskrivning av kraven i början av projektet. Då fångas idéer och koncept, och med ytterligare systemdokumentation som sker löpande vid varje sprint. En liknande slutsats kan även vi dra för Scrums tillvägagångssätt. Tvärs över alla tre begrepp vi undersökt, hävdar vi att alla typer av systemdokumentation i Scrum skapas löpande. Löpande i den mån att det skapas enligt principerna för Scrums sprintar.

Ambler (2014b) menar även att systemdokumentationen ska återhållas så länge som möjligt, för att inte dokumentera spekulativa begrepp, utan stabila. Detta anser vi stämmer bra överens med resultatet av undersökningen, med ett exempel från intervjun med A1 som säger en liknande sak gällande teknisk systemdokumentation: "*När det väl har snurrat ett par tag och börjat sätta sig ett tag och vi har nått en bra grund, så dokumenterar man det*" (A1, 79). Vi ser ett samband att fler också skapar systemdokumentation så sent som möjligt i en given sprint, även inom krav och arkitektur. Vad gäller krav, så skapas de så sent som möjligt i varje sprint: "*Vi skapade inte krav till exempel fyra månader innan det skulle implementeras, utan eftersom mycket förändras så satte vi kraven eller user storiesen så sent inpå sprinten som möjligt*" (A2, 78).

Empirin visar att agil arkitektur sällan beskrivs i formella dokument eller digitala representationer. Att arkitektur i Scrum är evolutionärt, är även något som stöds av empirin. Som den biologiska evolutionen, påbörjades den med en enkel form av liv. På liknande sätt skapar den agila arkitekturen en övergripande plan för systemet i början, ofta formad ur några idéer och koncept som oftast kommer uttryckt från den vision av systemet som kunden efterfrågat. Till exempel beskriver A3 några så kallade "visionsworkshops" där utvecklingsteamet i början av projektet i samråd med kund, tar fram en gemensam förståelse av vad kunden efterfrågar. Detta med syfte så de kan skapa en övergripande plan och vision för projektet.

Det finns ett undantag i undersökningen, som vi redogjort för i resultatet. I fallet för A2 har de i projektet fått en färdig systemarkitektur från en dedikerad arkitekt från en annan del av organisationen (som inte varit en officiell person i teamet). Detta har med anledning hänt eftersom A2 arbetat i ett Scrumprojekt i en organisation som utövar Vattenfallsmetoden. De hade samtidigt fria händer att utveckla och anpassa en arkitektur som passade deras projekt, som de skapade iterativt enligt Scrums sprintar. Detta talar verkligen för flexibiliteten i Scrum, eftersom de i projektet som A2 var delaktig i befann sig i en Vattenfallsomgivning. Trots det kunde de anpassa den tillämpade dokumentationsprocessen och de höga dokumentationskraven som ställdes på projektet, så systemdokumentationen skapades enligt principera för Scrum.

Vi resonerar därmed att systemdokumentation i Scrum är agil i den mån att den är iterativ och flexibel med hänsyn till efterfrågan på systemdokumentation. Undersökningen har också visat att agil systemdokumentation rent strategiskt varken ser systemdokumentation som ett processkrav eller ett affärskrav, då vi identifierat både kunders önskemål och organisationers interna krav på skapandet av systemdokumentation.

Undersökningen har bevisat att Scrum inte har en given dokumentationsprocess som efterföljs, till skillnad från flera teorier vi presenterat. Scrum stödjer istället en dokumentationsprocess som är flexibelt anpassningsbar till projeketet, baserat på de krav som kund och organisationen ställer på projektet. Därmed är det upp till respektive implementerad dokumentationsprocess att fullfölja den grad av systemdokumentation som projektet eftersträvar.

6 Slutsats

Detta kapitel sammanfattar studiens och redogör för dess slutsatser som dragits utifrån en analys av empirin och teorin, och besvarar studiens forskningsfråga.

Vi har i vår introduktion presenterat en problematisering som lett till en konkretisering av studiens forskningsfråga:
Hur värdesätts och hanteras skapandet av systemdokumentation i Scrum?

Vi har i studien genomgående granskat hur systemdokumentation skapas i Scrum. Vi har definierat systemdokumentation som ett samlingsbegrepp och identifierat tre stycken underordnade begrepp (se sektion 2.3.2). De tre begreppen undersöks med koppling till de tre aspekterna: strategi och tillvägagångssätt, roller och ansvarsfördelning, samt dokumentationstyper. Begreppen och aspekterna finns angivna i undersökningsmodellen, i vilken varit utgångspunkten för vår undersökning.

Scrum stödjer både projekt med professionella dokumenterare och projekt där utvecklare tillsammans ansvarar för skapandet av systemdokumentation. Givet ett projekt som värdesätter skapandet och kvalitén av systemdokumentation, har vi i föregående kapitel dessutom lyckats identifiera flera hjälpmedel, verktyg och aktiviteter som Scrum faktiskt stödjer. Bland annat digitala och fysiska user stories, acceptanskriterier och digitala webbverktyg. Vi resonerar att dessa kan stödja och kvalitetssäkra skapandet av systemdokumentation för utvecklare och mindre kvalificerade personer i projekt. Nedan sammanfattar vi några slutsatser som är relevanta till vår forskningsfråga.

Källkodskommentarer är överflödiga och skapar tung systemdokumentation. Scrum har visat sig ge stöd för koddokumentation, men undersökningen har visat att koddokumentation är relativt oanvänd i praktiken. Kod ska vara självbeskrivande, och därmed är källkodskommentarer överflödiga och tidskrävande att anpassa till den aktuella kodbasen.

Agil systemdokumentation skapas och lagras på webben. Samtliga i undersökningen utnyttjade en blandning av intranät och wikisidor för att skapa och samla sin systemdokumentation.

Majoriteten av intervjupersonerna använde sig av ett dokumenthanteringssystem för systemdokumentationen av krav. Dessutom använder de också wikisidor i kombination med dokumenthanteringssystem för att även kunna skapa och lagra övrig systemdokumentation.

Team organiseras dynamiskt i förhållande till kraven. Det finns tre angivna roller i Scrums ramverk, vars rollstruktur efterföljs. Men vad gäller i rollen för utvecklingsteamet, är dess interna organisering anpassningsbar. Uppsättningen i utvecklingsteamet är baserad på erfarenhet och kunskap enligt projektets krav. Detta sker enligt projektets egna implementation av Scrum, kundens önskemål, samt hur organisationen värdesätter skapandet av systemdokumentation.

Kunden och organisationen avgör projektets krav för hur systemdokumentation skapas. Majoriteten av intervjupersoner svarade att de hade någon form av riktlinje för hur de skulle dokumentera, oavsett vad kunden önskade. Detta motbevisar tidigare forskning att systemdokumentation skulle ses som affärskrav i agil metodik. Agil systemdokumentation skapas både utifrån affärskrav från kund, och från processkrav från organisationer och dess formella processer.

Agil systemdokumentation skapas iterativt och evolutionärt. Dokumentation skapas kontinuerligt. Vissa undantag förekommer vad gäller exempelvis arkitektur, men vi resonerar ändå att regeln består. Detta eftersom de fortfarande har skapat och anpassat befintlig systemdokumentation löpande enligt principerna för Scrums sprintar.

Avslutningsvis, som svar på forskningsfrågan hävdar vi att det inte finns ett tydligt sätt som alla intervjupersoner efterföljt i sina respektive projekt. Det finns inget som tyder på att skapandet av systemdokumentation skulle vara bristfälligt i Scrum, eftersom Scrum stödjer implementeringen av en egen dokumentationsprocess. Inte heller finns det någon gemensam värdering vad gäller skapandet av systemdokumentation. Det är istället upp till varje projekts implementation av Scrum att anpassa en dokumentationsprocess som är i linje med projektets affärs- och processkrav.

Bilagor

Bilaga 1 - Intervjufrågor

Bakgrund

1. *Om dig och din roll i företaget*
    a. Berätta lite kort om dig och om din roll i företaget
    b. Hur länge har du arbetat inom företaget och branschen som detta?
    c. Beskriv företaget kortfattat och ge gärna ett exempel på ett projekt som du varit delaktig i
2. Någon övrig information som du vill tillägga?

Dokumentationsprocessen hos er

Forskare har sedan en tid tillbaka kritiserat Scrums dokumentationsprocess och i hur människor väljer att hantera systemdokumentation i agil metodik. Det finns dock forskare som hävdar att dokumenteringen bara fått en ny skepnad, då många verksamheter övergått från traditionella dokumentdrivna metoder till agila, mer kommunikationsdrivna metoder. Vi tänker undersöka hur Scrum stödjer skapandet av systemdokumentation i förhållande till tre begrepp: krav, teknisk systemdokumentation, samt design och arkitektur.

Hjälpmedel och tillvägagångssätt

1. *I Scrum så talas det i kravsammanhang mycket om user stories, product backlogs, sprint logs, m.m. som en form av systemdokumentation.* Kan du ge några exempel på aktiviteter som ni dokumenterar? Givet tre begrepp som vi undersöker:
    i. *Krav*
    ii. *Teknisk systemdokumentation*
    iii. *Design och arkitektur*

2. *Forskning från Lund har visat att Scrum har ett bristfälligt stöd för systemdokumentation, och vissa företag inför därför en egen dokumentationsprocess.*

    a. Hur går ni tillväga när ni skapar systemdokumentation? Måla gärna upp ett scenario där ni skapar och dokumenterar exempelvis ett krav.
    b. Hur skapar ni övrig systemdokumentation? *Givet de tre begrepp som vi redogjort för.*

3. *Forskare talar om tekniska hjälpmedel såsom videofilmning och post-it lappar för att beskriva krav.* Kan du ge några exempel på dokumentationstyper, hjälpmedel eller annat som ni använder för att beskriva ett krav eller system?
    i. Källkodskommentarer
    ii. Externa systembeskrivningar
    iii. Tekniska hjälpmedel, t.ex. wikis eller ljudinspelningar
    iv. Modellering

4. Har ni några riktlinjer för hur systemdokumentation ska utformas och dokumenteras? Kan du placera var i projektlivscykeln ni skapar krav eller annan systemdokumentation?

Strategier

5. *En av det agila manifestets fyra kärnvärderingar är att prioritera kod över omfattande systemdokumentation. Flera forskare hävdar även att systemdokumentation ska vara kortfattad och efterfrågad eller att inte "göra mer än vad som krävs".*

    a. Hur värdesätter ni systemdokumentation?
        i. Krav från kund "affärskrav" vs. tekniskt krav "processkrav"
        ii. Verbal kommunikation vs. skriftlig kommunikation
    b. Hur prioriterar ni systemdokumentation?

Roller & ansvarsfördelning

6. *En av det agila manifestets tolv principer hävdar att krav, design och arkitektur bäst växer fram ur självorganiserande team.*

    a. Har ni några givna roller eller personer i era projekt? Beskriv hur ni organiserar era team.
    b. Hur ansvarar ni för dokumentering av respektive begrepp?
        i. Dedikerade roller vs. situationsanpassat ansvar
        ii. Eget ansvar vs. delat ansvar
    c. Vem brukar vanligtvis ansvara för dokumentering av exempelvis krav, design, arkitektur eller systemspecifikationer?

Bilaga 2 - Intervju A1

Utförd 02/05/14 13:00
A1 = Intervjuperson
O = Oskar Nielsen
M = Martin Lindqvist

M: Kan du berätta lite om dig?

A1: Ja, jag är anställd hos företag A, men är på uppdrag här hos en kund. Här jobbar jag som projektledare och agil coach. Jag har jobbat på det här uppdraget i snart två år. Tidigare har jag jobbat som ScrumMaster i några mindre interna projekt hos företag A, före jag började här.

M: Hur lång erfarenhet har du som ScrumMaster?

A1: Ungefär fyra år har jag jobbat som det.

M: Och kan du kort beskriva företag A och vad ni gör?

A1: Ja! Företag A är ett kunskapsföretag som specialiserar sig framförallt inom .NET med hjälp av Lean och agil metodik.

O: Utmärkt! En kort och liten beskrivning. Nu tänkte vi inleda resten av intervjufrågorna med vissa påståenden som vi sedan riktar med frågor till er.

O: I kravsammanhang talar man ofta om konkreta aktiviteter som product- och sprint backlogs, user stories m.m. som en form av systemdokumentation. Kan du ge några exempel på aktiviteter som ni dokumenterar? Givet dessa tre begrepp som vi redogjort för tidigare.

A1: Absolut. Det finns först ett par nivåer som ni måste förstå när ett projekt tas fram. Den första och högsta nivån är produktägarna och produktledningen, de kommer med sina övergripande visioner och så. Det dokumenterar de i s.k. themes. (A1 ritar på en whiteboard) Högst upp är themes och är väldigt högt och brett och kan vara hur fluffigt som helst.

O: Ett koncept?

A1: Just ja. Efter den kommer ett steg under som heter scenario, och det är fortfarande produktledningen och produktägaren som diskuterar detta. Därefter kristalliserar de till mer handgripliga saker som man faktiskt kan göra, och så har vi något som kallas feature set. De består i sin tur av olika features. Men det är först när det operativa teamet ska börja arbeta så är det features som de tittar på. Det är först när de bryter upp features som user stories kommer in, och tasks.

O: Så en feature är en enhet av ett antal krav som efterfrågas?

A1: Ja, så skulle man kunna säga. Så här längst uppe i themes skulle det kunna vara en plan om att göra en tabletlösning till produkten och sen bryts den ner i mindre bitar och utkristalliserar sig. Det är den kravstrukturen vi har på detta uppdraget.

O: Nu har du redan nosat på vår nästa fråga, men skulle du kunna kort måla upp ett scenario där ni tar fram ett krav?

A1: Nu har vi ju skapat olika personas. Beroende på var användaren är i systemet skapar vi krav. Rör det sig om en systemadministratör, eller en vanlig användare som ska använda produkten på en daglig basis. En persona har ju en viss egenskap, och då skapar vi krav utifrån vad de försöker göra.

O: Men om vi försöker koppla systemdokumentation, vad gäller exempelvis themes, hur får ni dessa instruktioner, från ledningen?

A1: Ja, det kan röra sig om vad som helst. Ett luddigt mail eller en power point presentation eller ett worddokument. Men den idén måste kristalliseras och bli tydligare, för när vi kommer till features, måste utvecklingsteamet veta vilken funktionalitet med mera som efterfrågas när de tar fram varje user story.

O: Men hur tydliga är era user stories?

A1: Dem är tydliga. När vi är på den nivån, använder vi user storyn som ett verktyg så alla vet vad som ska göras och kan gå dit och titta.

O: Så de finns skriftligt dokumenterade?

A1: Ja, det gör dem. Sen på daglig basis så har vi post-it lappar på fysiska tavlor, men vi har ändå ett centraliserat digitalt system där alla kraven finns. Produktägarna brukar vanligtvis sitta där och överse projekten på en lite högre nivå än enskilda user stories.

O: Hur rör du dig som ScrumMaster då?

A1: Jag arbetar i huvudsak med utvecklingsteamet och ser till så att arbetet rullar på och att de har vad de

behöver. Men sen så håller jag kontakten med den högre nivån av projektledning och produktägare, särskilt vid releaseplanering och sådant. De har ju oftast en enorm lista med saker som de vill ha, och då tittar vi på vad vi kan göra kanske ett halvår frammåt.

35

36 **O: Men vad dokumenterar du specifikt som ScrumMaster då, vad gäller exempelvis krav?**

37 A1: Jag behöver inte dokumentera särskilt mycket, mer än att sköta kommunikationen mellan nivåerna och individerna. Det är ju teamen själva som utvecklar user stories, så jag är mer av ett stöd så att de självorganiserade teamen kan lösa det. Men jag hjälper de skriva user stories bland annat.

38

39 **O: Toppen! Nu har vi varit inne på tekniska hjälpmedel och annat som hjälper er att utveckla kraven. Vi gör som innan och inleder med ett påstående.**

40 A1: Okej.

41

42 **O: Forskare har talat om tekniska hjälpmedel eller dokumentationssätt som kan hjälpa er utveckla krav. Det rör sig om att t.ex. videofilma användare eller använda enkla post-its för att samla in krav. Vilka verktyg använder ni?**

43 A1: Nu har jag inte varit jätteinvolverad, men vi har ju en hel UX-avdelning som arbetar med usability och framtagande av krav utifrån användarens behov.

44

45 **O: Men rent konkret nämnde du post-it lappar med user stories, och en digital kravlösning där ni samlar dem.**

46 A1: Jo, precis, det har vi. När vi faktiskt vet vad vi ska göra kan vi exempelvis filma eller skriva post-its som jag nämnde. De är bra hjälpmedel för att enkelt och snabbt skapa ett krav.

47

48 **O: Nu tänkte jag ställa en fråga för systemdokumentation, vilket är ett lite annorlunda ämne än vad gäller krav. Hur väljer ni att dokumentera systemen? Kommenterar ni exempelvis i källkoden eller använder ni några digitala plattformar för att beskriva ett system eller funktionalitet?**

49 A1: Nu har vi faktiskt en teammedlem som kommer ifrån systemdokumentation (en avdelning), som sköter mycket av systemdokumentationen. De är med från början när all planering sker, när alla user stories skapas, så är de fortfarande en del av teamet och dem aktiviteter som jag nämnde. När en user story är färdig, så har vi ett acceptanskriterie att det ska vara dokumenterat. Och systemdokumentation har sina egna user stories för att inte missa dem här sakerna. Sen är det ju olika vad för typ av systemdokumentation det gäller, så i fallet för användare av produkten, har vi en del av systemet med en onlinehjälp som de skapar, för just användaren. Där kan de även skriva saker som "vad behöver vi göra för att vidareutveckla systemet" och så vidare.

50

O: Onlinehjälp, kan du vara mer konkret vad ni gör där?

A1: Där finns all funktionalitet som vi utvecklar beskriven. Då har den här systemdokumentationspersonen varit en del av teamet, jobbat med att skriva user stories, också skrivit en del på denna onlinehjälp och dokumenterat vad vi gjort med systemet.

O: Men olika systemdokumentation är ju till för olika personer. systemdokumentation exempelvis riktar ju sig framförallt till utvecklare som står till uppgift att vidareutveckla ett system.

A1: Ja, och i den ändan har vi ju även då en intern Wiki som vi kallar Confluence. Vissa konsulter kan ju komma in och bygga en hel del funktionalitet, och efter x antal månader försvinner de kanske. Därför är det viktigt att deras tankar och hur funktionaliteten är tänkt ska fungera, dokumenteras så det sparas trots att personen har lämnat projektet. Och det är ju en del av en bra user story, att vi alltid utför lite systemdokumentation kontinuerligt, så att vi inte gör allt sista veckorna före vi släpper produkten.

O: Okej, då har vi en sista fråga vad gäller aspekten tillvägagångssätt. Har ni några riktlinjer för hur systemdokumentation ska skapas i allmänhet? Exempelvis dokument med mål eller mallar till hjälp för när ni utför systemdokumentationen?

A1: Där, återigen, ligger ansvaret på den teammedlemmen som kommer från systemdokumentation. De har ju det ansvaret och det är deras arbetsuppgifter, men jag har inte koll på huruvida de följer några mallar eller liknande.

O: Men vad gäller design och arkitektur, har ni några riktlinjer där då? Jag antar att ni modellerar för att beskriva en given arkitektur eller design.

A1: Mm, modellerar gör vi. Men vi har inga riktlinjer vad gäller design eller arkitektur. Det kan röra sig om en person som stått och ritat på en whiteboard och tagit en bild av den, och lagt upp på wikin. Och i det fallet kan det räcka och det som gäller tills vidare. I ett annat fall kan någon varit inne i Visio och skapat en digital modell. Så jag skulle inte säga att vi har något standardiserat sätt att arbeta med varken design eller arkitektur.

O: Men du nämnde några vanliga hjälpmedel för att modellera. Har ni några riktlinjer för hjälpmedel eller verktyg ni använder?

A1: Nej, men detta har på sistone varit en diskussion vi har haft.

O: Någon best practice, kanske?

A1: Just precis. När vi har kommit till vissa skeden i projekt och kanske inte hållit ihop så pass bra, behöver vi veta “vad är designen” och “hur tänkte ni” och därför är dessa svar på frågor som behöver dokumenteras. Det har tidigare egentligen bara rullat på enligt de tankar som en person i teamet eller så har haft vid ett skede.

68

69 **O: Då tänkte jag bara avsluta med en ytterligare fråga. Var i projektlivscykeln skulle du placera in systemdokumentation, gällande våra tre begrepp som vi redogjort för. Då exempelvis arkitektur och design som vi diskuterat nu?**

70 A1: Ja, i början när ingenting har gjorts är givetvis arkitektur någonting vi gör tidigt. Men det sker ju inkrement hela tiden. Visst, teman är ganska luddiga, men feature sets, beroende på vilken feedback vi och våra utvecklare får från olika team som utvecklat i iterationer, så sätts det allt på spel. Därför ser vi över och gör all systemdokumentation kontinuerligt.

71

72 **O: Och det gäller även systemdokumentation, som specifikationer och sådant som beskriver ett färdigt system?**

73 A1: Det är egentligen samma sak där. Det arbetet är en del och sker i varje iteration, så det anpassas hela tiden också utifrån de förändringar som uppstår.

74

75 **O: Som om en utvecklar en feature i systemet, har han som uppgift att uppdatera det så snart som möjligt i iterationer?**

76 A1: Precis.

77

78 **M: Då påpekar forskarna det här med att det behövs uppdateras väldigt ofta, och att man då behöver göra om det, och att det är onödigt, hur gör ni där? Skriver ni om det varje gång?**

79 A1: Där beror det på hur tidigt det är, vi kan fokusera just på en feature och försöker få ut den snabbt så att den fungerar, en bronsnivå. Så att vi inte lägger ner ett halvår eller så. Och när det väl har snurrat ett par tag och börjat sätta sig ett tag och vi har nått en bra grund, så dokumenterar man det. Vi gör det alltså inte direkt, utan det blir en större insats när vi har nått en bra grund.

80

81 **O: Okej, när det gäller arkitektur och design då? Är det samma sak där?**

82 A1: Ja, fast det kanske sitter lite mer som en grund, lite mer fast. Den ses självklart över också om det är något.

83

84 **O: Jo, men arkitektur ligger mycket till grund i projekt, i början.**

85 A1: Precis.

86

87 **O: En av det agila manifestets fyra kärnvärderingar är att prioritera kod över omfattande**

systemdokumentation. Flera forskare hävdar även att systemdokumentation ska vara kortfattad och efterfrågad eller att inte göra mer än vad som krävs. Och då är frågan, hur värdesätter ni systemdokumentation?

88 A1: Tidigare har varje produkt varit en pärm, här har ni, så här använder ni. Nu används det mer och mer en onlinetjänst. Man kan mäta mer vilka parti som används och behövs få hjälp om.

89

90 **M: Men vem skapar ni systemdokumentation för? I vilken utsträckning?**

91 A1: Gamla användare har säkert bra koll, men det är för att hjälpa nya att komma igång.

92

93 **O: Särskilt om det sitter en person som kan allt och sen slutar, då har hur mycket kunskap som helst gått förlorat.**

94 A1: Ja, och där är det också baserat på andra saker som att om du har varit duktig under hela tiden så är det mycket kontuerligt, design och så vidare.

95

96 **O: Hur skulle du säga att ni prioriterar systemdokumentation?**

97 A1: Mot, eller emot?...

98

99 **O: Ja, låt oss säga att en utvecklare har uppgift som att dokumentera så skulle han egentligen kunna skriva kod.**

100 A1: Jag skulle nog säga att det är väldigt viktigt, att om en utvecklare spenderar tid på dokumentera så är det inte så bra.

101

102 **O: Men hur är det verkligen?**

103 A1: Som jag kan se, så har det blivit bättre, men det är bristfälligt, det är ju alltid roligare att skriva kod än dokumentera.

104

105 **O: Är vissa moment viktigare att dokumentera än andra? Skulle du säga att arkitektur och så är viktigare..?**

106 A1: Absolut, för viss bra skriven kod, så talar den sitt tydliga språk, då ska det inte behövas så mycket annat.

107

M: Men om vi tar det här med källkodskommentarer, brukar ni kommentera mycket i själva koden?

A1: Jag är aldrig inne och kollar i koden, så det kan jag inte riktigt säga.

M: Dokumenterar ni för er själva eller är det ett krav från kunden?

A1: Det är både och. Vissa kunder är intresserade för den tekniska systemdokumentationen ifall de ska göra sina egna vidareutvecklingar eller använda produkten på ett annat sätt. Som ett affärskrav är det viktigt att det finns på plats för att det finns ett mervärde för kunden.

O: En av det agila manifestets tolv principer hävdar att krav, design och arkitektur bäst växer fram ur självorganiserande team. Frågan är då, har ni några givna roller eller personer för projektet? Hur ni organiserar era team?

A1: I en optimal värld så behövs inte jag finnas. Då är teamet så pass självorganiserade så de löser allt själva. Produktägaren ska ju aldrig komma med en lösning utan det är upp till teamet själva. Så i en optimal värld kan jag gå till andra projekt. Men här finns det ju testare, utvecklare och sen har vi en som sköter systemdokumentation och i vissa team har en UX som jobbar med det grafiska. Men samtidigt så är det inte så att ingen utvecklare kan inte testa. I ett team så..

O: Du är inte låst i en roll?

A1: Nä absolut inte. Visst det är folk som är anställda som utvecklare, men ingen som säger att den inte kan vara med och testa. Alla är med och försöker nå samma mål.

O: Låt oss säga att du är en utvecklare, ansvarar du för att dokumentera dina delar av koden, en funktion?

A1: Det är ju inte en utvecklare som skriver en funktion själv, utan det är teamet. Och i teamet har vi en som ansvarar för systemdokumentationen och enbart jobbar med det.

M: Så ni har en technical writer i varje team?

A1: Ja, precis. Vi har en skribent i varje team som sköter systemdokumentation.

O: Intressant, hur stora är era projekt, hur många medlemmar?

A1: 6-10 kanske.

M: Så utvecklarna utför egentligen ingen speciell systemdokumentation?

A1: Nä det är ju mer i design och arkitektur. Men de samarbetar med technical writers för att de är bra på olika saker.

O: Ni har alltså en roll som det faller på.

A1: Ja.

M: Scrum har väl egentligen inte en roll som det?

A1: Du kan ha.

O: Enligt litteratur och uppsatser så har vi fått uppfattningen att den egentligen inte finns.

A1: Okej, men technical writer i våra projekt ansvarar för systemdokumentation, vissa jobbar endast med användardokumentation och vissa med systemdokumentation.

O: Ni brukar ha en roll per projekt?

A1: Per team. Vissa delar av teamen skriver kanske inte så mycket om kod som användaren kan se utan mer backend. Varje team har alltså en utsedd technical writer som skriver det andra. De kan vara assignade till två eller tre teams för att det inte alltid finns tillräckligt med saker att göra i ett.

O: Hur delaktiga är de i skapandet av kraven och så vidare?

A1: De är med på alla planeringar t.ex.

O: Det var nog allt tror jag.

A1: Okej.

Bilaga 3 - Intervju A2

Utförd 02/05/14 14:30
A2 = Intervjuperson
O = Oskar Nielsen
M = Martin Lindqvist

O: Kan du berätta lite om dig och vad du gör på företag A?

A2: Jag är utbildad systemvetare från Lund, och har jobbat sedan 2005. Min roll på företaget är som specialiserad konsult inom krav och projektledning och lite annat. Jag har sedan tre år tillbaka arbetat ute hos en kund där jag arbetar som kravanalytiker.

O: Skulle du kunna kort beskriva företaget du arbetar på?

A2: Javisst! Företag A är ett specialistföretag som tillhandahåller konsulttjänster inom främst .NET och även Sharepoint där majoriteten av våra anställda är programmerare och andra typer av utvecklare. Så har vi även några få som jobbar i mitt område och även agil projektledning med ScrumMasters.

O: Ja, just precis. Vi har pratat med en av dina kollegor (från företag A) som faktiskt berättade att ni till nästan 90% bestod av utvecklare.

O: Vi tänkte bara till sist fråga om det är något du vill tillägga gällande din bakgrund, eller något du undrar över?

A2: Nej, det tror jag inte. Eller, jo, jag kan tillägga att min erfarenhet i det ämne som ni frågar mig om är ju egentligen inte baserat på företag A, eftersom vi inte är ett produktbolag.

O: Så du menar att du skickats ut till en kund, och din erfarenhet är hos kunden?

A2: Ja, precis. Företag A tar i vissa fall in uppdrag och utför dem internt, men i mitt fall har jag arbetat sedan tre år hos en kund.

O: Okej! Vi kommer öppna vissa frågor med påståenden, men i vilket fall, så här kommer den första frågan inom aspekterna hjälpmedel och tillvägagångssätt.

16

17 **O: I kravsammanhang så talar man om user stories, sprint- och product backlogs m.m. Kan du ge några exempel på aktiviteter som ni dokumenterar, eller som ni arbetar med? Om du kan ge några konkreta exempel?**

18 A2: Ja, precis, vi använde ju oss av de uppenbara sakerna som du precis nämnde, med sprint backlogs och burn-down charts. Vi använde oss av sprint backlogs, product backlogs och använde ett system, dvs ett kravhanteringssystem som hette Caliber. Det är i sig inte ett Scrum eller agilt verktyg, utan det är typ ett ärendehanteringssystem som hos kunden användes för kravhantering. Som en kravdatabas egentligen, för att koppla kraven till testfall som sen testarna skrev. Detta för att kunna följa upp och kunna se vilka testfall som var kopplade till vilka krav och så vidare. Men det som teamet använde för att få jobba med i varje sprint var ju egentligen lappar, fysiska lappar.

19

20 **O: Post-it lappar?**

21 A2: (Nickar) Så det blev ju lite redundant information, för man hade ju dels skrivit kraven fysiskt på lappar, som sedan sattes upp på backloggen på en vägg som sedan flyttades under tiden man jobbade. Men sedan hade vi ju även alla kraven i digital form med alla user stories och så vidare i det här Caliber, kravhanteringssystem. Enda syftet med det var ju att kunna koppla dessa till testfallen för att sen testavdelningen skulle kunna följa upp. Kunden var lite tokigt uppbyggd på ett konservativt sätt med den klassiska Vattenfallsmodellen där de hade en utvecklingsavdelning och en testavdelning, som två helt olika avdelningar med olika ansvar. Vi i vårt team jobbade med Scrum, som egentligen inte passade in i den där modellen (Vattenfall) där vi normalt ska ha testare i teamet. Istället fick vi hyra in testare från testavdelningen. Så vi fick anpassa verkligheten efter hur den ser ut i den uppbygga Vattenfallsmodellen och så de arbetar i resten av organisationen som de jobbade med.

22

23 **O: Då kan jag tänka mig att krav och systemdokumentationen av krav var väldigt viktigt.**

24 A2: Ja, precis med tanke på hur de jobbade i Vattenfall var det viktigt att varje krav skulle dokumenteras väl, och registreras i deras digitala verktyg Caliber.

25

26 **O: Okej! Nu har vi diskuterat väldigt mycket krav, men vad gäller systemdokumentation eller rena systemspecifikationer, kan du som tidigare ge exempel på hur ni beskriver ett system?**

27 A2: Vi arbetade aldrig riktigt på det sättet. Fokus låg väldigt mycket på att kommunicera oberoende krav eller user stories. Att de skulle vara små, oberoende, estimerbara och testbara, vilket vi följde ganska hårt. Det har ju så klart sina för- och nackdelar.

28

29 **O: Så väldigt detaljerade krav gällde?**

30 A2: Väldigt svartvita, dvs att funktionalitet inte ska vara snygg eller gå in på detaljer som att det ska finnas en röd knapp i högra hörnet. Utan dessa detaljer skulle lämnas från utvecklarna till UX-experterna som fanns

också. Det handlade om att man ska bygga en funktionalitet som användarna vill ha och den ska kunna vara testbar, och inte flummigt. Men varken en detaljerad kravspecifikation, eller en tjock lunta (systemdokumentation) som man tar fram innan utvecklingen, var aldrig något vi jobbade med. Detta eftersom världen är föränderlig och inte kan tre månader innan säga att "det är så här vi vill ha det". Saker kommer förändras.

31

32 **O: Det är intressant! Och vad gäller design och arkitektur, hur dokumenterades den?**

33 A2: Det teamet som jag jobbade med, vi jobbade med webbklienten till produkten som kunden arbetade med. De hade dels en tung klient, en Windowsklient, som större delen av organisationen jobbade med, och så hade vi vårt team som utvecklade webbklienten. Vi var ungefär 6 stycken utvecklare och kanske 2 testare. Vi levde vårt egna liv och kunde sköta och styra arkitekturen själva, och de besluten togs av utvecklarna tillsammans med någon övergripande systemarkitekt men som litade ganska mycket på utvecklarna.

34

35 **O: Så ni hade alltså någon sorts övergripande krav på arkitekturen.**

36 A2: Just, precis. Som kom från systemarkitekten.

37

38 **M: Så ni har dedikerade arkitekter som gör vad, exakt?**

39 A2: De sätter en roadmap för hur produkten ska utvecklas rent arkitektoniskt och försöker matcha det med krav från kunder och olika produktchefer och så vidare. De ser också till att arkitekturen efterföljs av utvecklarna, men de fokuserade mest på Windowsklienten, så vi i projektet med webbklienten hade mer fria händer. Men min uppgift där, var i huvudsak funktionalitet. Om en kund ville ha en viss feature, så var min uppgift att undersöka hur de ville att den skulle fungera exakt, och så skrev jag ett antal user stories, tillsammans med teamet.

40

41 **O: Ja, nu är du lite inne på vår nästa fråga. Vi tänkte nämligen fråga om du kunde måla upp ett scenario där du exempelvis skapar och dokumenterar ett krav?**

42 A2: Okej, först och främst hade vi en så kallad master backlog, som en product backlog. Den tog vi fram tillsammans med produktägaren och en produktchef, som hade koll på marknaden eftersom han var ute och besökte kunder och hade en bild av vad de eftersökte. Vi hade regelbundna möten där vi gick igenom den här product backloggen och tittade på vilka features som fanns.

43

44 **O: Och detta gjorde ni för att?**

45 A2: För att få en övergripande status på projektet, har det tillkommit något nytt, prioriteringar och så vidare. Det som var längst upp i backloggen var det som brann mest, så att säga, och produktägarna tog beslut om vilka vi skulle ta in på kommande sprintar.

46

O: Hur fanns denna backlog tillgänglig?

A2: Product backloggen fanns faktiskt på vårt intranät som körde Sharepoint, i form av en excel-liknande lista. Där flyttade vi runt prioriteter, och sen muntligen kom vi överens om vad vi skulle jobba vidare med och vad som brinner mest just nu.

O: Och där kommer nästa steg?

A2: Där kommer nästa steg. Där kom jag in som skulle genomdriva den här undersökningen tillsammans med utvecklare och produktchefen för att se exakt vilka features som man vill uppnå. Vad som kraven går ut på och bryter ner det till en "epic", beroende på hur stor den var. I vissa fall var en feature så liten så den fick plats i ett antal user stories per feature. Då hade jag möten med utvecklare, arkitekter, testare, alltså teamet, tills de kände att de var tillräckligt bekväma med kravet och kunde estimera det.

O: Men du ansvarade inte för user stories?

A2: Jag tog ju fram dem tillsammans med utvecklarna, och såg till så att de var beskrivna enligt kunds efterfrågan. Jag var så att säga personen mellan kunden och utvecklarna, som tolkade kundens krav. Det kan ibland vara svårt för en utvecklare att förstå vad det kunden vill ha. De är oftast inne i koden och vet vad som är bäst, och det stämmer kanske inte alltid överens med det som kunden vill ha. Och det kunden vill ha kanske inte alltid är det mest optimala, men vill ha det, så är det bara i bita i det sura äpplet och göra det den jobbiga vägen och inte den enkla. Så det var min roll att se till så att när utvecklarna har exempelvis ett förslag, så förmedlar jag det till produktägarna och produktcheferna.

O: Hur exakt förmedlar du detta förslag?

A2: Oftast bara muntligt, eller genom mötesanteckningar som jag tar med vidare till mötena med produktägarna. Men nästa steg var att ta in detta krav eller user story i en sprint då den var tydligt beskriven och estimerat av teamet, då följde vi Scrum helt enkelt. Produktägaren tog in den i sprinten och teamet commitade.

O: Och vad gäller design och arkitektur, hur går ni till väga för att skapa den?

A2: Där var jag inte inblandad särskilt mycket. Som jag nämnde togs en övergripande plan fram för hur utvecklingen skulle ske, med systemarkitekten och vissa seniora utvecklare. Men med vissa krav måste ju en viss arkitektur eller funktionalitet så klart följa, när det gäller implementationsbiten. Men i min roll var jag inte särskilt intresserad av hur man valde att implementera det, utan jag fokuserade på en funktionalitet som motsvarade det kunden ville ha. Så det var teamet som ansvarade för den biten, tillsammans med arkitekten. Jag kan inte svara jättemycket vad gäller systemdokumentation och arkitektur egentligen eftersom det inte låg i mitt ansvarsområde.

O: Nu kommer vi till en ny fråga som du delvis svarat på redan. Hur dokumenterade ni kraven? Använde

ni ut av några hjälpmedel, t ex post-it lappar eller filmade ni en user story?

63 A2: Vi använde oss av user story cards eller kort. Där skrev vi user storyn på ena sidan ("I want to, so that…") och på baksidan hade vi acceptanskriterier, på lapparna skrev vi estimat och så vidare. Sen hade vi som sagt de digitalt också i Caliber. Det ena syftet var ju att koppla kraven till ett testfall, men ett annat syfte var även för att vi hade en traditionell projektledare i Vattenfallsmodellen kunde följa statusen på kraven, t.ex. "redo att utvecklas", "redo för test", "testas", eller "godkänt". Olika states som man ska slå om i det här verktyget, bara så att projektledaren ska få ut en rapport som han visa för sina chefer. Så det var mycket overhead och onödigt arbete egentligen.

64

65 **O: Ja, det är verkligen ett intressant sätt att arbeta på. Det är lite som att de infört en hybrid Vattenfall-Scrum modell.**

66 A2: Jo, precis! Vi var faktiskt det enda teamet som körde Scrum på kontoret där åtminstone, övriga organisationen körde en egen modell som skulle vara agil, men som påminde mycket om Vattenfall. Så det blev ju en hybridlösning i sättet vi arbetade, att vi jobbade enligt Scrum i en Vattenfallsomgivning. Däremot försökte vi få så att utvecklingsteamet inte var påverkade alltför mycket hur det var utanför projektet. Men vi som hade roller som var mer "overhead-aktiga", vi fick anpassa oss till omgivningen och försöka jobba med den.

67

68 **O: Om jag får upprepa den tidigare frågan vad gäller tekniska hjälpmedel eller tekniker, använde ni källkodskommentarer eller liknande för att beskriva systemen?**

69 A2: Nej, egentligen vet jag inte hur de löste det i koden.

70

71 **M: Vad är det kunden oftast efterfrågar? När det gäller exempelvis källkoden.**

72 A2: De kraven som fanns på att dokumentera var att det skulle dokumenteras i deras program, Caliber. Om det inte hade varit ett krav så hade vi inte dokumenterat i caliber, utan i ett annat system. Det var en riktlinje för projektet. Även ifall källkodskommentarer inte är mitt område så är jag rätt säker på att det var ett krav. Det kommer att vara många utvecklare som jobbar inom samma delar med koden. Det fanns krav på att skriva systemdesigndokumentation, för varje implementation skulle man komplettera och uppdatera systemdokumentation, och det var utvecklarna som gjorde det.

73

74 **O: Det dokumenterades över intranät?**

75 A2: Ja precis, online worddokument som låg på sharepoint.

76

77 **O: Nästa fråga, vad gäller riktlinjer, kan du placera var i produktlivscykeln ni skapar krav eller annan form av systemdokumentation?**

78 A2: Arkitekturen är något som är mer långsiktigt. Tex. Roadmaps. Vad det gäller krav är det lite olika, allt från

några dagar innan sprinten kör igång tills två veckor innan. Vi gick igenom varje krav och uppdaterade dem för produktbackloggen. Vi skapade inte krav till exempel fyra månader innan det skulle implementeras, utan eftersom mycket förändras så satte vi kraven eller user storiesen så sent inpå sprinten som möjligt. När man har implementerat en ny feature som påverkar systemdesignen så ska systemdokumentationen uppdaters. Den uppdateras i slutet av en sprint.

79

80 **O: En av det agila manifestets fyra kärnvärderingar är att prioritera kod över omfattande systemdokumentation. Flera forskare hävdar även att systemdokumentation ska vara kortfattad och efterfrågad eller att inte göra mer än vad som krävs. Svenska lagom, det som gäller i scrum och många andra metodiker.**

81 A2: Intressant.

82

83 **O: Hur värdersätter ni systemdokumentationen?**

84 A2: Vi värdesätter, på företaget schneider, som jag nämnde innan dokumenterade vi kraven redudant, fysiskt och digitalt. Teamet värdesatte det högt att kraven var tydliga, att user storiesen var ordentliga så de visste exakt hur det skulle fungera. Vi hade user storiesen fysiskt på kort, vilket teamet uppskattade. Vi fokuserade mer på att förstå hur det ska fungera istället för hur det ska se ut. Kunden vill ha en funktionalitet och ska inte behöva se vad som pågår bakom kulisserna.

85

86 **O: Arkitektur eller?**

87 A2: Ja, arkitektur och så vidare, det vågar jag inte riktigt svara på det, eftersom det är inte mitt område.

88

89 **O: Men det är något som regel tillhör varje projekt?**

90 A2: Det gör det, och det är något som produktägaren ansvarade för i detta fallet. Jag vet att utvecklarna värdesatte systemdokumentationerna väldigt högt för att många olika utvecklarna kunde göra olika saker i koden. Det kan vara buggar också vidare. Det är väldigt viktigt att dokumentationen är uppdaterad och dessutom källkodkommentarerna. Testavdelning värdesatte också att genom tydliga acceptanskriterier veta vad de skulle göra. Så att de inte behövde testa en massa onödiga saker för att det kanske var otydliga krav.

91

92 **O: Precis, nästa fråga, hur prioriterar ni systemdokumentation? Som jag sa i det här påstående så är det agila att priotera kod istället för systemdokumentation, hur ser du på det?**

93 A2: Vi, i vårt team, körde nog enligt det agila manifestet. Att ha fungerande kod eller en produkt var viktigare än en massiv systemdokumentationsdatabas. Eftersom det kräver mycket tid att uppdatera.

94

95 **O: Påstående, en av det agila manifestets tolv principer hävdar att krav, design och arkitektur bäst**

växer fram ur självorganiserande team. Det är något de högt värderar. Har ni några givna roller eller personer i era projekt?

A2: Japp.

O: Kan du beskriva hur ni organisera era team?

A2: I vårt team körde vi en dedikerad scrummaster, en produktägare och en testledare som mer var ett krav. Sen hade vi min roll kravanalytiker. Dessa var rollerna som fanns.

O: Okej, en följdfråga då, hur ansvarar ni för dokumentering av de här respektive begreppen? Har ni en speciell person som ansvarar för systemdokumentation av kanske kod eller av en viss funktion? Eller skulle du säga att en utvecklare som utvecklar en funktion ansvarar för just den funktion som han utvecklar?

A2: När det gäller krav så var det mitt ansvar, att dokumentera dem och se till att de var tillräckligt bra beskrivna. När det gäller källkodskommentarer var det upp till just den utvecklare som utförde den funktionen. Han ansvarar också för att det uppdateras på den funktionen han uppdaterar.

O: Vi säger att det är två till tre utvecklar som ansvarar för en feature, ansvarar de då tillsammans i slutet av sprintet att det är dokumenterat?

A2: Jag är lite osäker om det. Men jag tror att det är hela teamet som ansvarar i en sprint att det är utfört. Sen att det är vissa som jobbar med olika saker under sprinten, men i slutet ansvarar teamet för att det är utfört i slutet av sprinten.

O: Ni hade ingen specifik roll för att arkitekturen ska vara på ett visst sätt?

A2: Vi hade en systemarkitekten som såg till för hela produktens arkitektur, och var det något som behövdes uppdaters där så var det han som utförde det. Det var en kommunikation mellan honom och utvecklarna om något behövdes ändras.

O: En form av informell kommunikation då?

A2: Ja precis, i vissa fall så blandade vi in systemarkitekten när vi undersökte och höll på att ta fram user stories, annars var det under sprinten.

O: Vem brukar vanligtvis ansvara för systemdokumentation för ex. krav, systemdokumentation eller design?

A2: Krav var jag som sagt, design och arkitektur var systemdesignern. Det finns en dedikerad organisation som heter technical publications som beskriver en användarmanual för produkten. Varje gång det tillkom en ny feature så skulle det tillkomma en systemdokumentation över “how-to”. En wiki, en webbaserad tjänst.

O: Som de leverade, det låg liksom utanför teamet?

A2: Ja precis, det var en helt separat process, tanken var att de skulle vara en del av teamet för att veta vad det var för något som skulle beskrivas. Men de blev oftast involverade i slutet av sprintarna. Det var en parallell process.

O: Så ni har alltså delvis ett antal dedikerade skribentroller fast med hänsyn till användardokumentation.

A2: Ja precis.

M: Det var alla våra frågor, tack för att du deltog.

A2: Ja, inga problem.

Bilaga 4 - Intervju A3

Utförd 05/05/14 12:20
A3 = Intervjuperson
O = Oskar Nielsen
M = Martin Lindqvist

O: Kan du berätta kort om vem du är och vad du gör här på företag A?

A3: Mm. Jag är en av grundarna till företag A. Varit med sen starten och har jobbat i mjukvaruutvecklingsbranschen i 13 år, mestadels som mjukvaruutvecklare. Jag är intresserad inte så mycket av processer, utan hur människor interagerar för att skapa en bra mjukvara. Därav har jag glidit in på det spåret och tittat på hur vi kan få människor till att samarbeta bättre. Jag har gjort lite allt möjligt som t.ex. agil coach, mentor, och i både tekniska och mjuka roller. Officiellt har jag rollen som ansvarig för applications, där jag är affärsområdesansvarig.

O: Mycket utveckling då?

A3: Custom utveckling, allt som inte har med mobilitet och Sharepoint.

O: Kan du nämna något kort om företag A också?

A3: Ja det kan jag väl göra. Vi är ett bolag som bygger högkvalitetsmjukvara på .NET-plattformen med hjälp av just agile och Lean metodik. Sen har vi tre affärsområden, applications, mobility, och collaboration.

O: Men ni är inte låsta till ert kontor? Jag menar som konsulter skickas man rimligtvis iväg till kunder också.

A3: Ja, vi har ungefär 50/50 in-house projekt och uppdrag där våra konsulter skickas ut till kudnder. Jag har erfarenhet av både att arbeta ute hos kund, senast på ett uppdrag i Helsingborg, men just nu sitter jag i ett projekt som vi utför här på kontoret i Lund.

O: Varför har ni detta upplägget?

A3: Det är för att vi inte ska bli instängda och köra ens egna metodik för mycket. Du lär dig inget nytt eller erfarenheter eller möjlighet att förbättra din process eller dig. Därför är det viktigt för oss att ha våra egna projekt internt som vi driver, men även att vi faktiskt är ute hos kund och tittar hur verkligheten ser ut och jobbar med dem och de problemen som de sitter i.

15 **O: Nu tänkte vi ta första frågan i vår faktiskta undersökning. Vi inleder med ett påstående och ställer frågan därefter.**

16

17 **O: I Scrums kravsammanang talar man ofta om aktiviteter som sprint- och product backlogs, user stories m.m. som en form av systemdokumentation. Hur och vad dokumenterar ni? Kan du ge några exempel på aktiviteter eller saker som ni dokumenterar?**

18 A3: Oj, först och främst borde jag nog förklara vår tolkning av Scrum. I mitt huvud åtminstone, är Scrum ett ramverk. Det talas mycket om saker som finns i Scrums ramverk, men väldigt lite om konkreta sätt hur man gör de på. Så det är väldigt mycket upp till ens egna implementation av Scrum. Såsom vi arbetar anpassar vi modellen till kunden vi arbetar med, och efter hur de vill jobba. Så finns det en product backlog, är det enkelt att göra en spint backlog. Jag tycker om att använda user stories som ett sätt att skapa ett underlag för en kravdiskussion. En user story är inte ett krav, utan ett underlag eller stöd. Eftersom en user story är ca tre rader text kan man inte riktigt beskriva ett krav. Men det är någonting jag kan ta med mig till kunden och sätta mig ned med honom, och fråga hur de tänkte när vi skrev denna user story. Genom den diskussionen kan man få ut acceptanskritetier som man kan dokumentera i samband med kravet. Där kommer de mer detaljerade kraven in i bilden. Min uppfattning av systemdokumentation är den saknar ett egenvärde. Är det ingen som tänker läsa systemdokumentationen, så tänker inte jag skriva den. Men mycket är beroende på projekt, på kund, och på kontext.

19

20 **O: Vad är viktigt för dig att dokumentera?**

21 A3: Ingenting. Ingenting är viktigt för mig att dokumentera. Vad som viktigt för mig är att skapa en bra mjukvara som kunden är nöjd med. All systemdokumentation är bara ett medel att nå dit. Oftast är det någon som ändrar sig och står i vägen för att skapa en bra mjukvara, kanske vi bör dokumentera den på slutet. Är det någon som inte riktigt vet vad de vill, är det kanske viktigt att vi dokumenterar kraven, och kanske viktigt att vi ritar och dokumenterar flöden och sånt. T.ex. pekar vi på ett flöde och berättar att dessa fyra user stories tillhör det. Men det är ju väldigt mycket beroende på. Är det någon kund som är villig att sitta här i knät på mig och har väldigt tydligt vad det är han vill ha, ser jag ingen poäng i att dokumentera alls, för då kan jag ju bara fråga honom. Så, vad som är viktigt för mig att dokumentera är det som hjälper mig skapa bra mjukvara, och det är väldigt mycket från situation till situation.

22

23 **O: Men kan du beskriva hur ni går tillväga när ni skapar exempelvis ett krav?**

24 A3: Ja, vi kan ta det projekt jag sitter i just nu som exempel. Där har vi ett antal user stories som fungerar som ett diskussionsunderlag. När vi sätter oss ner i iterationen så tittar vi på user stories, pratar igenom med kunden och frågar vad han menade, och sen kommer vi överens om vad som ska byggas.

25

26 **O: Ni har inte en högre nivå som ni arbetar med också? Jag tänker mig någon sorts vision för ett färdigt projekt. Kan du måla upp ett scenario från början?**

27 A3: Absolut! Ja, detta projekt då som exempel började som en kundvision på något som de vill bygga. Då börjar vi med att prata med kunden för att förstå vad denna visionen handlar om. Så det handlar om att vi

kallar kunden till oss där vi får utöva en visionsworkshop där vi fångar upp visionen och vad de ville bygga, och skrev ett par user stories. Sedan skapade vi ett projekt och backlog utifrån det. Därefter blev backloggen levande efter det, som genomgick kontinuerlig förändring. Den backloggen vi skapade vid visionsworkshoppen är väldigt annorlunda jämfört med den vi sitter med nu, halvvägs in i projektet.

O: Vad är syftet med denna visionsworkshop?

A3: Det är för att få alla att förstå vad det är vi ska bygga. Även user stories kan ju förändras, men visionen förblir densamma. Att krav och världen omkring förändras, det är ju så verkligheten ser ut.

O: Vad gäller design och arkitektur då, till exempel, hur går ni tillväga för att skapa arkitektur?

A3: Ja, jag personligen gillar att arbeta med agil arkitektur. Vilket är i princip ingen arkitektur alls. Det finns en grundtanke i att det finns vissa idéer och koncept som finns, men som kommer kanske att förändras över projektlivscykeln och hur saker skapas. Det gäller att ha ett modulärt perspektiv, som tillåter en att flytta kod och ändra saker, utan att bygga in dig i ett hörn. Det är som att bygga kod i vertikaler, vilket innebär att en funktion påverkar inte en annan jättemycket. Sedan handlar det om att skapa en infrastruktur som kan utnyttja de oberoende funktionerna. Det gör att du evolverar fram en arkitektur. Det gör att du i princip har fungerande kod redan efter en vecka att demonstrera för kunden. Till skillnad från människor som ger sig in i stora arkitekturella diskussioner som försöker modellera upp allting i förväg, sitter sex månader in i projekt med en pappersprodukt.

O: Men någonting konkret som andra vi pratat med har nämnt är begreppet roadmaps, och att skapar en övergripande arkitektur tidigt. Förekommer det även hos er?

A3: Ja, det gör det. Men det gäller att skjuta på alla beslut till det sista ansvarsfulla ögonblicket. Behöver vi veta någonting om tre veckor, väntar vi hellre med att göra det så sent som möjligt. Vi behöver inte ödlsa tid på att fundera på det nu.

O: Finns det några risker eller nackdelar med det tillvägagångssättet?

A3: Ja, risker finns om man inte gjort det tidigare, så är det lätt att man bygger in sig i ett hörn. Att man bygger saker som inte är modulära och tänker på hur jag faktiskt kan förändra eller plugga ut denna funktionen och göra om den till någonting annat. Man ska därför designa och bygga saker modulära så de fungerar bra. Använd SOLID-principerna och använd den typen av arkitektur, använd testdriven utveckling så du snabbt kan fakturera saker. Använd mycket av dessa tekniska best practices som gör att vi kan få det som kallas agil arkitektur. Vilket är en kodbas som snabbt kan förändras. Det finns ingen given mall, struktur eller riktlinjer för vad agil arkitektur är.

O: Och vad gäller ren systemdokumentation då?

A3: Det händer väldigt sällan. Jag brukar fråga kunden vad det är han ska ha systemdokumentationen till, och kan de inte riktigt kan svara på den frågan så ställer vi det på tvären och vägrar skriva den. Inte riktigt så hårt, kanske. Men man måste försöka prata med kunden och fråga om vem som är konsumenten, alltså vem som tänker läsa den, och om de vet varför de behöver den. Man måste fråga de om de är villiga att betala si och så mycket pengar för att vi ska ta fram den. Jag kan ta fram all systemdokumentationen, men finns det ingen konsument av den eller vet vad de ska ha den till, så fyller det inget värde att skapa den. Det handlar hela tiden om att skapa och leverera värde. Men jag vet enstaka projekt som har haft de kraven på sig, att leverera en detaljerad systemdokumentation.

O: Men du nämnder det som ett krav?

A3: Ja, eftersom det inte finns något egenvärde i att skapa systemdokumentation.

O: Har ni har inte några riktlinjer för att skapa systemdokumentation i eller utanför systemet för att beskriva någonting svårt eller komplext? Det kanske kan röra sig om någon svår funktion eller algoritm i koden.

A3: Alltså har du skrivit en svår funktion, har du gjort fel. Jag säger så här till andra utvecklare, kan inte jag förstå din kod eller vad den gör, så har du gjort fel. Gör om, gör rätt. Jag som normalfuntad utvecklare med den erfarenhetsnivån jag har måste kunna sätta mig och läsa koden och kunna avgöra vad den gör. Skulle jag behöva sitta och beskriva allting som jag gör i koden, är jag inte tillräckligt bra som utvecklare. Dessutom så skapar det en del problem, eftersom koden uppdateras kontinuerligt, medan den systemdokumentation kommer ingen att skriva om. Så vad händer om en ny person kommer in i projektet, och tar fram systemdokumentationen som inte alls stämmer överens med verkligheten? Då är det bättre att han sätter sig i koden och lär sig vad som görs. Kod ska vara enkel och självbeskrivande.

O: Och vad gäller mer mjuka delar av utveckling som design då? Exempelvis UI/UX.

A3: Ja, då finns ju användarmanualer, men det är ingenting som vi gör. I många fall så har en kund arbetat med en designbyrå och framtagit en grafisk manual eller riktlinjer för hur mjukvaran ska se ut, och så arbetar vi därefter.

O: Vad har du för förhållningssätt gällande systemdokumentation?

A3: Min inställning till systemdokumentation är att finns det ett syfte, så dokumenterar vi. Ställer kunden ett krav på att dokumentera och kan uppge ett skäl, så gör vi självklart det. Det gäller i stort sett all typ av systemdokumentation. Jag tvingar faktiskt våra kunder att ha backlogs, men vad exakt du har i den spelar mindre roll, som t.ex. user stories eller punktlistor. När kunder frågar oss vilket sätt som är bra, så brukar vi rekommendera user stories osv.

O: Har ni några tekniska hjälpmedel eller liknande för att beskriva era user stories? T.ex. filmar ni

ett samtal med en kund?

A3: Så som jag brukar göra det, är att sätta sig ner med en kund och prata med dem.

O: Och då skapar du en user story utifrån det?

A3: Ja, jag sitter och pratar med dem och skriver user stories samtidigt och sen visar det för dem och bekräftar, är det här det vi har pratat om. Det här kan man göra via en workshop eller “one on one”, det finns massa sätt. User storiesen är ett sätt att dokumentera samtalet.

O: Om vi tar det andra begreppet, arkitektur, är det samma sak där?

A3: Arkitektur är mer flyktigt, den ändras flera gånger i veckan, den är inte fast.

O: Om man tar en roadmap eller en vision för en kodbas.

A3: Ja, den finns i kodbasen, det nuvarande tillståndet som finns i kodbasen. Sen kanske du behöver någon att diskutera med, dem som sitter i ditt team, kanske rita på whiteboard. Du kan ta foto på whiteboarden, jag har faktiskt gjort det och testat att ta foto på whiteboarden för att se hur många gånger jag behövde gå tillbaka för att titta. Och det behövde jag nästan aldrig.

O: Vi har faktiskt sett ett exempel när de har fotat och sen lagt upp det på intranät.

A3: Ja det är en annan femma, om man jobbar geografiskt distruberat.

O: Ni kör en centraliserad modell?

A3: Ja, vi kör väldigt centraliserade teams. Vi har inga teams som går över våra kontor, och när det väl skulle hända så kommer det få helt andra konsekvenser.

O: Då är wikis inte något ni använder?

A3: Jo, det gör vi.

O: Kan du beskriva vad ni använder det till?

A3: Vi använder det vid vissa projekt där det kommer en grej som behöver fixas nån gång om året. Då kan det vara bra att det finns kontaktinfo om kunden. Eller vilka saker som ändrades sen senaste releasen. Sådana saker kan vara bra att skriva upp och då är wiki bra. Inloggningar och andra saker.

O: Men inte systemdokumentation som beskriver?

A3: Nä inte beskrivande systemdokumentation. Utan saker som kan vara bra att ha om du går in en gång om året i kodbasen och ska kolla. Så du inte är helt borta när du kollar i kodbasen. Hur du får koden att bygga, det kan vara svårt att se. Du kanske behöver koppla in plugins eller liknande, det kan vara bra att skriva upp för det kommer ingen att kunna klura ut. Samtidigt dock om du behöver skriva upp det så har du gjort något fel. När du sitter och skriver en systemdokumentation ska du försöka förstå varför du skriver den här och ifall det är onödigt så har du gjort något fel i koden, gå tillbak och gör det enkelt så slipper du. Men ja, det är ju långt ifrån idealt i alla projekt, så klart det finns wikis eller readme filer. Lättviktigt ska det vara i alla fall.

M: Har ni några riktlinjer för hur systemdokumentationen ska utformas? Vad gäller krav t.ex. Ni har ingen mall eller policy över det här ska göras?

A3: Det beror på projektet, är det ett projekt som vi ansvarar för så har vi vissa grundkriterier som måste vara uppfyllda. Det har dock väldigt lite med produkten att göra utan istället en systemdokumentation för projektet. Alltså vad driver projektet, vad är viktigt för kunden, hur ser framgång ut, den typen av frågor. Det är något som vi vill veta för det hjälper oss att lyckas med projektet.

O: Det är mer luddiga begrepp eller värderingar?

A3: Mm, typ drivers brukar vi prata om, vad driver detta projektet, är det budget eller är det tid? Ska det ha fastna leveransdatum.

M: Ni kategoriserar projektet alltså?

A3: Ja precis, lite vad som är viktigt för kunden så att vi kan jobba utifrån hur de bedömmer projektet.

O: Men mindre hur systemet ska vara uppbyggt?

A3: Väldigt lite, det är mer hur projektet är designat. Det har vi väldigt mycket guider, det ska finnas en a4 sida för varje projekt. Det är ett arbete som görs ihop med kunden. Sen är det ett samspel med teamet och kunden om hur mycket de ska dokumentera och vad det ska dokumentera.

O: En ytterligare fråga då, kan du placera in var i produktlivcykeln dokumenterar ni?

A3: Löpande.

O: Löpande? Men hur tidigt sker t.ex. design och arkitektur?

A3: Löpande.

99

100 **O: Vad är det första ni gör när ni initerar ett projekt?**

101 A3: Det första man gör är att man bestämmer vad man ska bygga. Vi pratar med en kund, kommer överens om det här är vår backlog och så här ser den ut. Prioriterar backloggen utifrån affärsvärde och vad de tycker är viktigt, och utifrån risker. Sen väljer vi det här ska vi bygga nästakommande vecka eller de två nästa veckorna.

102

103 **O: Och då bygger ni upp en sprint?**

104 A3: Precis, då börjar vi bygga. Man ska t.ex. bygga en funktion där man ska kunna lägga till en person i systemet. Då bygger vi det simpelt, som man ska kunna byta ut om man vill längre fram i projektet. Sen hela tiden ha den strategin och därpå börjar det kristalliseras en arkitektur, och därför säger jag dokumentera löpande för om en kund vill ha en detaljerad en systemdokumentation så är det lättare att skriva den, alltså kommer du att få skriva om den tusen gånger.

105

106 **O: Så då är det lika bra att skriva den när det är satt?**

107 A3: Nej, det är lika bra att skriva om den tusen gånger. Poängen med ett agilt projekt är att du ska kunna efter en iteration säga att du är färdig. Och det kan du inte göra om du har bara systemdokumentationen kvar, om kravet finns att systemdokumentationen ska finnas. Då måste den finnas löpande, för det är ingen som vill sitta i tre veckor och dokumentera utan istället sitta lite varje dag.

108

109 **O: Är det inte lite bortkastad tid att sitta och dokumentera något spekulativt?**

110 A3: Men det gör du inte. Du förutsätter att det är onödigt. Man får fråga sig är det här viktigt för mig, systemdokumentationen. Då är det en del av leverablen och är det en del av leverablen så ska det vara klart vid iterationen.

111

112 **M: Föregående som vi intervjuade sa att de fokuserade på att få ut en fungerade bronsnivå av funktionen och dokumenterade inte förrän de har en stabil version.**

113 A3: Jag säger som så, om kunden påstår att de vill ha systemdokumentationen så dokumenterar man löpande. Sen tror jag att det låter lite blodigare än vad det är, att det skulle förändras så mycket att du skulle behöva kasta all systemdokumentation, utan det är mest att något litet behöver uppdateras. Det är samma med användarmanual, det är bäst att göra det löpande istället för att göra det i slutet.

114

115 **O: I en annan aspekt, strategi, är vår fråga: Hur värdesätter ni systemdokumentation? En av det agila manifestets fyra kärnvärderingar är att prioritera kod över omfattande systemdokumentation. Flera forskare hävdar även att systemdokumentation ska vara kortfattad och efterfrågad eller att inte göra mer än vad som krävs.**

116 A3: Det det handlar om är om man hamnar i en prioriteringssituation där jag måste prioritera mellan systemdokumentation och fungerande mjukvara så måste jag prioritera fungerande mjukvara men det betyder inte att den är mer viktig. Det är ett förhållningssätt. Det finns ingen anledning att dokumentera bara för att dokumentera. Men kan vi se en risk med att vi inte dokumentera, behövs det göra för en överlämning finns det ett lagkrav, alltså kan någon berätta för mig att varför vi ska dokumentera. Jag kommer att vara först med att dokumentera, jag har inga problem med det, men det måste hela tiden finnas ett syfte, och strategin kommer ifrån syftet.

117

118 **O: Okej, hur skulle du om du fick prioritera skriftlig eller verbal kommunikation, vad skulle du välja?**

119 A3: Två personer vid en whiteboard är den bästa kommunikationen, det är research, jag håller vid det, det är svårslaget.

120

121 **M: Okej, rörande roller då, börjar med ett påstående då: En av det agila manifestets tolv principer hävdar att krav, design och arkitektur bäst växer fram ur självorganiserande team.**

122 A3: Det de handlar om är att du ska ha en teamsammansättning som har all den kompetensen för att klara av den uppgiften de ska klara av. När du har pratat med kunden och har förstått vad kunden vill ha, så får du designa teamet utifrån det. Säger då kunden att de vill ha jättemycket användarmanual, då hade jag sagt att vi hade fått ta in en teknisk dokumentatör på heltid för att du behöver så mycket systemdokumentation så då är det läge att ha en som kan göra det. Det är samma med UX. Det blir som sagt mycket utifrån kraven.

123

124 **O: Men om ni har rättsliga skäl om ni gör en applikation t.ex. vem gör det?**

125 A3: Men vad är lagkravet? Då får man gå tillbaka till kraven och lösa det där eftersom. Exempel finns det ju automatiserad systemdokumentation, det kan man komma rätt långt med.

126

127 **O: Okej, tack. Nu har vi de svar vi sökte. Då tycker jag vi sätter punkt för intervjun.**

128 A3: Tack.

Bilaga 5 - Intervju B1

Utförd 09/05/14 11:10
B1 = Intervjuperson
O = Oskar Nielsen
M = Martin Lindqvist

1 **O: Berätta lite kort om dig och om din roll i företaget**

2 B1: Jag är ETL-utvecklare, kundansvarig för en av affärerna som har ett DW hos oss samt scrummaster för hela Business Intelligencegruppen. Gruppen består av ca 7-8 personer.

3

4 **O: Hur länge har du arbetat inom företaget och branschen som detta?**

5 B1: Jag har arbetat på företag B i 2 år, och i branschen ungefär 7-8 år.

6

7 **O: Beskriv företaget kortfattat och ge gärna ett exempel på ett projekt som du varit delaktig i.**

8 B1: Vi är ett shared servicescenter för ett 20-tal mediaabolag. Vi är mycket inblandade i både bolagsspecifika projekt och sådana som innefattar hela koncernen eller en delmängd av bolagen. Just min grupp jobbar med BI och uppföljning både på en övergripande nivå och hela vägen ner till smådetaljer. Ex. vi har infört ett koncerngemensamt CRM-system och annonsbokningssystem, där jag haft ansvaret för uppföljningsdelarna.

9

10 **M: Någon övrig information som du vill tillägga?**

11 B1: Nej, det behövs inte.

12

13 **O: Vi inleder med ett påstående. I Scrum så talas det i kravsammanhang mycket om user stories, product backlogs, sprint logs, m.m. som en form av systemdokumentation. Kan du ge några exempel på aktiviteter som ni dokumenterar? Givet tre begrepp som vi undersöker: Krav, teknisk systemdokumentation och design & arkitektur.**

14 O: Vi börjar med krav t.ex.

15 B1: Vi skapar först en user story. När kraven formulerats klart adderas de till user storyn eller så hänvisar den till en annan user story. Vi lider av en ganska dålig kravkultur generellt, så det är inte så formaliserat som det kanske borde vara, ofta skriver jag kravet efter en diskussion med verksamheten, och de okejar formuleringen. Inte så detaljnoga.

O: Okej, om vi tar för systemdokumentationen då?

B1: Vi använder oss av confluence, en wiki egentligen. Här läggs systemdokumentationen upp under respektive område/system i den mån den finns.

O: Sen har vi då design & arkitektur, hur gör ni där?

B1: Det här dokumenteras mycket dåligt. Jag har påbörjat ett arbete med en ”best practice”-systemdokumentation som skall användas av alla utvecklare och konsulter, men idag existerar den framför allt på en muntlig nivå. Jag tar vissa beslut och skriver dem i user storyn, andra är upp till utvecklaren.

M: Vi inleder med ett till påstående, Forskning från Lund har visat att Scrum har ett bristfälligt stöd för systemdokumentation, och vissa företag inför därför en egen dokumentationsprocess. Hur går ni tillväga när ni skapar systemdokumentation? Måla gärna upp ett scenario där ni skapar och dokumenterar exempelvis ett krav.

B1: Vi bara skriver upp kravet mycket enkelt, ibland skickar vi med en skiss på ett diagram eller en rapport. Vi har inga speciellt kravprocesser, men vi ser det som en brist och hoppas bli bättre på det.

M: Hur skapar ni övrig systemdokumentation?

B1: Vi har egentligen allt i confluence/wikin som skall fungera som ett stöd för att hitta orsaken till eventuella buggar, och som stöd för nyanställda. Här ser man övergripande kodbeskrivningar, lite verksamhetsförklaringar etc. Gärna också lösningar på vanliga fel. Utöver det har vi ett ärendehanteringssystem (JIRA) där alla user stories står, och där saker beskrivs i mer detalj. Dessutom använder vi oss av en chatt där vi diskuterar vissa saker, som vi kan gå tillbaka och söka i.

M: Hur beskriver ni userstories rent praktiskt? Lät som i fysisk “lappform” och i digital form? Anser du att krav kan vara redundant (att dokumentera)? T.ex: Tar ni alltid fram userstories och beskriver dem i ärendehanteringssystemet, trots att det gäller en feature som du kan utveckla över en helg?

B1: Vi använder JIRA och vi skriver in våra userstories där. Vi tar också ut en fysisk lapp och har på den fysiska tavlan, fast vi kommer ev. att skrota den fysiska tavlan så småningom och gå helt på den digitala tavlan. Kraven använder vi mer medan vi utvecklar, när det är klart och godkänt av kund så tar systemdokumentationen över, så att säga. Krav är dock vår stora svaghet, de är inte väldokumenterade och de lämnar mycket utrymme för tolkning. Så det enklaste för oss är täta avstämningar och ett OK från kund när det är klart, mer än tydliga krav och ett fast leveransdatum.

O: Okej. Forskare talar om tekniska hjälpmedel såsom videofilmning och post-it lappar för att beskriva krav. Kan du ge några exempel på dokumentationstyper, hjälpmedel eller annat som ni använder för att beskriva ett krav eller system?

O: Vi kan börja med källkodskommentarer.

B1: Vi försöker skriva en kort beskrivning i koden, det är det enklaste att få igenom. Det är bara i text.

O: Och externa systembeskrivningar?

B1: Här brukar vi begära in från systemleverantören, ofta ett klassiskt pdf-dokument.

O: Tekniska hjälpmedel, t.ex. wikis eller ljudinspelningar?

B1: Wikin används mycket, chatt.

O: Modellering då?

B1: Mycket foton på skisser på whiteboards. Integrationsbeskrivningar och modelleringstänk och planer framåt är ofta på fotostadiet. Meningen är att det ska koka ner till en klassisk systemdokumentation, men det sker sällan.

M: Har ni några riktlinjer för hur systemdokumentation ska utformas och dokumenteras? Kan du placera var i projektlivscykeln ni skapar krav eller annan systemdokumentation?

B1: Det är tyvärr mycket upp till var och en, till projektledaren eller till utvecklaren eller till systemansvarig. Riktlinjerna är att all systemdokumentation skall finnas i confluence.

O: Vi undrar lite hur ni ser på exempelvis systemdokumentation, om det är något som skapas i samråd med kund som efterfrågar det, eller för att er organisation säger så (från policy, ledning, formella processer eller dylikt)? Du nämnde att ni har vissa riktlinjer att beskriva viss funktionalitet under delar av Confluence. Kan du utveckla det?

B1: dokumentationskrav har inte kunden alls, de vill bara ha funktionaliteten. Här är mer målet att vi skall kunna kundskapsöverföra enkelt. Organisationen som sådan lämnar det delvis upp till varje grupp, men riktlinjen är att systemdokumentationen skall finnas i confluence på nivån att "andra kan använda det för felsökning". Men vi vill ju gärna även kunna använda det för att sätta in nyanställda i våra system. Idag är vi inte där på långa vägar, vi har möjligen en kort beskrivning och kontaktpersoner.

53 **M: Har ni samma riktlinjer vad gäller krav och arkitektur också, eller kan ni anpassa er till projektets omfattning och kundens önskemål?**

54 B1: Arkitekturen rår vi som grupp själva över utan att någon blandar sig i. Riktlinjen är mest att det skall vara enkelt att förvalta. Våra system är ju ett hopplock från fler bolag från början, så vi håller fortfarande på med ett likriktningsarbete, och vi kommer i år att skapa ett Best practicedokument.

55

56 **M: En av det agila manifestets fyra kärnvärderingar är att prioritera kod över omfattande systemdokumentation. Flera forskare hävdar även att systemdokumentation ska vara kortfattad och efterfrågad eller att inte göra mer än vad som krävs.**

57 B1: Okej.

58

59 **M: Hur värdesätter ni systemdokumentation? T.ex. är det krav från kund "affärskrav" eller tekniska krav?**

60 B1:Affärskrav är viktiga i vårt område, eftersom vi måste ha en stor affärsinsikt för att förstå datat. Så oftast värdesätter jag verksamhetsbeskrivningar mer en tekniska krav, men det är ju lite annorlunda i traditionell systemutveckling.

61

62 **M: Verbal kommunikation vs. skriftlig kommunikation hur gör ni där?**

63 B1: Verbal kommunikation är det viktigaste för att man då fångar upp nyanser och kan ställa följdfrågor och What If-frågor. Däremot måste det ju därefter landa skriftligt – men skulle jag få välja den ena väljer jag det verbala.

64

65 **O: Hur prioriterar ni systemdokumentation?**

66 B1: Alldeles för lågt. Är det mycket att göra prioriteras den tyvärr bort. Ofta får man försöka komma ikapp på till exempel sommaren när det är lite lugnare.

67

68 **O: En av det agila manifestets tolv principer hävdar att krav, design och arkitektur bäst växer fram ur självorganiserande team. Har ni några givna roller eller personer i era projekt? Beskriv hur ni organiserar era team.**

69 B1: Vi har givna roller, fast många sitter på flera stolar. Jag äger backloggen, jag utvecklar, och jag ansvarar för scrummötena och sprintplaneringen, inkl grooming och reflektion. Vi har en indelning dels i teknisk kompetens, dels i verksamhetskompetens, men vi skulle vilja bli mer lättrörliga över gränserna. Tyvärr är det just nu ganska låst.

70

71 **O: Hur ansvarar ni för dokumentering av respektive begrepp?**

72 B1: Det är jag som ansvarar för att allt finns, men jag kan ju delegera det. Egentligen ingår syssystemdokumentation i definition of done, men det görs ofta avkall på. Kravdokument skall finnas innan jag lyfter in det i en sprint – men även det slarvas det med. Övrig dok finns inte hos oss just nu.

73

74 **O: Om vi tar dedikerade roller vs. situationsanpassat ansvar då?**

75 B1: Vi vill gå mer åt situationsanpassat, fast vi är långt ifrån där. Folk hittar ofta en roll där de trivs och vill inte riktigt gå utanför sin comfort zone.

76

77 **O: Om vi delar upp det i eget ansvar vs. delat ansvar**

78 B1: Vi har mycket eget ansvar, alla ansvarar för att driva sin del vidare, och det är ett måste i vår organisation. Jag har dock ansvaret gentemot kund.

79

80 **M: Okej, men vem brukar vanligtvis ansvara för dokumentering av exempelvis krav, design, arkitektur eller systemspecifikationer?**

81 B1: I nyutvecklingsprojekt är det projektledaren som bestämmer kravansvarig. Finns det inte en speciell kravansvarig landar det på respektive områdesansvarig och PL. Systemspecen är det utvecklarna själva som ansvarar för att uppdatera, fast områdesansvarig skall se till att det finns någon till att börja med.

82

83 **M: Okej, då jag tror vi har allt. Tack för att du kunde delta!**

84 B1: Tack själva.

Referenser

Ambler, S., 2002. *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process*. 1st Edition. Wiley.

Ambler, S., 2012a. *Agile Architecture: Strategies for Scaling Agile Development*. Tillgänglig på: http://www.agilemodeling.com/essays/agileArchitecture.htm. Hämtad den 9/4-14 12:58.

Ambler, S., 2012b. *Agile/Lean Documentation: Strategies for Agile Software Development*. Tillgänglig på: http://www.agilemodeling.com/essays/agileDocumentation.htm. Hämtad den 7/4-14, 15:16.

Ambler, S., 2014a. *Agile Modeling: Where Do I Start?*. Tillgänglig på: http://www.agilemodeling.com/essays/whereDoIStart.htm#RequirementsAnalyst. Hämtad den 8/4-14, 17:26.

Ambler, S., 2014b. *Best Practices for Agile/Lean Documentation*. Tillgänglig på: http://www.agilemodeling.com/essays/agileDocumentationBestPractices.htm. Hämtad den 8/4-14, 16:16.

Andersson, B-E, 1985. *Som man frågar får man svar - en introduktion i intervju- och enkätteknik*. Stockholm: Raben & Sjögren.

Ascezen Consulting, 2014a. *Documentation Development in Agile Development Models*. Tillgänglig på: http://www.ascezen.com/2013/11/documentation-development-agile-development-models/. Hämtad den 6/4-14, 14:23.

Ascezen Consulting, 2014b. *Mapping Documentation Development Life Cycle (DDLC) with SDLC*. Tillgänglig på: http://www.ascezen.com/2013/11/mapping-documentation-development-life-cycle-ddlc-sdlc/. Hämtad den 6/4-14, 14:05.

Avison, D., Fitzgerald, G., 1995. *Information Systems Development: Methodologies, Techniques and Tools.* 2nd edition. Maidenhead: McGraw-Hill.

Backman, J., 1998. *Rapporter och uppsatser*. Lund: Studentlitteratur.

Barker, T., 2002. *Writing Software Documentation: A Task-Oriented Approach (Part of the Allyn & Bacon Series in Technical Communication) (2nd Edition)*. 2nd Edition. Longman.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D., (2001): The agile manifesto. Tillgänglig på: http://agilemanifesto.org/ Hämtad 10/4-14 18:14.

Cho, J., *2008*. *Issues and Challenges of Agile Software Development with Scrum*. IACIS. Issues in Information Systems. Stillwater, OK., USA.

Cockburn, A., 2001. *Agile Software Development*. Edition. Addison-Wesley Professional.

Cockburn, A., Highsmith, J., 2001. *Agile software development: the business of innovation.* Tillgänglig på: http://alistair.cockburn.us/Agile+software+development%3A+the+business+of+innovation. Hämtad 29/4-14 17:46.

Denscombe, M., 2000. *Forskningshandboken – för småskaliga forskningsprojekt inom samhällsvetenskaperna*, Studentlitteratur, Lund.

DocForge, 2013. *Code documentation - DocForge*. Tillgänglig på: http://docforge.com/wiki/Code_documentation. Hämtad 3/4-14 13:32.

Gugenberger, P., 2007. The waterfall accident. Tillgänglig på: http://pascal.gugenberger.net/thoughts/waterfall-accident.html Hämtad 4/4-14 20:13.

Gustavsson, T,. 2007. Agile: konsten att slutföra projekt. Karlstad: TUK förlag.

Highsmith, J., 2002. *Agile Software Development Ecosystems*. 1st Edition. Addison-Wesley Professional.

Jacobsen, I., 2002. *Vad, hur och varför? Om metodval i företagsekonomi och andra samhällsvetenskapliga ämnen,* Studentlitteratur, Lund.

Lacey, M., 2012. *The Myth of No Documentation in Scrum Projects.* Tillgänglig på: http://www.informit.com/articles/article.aspx?p=1851231&seqNum=2. Hämtad 29/3-14 18:46.

Magnusson, A., 2004. *Användningen och nyttan av systemdokumentation.* Handelshögskolan vid Göteborgs universitet.

MaRSDD, 2009. *Product development: Waterfall Model | Software development methodology | Entrepreneur's Toolkit*. Tillgänglig på : http://www.marsdd.com/articles/product-development-waterfall-methodology/. Hämtad den 3/4-14, 11:50.

McConnell, S., 1996. *Rapid Development: Taming Wild Software Schedules*. 1 Edition. Microsoft Press.

Petersen, K., Wohlin, C., Baca, D., 2009. *The Waterfall Model in Large-Scale Development - Springer*. Tillgänglig på: http://link.springer.com/chapter/10.1007%2F978-3-642-02152-7_29#page-5. Hämtad den 3/4-14, 11:55.

Prause, C.R.; Durdik, Z., "Architectural design and documentation: Waste in agile development?," *Software and System Process (ICSSP), 2012 International Conference on* , vol., no., pp.130,134, 2-3 June 2012.

Pressman, R., 2009. *Software Engineering: A Practitioner's Approach*. 7 Edition. McGraw-Hill Science/Engineering/Math.

Rakitin, Steven R., 2001. Manifesto Elicits Cynicism: Reader's letter to the editor by Steven R. Rakitin. *IEEE Computer* 34: 4

Rüping, A., 2003. *Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects*. 1 Edition. Wiley.

Sommerville, I., 1992. Software Engineering. 4th edition. Harlow: Addison- Wesley.

Sommerville, I., 2010. *Software Engineering (9th Edition)*. 9th Edition. Addison-Wesley

Sutherland, J., Schwaber, K., 2013. Scrumguiden, Den definitiva guiden till Scrum: Spelets regler. Tillgänglig på: http://www.scrumguides.org/static/51e3f87ce4b0031a73dac256/t/52d98265e4b0a29115b954b9/1389986405877/Scrum_Guide.pdf#zoom=100. Hämtad den 4/4-14, 15:28.

Uikey, N., Suman, U., & Ramani, A. K., 2011. A Documented Approach in Agile Software Development. *International Journal of Software Engineering (IJSE)*, *2*(2), 13-22.

VersionOne, 2013. *8th Annual State of Agile Survey*. Tillgänglig på: www.versionone.com/pdf/2013-state-of-agile-survey.pdf. Hämtad 23/5-14 11:23.

Wyssocky, L., 2003. *Source Code Documentation*. Tillgänglig på: http://www.qualityprogramming.org/implementation/SourceCodeDocumentation/So urceCodeDocumentation.html. Hämtad 5/4-14 15:43.