LUNDS UNIVERSITET
Ekonomihögskolan

Institutionen för informatik

Dokumentation inom agila mjukvaruutvecklingsprojekt

Kandidatuppsats: 15 högskolepoäng, SYSK02 i informatik

Författare: Nathalie Monroy
Emma Pehrsson
Elin Strid

Handledare: Mirella Muhic

Examinatorer: Umberto Fiaccadori
Benjamin Weaver

Slutseminarium: Augusti 2015

Dokumentation inom agila utvecklingsmetoder

Författare Nathalie Monroy
Emma Pehrsson
Elin Strid

Utgivare Institutionen för informatik

Handledare Mirella Muhic

Examinatorer Umberto Fiaccadori
Benjamin Weaver

Slutseminarium 21 augusti 2015

Uppsatstyp Kandidatuppsats

Nyckelord Dokumentation, utvecklingsmetoder, mjukvara, agilt, vattenfall

Sammanfattning

Idag är de traditionella och agila utvecklingsmetoderna de mest använda metoderna inom mjukvaruutveckling. I grunderna för de olika metoderna läggs det olika stor vikt på dokumentation. Inom traditionella metoder uppmuntras omfattande dokumentation, något som skiljer sig från de agila metoderna där fungerande mjukvara prioriteras före omfattande dokumentation. I denna studie kommer vi främst fokusera på dokumentation inom agila mjukvaruutvecklingsprojekt men vi kommer även att gå in på synen på dokumentation inom traditionella utvecklingsmetoder. Detta för att skapa en övergripande bild av hur dokumentation behandlas samt anses ha för nytta inom de olika metoderna. Vi har undersökt hur den dokumentation som skapas inom agila projekt används och på så sätt identifierat huruvida den har negativ eller positiv inverkan på projekt som utförs enligt någon av de agila metoderna. Genom vår litteraturstudie identifierade vi sex faktorer inom agila projekt som kan påverkas av dokumentation. Dessa faktorer jämfördes med vårt empiriska resultat från sju olika företag som arbetar agilt för att kartlägga dokumentationens faktiska inverkan på dessa faktorer samt hur agila utövare ställer sig till dokumentation. Resultatet av studien visar att användare av agila utvecklingsmetoder ställer sig positiva till dokumentation. Samt att dokumentation stödjer de faktorer inom agil mjukvaruutveckling som vi funnit genom litteraturstudien.

Innehållsförteckning

1 Inledning 5
1.1 Bakgrund 5
1.2 Problemdiskussion 6
1.3 Forskningsfråga 7
1.4 Syfte 7
1.5 Avgränsningar 7
2 Litteraturgenomgång 8
2.1 Mjukvaruutvecklingens livscykel 8
2.2 Traditionella utvecklingsmetoder 9
2.3 Vattenfallsmetoden 10
2.3.1 För- respektive nackdelar med vattenfallsmetoden 11
2.4 Agila utvecklingsmetoder 12
2.4.1 Scrum 13
2.4.2 Extreme programming 15
2.4.3 Lean 16
2.4.4 För- respektive nackdelar med agila metoder 16
2.5 Dokumentation inom mjukvaruutveckling 17
2.5.1 Dokumentation inom traditionella utvecklingsmetoder 19
2.5.2 Dokumentation inom agila metoder 21
2.6 Presentation av faktorer 23
2.6.1 Flexibilitet 23
2.6.2 Produktivitet 24
2.6.3 Kunskapsspridning 24
2.6.4 Förvaltning 24
2.6.5 Dokumentationskvalitet 25
2.6.6 Kommunikation 25
3 Metod 26
3.1 Litteraturstudie 26
3.2 Metod för insamling av empirisk data 26
3.2.1 Forskningsansats 26
3.2.2 Urval 27
3.2.3 Intervjuguide 27
3.2.4 Genomförande 28
3.2.5 Tolkning av empiri 29

3.2.6 Etik 29
3.2.7 Reliabilitet och validitet 30
4 Informanternas syn på dokumentation 31
4.1 Informanter 31
4.2 Sammanfattning av informanternas svar 32
4.2.1 Flexibilitet 32
4.2.2 Produktivitet 33
4.2.3 Kunskapsspridning 34
4.2.4 Förvaltning 35
4.2.5 Dokumentationskvalitet 37
4.2.6 Kommunikation 39
5 Resultat 40
5.1 Analys 40
5.1.1 Flexibilitet 41
5.1.2 Produktivitet 42
5.1.3 Kunskapsspridning 43
5.1.4 Förvaltning 44
5.1.5 Dokumentationskvalitet 46
5.1.6 Kommunikation 47
5.2 Diskussion 49
6 Slutsats 52
6.1 Teoretiskt bidrag 52
6.2 Empiriskt bidrag 52
6.3 Förslag på vidare forskning 53
7 Bilagor 54
7.1 Intervjuguide 54
7.2 Transkribering informant 1 55
7.3 Transkribering informant 2 59
7.4 Transkribering informant 3 61
7.5 Transkribering informant 4 62
7.6 Transkribering informant 5 63
7.7 Transkribering informant 6 66
7.8 Transkribering informant 7 70
8 Referenslista 74

Tabellförteckning

Tabell 2.1 Faktorer i agila projekt som påverkas av dokumentation 23
Tabell 4.1 Presentation av informanter 31
Tabell 4.2 Sammanfattning av informanternas svar gällande dokumentationens påverkan på flexibilitet 32
Tabell 4.3 Sammanfattning av informanterna svar gällande dokumentationens påverkan på produktivitet 33
Tabell 4.4 Sammanfattning av informanternas svar gällande dokumentationens påverkan på kunskapsspridning 34
Tabell 4.5 Sammanfattning av informanternas svar gällande dokumentationens påverkan på förvaltningsarbete 35
Tabell 4.6 Sammanfattning av informanternas svar gällande dokumentationskvalitet 37
Tabell 4.7 Sammanfattning av informanternas svar gällande dokumentationens påverkan på kommunikation 39
Tabell 5.1 Analys av informanternas svar gällande flexibilitet 41
Tabell 5.2 Analys av informanternas svar gällande produktivitet 42
Tabell 5.3 Analys av informanternas svar gällande kunskapsspridning 43
Tabell 5.4 Analys av informanterna svar gällande förvaltning 44
Tabell 5.5 Analys av informanternas svar gällande dokumentationskvalitet 46
Tabell 5.6 Analys av informanternas svar gällande kommunikation 47

Figurförteckning

Figur 2.1 Mjukvaruutvecklingens livscykel 8
Figur 2.2 Vattenfallsmetoden 11
Figur 2.3 Scrumprocessen 15
Figur 2.4 Dokumentation vattenfallsmetoden 20
Figur 6.1 Dokumentationsnyttan inom agila projekt 53

1 Inledning

1.1 Bakgrund

Mjukvaruutvecklingens livscykel började användas inom mjukvaruutveckling på 1950-talet. Tidigare hade den använts för problemlösning inom verksamhetsforskning. Behovet av att förbättra strukturen av utvecklingsprojekt ledde till att livscykeln utvecklades i syfte att användas vid mjukvaruutveckling (Fitzgerald, Russo & Stolterman, 2002). Idag är de traditionella och agila utvecklingsmetoderna de mest använda metoderna inom mjukvaruutveckling (Leau, Loo, Tham & Tan, 2012).

De traditionella utvecklingsmetoderna, även kallade tungviktiga, grundar sig på en sekventiell serie av steg där kraven bestäms i början av ett projekt. Stegen kan vara kravspecifikationer, lösningsbyggnad, test och driftsättning. Den dokumentation som skapas under ett projekt anses vara betydande i traditionella utvecklingsprojekt. Det traditionella arbetssättet är processorienterat, det vill säga processen är väl planerad (Awad, 2005). Vattenfallsmetoden ingår i de traditionella metoderna och fokuserar på leverans långt fram i projektet. Agila metoder uppstod under tidigt 2000-tal som svar på den kritik som vattenfallsmetoden fått (Greer & Hamon, 2011).

Grunden till att de agila metoderna utvecklades berodde på frekventa kravförändringar samt behovet av att snabbt utveckla mjukvara (Greer & Hamon, 2011). Det agila arbetssättet skiljer sig från traditionella sätt att arbeta med mjukvaruutveckling (Gustavsson, 2013). Inom agila metoder förespråkas en enkel utvecklingsprocess. Syftet är att löpande producera fungerande mjukvara till kund i så kallade iterationer. Dessa löpande leveranser som agilt projektarbete bygger på ger kunden möjlighet att modifiera kraven på produkten vid de olika delleveranserna (Matook & Vidgen, 2014). År 2001 tog 17 systemutvecklare fram grunden för de agila metoderna. De ansåg att det behövdes ett nytt sätt för att ta fram mjukvara och skapade därför det agila manifestet. I manifestet förklarades de punkter som de ansåg var centrala för en god mjukvaruutveckling. Dessa är:

Individer och interaktioner **före** processer och verktyg
Fungerande mjukvara **före** omfattande dokumentation
Kundsamarbete **före** kontraktsförhandling
Förändringsbenägenhet **före** att följa en plan
(Beck et al, 2001)

Till vänster står det som prioriteras högst inom agila metoder. Dokumentation är inte en huvudprioritet inom agil utveckling men det innebär inte att dokumentation är oviktig eller att dokumentation inte behövs (Rüping, 2003). Dokumentation ska innehålla information om hur systemet fungerar, hur systemet är strukturerat samt varför den aktuella strukturen och designen har valts (Stettina & Heijstek, 2011). Rüping (2003) menar att ett "light but sufficient" synsätt på dokumentation är att föredra inom agila utvecklingsmetoder.

1.2 Problemdiskussion

Vattenfallsmetoden är en av de äldsta metoderna inom mjukvaruutveckling (Munassar & Govardhan, 2010). Vattenfallsmetoden har vanligtvis väldefinierade gränser och ansvarsområden för de som arbetar inom samma projekt (Zhang, Hu, Day & Li, 2010) vilket gör den till en lämplig metod för svaga projektteam. Vid arbete enligt vattenfallsmetoden ingår omfattande dokumentation av hela utvecklingsprocessen (Munassar & Govardhan, 2010). Leau (2012) menar att dokumentationen ska fungera som kommunikationsmedel i projekt samt för kunskapsspridning bland projektmedlemmar. Vattenfallsmetoden har dock kritiserats för att kräva allt för omfattande dokumentation (Munassar & Govardhan, 2010). Dokumentation måste vanligtvis verifieras mellan varje steg vilket leder till att arbetet med utvecklingen vanligtvis inte kan fortskrida förrän dokumentationen från förgående fas är färdig (Zhang et al, 2010).

Inom vattenfallsmetoden ligger arbetets fokus på leverans långt fram i projektet (Gustavsson, 2013). Vid användning av traditionella metoder, som till exempel vattenfallsmetoden är sannolikheten stor att kravändringar sker innan slutprodukten har levererats. Vattenfallsmetoden har kritiserats för att vara oflexibel då det inte är lätt att anpassa produkten till ändrade affärskrav som kan komma under arbetets gång (Chasalow, 2008).

Enligt Dybå & Dingsøyr (2008) är agila metoder idag de utvecklingsmetoder som dominerar utvecklingen av informationssystem. Branschundersökningar som gjorts pekar på att mer än 88 % av företag arbetar agilt. De löpande leveranserna av system ger kunden möjlighet att modifiera sina produktkrav allt eftersom de olika delleveranserna presenteras (Matook & Vidgen, 2014). Iterationerna leder till att kunden i slutet av projektet får en produkt som är i linje med vad som efterfrågats, det vill säga i linje med de krav som har ställts på produkten (Gustavsson, 2013). I agila projekt förespråkas små självorganiserade team, kontinuerlig designförbättring, testdriven utveckling och löpande kundintegration. Kunskapshantering inom agila projekt tenderar att vara implicit och face-to-face kommunikation prioriteras framför omfattande dokumentation (Greer & Hamon, 2011).

Dokumentation inom agila utvecklingsmetoder har fått stor uppmärksamhet i diskussioner gällande agila metoder. Förespråkare för agila metoder diskuterar huruvida dokumentation överhuvudtaget är nödvändigt. Andra menar att det är viktigt för att bibehålla viktig information i företaget och att många organisationer kräver mer dokumentation än vad som anses nödvändigt (Basili, Boehm, Costa, Dangle, Lindvall, Shull, Tesoriero, Williams & Zelkowitz, 2002).

Inom litteraturen finns det motsättningar gällande hur omfattande dokumentation inom agila projekt bör vara. Vi vill därför identifiera vilka faktorer dokumentation kan påverka inom agila utvecklingsprojekt samt hur företag som arbetar agilt ställer sig till dokumentation.

1.3 Forskningsfråga

Hur ställer sig användare av agil mjukvaruutveckling till dokumentation: och vilka faktorer påverkas av dokumentation?

1.4 Syfte

Syftet med denna studie är att undersöka vad dokumentation har för påverkan på olika faktorer inom agila mjukvaruutvecklingsprojekt. Vi vill undersöka hur den dokumentation som skapas används och på så sätt ta reda på om dokumentation har positiv eller negativ inverkan på arbetet med agil mjukvaruutveckling.

1.5 Avgränsningar

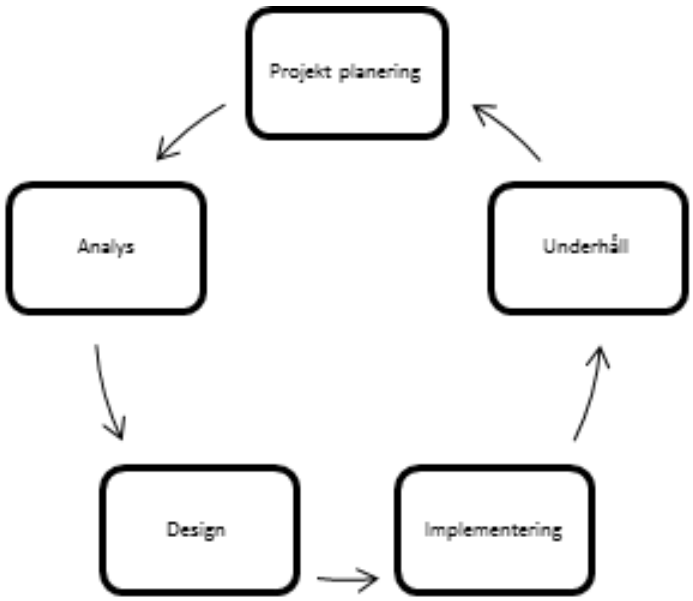
Studien kommer endast att behandla dokumentation inom agila projekt hos företag som tillhandahåller mjukvarusystem till externa kunder. För att få en bild av hur dokumentationen sker i utvecklingsprojekt har vi valt att även presentera vattenfallsmetoden, som en representant för ett traditionellt synsätt på dokumentation. Den dokumentation vi fokuserar på är dokumentation som genomförs under hela mjukvaruutvecklingens livscykel. Dessa är krav-, design- och implementationsdokument samt test- och koddokument.

2 Litteraturgenomgång

I detta kapitel redovisar vi centrala begrepp för uppsatsen. Avsnittet inleds med en redogörelse för mjukvaruutvecklingens livscykel. Därefter presenteras de två vanligaste metoderna, traditionella samt agila metoder. Vidare ges en övergripande förklaring till vad dokumentation inom mjukvaruutveckling är. Samt hur dokumentation behandlas vid arbete enligt traditionella respektive agila metoder.

2.1 Mjukvaruutvecklingens livscykel

Mjukvaruutvecklingens livscykel (se figur 2.1) är en process som täcker alla stadier i ett systems utveckling, från start med kravdefinitioner fram till implementation samt underhåll (Ruparelia, 2010., Leau, et al, 2012).



Figur 2.1 Mjukvaruutvecklingens livscykel (Zhang, Carey, Te'eni & Tremaine, 2004, modifierad)

Användandet av livscykeln infördes inom mjukvaruutveckling i början av 1950-talet, innan dess hade den används inom verksamhetsforskning där dess principer användes för problemlösning. Anledningen till att livscykeln utvecklades för att användas till utveckling av mjukvara var att det fanns ett behov av att strukturera samt förbättra hanteringen av utvecklingsprojekt (Fitzgerald et al, 2002).

Det finns ett flertal modeller som beskriver olika sätt att förhålla sig till SDLC-processen. Dessa modeller används för att beskriva vilka steg i SDLC som följer varandra, det vill säga vad som ska göras (Ruparelia, 2010). De modeller som används i dagsläget bygger på principer från de tidigare modellerna (Leau et al, 2012) och de modeller som finns brukar delas in i tre huvudgrupper:

- Linjära modeller
- Iterativa modeller
- Kombinationsmodeller

I linjära modeller följer alla steg en bestämd ordningsföljd, där ett steg måste avslutas innan nästa kan påbörjas. I de iterativa modellerna är det å andra sidan tänkt att under utvecklingsarbetet gå tillbaka till tidigare genomförda steg och vidareutveckla och förbättra det som gjorts tills en produkt är färdig. Kombinationsmodeller kombinerar det iterativa och linjära tillvägagångssättet (Ruparelia, 2010).

Traditionella metoder (metoder beskriver hur något ska göras (Ruparelia, 2010)) används för att genomföra arbete som är strukturerade linjärt, medan agila metoder används vid utvecklingsarbete som är iterativa. Traditionella och agila utvecklingsmetoder är de som i dagsläget är mest använda inom mjukvaruutveckling (Leau et al, 2012). Nedan beskriver vi den vanligaste traditionella metoden samt de vanligaste agila metoderna som används vid utveckling av system och hur dessa skiljer sig gällande dokumentation.

2.2 Traditionella utvecklingsmetoder

Traditionella utvecklingsmetoder, som även kallas tungviktiga metoder, grundar sig på en sekventiell serie av steg. Exempel på dessa är kravspecifikationer, lösningsbyggnad, test och driftsättning. Traditionella metoder är processorienterade och målet är att definiera en process som kommer att fungera för vilken användare som helst. En stor del av processen planeras i detalj. Projektutvecklingen blir på så sätt förutsägbar (Awad, 2005).

2.3 Vattenfallsmetoden

Vattenfallsmetoden är en av flera metoder inom traditionell utveckling (Awad, 2005) och är en av de äldsta metoderna inom mjukvaruutveckling (Munassar & Govardhan, 2010). Vattenfallsmetoden består av ett antal faser som följer varandra i en bestämd ordning (Avison & Fitzgerald, 2003). Varje fas består av en uppsättning aktiviteter som måste slutföras innan nästa fas kan påbörjas (Awad, 2005). Det sker således ingen överlappning mellan de olika faserna i metoden (Munassar & Govardhan, 2010).

Vattenfallsmetoden har kritiserats för att vara alltför rigid som inte kan anpassa sig till snabbt ändrade affärskrav. Vid användning av denna metod är sannolikheten stor att kravändringar sker innan leverans av slutprodukt. Detta beror på de långa ledtider som råder mellan kravinsamling, design och kod. Trots detta används den fortfarande i verksamheter (Chasalow, 2008).

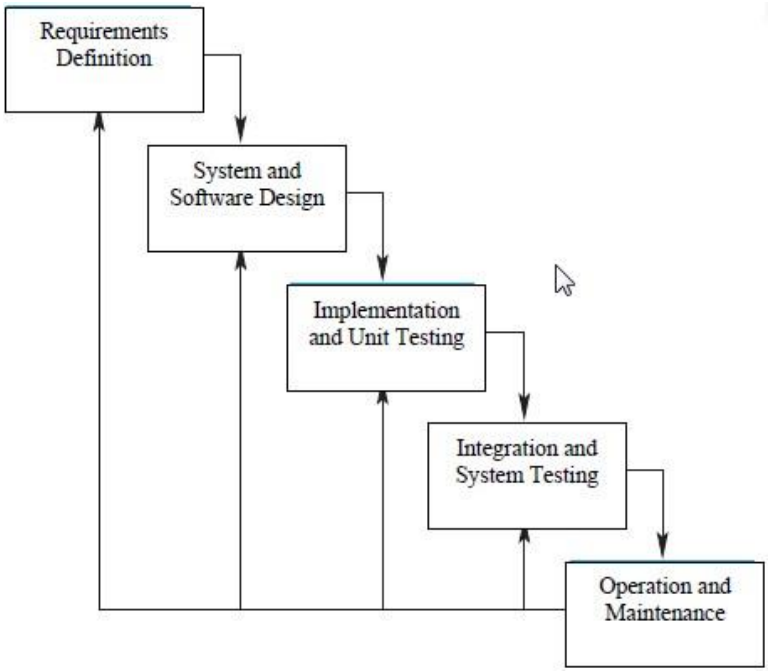
I grunden omfattar vattenfallsmetoden fem faser. Dessa är analys, design, implementation, testning och underhåll. Den första fasen, analysfasen, mer känd som kravspecifikation, består av omfattande beskrivningar av funktioner samt beteende hos systemet som ska utvecklas. Kraven som tas fram är både funktionella och icke-funktionella krav. De funktionella kraven är utformade som användarfall och beskriver de tänkta användarnas interaktion med systemet. Dessa omfattar krav som till exempel, syfte, omfattning, gränssnitt och interaktion. Icke-funktionella krav består av design- och driftkrav. Dessa omfattar krav på mjukvarans tillförlitlighet, skalbarhet, testbarhet, tillgänglighet, underhåll, prestanda och kvalitet (Bassil, 2012).

Efter att arbetet med analysfasen är klart, går utvecklingsarbetet vidare till designfasen. I designfasen bestäms det hur systemets design kommer att se ut (Awad, 2005). Det innebär att utvecklare tillsammans med designers definierar en lösning som omfattar bland annat mjukvaruarkitektur och konceptuella databasmodeller (Bassil, 2012).

När arbetet med att bestämma systemdesign avslutas påbörjas arbetet med att bygga systemet. Detta sker i implementationsfasen. Arbetet som genomför här är förverkligandet av de krav och specifikationer som tagits fram i de föregående faserna. Arbetet i denna fas resulterar i fungerande program såsom databaser, webbsidor eller olika komponenter som har programmerats (Bassil, 2012).

Efter implementationsfasen följer testfasen. I testfasen ingår verifiering samt validering av det som utvecklats i implementationsfasen. Syftet är att säkerställa att systemet möter de krav samt specifikationer som togs fram i de två första faserna. Utöver detta genomförs i denna fas felsökningar för att hitta och åtgärda eventuella buggar som finns i systemet (Bassil, 2012).

Underhållsfasen är den sista fasen och pågår från och med att systemet har levererats och satts i drift. Arbetet i denna fas utgörs vanligtvis av att förbättra data som systemet tillhandahåller. Det kan till exempel vara felkorrigeringar eller prestanda- och kvalitetsförbättringar. Arbetet i denna fas kan även omfatta anpassning av systemet till den miljö den ska användas i, arbete med nya användarkrav samt arbete med att öka systemets tillförlitlighet (Bassil, 2012).



Figur 2.2 Vattenfallsmetoden (Munassar & Govardhan, 2010, s.95)

Vattenfallsmetoden som illustreras i figur 2.2 betonar viktiga steg i projektutvecklingen (Munassar & Govardhan, 2010). Figuren visar även på de iterativa relationerna mellan de olika faserna. Det sker iterationer mellan en viss fas förgående och efterföljande fas, men sällan mellan mer avlägsna faser (Royce, 1970). Framgången för ett projekt som utgår från ovanstående modell bygger på att känna till alla krav innan utvecklingen börjar.

2.3.1 För- respektive nackdelar med vattenfallsmetoden

Vattenfallsmetoden största svaghet är att den är oflexibel (Zhang et al, 2010). Metoden har svårt att hantera förändringar under projektets gång (Leau et al, 2012). Som tidigare nämnt så sker iterationer vanligtvis endast mellan de faser som kommer före eller efter en viss fas. Detta är en bidragande orsak till bristen på flexibilitet. En annan faktor som bidrar till att flexibiliteten minskar är att utvecklingen och testningen av systemet kommer sent i projektet. Detta leder till att när fel identifieras måste tidigare faser revideras vilket ofta leder till att projekt överskrider tid- och budgetramarna. Eftersom iterationer till någon av de tidigare faserna i projektet leder till att tid och budget inte hålls är det ofta svårt att reagera effektivt på förändrade kundkrav (Zhang et al, 2010, Munassar & Govardhan, 2010). Vattenfallsmetoden

har även fått kritik för att kräva allt för omfattande dokumentation (Munassar & Govardhan, 2010).

Fördelarna med vattenfallsmetoden är att den är lätt att förstå och implementera (Munassar & Govardhan, 2010). Den gör det lättare att fastställa kostnaderna för projektet, sätta upp en projektplan samt fördela resurser över projektets olika faser (Leau et al, 2012). Den understryker också viktiga projektstadier som måste beaktas i alla utvecklingsprojekt oberoende av vald metod. Den uppmuntrar till att definiera ett system innan designförslag utformas, och att kod inte skrivs innan designbeslut är fastställda. En annan fördel är att leverabler och milstolpar definieras tidigt i projektet. Vattenfallsmetoden anses fungera bra för arbete med mogna produkter där projektteamet är svagt (Munassar & Govardhan, 2010).

2.4 Agila utvecklingsmetoder

De agila utvecklingsmetoderna skiljer sig från traditionella utvecklingsmetoder. Något som skiljer de agila metoderna från de traditionella är acceptansen av förändring. Inom traditionella projektmetoder som till exempel vattenfallsmetoden ligger arbetets fokus på leverans långt fram i projektet. Kravdokument, som anses vara en viktig del inom traditionell utveckling, fastställs tidigt i projektet. Traditionella metoder anpassar sig inte till förändringar i lika stor utsträckning som agila metoder vilket kan vara avgörande för huruvida ett projekt uppnår sina mål eller inte (Awad, 2005). Frekventa kravförändringar samt behovet av att snabbt utveckla mjukvara är grunden till att de agila metoderna utvecklades (Berger & Flouri, 2010). Några av de agila metoderna är Lean development, Scrum, Extreme programming, Dynamic system development och Crystal methods (Awad, 2005).

I början av 2000-talet formade den agila alliansen det agila manifestet. Med det agila manifestet ämnade grundarna göra det lättare för utvecklare att snabbt utveckla fungerande mjukvara samt göra det möjligt för utvecklingsteamet att effektivt reagera på förändrade systemkrav (Martin, 2003). Agila utvecklingsmetoder, även kallade lättviktiga utvecklingsmetoder, används av systemutvecklare för att bygga snabba lösningar och för att enklare kunna anpassa sig till kravändringar. De agila metoderna fokuserar på utveckling baserat på korta livscykler där kunderna är involverade i processen (A. I. Khan, Qurashi & U. A. Khan, 2011). Genom att lista värderingar och principer för att möjliggöra detta skapades grunderna för de agila utvecklingsmetoderna.

Grundprinciperna i agila utvecklingsmetoder är att människorna värderas före processer och verktyg (Beck et al, 2001). Den främsta drivkraften för ett lyckat agilt mjukvaruutvecklingsprojekt är de individer som bildar det agila teamet. Processer och verktyg kan inte föra ett projekt framåt om de rätta kompetenserna inte finns inom teamet (Berger & Flouri, 2010). Den kompetens som värderas högst inom agila metoder är förmågan att arbeta i grupp. En stark medarbetare är inte nödvändigtvis den som besitter mest teknisk kunskap utan

det är den som har goda kommunikationsegenskaper och som interagerar med sina kollegor. Det är mer sannolikt att ett agilt team som består av individer med goda kommunikationsförmågor lyckas än ett team med tekniska experter som inte kommunicerar alls (Martin, 2003).

Fungerande mjukvara förespråkas före omfattande dokumentation (Beck et al, 2001). Att ha för omfattande dokumentation anses inom agila metoder vara värre än att inte ha någon dokumentation alls. Agila förespråkare menar på att en alltför omfattande dokumentation inte bara är tidskrävande utan kan, om den inte underhålls i takt med att systemet utvecklas, vara missvisande (Martin, 2003).

Nära kontakt och frekvent samarbete med kund förespråkas inom agila metoder framför kontrakt där krav, schema och kostnad är fastställt från projektstart. Detta beror på att det som skrivits i kontraktet i de flesta fall blir förlegat innan projektet avslutats (Martin, 2003). Inom agila projekt sker arbetet istället inkrementellt och iterativt (Beck et al, 2001). Målet med att arbeta iterativt är att för varje iteration bygga och tillhandahålla ett inkrement av kod som är värdeskapande för kund (Leffingwell, 2011). Att arbeta inkrementellt innebär att bitar av ett system utvecklas i olika takt eller under olika tillfällen för att sedan integreras under utvecklingens gång. Efter att en del av ett system har byggts utvärderar teamet sin arbetsprocess för att förbättra denna samt designen av systemet (Cockburn, 2002).

Den sista värderingen som förespråkas inom agila metoder är förändringsbenägenhet. Förändringsbenägenhet ska inom agila metoder gå före uppföljning av en plan (Greer & Hamon, 2011). Det är förmågan att reagera på förändring som avgör framgången i ett mjukvaruprojekt (Martin, 2003). Problematiken i att skapa en utförlig projektplan för mjukvaruutvecklingsprojekt är att det finns för många variabler inom projekt som inte kan estimeras (Berger & Flouri, 2010). Ett exempel på en sådan variabel är företagsklimatet som under projektets gång kan komma att förändras vilket leder till att kraven på systemen också ändras. En ytterligare variabel som kan komma att leda till ändringar är att kunderna ändrar sina krav när de ser de första delarna av systemet. Även om kraven inte ändras är det svårt att utforma en plan för ett mjukvaruprojekt då det sällan går att estimera hur lång tid det kommer ta att uppfylla de krav som ställts (Martin, 2003).

2.4.1 Scrum

Den mest använda agila metoden är Scrum. Scrum är en strömlinjeformad process för att styra team. Konceptet är att mjukvaruutvecklingsprocessen ska ske enkelt (Stober & Hansmann, 2010). Scrum baseras på hur medlemmarna i ett utvecklingsprojekt arbetar för att skapa flexibilitet i en ständigt förändrande miljö. Metoden koncentrerar sig därför inte på en specifik utvecklingsmetod utan snarare på olika strategier och verktyg för ledning i olika faser av projektet (Awad, 2005).

Scrum utvecklades för att hantera utvecklingsprocesser som sker i en instabil och föränderlig miljö. Scrum fokuserar huvudsakligen på flexibilitet, anpassningsförmåga och produktivitet. Detta gör att utvecklare har möjligheten att välja de verktyg och metoder som passar bäst för det som ska utvecklas. Genom att använda Scrum kan eventuella brister identifieras under utvecklingsprocessen och i de metoder som används (Dingsøyr, Nerur, Balijepally & Moe, 2012). Nedan följer de viktigaste delarna (se figur 2.3) av Scrum:

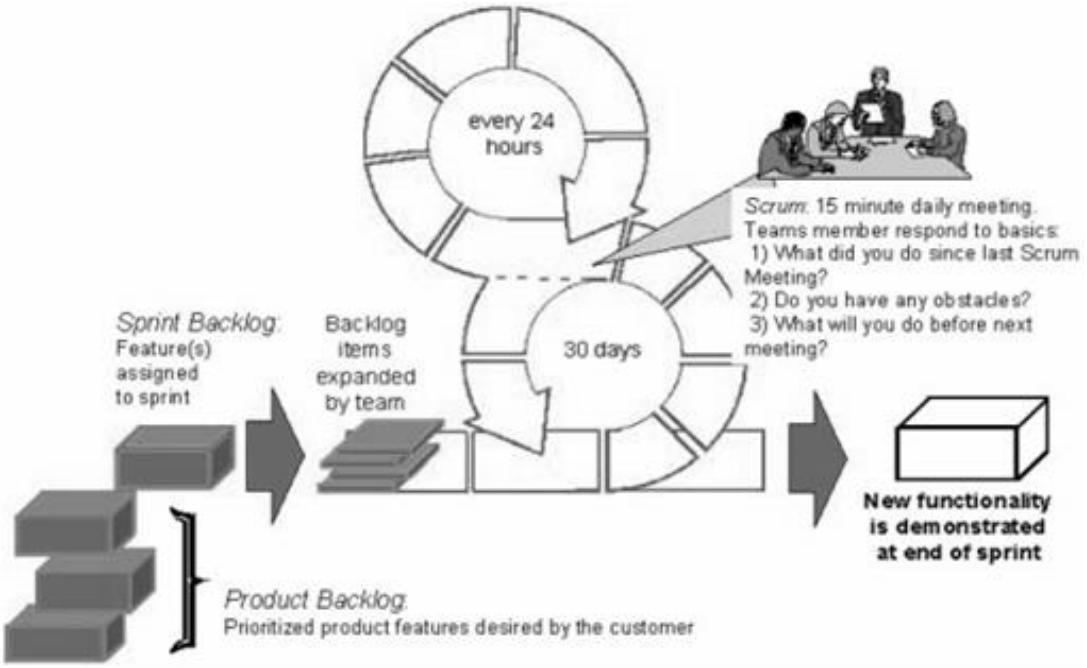
Product backlog: En lista på de funktioner och egenskaper som ska utvecklas i systemet. Funktionerna får en prioriteringsordning för att utvecklarna ska veta vad de ska börja med. Produktägaren sköter underhållet av listan som utgörs av krav från olika aktörer inom projektet (Awad, 2005).

Sprint planning meeting: Möte med samtliga i teamet där mål sätts upp för kommande sprint. Mötena sker innan varje ny sprint och i mötet skapas en sprint backlog (Drudy-Drogan, 2013).

Sprints eller iterationer: En sprint är en del av ett projekt där en del av ett program ska utvecklas. Utvecklarna följer den product backlog som ska utföras under respektive iteration. Sprintarna har en bestämd längd beroende på projektet och funktionernas svårhetsgrad. När sprinten är klar ska den bestämda delen av systemet levereras. Ofta har teamet då ett möte där utvecklarna går igenom sprinten för att se vad som kan göras bättre till nästa delleverans (Ruler, 2014).

Sprint backlog: En lista på mål som ska genomföras i en viss sprint. När listan är klar ska en ny iteration genomföras (Awad, 2005). I Sprintens backlog ska innehålla alla funktioner och krav för produkten. En sprint ska börja med en genomgång av sprintens backlog med produktägaren. Samtliga funktioner prioriteras och teamet bestämmer hur många funktioner som de tror kommer att kunna levereras inom en sprint (Kayes, Sarker & Chakareski, 2014).

Daily Scrum: Dagliga möten som är cirka 10-15 minuter långa. På mötena redovisas framsteg och eventuella problem som uppstått med det senaste arbetet (Awad, 2005). Vanligtvis genomförs mötena varje dag vid samma tid och de medverkande står ofta upp. Under mötet får varje projektmedlem berätta vad som genomförts sen senaste mötet samt vad som ska genomföras till nästa. Medlemmarna ska även berätta om det finns något som gör att de kan komma att få problem med det som ska genomföras under dagen (Drudy-Drogan, 2013).



Figur 2.3 Scrumprocessen (Awad, 2006, s.11)

I arbetet med Scrum ser rollerna annorlunda ut än i traditionella projekt. Teamet organiserar sig själv och tar beslut om vad som ska göras vilket vanligtvis sköts av ledningen. Projektledaren kallas Scrum master och är till för att lösa olika problem som uppstår under projektet (Awad, 2005). Ett problem som uppstod när ett utvecklingsprojekt arbetade agilt var enligt Drury et al (2012) att det var svårt för teamet att fatta ett beslut. När beslutet inte kunde tas så föll på Scrum mastern. Detta trots att arbetsmetoden bygger på att gruppen är självorganiserad. Detta ledde till att beslut blev uppskjutna och projektet försenat (Drury et al, 2012).

2.4.2 Extreme programming

Extreme programming (XP) är en agil metod som har utvecklats såsom övriga agila metoder som ett resultat av kritiken på de långa utvecklingscyklerna hos de traditionella utvecklingsmetoderna. Extreme programming utmärker sig för sina korta utvecklingscykler, inkrementella planering, frekventa återkoppling, kommunikation och design (Cusumano, 2007).

Livscykeln för ett projekt med Extreme programming delas upp i sex delar. Dessa är explorering, planering, iterationer, produktion, underhåll och död. I exploreringsfasen beskriver kunden vad de önskar ska finnas med i systemet. Planeringsfasen innebär att teamet gör en prioritering av vilka funktioner som ska göras i vilken ordning. Det första schemat över utvecklingen släpps i planeringsfasen. I iterationsfasen börjar utvecklarna att utveckla de

första delarna av systemet. Den ska ha samma arkitektur som resten av programmet. I produktionsfasen dokumenteras funktioner och idéer som kommit fram under projektets livscykel. Dessa idéer ska sedan implementeras i underhållsfasen. I dödsfasen har kunden inte längre några nya krav att lägga till i systemet, allt är dokumenterat (Awad, 2005).

2.4.3 Lean

Lean software development är grundprinciper som fokuserar på att skapa värde för kunden, minimera överflöd, optimera värdeskapandeflöden, ge makt till projektteamet och att kontinuerligt förbättra (Ebert, Abrahamsson & Oza, 2012). Lean software development är inte en metod i sig men erbjuder dessa principer som kan appliceras i alla miljöer för att förbättra mjukvaruutveckling (Awad, 2005). Lean software development kännetecknas av att den har mindre av allt i jämförelse med andra produktionssätt. Den betonar att den mänskliga insatsen i produktionen, tillverkningsutrymmet, produktionshantverken och produktionstimmarna ska halveras (Pakdil & Leonard, 2014).

2.4.4 För- respektive nackdelar med agila metoder

Ett grundläggande problem med de agila utvecklingsmetoderna är enligt Chasalow (2008) att automatiskt deklarera de agila utvecklingsmetoderna som bättre än de traditionella. Detta då olika utvecklingsteam har olika egenskaper. Således är de agila utvecklingsmetoderna inte alltid de mest lämpliga. Enligt Awad (2005) är projektledning ett av de agila metodernas stora problem. Ju större teamen blir och ju mer systemen växer, desto större är risken att kommunikationen som agila metoder förespråkar brister. Vidare menar Awad (2005) att en av grundarna till det agila manifestet har ifrågasatt hur säkert och tillförlitligt arbetet genomförs inom agila metoder. Ifrågasättandet grundar sig enligt Awad (2005) på att de agila metoderna inte har traditionella "walkthrough" och kodinspektionerna under projektets livscykel.

En fördel med de agila utvecklingsmetoderna är däremot att det blir enklare att genomföra produktförändringar i projektets testfas eftersom leverans inte endast sker i slutet av ett projekt såsom i de traditionella arbetssätten (Gustavsson, 2013).

En annan fördel är att det i en studie visat sig att de flesta utvecklarna finner att face-to-face kommunikationen inom agila utvecklingsmetoder är ovärderliga när det gäller att sprida information inom utvecklingsprojekten (Williams, 2012). Williams (2012) beskriver att de agila grundprinciperna har ett stort stöd hos utvecklarna. Chasalow (2008) menar på att i det projekt han studerade ansåg projektmedlemmarna att face-to-face kommunikationen var ett effektivt sätt att sprida information. Dock ansåg de att det ibland kunde vara för mycket information att ta in på genom endast informell kommunikation.

2.5 Dokumentation inom mjukvaruutveckling

Den satsning som läggs på dokumentation är en central kostnadsdrivare vid mjukvaruutveckling. Studier visar att cirka 11 % av ett projekts totala kostnad läggs på dokumentation (Zhi, Garousi-Yusifo, Sun, Garousi, Shahnewaz, Ruhe, 2014).

En bestående fråga inom mjukvaruutveckling är hur mycket dokumentation som anses tillräcklig under ett utvecklingsprojekt. Det finns ett flertal fall som beskriver avsaknad av samt bristfällig dokumentation hos system, speciellt i gamla system (legacy systems) (Garousi, Garousi-Yusifoğlu, Ruhe, Zhi, Moussavi, Smith, 2014).

Dokumentation av programvara är en formell skrift som stödjer användning samt utveckling av mjukvara (Ding, Liang, Tang, Vliet, 2014). År 2014 gjordes en systematisk kartläggning av litteratur som behandlar dokumentation. I studien listades vad dokumentation inom mjukvaruutveckling är.

- Dokumentation beskriver ett mjukvarusystem.
- Dokumentation förväntas tillhandahålla exakt information av ett system.
- Dokumentation kan vara produktmanualer som utvecklare skapat till övriga användare.
- Dokumentation av utveckling eller teknisk dokumentation används som kommunikationsmedel mellan projektmedlemmar.
- Dokumentation kan exempelvis innefatta krav, design, kod, kommentarer och testfall.
- Dokumentation kan presenteras på olika sätt, såsom traditionellt skriven text och grafiska modeller som till exempel UML-diagram.
  (Zhi et al, 2014)

Zhi et al (2014) skriver om Forward som definierar dokumentation av programvara som en artefakt vars syfte är att användas som en form av kommunikation mellan projektmedlemmar. Parnas definierar dokumentation som en skriftlig beskrivning som har en officiell status eller auktoritet och kan användas för att tillhandhålla noggrann information om systemen (Zhi et al, 2014).

Den dokumentation som skapas under ett mjukvaruutvecklingsprojekt leder till ökad användbarhet om den håller viss kvalitet (Garousi et al, 2014). Användbarheten av dokumentation beskrivs enligt Garousi et al (2014) som: *”A measure of how well a document’s content can provide knowledge, information and/or insight to its reader with respect to other available artifacts considering both the activity and value of the artifact”*.

Dokumentation av programvara delas i stort sätt in i teknisk och icke-teknisk dokumentation. Icke-teknisk dokumentation kan till exempel vara användarmanualer medan teknisk dokumentation handlar om kravspecifikationer, kommentarer av källkod, testdokument och processbeskrivningar. Teknisk dokumentation inkluderar den dokumentation som är kopplat till det som skapas under programvarans livscykel (Garousi et al, 2014). Den tekniska dokumentationen brukar ofta komma med beskrivande text, detta för att underlätta förståelsen för icke-tekniska läsare (Antoniol, Canfora, Casazza, De Lucia, Merlo, 2002).

Dokumentationen som utformas under ett mjukvaruutvecklingsprojekt har olika syften under olika delar av systemets livscykel (Garousi et al, 2014). Garceau och Jancura (1990) menar att fördelarna med att effektivt arbeta med dokumentation under hela mjukvaruutvecklingens livscykel är att stödet och godkännandet från högsta ledningen samt mellanchefer ökar samt att mjukvaran som skapas blir mer hållbar, det vill säga, kan användas längre om det finns välutformad dokumentation.

När ett system ska implementeras kan dokumentation från de tidigare faserna av projektet vara till nytta för utvecklarna. Befintliga kognitionsmodeller är överens om att utvecklare använder olika typer av kunskap för att skapa förståelse för ett system eller olika delar av det. Denna kunskap är alltifrån domänspecifik kunskap till allmän kunskap gällande programmering (Antoniol et al, 2002). Garousi et al (2014) menar att programmerare även använder sig av intern dokumentation som hjälpmedel för att skapa ytterligare systemförståelse. Även Antoniol et al (2002) menar att spårbarhet mellan kod och olika typer av dokumentation såsom manualer, designdokument samt kravspecifikationer hjälper programmerare att skapa förståelse för ett system. Utöver det stöd i utvecklingen som dokumentation bidrar med under utvecklingsfasen menar Garousi et al (2014) att dokumentation även är till hjälp under testfasen. UML-diagram kan då användas som stöd för enhets- samt regressionstestning.

Vid underhåll av system kan dokumentation användas för att underlätta systemförståelse (Garousi et al, 2014). Den kan användas för att se hur olika delar av systemet påverkar varandra, hur olika delar av ett system skulle reagera på ett förändringsförslag (Antoniol et al, 2002) och för att identifiera var i koden som ändringar ska genomföras (Garousi et al, 2014). Utan dokumentation kan förståelse för ovanstående endast erhållas genom att läsa källkoden, vilket kan leda till att det tar lång tid att hitta relevant information och få en förståelse för hur systemet fungerar (Garousi et al, 2014).

Utöver ovanstående användningsområden kan dokumentation användas för att öka kunskapsspridning. Kunskap kan delas in i implicit kunskap och explicit kunskap. Explicit kunskap innefattar det som är kodat i en särskild form som till exempel ett dokument eller en modell. Den är enkel att använda och återanvända. Implicit kunskap är den information som finns i människors huvud och som kan vara svår att uttrycka eller visualisera. Det finns risk för att tyst kunskap försvinner med tiden eftersom den som innehar kunskapen kanske lämnar teamet. Dokumentation gör att implicit kunskap blir explicit kunskap vilket leder till att alla projektmedlemmar kan ta del av den (Ding et al, 2014).

Kunskapsspridning genom dokumentation är användbart eftersom det, speciellt i stora team, kan vara svårt att dela personlig kunskap mellan alla projektmedlemmar. Beslut som dokumenteras under design-, utvecklings- och underhållsfasen kan också vara till hjälp då nya projektmedlemmar ansluter till ett projekt eller då utvecklare ska bygga vidare på eller underhålla ett system som de själva inte har varit fullt delaktiga i (Garousi et al, 2014).

Som tidigare nämnts är dokumentation inom mjukvaruutvecklingsprojekt en central kostnadsdrivare, cirka 11 % av ett projekts totala kostnad läggs på dokumentation. Det är därför, med tanke på kostnaden, naturligt att inom ett projekt förvänta sig att dokumentationen någon gång under utvecklings- eller underhållsfasen bidrar till att ett flertal aspekter inom arbetet med systemet underlättas (Zhi et al, 2014). Zhi et al (2014) menar att dokumentation bör bidra med förbättringar inom alla aspekter som är centrala för mjukvaruutveckling. Exempel på aspekter som med hjälp av dokumentation bör kunna underlättas är tiden det tar att slutföra en uppgift, förbättrad kvalitet på kod och högre produktivitet (Zhi et al, 2014). En svårighet med dokumentation är att det av vissa uppfattas som kostsamt och svårt att underhålla i projekt.

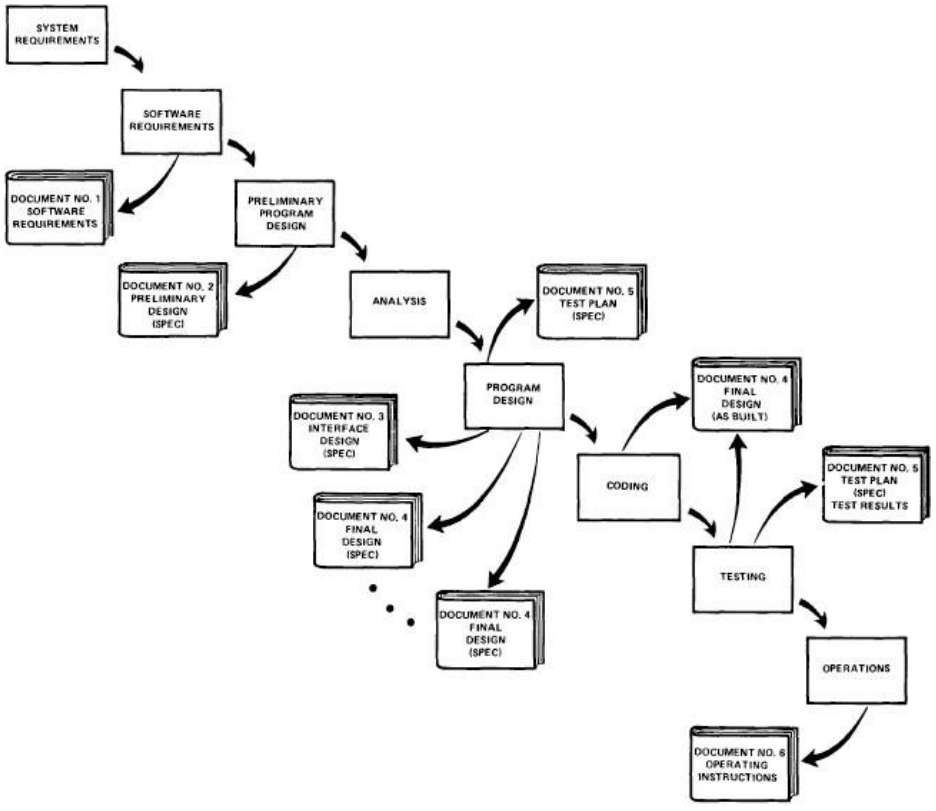
2.5.1 Dokumentation inom traditionella utvecklingsmetoder

Vattenfallsmetoden har vanligtvis väldefinierade gränser och ansvarsområden för de som arbetar inom projektet och utvecklingsprocessen dokumenteras väl under hela arbetets gång (Zhang et al, 2010). Dokumentationen måste vanligtvis verifieras mellan varje steg vilket leder till att arbetet med utvecklingen vanligtvis inte kan fortskrida förrän dokumentationen från förgående fas är färdig (Zhang et al, 2010). Motivationen till att utforma dokumentation efter att en fas genomförs och innan arbetet går vidare i nästa fas är att dokumentationen fungerar som kommunikationsmedium för kunskapsspridning gällande projektet mellan de individer som arbetar med det. Kunskapen som finns i dokumentationen är i form av riktlinjer och klargörande över projektdetaljer (Leau et al, 2012).

En fördel med vattenfallsmetoden är att den ger en tydlig struktur för att organisera och styra aktiviteter under hela projektets gång. Dokumentation av funktioner som ska tillämpas är detaljerad och underlättar på så sätt för distribuerade team genom att de kan arbeta mer självständigt. Detta minskar problem och risker som är kopplade till kommunikation på distans eftersom kommunikationen mellan projektmedlemmarna är minimal (Papadopoulos, 2015).

Royce (1970) menar på att minst sex olika typer av dokumentation bör utformas under utvecklingens gång (se figur 2.4).

1) Kravdokument, som bör utvecklas under analysfasen.
2) Dokumentation över preliminär design under designfasen.
3) Gränssnittsdokumentation designfasen.
4) Slutgiltig designdokumentation i designfasen.
5) Framtagande och dokumentation av en testplan under designfasen som sedan uppdateras under testfasen.
6) Användarmanualer samt instruktioner som tas fram under implementationsfasen.



Figur 2.4 Dokumentation vattenfallsmetoden (Royce, 1970, s.333)

2.5.2 Dokumentation inom agila metoder

Att ta beslut kring hur mycket dokumentation som ska utformas under agila utvecklingsprojekt är fortfarande ett problem i många agila projekt (Garousi et al, 2014). Enligt det agila synsättet prioriteras utvecklingen av mjukvara framför dokumentation av systemet. Detta på grund av att det enligt det agila synsättet ska produceras mjukvara i systemutvecklingsprojekt, inte dokumentation (Kakar, D. Hale & J. Hale, 2012). Förespråkare för agila metoder menar att detaljerad dokumentation i vissa fall kan vara nödvändigt, men att koncis dokumentation är att föredra då den är lättare att ta till sig. Detaljer i ett projekt ändras snabbt och dokumentation hinner oftast inte uppdateras och blir således irrelevant för projektet. Projektdetaljer kommuniceras på bästa sätt via samtal projektmedlemmar emellan, face-to-face (Rüping, 2003). Awad (2005) skriver om Scott Ambler som menar att organisationer ofta begär mer dokumentation än som behövs.

Enligt Rüping (2003) finns det tre tekniker som kan användas för att arbeta fram rätt typ av dokumentation vid agila utvecklingsprojekt. Den första tekniken är att dokumentationen ska vara strukturerad på ett användbart sätt så att läsare ska kunna ta till sig informationen. Den andra tekniken menar på att diagram i dokumentationen kan underlätta och användas under face-to-face kommunikationen under projekten. Den sista tekniken som Rüping (2003) nämner är att dokumentationen ska innehålla relevant meta-information så att det är lätt att avgöra om dokumentationen är relevant för just det system som byggs. Rüping (2003) menar vidare att ett "light but sufficient" synsätt till dokumentation är att föredra inom de agila utvecklingsmetoderna eftersom det bidrar till att projektteamet inte lägger för mycket tid och möda på den. Andra menar att "light but sufficient" dokumentation gör det lättare för användaren att ta den till sig. Således blir denna typ av dokumentation mer användbar under projektet.

Enligt Stettina & Heijstek (2011) behövs det mer dokumentation än vad som används inom agila utvecklingsprojekt idag. Anledningen till dokumentationsbrist beror på att de agila metoderna inte förespråkar dokumentation. Hur mycket som ska dokumenteras är upp till utvecklingsgruppen och tolkningsbart efter den agila metodens riktlinjer för dokumentation vilket är "light but sufficient" (Stettina & Heijstek, 2011). Samtidigt menar Prause & Durdik (2012) att problemet inte är att agila metoder inte förespråkar viss dokumentation utan att den negligeras.

Förespråkare för dokumentation inom agila mjukvaruutveckling menar att dokumentation kan användas för att utbilda nya projektmedlemmar, motivera medlemmarna i projektet samt preservera kunskap inom teamet (E. Rubin & H. Rubin, 2011). Dokumentation kan även förhindra att viktig kunskap förloras om nyckelpersoner inom projektet lämnar teamet eller är otillgängliga (Kakar et al, 2012). Analysfasen är den fas då de traditionella utvecklingsmetoderna kommer fram till vilka funktioner systemet ska ha. Inom traditionella utvecklingsmetoder skapas viktig dokumentation under analysarbetet. Dessa dokument och

den processen saknas inom agila metoder. E. Rubin & H. Rubin (2011) hävdar att mer dokumentation skulle underlätta det agila utvecklingsarbetet då det ofta bristande kravdokumentation som gör att problem uppstår inom mjukvaruutveckling. Vidare kommenterar Chasalow (2008) på ett utvecklingsprojekt där en av de kritiska punkterna i projektet var dokumentationen. Teamet hade arbetat enligt Scrum och menade på att det behövdes dokumentation både för applikationssupport och eventuella förbättringar. Problematiken låg bland annat i att det under projektet inte funnits tid för dokumentation vilket i vissa fall ledde till att ingen dokumentation utformades.

Vissa förespråkare av agila metoder menar på att källkoden ska vara den enda dokumentationen som behövs. E. Rubin & H. Rubin (2011) menar dock på att källkoden inte helt kan ersätta traditionella designdokument eftersom det kan leda till kunskapsförlust. En annan anledning till att använda sig av traditionella designdokument tillsammans med källkod är att dokumentationen gör det lättare för projektteamet att kommunicera med kunden (E. Rubin & H. Rubin 2011). De alternativ, utöver dokumentation, som finns för att lära sig koden är att få den förklarad för sig. Medlemmarna i projektet får då undersöka koden och försöka förstå hur den är strukturerad (Selic, 2009). Att förlita sig på endast samarbete inom projektgruppen innebär att alla medarbetare måste besitta samma kunskap och språk för att kommunikationen ska fungera optimalt (E. Rubin & H. Rubin, 2011). De experter som lär ut systemets funktionalitet blir tvungna att göra om detta varje gång det ansluter en person till projektet och informationen blir ofta bristfällig (Selic, 2009).

I en litteraturstudie av Kakar et al (2012) diskuteras även att agila metoder saknar grunder för att bygga långvariga system, som till exempel dokumentation av övergripande mjukvaruarkitektur och processmodeller. Saknaden av dokumentation kan på sikt leda till problem hos företag som anammar agila metoder då sakna av dokumentation ofta leder till problem med att upprätthålla samt underhålla system, något som de flesta företag någon gång måste göra. Att undvika dokumentation eller att dokumentera så lite som möjligt kan medföra att arbetet med utvecklingsarbetet går snabbare, men i längden leda till att kostnaden för ett system kostnaden blir högre.

2.6 Presentation av faktorer

Nedan presenteras resultatet från den litteraturstudie som genomfördes på 20 artiklar. Studien genomfördes för att identifiera faktorer som påverkas av dokumentation inom agila utvecklingsprojekt. Vi går därefter igenom varje faktor utifrån den relevanta forskningen inom ämnet.

Tabell 2.1 Faktorer i agila projekt som påverkas av dokumentation

	Rubin et al, 2010	Santos et al, 2014	Garousi et al, 2014	Moe et al, 2012	Zhi et al, 2014	Selic et al, 2009	Turk et al, 2104	Stettina et al, 2012	Hall, 2014	Parker et al, 2015	Bonner et al, 2010	Petersen et al, 2012	Holten et al, 2015	Ahimbisibwe et al, 2014	Estler et al, 2013	Layman et al, 2006	Lee & Xia, 2010	Gandomani et al, 2013	Rao et al, 2011
Flexibilitet		X	X	X									X	X	X		X	X	X
Produktivitet		X		X	X										X			X	
Kunskapsspridning	X	X	X	X			X	X		X	X	X		X				X	
Förvaltning			X			X	X	X											
Dokumentationskvalitet					X				X				X	X			X	X	X
Kommunikation	X	X	X			X	X	X			X		X	X	X	X	X	X	X

2.6.1 Flexibilitet

Korta iterationscykler ger den flexibilitet som behövs för att tillgodose förändringar i agila processer. Att ändra kraven betraktas som en möjlighet att tillfredsställa kundernas behov, vilket är anses bättre än att hålla fast vid gamla krav. Om kundernas behov förändras sent i projektet, är det viktigt att projektteamet anpassar sig till dessa förändringar för att lyckas med projektet (Turk, France & Rumpe, 2014).

Agila metoder handlar om att skapa och svara på förändringar för att nå framgång i en miljö som ständigt förändras. Förmågan att balansera flexibilitet och stabilitet. Självorganiserade team är nyckeln för flexibilitet (Lee & Xia, 2010). Agila metoder baseras på iterativa och anpassningsbara cykler och är designade för att anpassas till förändringar. Den iterativa processen fokuserar på att anpassa projektet vid varje iteration beroende på om nya kravändringar uppkommer (Ahimbisibwe, Cavana & Daellenbach, 2014).

2.6.2 Produktivitet

Agila metoder baseras på aktiviteter som stödjer snabb utveckling och hög kvalitetsproduktion. Estler, Nordio, Furia, Meyer & Schneider (2014) nämner att i en studie som involverade studenter och professionella programmerare visade det sig att det inte fanns något bevis på att parprogrammering (där dokumentationen är minimal) skulle förbättra produktiviteten i företaget. Dokumentation kan underlätta att kunskap sprids och på så sätt förbättra organisationens produktivitet (Santos et al, 2014).

2.6.3 Kunskapsspridning

En faktor där dokumentation spelar en roll är vid kunskapsspridning (E. Rubin & H. Rubin, 2011). Alla dokument som skapas ska hjälpa utvecklarna att förstå ett system så att de i sin tur kan utföra sina uppgifter på ett bättre och mer effektivt sätt. Dokumentationen kan användas för att utbilda nya medlemmar i projektet. De kan genom dokumentationen förstå mål samt de tekniska lösningarna samt se vart i koden som ändringar ska genomföras (Garousi, G., Garousi-Yusifoglu, V., Ruhe, G., Moussavi., Smith, B, 2014). För att kunderna ska förstå hur utvecklingsprojektet går så sker kommunikation mellan projektteamet och kunderna. För lite dokumentation kan leda till att denna kommunikation blir bristfällig då det blir svårare att förklara systemet på ett sätt så att kunden förstår (Layman, Williams, Damian, Bures, 2006). Kunskapsspridning och kommunikation inom agila projekt fungerar även som en positiv faktor i att utveckla en innovativ miljö (Moe, Barney, Aurum, Khurum, Wohlin, Barney, Gorschek & Winata, 2012).

2.6.4 Förvaltning

Experter inom mjukvaruutveckling menar på att underhållskostnaden är 75-80 % av projektets totala kostnad (Brandt, 2005). SouarIssa och Stenlund (2012) har funnit att vanligt förekommande problem inom systemförvaltning är obefintlig eller bristande systemdokumentation, komplex och ostrukturerad kod, svårigheter att förstå användarnas förväntan och brist på erfarenhet när systemförvaltare lämnar projektgruppen.
Turk et al (2014) sammanfattade i sin rapport de agila principerna som presenterats av den agila alliansen 2001 och deriverade utifrån dem antaganden som kan göra att projekt misslyckas. Ett av antagandena de hittade inom agila metoder var den om att dokumentation och mjukvarumodeller är kontraproduktivt. Författarna menar att det finns situationer där modeller är värdefulla, och att det är fördelaktigt att behålla dessa modeller för framtida bruk. En av de situationer som författarna menade att dokumentationen och modellerna är användbara vid är under förvaltning av system. Finns dokumentation tillgänglig när ett system ska modifieras kan kostnaderna samt arbetet som måste läggas på ändringarna minska. Utan modeller är utvecklare tvungna att analysera källkod för att förstå den och bedöma effekten av

förändringar (Turk et al, 2014). Dokumentation kan således underlätta arbetet när förändringar av ett system ska genomföras.

Dock visar en undersökning av Garousi et al (2014) att teknisk dokumentation används som mest som en informationskälla under utvecklingsprojektet och minst under systemförvaltning.

2.6.5 Dokumentationskvalitet

De kvalitetsattribut som är mest förekommande inom forskning gällande dokumentation inom mjukvaruutveckling är att den dokumentation som tas fram ska vara 1) lättillgänglig 2) komplett 3) konsekvent (Zhi et al, 2014). Zhi (2014) menar att kvalitetskriterierna influeras av hur mycket tid och arbete som läggs på dokumentation.

2.6.6 Kommunikation

Det agila arbetssättet förespråkar konstant kommunikation bland projektmedlemmar och kunder där face-to-face-kommunikation är särskilt viktig. Agil mjukvaruutveckling beskrivs som "a cooperative game of invention and communication" (Hummel, Rosenkranz & Holten, 2015). Ahimbisibwe et al (2014) listar 37 kritiska framgångsfaktorer för ett mjukvaruutvecklingsprojekt. Dessa faktorer har identifierats eftersom de anses ha betydelse för ett projekts framgång. Av dessa hamnar kommunikation på tionde plats (Ahimbisibwe et al, 2014). Kommunikation kan delas in i direkt och indirekt. Direkt kommunikation kan till exempel omfatta face-to-face, telefon eller annan media. Indirekt kommunikation involverar det som skrivs såsom dokument, email eller böcker (Hummel et al, 2015). Strukturerade processer understryker ofta vikten av att upprätthålla dokumentation (Estler et al, 2014). Kommunikation är en betydande komponent vad gäller samarbete inom mjukvaruutveckling. Empiriska studier visar på att utvecklare är beroende av formell kommunikation såväl som informell kommunikation. Projektmedlemmar som är utspridda över olika länder har ofta en bristfällig kommunikation, vilket resulterar i dålig kännedom kring arbetsprocessen och de framsteg som görs (Layman et al, 2006).

3 Metod

I detta kapitel redogör vi för den undersökningsmetod som använts i studien samt hur vi har arbetat för att analysera våra empiriska data. Avsnittet avslutas med en redogörelse för studiens reliabilitet samt validitet.

3.1 Litteraturstudie

För att finna relaterade artiklar till problemområdet genomfördes en litteraturstudie. De verktyg vi använde oss av för att hitta relevanta artiklar var olika sökmotorer för akademiska artiklar. Den största delen av artiklarna som användes för litteraturstudien hittades genom Senior Scholars' AIS, Google Scholars och Lubsearch. Sökningarna bestod av nyckelord relevanta för problemområdet.

Vid analys av den litteratur vi fått fram identifierades de sex faktorer inom agila mjukvaruutveckling som dokumentation kan påverka. Dessa sex faktorer presenteras i litteraturgenomgången och ligger till grund för utformandet av vår empiriska undersökning.

3.2 Metod för insamling av empirisk data

3.2.1 Forskningsansats

Vår forskningsfråga är: *Hur ställer sig användare av agil mjukvaruutveckling till dokumentation: och vilka faktorer påverkas av dokumentation?*

Målet med denna studie är att tolka och skapa förståelse för dokumentation inom agila projekt och dess inverkan på sex faktorer. För att besvara den explorativa forskningsfrågan vi utgår från i denna uppsats anser vi att ett kvalitativt upplägg är mest lämpligt. Den kvalitativa undersökningen ger en nyanserad beskrivning av ett ämne och är passande för att få större klarhet i ett ämne (Jacobsen, 2002). Genom ett kvalitativt upplägg kan vi skapa förståelse för det sammanhang som dokumentationen av systemutvecklingen sker i och är således ett lämpligt metodval för denna studie.

Vår studie har därför lagts upp som små-N-studier, detta upplägg leder till en rikare och bredare beskrivning av ett fenomen än om endast ett fall hade studerats (Jacobsen, 2002). Vi har i denna studie valt ut sju företag för att gå på djupet med samt skapa förståelse för dokumentationens användning inom agila projekt samt hur dokumentation kan påverka olika

faktorer inom agila systemutvecklingsprojekt. För att få en så nyanserad bild av fenomenet som möjligt har en kvalitativ ansats med öppna intervjuer använts för insamling av empiri.

3.2.2 Urval

Vid urvalet i denna studie var första steget i urvalsprocessen att få en överblick över företag som var relevanta för studien, det vill säga den teoretiska populationen. Efter att vi undersökt alla potentiella enheter satte vi upp fyra kriterier som vi ville att företagen skulle uppfylla. Detta för att finna de mest lämpade enheterna för studien (Jacobsen, 2002). Kriterierna var att företagen skulle vara verksamma i Sverige, de skulle arbeta agilt, arbeta med utveckling och vara ett produktföretag som levererade mjukvara till kund.

Vi kontaktade, via mail, de enheter som uppfyllde ovanstående kriterier och som vi ansåg kunde ge oss nyanserad och god information gällande agila metoder (Jacobsen, 2002). Företagen som svarade bildade en grupp av företag som arbetade enligt någon av de agila metoderna. Dessa företag var både mindre konsultföretag samt globala IT-företag.

Informant 1: Scrum master på ett mindre utvecklingsföretag
Informant 2: En utvecklare samt agil coach på ett världsledande konsultföretag
Informant 3: Utvecklare på ett medelstort företag
Informant 4: Produktägare på kravsidan
Informant 5: Projektledare
Informant 6: Utvecklare på ett medelstort företag
Informant 7: Expert inom projektledning och kravhantering

3.2.3 Intervjuguide

Vi utformade våra intervjufrågor för att för en djupare inblick i dokumentationens betydelse i agila projekt. Vi ville utforska de olika faktorer inom agila projekt som vi sett var återkommande i litteraturen:

Flexibilitet, produktivitet, kunskapsspridning, förvaltning, dokumentationskvalitet och kommunikation.

Frågorna formades för att få en djupare inblick i dokumentationens betydelse i relation till ovanstående faktorer. Vi ansåg att det var nödvändigt att göra denna avgränsning för kunna besvara forskningsfrågan. Detta eftersom vi endast fokuserar på de faktorer vi tagit fram i litteraturstudien i relation till dokumentation inom agila projekt och inga andra aspekter som kan påverka dessa faktorer. Vi har i intervjuguiden inte följt den exakta ordningen av faktorer listade ovan. Utan vi har blandat dem för att minimera risken att påverka informanternas svar. På grund av att intervjuguiden inte var uppdelad enligt en tydlig struktur var vi tvungna att under intervjuernas gång vara lyhörda på att vi fick information gällande hur de olika faktorerna påverkas av dokumentation.

3.2.4 Genomförande

För att besvara vår forskningsfråga har vi genomfört kvalitativa intervjuer. Jacobsen (2002) menar att kvalitativa intervjuer inte bör vara helt ostrukturerade. Därför har intervjuerna i denna studie utformats på ett semistrukturerat vis. Detta innebär att vi utifrån en intervjuguide (se bilaga 7.1), öppet samtalat kring ämnen som är relevanta för att besvara vår forskningsfråga. Fördelen med att utforma en intervjuguide utan fast ordningsföljd och med frågor som enbart kan besvaras med öppna svar är att intervjuobjektet har få eller inga begränsningar i sina svarsformuleringar (Jacobsen, 2002).

Samtliga intervjuer inleddes med en kort beskrivning av studiens syfte. Därefter har vi har i enlighet med rekommendationer i Jacobsen (2002) inte haft någon förutbestämd ordningsföljd av frågor under intervjuerna utan ämnena har bearbetats och styrs utifrån informanternas svar. Intervjuguiden har endast används när informanterna inte själv diskuterat kring ett visst ämne. Upplägget har gett oss möjligheten att ställa följdfrågor under intervjuernas gång, något vi inte hade kunnat göra i samma utsträckning om vi valt en kvantitativ ansats.

Vid genomförandet av intervjuerna har vi anpassat oss efter enheternas önskemål om tid och plats. Majoriteten av intervjuerna har genomförts på uppgiftslämnarnas arbetsplats och ett fåtal intervjuer har genomförts via Skype.

3.2.5 Tolkning av empiri

För att minska risken att gå miste om information så spelades alla intervjuer in för att sedan transkriberas. Transkriberingarna, det vill säga de tjocka beskrivningarna, reducerades sedan genom kodning. Detta på grund av att de tjocka beskrivningarna är ostrukturerade. Systematisering genom kodning är därför viktig för att förmedla vad vi funnit under vår empiriska undersökning (Jacobsen, 2002).

Den kodade data använde vi sedan för att avlägsna analysen från de enskilda informanterna till att sätta fokus på de för studien relevanta fenomen. Vi kopplade samman information från de olika informanterna för att finna samband mellan de olika fenomenen.

3.2.6 Etik

Enligt Vetenskapsrådets (2002) forskningsetiska principer finns det inom humanistisk och samhällsvetenskaplig forskning fyra huvudkrav att ta ställning till; *informationskravet, samtyckeskravet, konfidentialitetskravet* samt *nyttjandekravet*.

Informationskravet innebär att intervjuenheterna har information om forskningens syfte. Detta krav innebär att forskarna ska tillhandahålla all information som kan komma att påverka enheternas beslut att delta i undersökningen. Kravet innefattar även att intervjuobjekten ska vara medvetna om deras uppgift i projektet samt vilka rättigheter de har inom det (Vetenskapsrådet, 2002).

Samtyckeskravet innebär att enheterna har rätt att bestämma över sin medverkan, hur länge de ska delta samt på vilka villkor. Avbrott under deltagandet ska inte under några omständigheter påverka enheterna och innan undersökningen ska enheterna lämnat samtycke för deltagande till forskarna (Vetenskapsrådet, 2002).

Konfidentialitetskravet innebär att de personuppgifter samt etiskt känsliga uppgifter som lagras om enheterna under undersökningen ska behandlas och lagras på ett sätt som gör det omöjligt för enskilda deltagare att identifieras av personer utanför forskningsteamet (Vetenskapsrådet, 2002).

Nyttjandekravet innebär att personuppgifter som samlats in under forskningsarbetet inte får användas för kommersiellt bruk eller andra icke-vetenskapliga ändamål. (Vetenskapsrådet, 2002). Vi har vid arbetet med denna studie följt ovanstående etiska riktlinjer.

3.2.7 Reliabilitet och validitet

Enligt Holme och Solvang (1997) bestäms en studies reliabilitet av hur mätningarna har utförts samt graden av noggrannhet vid bearbetning av insamlad data. För att åstadkomma reliabilitet har vi under arbetets gång varit noga med vår datainsamling samt analys av denna. För att stärka reliabiliteten har vi även redovisat vår arbetsgång för att ge en tydlig bild av vårt tillvägagångssätt. Vi har låtit vår studie granskas av utomstående för att hitta svagheter i den. Dessa personer har varit vår handledare samt opponentgrupper vid institutionen för informatik vid Lunds universitet. En ytterligare åtgärd som har vidtagits för att uppnå hög reliabilitet i denna studie är att vi har låtit alla informanterna ta del av intervjutranskriberingar för att säkerställa att inga missuppfattningar skett.

Validitet innebär enligt Höst och Runeson (2006) att forskarna under studiens gång mäter det som de avser att mäta. Informationen som mäts inom studien ska kunna användas för att pröva frågeställningen (Holme & Solvang, 1997). För att uppnå hög validitet har vi därför kritiskt granskat att alla delar och moment. För att säkerställa att alla den information som presenteras och de intervjuer vi genomfört är relevanta för att besvara frågeställningen.

4 Informanternas syn på dokumentation

I detta kapitel redovisar vi för den empiriska undersöknings resultat. Vi börjar med att presentera informanterna. Sedan följer en presentation av en individuell sammanställning av informanternas svar (för hela transkriberingar se bilaga 7.2 – 7.8).

4.1 Informanter

Tabell 4.1 Presentation av informanter

Typ av företag	**Befattning**	**Intervjumetod**	**Tid**
Konsultföretag med fokus på utveckling	Scrum master	Personligt möte	20 minuter
Världsledande konsultföretag inom IT	Agil coach och utvecklare	Personligt möte och Skype	13 minuter
Konsultföretag inom IT och verksamhetsutveckling	Utvecklare	Personligt möte och Skype	12 minuter
Utvecklingsföretag	Produktägare på kravsidan	Telefon	27 minuter
Konsultföretag inom design och konstruktion av webbportaler	Projektledare	Personligt möte	14 minuter
Konsultföretag med fokus på applikationssäkerhet	Utvecklare	Personligt möte	26 minuter
Konsultföretag inom business intelligence	Expert inom projektledning och kravhantering	Personligt möte	30 minuter

4.2 Sammanfattning av informanternas svar

4.2.1 Flexibilitet

Tabell 4.2 Sammanfattning av informanternas svar gällande dokumentationens påverkan på flexibilitet

Företag	**Sammanfattning**
Konsultföretag med fokus på utveckling	Informanten menar att med rätt mängd dokumentation ökar flexibiliteten då det blir lättare att ta in nya resurser och förändra projektets struktur. Däremot menar informanten att för mycket dokumentation är tidskrävande och gör att arbetet försvåras. Samt att allt för omfattande dokumentation lätt blir förlegad.
Världsledande konsultföretag inom IT	Informanten säger att dokumentation underlättar flexibiliteten samt att det blir lättare att göra förändringar i systemet, då den som genomför en viss ändring inte nödvändigtvis är samma person som utvecklade en viss applikation från början. Informanten anger att dokumentation på en lagom nivå behövs så att andra förstår de ändringar som har gjorts.
Konsultföretag inom IT och verksamhetsutveckling	Informanten anser att utan dokumentation hade det blivit svårare att driva projektet. Förutsatt att dokumentationen fungerar smidigt då det är en del av processen. Informanten tycker att dokumentation gör det betydligt enklare genomföra förändringar. Vissa funktioner är beroende av varandra och görs ändringar i en av dem kan det vara så att de andra slutar fungera om dokumentation inte finns.
Utvecklingsföretag	Informanten tycker inte att dokumentation gör någon större skillnad vad gäller flexibilitet. Antingen utformas dokumentation som beslut tas utifrån eller så görs lösningar för stunden.
Konsultföretag inom design och konstruktion av webbportaler	Informanten anser att flexibiliteten ligger i sprintarna, och att flexibiliteten varken ökar eller minskar med dokumentation. Dock anger informanten att det går mycket smidigare att genomföra förändringar om dokumentation finns då det är lättare att hitta vem som gjort vad och när. Med mer dokumentation menar informanten att det hade varit lättare att införa nya saker.
Konsultföretag med fokus på applikationssäkerhet	Informanten anser att dokumentation inte gör arbetet mer flexibelt. Dokumentationen påverkar inte om någonting ska göras om eller inte. Informanten fortsätter med att förklara att dokumentation kan vara svår att uppdatera och på så sätt är den inte aktuell när nästa person ska sätta sig in i systemet.
Konsultföretag inom business intelligence	Informanten förklarar att om projektteamet vet vilka ramar dem rör sig inom så ökar flexibiliteten eftersom beslut kan tas snabbare och förståelsen för vad som ska göras. Informanten tycker att det blir enklare att genomföra förändringar i systemet om det finns dokumentation.

4.2.2 Produktivitet

Tabell 4.3 Sammanfattning av informanterna svar gällande dokumentationens påverkan på produktivitet

Företag	**Sammanfattning**
Konsultföretag med fokus på utveckling	Informanten menar på att om dokumentation används för mycket blir det plottrigt i koden och det blir ofta förlegat vilket gör att det blir svårare att jobba med och det tar tid.
Världsledande konsultföretag inom IT	Informanten säger att om dokumentation skapas på rätt sätt, ökar produktiviteten. Det ska inte dokumenteras mer än vad som är nödvändigt.
Konsultföretag inom IT och verksamhetsutveckling	Informanten anser att dokumentation gör att produktiviteten minskar på kort sikt. Däremot menar informanten att dokumentation gör att det blir mer effektivt på lång sikt eftersom de som arbetar i projekten inte behöver sätta sig in i områdena på nytt.
Utvecklingsföretag	Däremot anser informanten att rätt mängd dokumentation är att föredra för effektiviteten. Informanten menar att om det finns dokumentation och således vet vilka ramar som ska hållas inom projektet blir det enklare att börja koda med en gång.
Konsultföretag inom design och konstruktion av webbportaler	Informanten anser att dokumentation minskar produktiviteten initialt eftersom tid måste läggas på att komma ihåg den. Däremot förklarar informanten att produktiviteten ökar på sikt förutsatt att dokumentationen är välformulerad, vilket leder till att det blir lättare att hitta i de olika releaserna för kunderna.
Konsultföretag med fokus på applikationssäkerhet	Informanten svarar att dokumentation påverkar produktiviteten eftersom en ändring som görs i koden även måste dokumenteras. Det blir alltså mindre produktivt.
Konsultföretag inom business intelligence	Informanten anser att produktiviteten ökar eftersom snabbare beslut kan tas och förståelse om vad som ska göras och vilka vägar som kan tas.

4.2.3 Kunskapsspridning

Tabell 4.4 Sammanfattning av informanternas svar gällande dokumentationens påverkan på kunskapsspridning

Företag	**Sammanfattning**
Konsultföretag med fokus på utveckling	Informanten menar på att kunden ser vad de lagt timmar på genom offerten och på det viset delar företaget information med dem. Finns en API så får den uppdateras och beskriva för andra hur det fungerar. På informantens företag använder de sig av wikis, både för enskilda projekt och internt där kod som är krånglig dokumenteras. Kunden äger koden så även kunden ska förstå koden.
Världsledande konsultföretag inom IT	Informanten berättar att de använder user stories som agerar kommunikation till kund. Dessa blir sedan funktioner. Utvecklarna hanterar det mesta arbetet i koden men det är viktigt att få ner vad som gjort och hur. Viktigaste att kunden förstår dokumentationen. Dokumentationen handlar om att andra ska förstå det utvecklaren gjort. Om det är flera människor som arbetar med en kod så behövs dokumentation för att alla ska förstå.
Konsultföretag inom IT och verksamhetsutveckling	Informanten menar på att företaget alltid har reviews på andras arbete för att sprida informationen så mycket som möjligt. Alla ska kunna täcka upp för varandra. När utvecklaren slutför en produkt brukar mycket dokumentation ske för att kunna visa för kund. Vidare berättar informanten att dokumentationen kring hur systemet ska fungera är viktigt för kund och systemteknisk dokumentation är viktigt för utvecklarna för att de ska hålla koll på vad som gjort och varför. Det är viktigt med information när en ny medarbetare kommer in i projektet.
Utvecklingsföretag	Informanten arbetar med parprogrammering för att sprida kunskap kring systemet. Dokumentation används således inte för att sprida kunskap inom projekten.
Konsultföretag inom design och konstruktion av webbportaler	Informanten hade önskat att det fanns mer dokumentation för att kunna sprida kunskap till nyanställda om hur saker och ting fungerar. På informantens företag används dokumentation till att förklara hur saker fungerar snarare än vilka funktioner som finns.
Konsultföretag med fokus på applikationssäkerhet	Informanten menar på att dokumentationen används till för att visa för nästa som ska programmera hur det fungerar.
Konsultföretag inom business intelligence	Informanten menar på att de inom projekten litar på att alla kan det dem ska kunna. Vidare menar informanten dock att dokumentation används när en ny resurs ska fortsätta på ett system eller göra om någonting. För den är det då viktigt att förstå hur den tidigare utvecklaren tänkte.

4.2.4 Förvaltning

Tabell 4.5 Sammanfattning av informanternas svar gällande dokumentationens påverkan på förvaltningsarbete

Företag	**Sammanfattning**
Konsultföretag med fokus på utveckling	Informanten anger att 25 % av deras tid i projekt läggs på systemvård och att ett projekt sällan är 100 % klart. Dokumentationen av systemet används inte i speciellt stor utsträckning av supportteamet utan den dokumentation som skapats är främst till för om kunden vill ta över produkten själv, då menar informanten att de som utvecklat den inte ska behöva sitta och konsultera i 2 år utan kunden ska kunna använda, supporta och underhålla sin produkt direkt. Informanten anger dock att tiden som läggs på underhåll minskar om god dokumentation finns, speciellt i de fall då dem tar över andras projekt.
Världsledande konsultföretag inom IT	Informanten menar att under förvaltning har dokumentation en positiv inverkan på tid om det är nya personer som kommer in i ett gammalt projekt samt att dokumentation antagligen minskar kostnaden. Normalt sett så utvecklar dem en produkt, sen så underhålls den. Informanten anger att det blir det lättare i förvaltning om dokumentationen finns eftersom det antagligen inte är samma person som utvecklade produkten som sitter med förvaltningen.
Konsultföretag inom IT och verksamhetsutveckling	Informanten anger att systemen dem tillhandahåller sina kunder underhålls och vidareutvecklas tills dem lämnar över systemet. Av företagets totala arbete består 50 % av underhåll. Med hjälp av dokumentation menar informanten att arbetet med underhåll och förvaltning kan spara veckors arbete samt att kostnaden för underhåll minskar. Detta på grund av att det i dokumentationen finns information kring hur olika funktioner på verkar varandra samt att grund strukturen kan följas. Detta leder också till att de risker som förknippas med underhåll kan minimeras.
Utvecklingsföretag	Informanten menar att de mest sysslar med förvaltning då dem har en färdig produkt. Det som finns dokumenterat är bakgrunds ledtrådar och user stories. Informanten menar vidare att utvecklingsarbetet hade fungerat utan dokumentation men att det hade kunnat skapa en teknisk skuld, att det görs en lösning för stunden men inte på bästa sätt. Således är dokumentation runt de övergripande delarna viktig för att minimera skulden.
Konsultföretag inom design och konstruktion av webbportaler	Informanten anger att det läggs lite tid på underhåll av kod. Supporten på företaget lägger lite tid på underhåll men fixar saker som gått sönder i systemet. Informanten menar dock att med hjälp av dokumentation så blir det lättare att lösa felen som uppstår. När supporten behöver gå in och lösa ett problem så vet dem exakt var dem ska leta. Således minskar kostnaden i och med att supporten slipper sitta och leta i koden.
Konsultföretag med fokus på applikationssäkerhet	Informanten menar att de lägger mycket tid på underhåll. Men att dokumentation inte påverkar kostnaden för underhållet mer än att det blir dyrare för kunden så dem måste betala per timme.

Konsultföretag inom business intelligence	Informanten menar att det blir en signifikant skillnad i förändringsarbetet om det finns god dokumentation. Med dokumentation kan de som arbetar i ett visst projekt se vilka principer som tillämpas och kan således följa det istället för att hitta på något nytt. Detta leder till att förändringar eller vidareutveckling harmonierar med det tidigare utvecklade. Samt att det går snabbare att hitta i koden. Informanten anger att dokumentation har positiva effekter på kostnaden och tiden vid underhåll.

4.2.5 Dokumentationskvalitet

Tabell 4.6 Sammanfattning av informanternas svar gällande dokumentationskvalitet

Företag	**Sammanfattning**
Konsultföretag med fokus på utveckling	Informanten menar att dokumentation är viktigt och att det genomförs löpande samt att mycket tid läggs på dokumentation (ca 10 % av projekttiden). Företagets standardlösning finns dokumenterad sedan tidigare och utöver den dokumentationen måste även ändringar som är svåra att förstå enbart genom att kolla på koden dokumenteras. Dock är deras mål att koden i sig ska vara så lättförståelig som möjligt. De riktlinjerna för dokumentation som finns på informantens företag säger att om koden är svårskriven ska den göras om, men om den redan är gjord på bästa möjliga sätt och den fortfarande är komplicerad så ska det dokumenteras.
Världsledande konsultföretag inom IT	Informanterna menar att det i början av projektet läggs ner tid på att få ner user stories som rör tre frågor gällande slutprodukten som sedan ska blir funktionalitet. Det finns dedikerade personer som sköter dokumentationen. Dokumentationen lagras i ett speciellt system och det är relativt uppstyrt vad gäller utformningen av den. Informanterna menar att det är viktigt att i ett projekt veta vad som är gjort och hur det har gjorts. Dokumentation och specifikationer görs kontinuerligt i projektet men informanterna menar att det är onödigt att dokumentera i början för då är projektet oroligt. Det är bättre att utforma dokumentationen när projektet är relativt stabilt. En av informanterna anger att under projektets gång måste tänka på att dokumenterar på rätt nivå.
Konsultföretag inom IT och verksamhetsutveckling	Informanten menar dokumentation är ett levande dokument som måste uppdateras hela tiden, inte bara i början eller slutet av ett projekt. Både delkrav och mer tekniska bitar dokumenteras för att relevant information alltid ska finnas tillgänglig. Informanten anger att all dokumentation är viktig men att de olika typerna används på olika sätt, vissa är endast värdeskapande för kunden mena annan endast används internt. På företaget användes mallar för olika typer av dokumentation. Dock finns det inget krav på att dokumentera utan endast generella riktlinjer för projekt.
Utvecklingsföretag	Informanten anger att inget dokumenteras under utvecklingstiden utan att koden är dokumentationen. För att få ökad kvalitetshöjning på det som utvecklas finns dokumentation med riktlinjer och det finns uppritat hur arkitekturen ser ut, hur allting hänger ihop. Den dokumentation som finns förfinas då dem försöker vara så lättviktiga och pragmatiska som möjligt. Det som dokumenteras mer ingående är när det är icke-funktionella krav eller tekniska krav.

Konsultföretag inom design och konstruktion av webbportaler	Informanten menar att grundplattformen inte är dokumenterad, något informanten gärna hade haft. Det som dokumenteras är de förändringarna som görs i grundplattformen. Informanten anger att 10 % av projekttiden läggs på att utforma dokumentation samt att alla anställda är tvungna att dokumentera enligt de riktlinjer som finns på företaget. Både specifika kundförändringar samt ändringar i grundplattformen måste dokumenteras. Informanten menar att det är viktigt att hålla koll på vilka kodändringar som görs så att alla på företaget vet vad som ska ingå i en ny kundrelease. Informanten hade utöver den dokumentation som redan finns gärna sett att det fanns dokumentation över hur saker ska göras.
Konsultföretag inom business intelligence	Informanten anger att dem lägger 4-5 % av den fullständiga projekttiden på dokumentation och att det under olika delar av projektet är viktigt med olika typer av dokumentation. På informantens företag finns inga riktlinjer för hur dokumentationen ska vara utformad och att vissa typer av dokumentation görs i mån av tid. Oftast använder deras utvecklare någon standard dem är vana vid. Hur utförlig dokumentationen är beror på vem det är som i slutet ska använda den. För externa resurser är den mer utförliga än vad den som utformas för internt bruk är. Informanten menar att det finns positiva kringeffekter av att ha dokumentation men att det måste finnas en avvägning, så att den inte blir för omfattande eller djup.

4.2.6 Kommunikation

Tabell 4.7 Sammanfattning av informanternas svar gällande dokumentationens påverkan på kommunikation

Företag	**Sammanfattning**
Konsultföretag med fokus på utveckling	Informanten nämner att de kommunicerar med kunderna under sprintmöten samt då förändringar sker eller om något skiljer sig från planen. I stora projekt kommunicerar informanten med kunden en gång om dagen. Hur kontakten sker är upp till kunden.
Världsledande konsultföretag inom IT	Informanten nämner att kommunikationen sker främst via telefon, Skype eller chatt eftersom de sitter i distribuerade team. Informanten nämner också att de har Scrum-möten varje dag för att hålla sig uppdaterade om vad de andra arbetar med.
Konsultföretag inom IT och verksamhetsutveckling	Informanten förklarar att de har dagliga informella stand-ups. Då går de igenom vad alla jobbar med, eventuella problem och vad som ska arbetas vidare med för att komma vidare. De har även frekventa sprintmöten för återkoppling med kund. Kommunikationen med kund sker face-to-face eller över telefon varje vecka. Mot produktägaren sker kommunikation varannan vecka.
Utvecklingsföretag	Informanten förklarar att kommunikationen är tät mellan utvecklarna eftersom mycket arbete görs tillsammans.
Konsultföretag inom design och konstruktion av webbportaler	Dokumentationen är till för att alla ska se vad som ska ingå i en viss release.
Konsultföretag med fokus på applikationssäkerhet	Använder dokumentation för att förklara för kunden hur systemet ska komma att se ut.
Konsultföretag inom business intelligence	Informanten förklarar att dokumentation i form av specifikationer måste vara väl utformade särskilt i projekt där utvecklarna sitter distribuerat i olika länder. Däremot i interna projekt förlitar de sig på vanlig face-to-face-kommunikation.

5 Resultat

Vi kommer i detta kapitel, utifrån teori och empiri, diskutera samt analysera resultaten av vår studie där vi ämnar besvara forskningsfrågan:

Hur ställer sig användare av agil mjukvaruutveckling till dokumentation: och vilka faktorer påverkas av dokumentation?

5.1 Analys

För att analysera de empiriska resultaten med teori utgår vi från tidigare presenterade faktorer. Avsnittet inleds med en analys av varje enskild faktor i förhållande till empirin.

5.1.1 Flexibilitet

Tabell 5.1 Analys av informanternas svar gällande flexibilitet

	Konsult-företag med fokus på utveckling	**Världsledande konsult-företag inom IT**	**Konsult-företag inom IT och verksamhetsutveckling**	**Utvecklings-företag**	**Konsult-företag inom design och konstruktion av webbportaler**	**Konsult-företag med fokus på applikations-säkerhet**	**Konsult-företag inom business intelligence**
Underlättar dokumentation förändring	Ja	Ja	Ja	Nej	Nej	Nej	Ja
På vilket sätt	Det blir lättare att ta in nya resurser och förändra projektets struktur.	Andra personer ska förstå det som har gjorts. Ändringar görs enklare om det finns dokument-ation på rätt nivå.	Informanten tycker att dokument-ation gör det betydligt enklare genomföra förändringar.	Informanten tycker inte att dokument-ation gör någon större skillnad vad gäller flexibilitet.	Anser att flexibiliteten ligger i sprintarna	Anser att dokument-ation inte gör arbetet mer flexibelt.	Snabbare beslut och förståelse om vad som ska göras och vilka vägar som kan tas.

Förändringsbenägenhet ska inom agila metoder gå före uppföljning av en plan (Greer & Hamon, 2011). Turk et al (2014) beskriver att de korta iterationscyklerna ger den flexibilitet som behövs för att tillgodose förändringar i agila metoder. Att ändra kraven betraktas som en möjlighet att tillfredsställa kundernas behov. Om kundernas behov förändras sent i projektet, är det viktigt att projektteamet anpassar sig till dessa förändringar (Turk et al, 2014). Det är förmågan att reagera på förändring som avgör framgången i ett mjukvaruprojekt (Martin, 2003).

Vi har genom vår empiriska studie funnit att dokumentation till viss del kan stödja flexibiliteten inom agila metoder. Majoriteten av de större företagen anser att dokumentation kan stödja flexibiliteten inom agila projekt samt att den kan bidra till att projektet hanterar förändringar mer effektivt. Hos de mindre företagen varierar informanternas syn på hur dokumentationen påverkar flexibiliteten. En av informanterna hos de mindre företagen menar på att flexibiliteten ligger i sprintarna och att dokumentationen inte påverkar dem. En annan informant menar på att flexibiliteten även ligger i arbetet mot kund och att dokumentationen kan användas för att göra samarbetet mer flexibelt.

5.1.2 Produktivitet

Tabell 5.2 Analys av informanternas svar gällande produktivitet

	Konsult-företag med fokus på utveckling	**Världsledande konsult-företag inom IT**	**Konsult-företag inom IT och verksamhetsutveckling**	**Utvecklings-företag**	**Konsult-företag inom design och konstruktion av webbportaler**	**Konsult-företag med fokus på applikations-säkerhet**	**Konsult-företag inom business intelligence**
Underlättar dokumentation produktiviteten	Ja, men det ska vara lagom.	Ja	Ja, på längre sikt.	Ja	Ja, på längre sikt.	Nej	Ja
På vilket sätt	För mycket dokument-ation kan göra att den blir förlegad vilket gör att det blir svårt att jobba med och tar tid, vilket inte är produktivt.	Dokument-ation av systemet ökar produktivitet-en för att information finns.	Det blir mer effektivt eftersom det finns dokument-ation som beskriver olika funktioner.	Man kan gå rakt in i kodandet istället för att lägga ner tid på att sätta sig in i systemet igen.	För att göra processer effektivare genom att ha information dokument-erad. Både för kund och internt	Produktivitet-en försvåras eftersom en ändring som görs i koden måste dokument-eras. Det blir dubbelarbete.	Snabbare beslut kan tas och förståelse om vad som ska göras.

Zhi et al (2014) menar att dokumentation bör bidra med förbättringar inom alla aspekter som är centrala för mjukvaruutveckling. Exempel på aspekter som med hjälp av dokumentation bör kunna underlättas är: tiden det tar att slutföra en uppgift, förbättrad kvalitet på kod och därmed högre produktivitet (Zhi et al, 2014). Även Santos et al (2014) betonar dokumentationens betydelse för kunskapsspridning som i sin tur underlättar för organisationens produktivitet.

Två av informanterna menar på att dokumentation inte nödvändigtvis bidrar till produktivitet eftersom de ändringar som görs i koden leder till att utvecklaren därmed måste göra dubbelarbete. En av dem menar dock att dokumentation på längre sikt kan leda till en ökad produktivitet. Majoriteten av informanterna anser dock att dokumentationen kan bidra till ökad produktivitet eftersom att dokumentationen av systemet kan göra arbetet effektivare genom att utvecklarna kan få tag i information kring systemet snabbare. En informant menar bland annat att de har utvecklat ett system där de kan söka relevant dokumentation kring systemet beroende på om det är mot kund eller internt som den ska användas. Därmed blir processer mer effektiva och produktiviteten ökar.

5.1.3 Kunskapsspridning

Tabell 5.3 Analys av informanternas svar gällande kunskapsspridning

	Konsult-företag med fokus på utveckling	**Världsledande konsult-företag inom IT**	**Konsult-företag inom IT och verksamhetsutveckling**	**Utvecklings-företag**	**Konsult-företag inom design och konstruktion av webbportaler**	**Konsult-företag med fokus på applikations-säkerhet**	**Konsult-företag inom business intelligence**
Sker kunskaps-spridning genom dokumentation	Ja	Ja	Ja	Ja	Ja	Ja	Ja
Användning av dokumentation vid kunskaps-spridning	Både mot kund samt internt mellan projekt-medlemmar.	User stories mot kund samt dokumentation av kod för att projekt-medlemmar ska förstå.	Systemteknisk för utvecklare samt nya projekt-medlemmar. Dokument-ation i slutet för att förklara för kund.	Arkitektur-specifik för att få riktlinjer hur systemet ska se ut.	Inte genom dokument-ation utan genom kommunik-ation.	För att visa andra medlemmar hur systemet fungerar.	Inte under projekt men ibland när det kommer nya resurser.

En faktor där dokumentation spelar roll är vid kunskapsspridning (E. Rubin & H. Rubin, 2011). Alla dokument ska hjälpa utvecklarna och förstå ett system så att de i sin tur kan utföra sina uppgifter på ett bättre och mer effektivt sätt. Dokumentationen kan användas för att utbilda nya medlemmar i projektet. Med hjälp av dokumentation kan projektmedlemmar skapa förståelse för mål, tekniska lösningar samt se var i koden ändringar har genomförts (Garousi et al, 2014). Enligt Layman et al (2006) kan för lite dokumentation leda till att kommunikationen blir bristfällig då det blir svårare att förklara systemet på ett sätt så att kunden förstår.

Samtliga informanter använder sin dokumentation för att sprida kunskap internt samt mot kund. Mot kunden används den för att lättare kunna förklara systemet och även underlätta kravställandet från kunden. Internt används dokumentation för att nå information om systemet på ett mer effektivt sätt och kan på så vis användas för att utbilda projektmedlemmar, nya som gamla om systemet. En informant berättar att de ändringar som ska göras i koden antagligen ska göras av någon annan än den första utvecklaren. Då behöver den personen förstå vad som gjorts tidigare för att kunna ta över arbetet.

En informant angav att den dokumentation som gjorts tidigare i projektet senare användes för att skapa ett nytt dokument som användes vid kommunikation mot kund.

5.1.4 Förvaltning

Tabell 5.4 Analys av informanterna svar gällande förvaltning

	Konsult-företag med fokus på utveckling	**Världsledande konsult-företag inom IT**	**Konsult-företag inom IT och verksamhetsutveckling**	**Utvecklings-företag**	**Konsult-företag inom design och konstruktion av webbportaler**	**Konsult-företag med fokus på applikations-säkerhet**	**Konsult-företag inom business intelligence**
Arbetar med förvaltning	Ja	Ja	Ja	Ja	Nej, endast supporten.	Ja	Ja
Användare av dokumentation	Främst av kund vid övertag av system.	Intern användning	Intern användning	Intern användning	Intern användning	Främst för kund	Intern användning
Fördelen med dokumentation	Blir lättare för kund att ta över projekten.	Arbetet med förvaltningen underlättas.	Minimerar risker som kan förknippas med underhåll.	Minimerar den tekniska skulden.	Lättare att hitta när något gått fel i systemet.	Ser inga fördelar.	Lättare att se vilka principer som använts så allt harmonierar.
Påverkan på tid och kostnad	Positiv	Positiv	Positiv	-	Positiv	-	Positiv

Experter inom mjukvaruutveckling menar på att underhållskostnaden är 75-80 % av projektets totala kostnad (Brandt, 2005). SouarIssa och Stenlund (2012) har funnit att vanligt förekommande problem inom systemförvaltning är obefintlig eller bristande systemdokumentation, komplex och ostrukturerad kod, svårigheter att förstå användarnas förväntan och brist på erfarenhet när systemförvaltare lämnar projektgruppen.
Dock visar en undersökning av Garousi et al (2014) att teknisk dokumentation används som mest som en informationskälla under utvecklingsprojektet och minst under systemförvaltning.

Alla informanter i studien arbetar med någon form av förvaltning eller support. Majoriteten av företagen använder vid förvaltning den dokumentation som finns tillgänglig vid systemunderhåll. Endast två av informanterna angav att något annat medel för systemförståelse användes vid förvaltning. En av de som inte använde sig av dokumentationen angav att dem jobbar aktivt med intern kunskapsspridning och den andra angav att vid underhåll så är det effektivaste sättet att se hur ett system fungerar genom att gå in och prova sig fram. Informanten som angav att kunskapsspridning är mest effektivt vid systemförvaltning menar dock att dokumentation som finns om ett kundprojekt är viktig att ha om kunden helt vill ta över arbetet med systemet, då menar informanten att det inte ska kräva att de som företag ska behöva konsultera under en längre tid innan kunden kan ta över sitt egna system.

Resterande informanter belyser dock fördelar med att använda sig av dokumentation vid systemförvaltning. Överlag används dokumentation vid förvaltning för att se ett systems övergripande struktur vilket leder till att det som uppdateras eller läggs till i systemet över stämmer med resten av systemet vilket leder till att utvecklarna inte behöver uppfinna hjulet på nytt. De manar även på att tiden som läggs på förvaltning leder till att tiden som läggs på underhåll samt kostnaden minimerar då utvecklarna lättare kan sätta sig in i hur ett system fungerar om dokumentation finns.

5.1.5 Dokumentationskvalitet

Tabell 5.5 Analys av informanternas svar gällande dokumentationskvalitet

	Konsult-företag med fokus på utveckling	**Världsledande konsult-företag inom IT**	**Konsult-företag inom IT och verksamhetsutveckling**	**Utvecklings-företag**	**Konsult-företag inom design och konstruktion av webbportaler**	**Konsult-företag med fokus på applikations-säkerhet**	**Konsult-företag inom business intelligence**
Vem dokument-ationen överlag mest är till för	Kunden	Kunden	Interna resurser och kunden.	Interna resurser	Interna resurser	Kunden	Interna resurser och kunder.
Dokumentations-krav	Ja	-	Nej	Nej	Ja	Kunden avgör	-
Riktlinjer	Ja	Ja	Nej	Nej	Ja	Kunden avgör	Nej
Tid som läggs på dokumentation	Ca 10 % av projekttiden.	Löpande	Löpande	0 – 5 minuter per dag.	Ca 10 % av projekttiden.	Kunden avgör	Ca 4 - 5 % av projekttiden.
Tillgänglighet	Sparas i repositorys och interna wikis.	I ett speciellt system.	-	I ett speciellt system.	Listor på intranätet.	-	-

De kvalitetsattribut som är mest förekommande inom forskning gällande dokumentation inom mjukvaruutveckling är att den dokumentation som tas fram ska vara 1) lättillgänglig 2) komplett 3) konsekvent (Zhi et al, 2014). Zhi (2014) menar att kvalitetskriterierna influeras av hur mycket tid och arbete som läggs på dokumentation.

Gällande tiden som läggs på dokumentation så varierar den mellan de olika företagen i studien. Det vanligaste är att dokumentation utgör ca 10 % av den totala projekttiden eller att dokumentationen sker löpande under projektets gång. En minoritet av företagen har krav på dokumentation men hos majoriteten av företagen finns det riktlinjer för hur dokumentationen ska vara utformad. Det vanligaste är att företaget själv formar riktlinjerna för hur dokumentationen ska se ut men det förekommer även att kunden har krav på utformningen av dokumentationen, detta anser vi beror på att dokumentationen till största del utformas åt kunderna. Vad gäller tillgänglighet så lagrar de flesta av företagen sin dokumentation i någon typ av system eller intranät så att alla medarbetare har tillgång till det.

5.1.6 Kommunikation

Tabell 5.6 Analys av informanternas svar gällande kommunikation

	Konsult-företag med fokus på utveckling	**Världsledande konsultföreta g inom IT**	**Konsult-företag inom IT och verksamhetsu tveckling**	**Utvecklings-företag**	**Konsult-företag inom design och konstruktion av webbportaler**	**Konsult-företag med fokus på applikations-säkerhet**	**Konsult-företag inom business intelligence**
Sker kommunikation	Ja	Ja	Ja	Ja	Nej	Ja	Ja
När används kommunikation	Kommunikat-ion sker mot kund.	Internt och vid internationella projekt.	Inom team Mot kund varje vecka och varannan mot produktägare.	Tät kommunika-tion mellan utvecklarna.	Använder inte dokument-ation för kommunik-ation.	Mot kund	Internt och mot kund.
Kommunikations sätt	Sprintmöten	Scrum-möten internt. Daily stand-ups Internationellt Skype, chat eller telefon.	Stand ups internt.	Stand-ups	-	Stand-ups	Face-to-face internt. Mot kund med specifikationer.
Dokumentation vid kommunikation	För att kunden ska kunna kommunicera med andra aktörer kring systemet.	Till exempel mock-ups till kund för att förklara funktionalitet. För att stödja kommunika-tionen kring krav.	För att kunna kommunicera kring systemet med kunden.	-	Stödja kommunika-tion med kund.	För att underlätta kommunika-tionen med kunden	Använder inte dokument-ation för att stödja dokument-ation.

För att kunderna ska förstå hur utvecklingsprojektet går så sker kommunikation mellan projektteamet och kunderna (Layman et al, 2006). Samtliga informanter utom en använder sin dokumentation för att främja kommunikation mot kund i de agila projekten. Det skiljer sig dock vilken typ av kommunikation som informanterna använder sig av.

Kommunikation kan delas in i direkt och indirekt. Direkt kommunikation kan till exempel omfatta face-to-face, telefon eller annan media. Indirekt kommunikation involverar det som skrivs såsom dokument, email eller böcker (Hummel et al, 2015). En informant betonar vikten av att använda dokumentationen i specifikationerna i internationella projekt där kommunikation ofta kan vara en utmaning och kommunikationen är indirekt. Övriga informanter använder inte sin dokumentation för att stödja den interna kommunikationen.

Korta iterationscykler ger den flexibilitet som behövs för att tillgodose förändringar i agila processer (Turk et al, 2014). Ahimbisibwe (2014) listar 37 kritiska framgångsfaktorer för ett mjukvaruutvecklingsprojekt. Dessa faktorer har identifierats eftersom de anses ha betydelse för ett projekts framgång. Av dessa hamnar kommunikation på tionde plats (Ahimbisibwe et al, 2014).

Gemensamt för de flesta informanter och där vi ser ett mönster är att dokumentationen används för att underlätta kommunikationen kring systemet mot kunden. Dokumentationen av systemet gör så att kommunikationen med kunden underlättas då den ger en bild av systemet så att det lättare går att förklara för kunden vad som har utvecklats. Därmed kan kunden ta bättre beslut i form av krav. Det är endast en av informanterna som använder dokumentationen för att stödja kommunikationen internt inom projekten och det är ett större företag där projektteamet inte sitter på samma plats och arbetet ibland är internationellt. Majoriteten använder sig inte av dokumentation för att stödja den interna kommunikationen utan den sker genom olika direkta kommunikationsformer.

5.2 Diskussion

Vi har sett att det råder skillnader mellan de agila och traditionella utvecklingsmetoderna, inte minst vad gäller synen på dokumentation. Medan de traditionella utvecklingsmetoderna förespråkar och använder dokumentation flitigt, är det något som inte förespråkas inom de agila metoderna. Trots att dokumentation inte förespråkas inom agila metoder råder det motsättningar inom agila diskussioner gällande vilken dokumentationsmängd som anses lämplig. Vi har genom vår undersökning kommit fram till att dokumentation kan underlätta olika aspekter inom agila utvecklingsprojekt.

Inom agil utveckling förespråkas flexibilitet då utveckling av system sker i en ständigt förändrande miljö. Bland de stora företagen var det tre av fyra som ansåg att flexibiliteten ökade med rätt mängd dokumentation. Dessa menar att projektmedlemmar genom dokumentation lättare kan se vilka ramar dem ska röra sig inom. Anledningen till att dokumentation ökar flexibiliteten i stora företag är att de inte arbetar lika nära varandra, ibland internationellt, samt att det är fler personer som arbetar med ett visst projekt. Det leder till att det genom dokumentation blir lättare att kommunicera ut information till nya personer i ett projekt då det finns dokumentation kring tidigare lösningar i systemet. En viktig aspekt gällande dokumentation i relation till flexibilitet är att den inte får bli allt för omfattande, dokumentationen ska således inte vara detaljerade beskrivningar över allt i ett system utan dokumentation ska omfatta övergripande struktur.

Utifrån vår studie kan vi se att produktiviteten ökar vid intern användning av dokumentation. Informanterna menar på att dokumentation av både arkitektur och kod stödjer arbetet med att utbilda nya utvecklare i ett projekt. Genom dokumentation får de snabbare en överblick av funktioner i systemet, det blir lättare att hitta information och på så sätt effektiviseras den processer som innebär att leta information i tidigare utvecklad kod. Informanter anger även att dokumentation av systemet kan bidra till att tid inte går förlorad på att sätta sig in i koden.

De flesta av informanterna menar på att det agila arbetet blir mer effektivt med dokumentation. Flera av informanterna betonar dock vikten av att inte ha för mycket dokumentation då den snabbt blir förlegad samt är tidskrävande att utforma. Blir dokumentationen för omfattande och förlegad försvinner dokumentationens positiva påverkan på produktiviteten inom projekten och blir istället en börda. Vår studie visar alltså att produktiviteten kan öka med hjälp av dokumentation av systemet. Både för utvecklarna inom projektet samt nya medlemmar.

Av informanterna svarade fler än hälften att de använde dokumentation kring systemet för att underlätta kommunikationen med kund. Bland annat för att kunna förklara hur systemet byggs upp samt för att kunna kommunicera så att alla förstår kravförändringar. Ingen av informanterna använder sin dokumentation till kommunikationen inom projektteamen. För att stödja intern kommunikation används till exempel Scrum-möten, daily stand-ups eller face-to-

face-kommunikation. Vi drar därför slutsatsen att dokumentationen kan underlätta och stödja kommunikationen med kunder men att den inte är avsedd att stödja den interna kommunikationen inom projekten.

En faktor som är kopplad till kommunikation är kunskapsspridning. Eftersom nära kontakt samt kravförändringar förespråkas inom agila metoder behövs kommunikation och kunskapsspridning mot kund. Kunskapsspridning används således inom utvecklingsprojekt för att stödja arbetet och sprida information kring systemet inom projekten. En större del av informanterna påvisar att dokumentation kan bidra till kunskapsspridning. Dokumentation användes av fler än hälften av informanterna för att underlätta och stödja kunskapsspridning och kommunikation mot kunden i projekten. Dokumentation som är av störst vikt för kunskapsspridningen är dokumentation gällande kod och arkitektur. Om denna information finns dokumenterad blir det enligt informanterna lättare att ta in nya medlemmar i projekt och utbilda dem kring systemet. Större delen av informanterna använder dokumentation för att visa hur systemet fungerar för övriga projektmedlemmar inklusive nya medlemmar som ska ta över ett påbörjat arbete. Vi anser att detta visar att dokumentation bidrar positivt och stödjer arbetet med kunskapsspridning både inom projektet men även mot kund.

Mer intern användning av dokumentation sker vid systemförvaltning och support. Trots att det råder små skillnader vid användandet av dokumentation vid systemförvaltning kan vi se ett mönster bland informanternas företag där majoriteten av de företag som har kontor på olika orter anser att dokumentationen är användbar vid förvaltning. Vi kan därför dra slutsatsen att på företag med kontor på olika platser i landet eller världen i större utsträckning internt använder dokumentation vid förvaltning. Majoriteten av företagen ansåg, i linje med teorier, att dokumentation har en positiv inverkan på kostnad och tid som läggs på underhåll. Ett av företagen som angav att de inte använder dokumentation vid förvaltning är ett mindre utvecklingsföretag som har kontor på endast ett ställe varav det andra företaget som angav att de inte använder dokumentation vid förvaltning har kontor på sex platser i landet. Det mindre företaget med endast ett kontor jobbar aktivt med intern kunskapsspridning och vi anser att detta är en faktor som spelar in för användandet av dokumentation vid förvaltning. En gemensam nämnare för de företag som inte ansåg att dokumentation var lika viktigt som resterande informanter är att de lägger stor vikt på att koden ska vara så tydlig som möjligt. Dock ser båda informanterna på de företagen fördelen som dokumentationen bidrar med till kunderna under förvaltningen av systemet.

Vi har sett att omfattande dokumentationen inte uppdateras som den ska vilket leder till att den blir förlegad och opålitlig. Dokumentationen får då en kontraproduktiv effekt på utvecklingsarbetet. Vad gäller dokumentationsvolymen är det skillnad mellan företag som tillhandahåller en grundprodukt med tillägg till sina kunder och de som utformar helt nya system till kunderna. Hos de företag där unika system tas fram till kunden är det i större utsträckning kunden som avgör hur mycket dokumentation som ska utformas samt vilken typ. Den interna dokumentationen varierar dock inte beroende på vilken typ av produkt som utformas utan det finns då riktlinjer för hur dokumentationen ska vara utformad. Trots att

många informanter anger att det inte finns något uttalat krav om dokumentation ser vi utifrån empirin att det dokumentationen bidrar till är att olika delar av utvecklingsprocessen underlättas och anser utifrån vår studie att medarbetare utgår från sunt förnuft, kommunikation samt avvägning vad gäller dokumentationsmängden för att avgöra när samt hur mycket som ska utformas.

6 Slutsats

I detta kapitel redovisas studiens slutsats. Det teoretiska och empiriska bidraget presenteras följt av förslag till vidare forskning.

6.1 Teoretiskt bidrag

En uppmärksammad diskussion kring agila utvecklingsmetoder är dokumentation. Det har diskuterats hur mycket dokumentation som är nödvändigt, om det är nödvändigt överhuvudtaget. De teorier som vi har studerat visar på olika ställningstaganden inom området. Agila förespråkare menar på att det ofta krävs för mycket dokumentation medan andra pekar på vikten av att behålla information i företaget genom dokumentation.

Efter att ha undersökt forskning kring agila metoder och dokumentation fann vi att det inte fanns en samanställd bild av vad dokumentation kan bidra med i agila projekt. Den forskning vi fann fokuserade endas på ett fåtal faktorer som kan påverkas av dokumentationen men dock inte på en sammanställd bild av vad dokumentation kunde ha för positiv eller negativ påverkan på dessa faktorer.

Vi har i denna studie tagit fram de, inom forskningen, mest förekommande faktorerna och i sex företag samlat empiriska data för att kartlägga hur dessa faktorer påverkas av dokumentation inom agila projekt för att få en samlad bild dokumentationens inverkan på dessa. Vi har genom studien kommit fram till att dokumentation för majoriteten av faktorerna bidrar till både intern- och kundnytta. Den enda av de sex faktorerna som genom dokumentation endast bidrog till kundnytta var kommunikation.

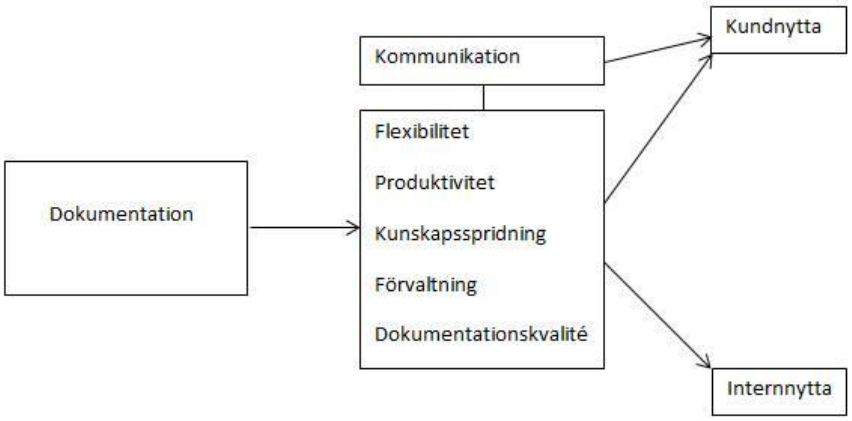
6.2 Empiriskt bidrag

Grunderna för de agila metoderna har inga tydliga riktlinjer för hur dokumentation inom agila projekt ska vara utformad eller vad den ska användas till. Endast att fungerande mjukvara ska går före omfattande dokumentation. De empiriska resultaten i denna studie visar på att dokumentation påverkar mjukvaruutveckling inom agila projekt.

Resultaten i denna studie visar på att informanterna är medvetna om de fördelar som dokumentation kan bidra med. Den skillnad vi ser mellan de agila och traditionella metoderna är att i de traditionella metoderna var dokumentation en inräknad del av processen. I agila metoder är dokumentation inte det vilket leder till att beslut gällande dokumentation ligger på de företag som väljer att arbeta agilt. Dock kan vi med stöd av empirin dra slutsatsen att dokumentation inte fördröjer den agila arbetsprocessen utan snarare stöttar den.

Informanternas svar visar på att företagen använder sig av egna interna regler eller krav från kund för att avgöra mängden dokumentation i ett projekt. Detta anser vi kan vara en bidragande faktor till att företagen i studien ser nyttan av dokumentationen. Den frihet som de agila metoderna lämnar till företagen gällande mängden dokumentation leder således till skillnad från de traditionella metoderna.

Vår slutsats är att majoriteten av företag ställer sig positiva till dokumentation. Vi kan även dra slutsatsen att användning av dokumentation stödjer flexibilitet, produktivitet, dokumentationskvalitet, kunskapsspridning och kommunikation på företag som arbetar agilt. För att få en övergripande och sammanfattande bild över faktorerna och dess påverkan har vi utformat en modell. Modellen visar de centrala faktorer inom agil mjukvaruutveckling som dokumentation kan stödja. Som modellen visar bidrar dokumentation endast till förbättrad kommunikation mot kund, inte internt inom projekten. Utöver det påverkar dokumentation både kund och internnyttan för resterande faktorer.



Figur 6.1 Dokumentationsnyttan inom agila projekt

6.3 Förslag på vidare forskning

Vi föreslår även vidare forskning kring dokumentation utifrån de faktorer som vi tagit fram i denna studie. Vi tror att en vidare och större studie kring dessa faktorer i förhållande till dokumentation skulle kunna leda till ett resultat som ytterligare kan bekräfta och fördjupa insikten om betydelsen av dokumentation inom agila utvecklingsmetoder. Ytterligare förslag till vidare forskning är att genomföra intervjuer på företag som arbetar enligt andra metoder för att få en nyanserad bild av hur dokumentation påverkar dessa faktorer inom utvecklingsprojekt.

7 Bilagor

7.1 Intervjuguide

Q1. Hur mycket tid läggs, per dag, på dokumentation under projektets gång?

Q2. Är era anställda tvungna att dokumentera all programvara?

Q3. Har ni några riktlinjer för hur dokumentationen ska vara utformad?

Q4. Vilken typ av dokumentation är mest värdefull; teknisk eller arkitekturspecifik?

Q5. Vem är dokumentationen främst till för?

Q6. Anser ni att flexibiliteten i projektet minskar eller ökar på grund av dokumentationen?

Q7. Anser ni att produktiviteten i projektet minskar eller ökar på grund av dokumentationen?

Q8. Hur mycket lättare blir det att genomföra förändringar i systemet när det finns noggrann dokumentation?

Q9. Hur mycket tid lägger ni uppskattningsvis på underhåll av system som levererats till kund?

Q10. Hur påverkar god dokumentation underhållskostnaden för ett system? Hur påverkar god dokumentation tiden som måste läggas på underhåll?

Q11. Tycker ni att man ser för lätt på dokumentation inom agila metoder?

7.2 Transkribering informant 1

Hur mycket tid lägger ni på dokumentation under ett projekt?

Dokumentation sker löpande, så om man skriver en metod eller man har något lösenord eller något som behövs skrivas ner så skrivs det ner. Om det är så att det inte behöver skrivas ner så skrivs det inte ner. Vårt mål är att koden ska vara så läsbar som möjligt. Så det ska inte dokumenteras mer än nödvändigt, det ska inte blir bubbelarbete. Men vi vet också att en kommentar här och där i koden är bra. Snarare "här börjar koden för detta än det här" än det här och det här gör den här koden. För om man säger det här gör den här koden blir det förlegat inom bara några veckor i ett nyutvecklingsprojekt för att saker hamnar där och förändras eller kraven förändras så då finns det risk att kommentarer inte följer med, men om du kommenterar att här börjar den här och den här koden så har man den typen av dokumentation. Då blir det lättare att hitta och läsa koden för du behöver inte läsa alla kod och det blir inte förlegat lika snabbt. Men absolut det dokumenteras. Sen har vi en wiki för varje projekt både i det vi kallar repository där vi har all kod och versionshantering. Där skriver vi ner till exempel om vi har någon testmiljö eller något eller en produktionsmiljö som det ska publiceras till. Så har vi lösenord och alla inloggningar och så där och ser till att det finns närvarande och även information om hur man kommer in på … vi hostar ju det mesta i Acher tillexempel och hur man kommer in på Acher och vi ställer in vad man har för möjligheter där.

Är era anställda tvungna att dokumentera viss programvara?

Riktlinjerna säger att är det något ni själva känner är krångligt, om koden är krånglig fundera på om du ska skriva om den om den är gjord på ett så bra sätt som möjligt eller det inte finns tid att skriva om den så dokumentera den. Är det inloggningsuppgifter eller hur man sätter upp en ny utvecklingsmiljö till exempel om man får in en nyutveckling av någonting då måste man göra det direkt. Och det gör man alltid och oftast är det så att vi börjar utveckla på något, säg att vi sätter upp en utvecklingsmiljö, och så inser vi det här fungerar och så sätter vi upp nästa utveckling, då sätter vi upp resterande utvecklingsmiljöer, då dokumenterar vi dem vi sätter upp där och sen får man förändra i det dokumentet löpande. Så vi har en wiki internt och vi har en wiki för projektet. Så vi har två olika wikis där man kan skriva. Men det finns inget tvång att det här måste du dokumentera men ibland säger jag: "Dokumentera!".

Så det finns inte några riktlinjer för hur dokumentationen ska vara utformad?

Det finns riktlinjer att dokumentera om det är så att koden är svår. Tänk: "Hur skulle en utvecklare som kommer in i den här koden se den?", är den krånglig? Så det försöker vi föra med oss i alla delar. Men det är också det att det ska ju skrivas så pass lättläst kod så att det inte ska behöva dokumenteras. Om det inte gör det då kodar vi fel. Så vi gör ju code reviews regelbundet för att se till att koden är läsbar och att den håller samma struktur och format och istället för att göra speciallösningar så försöker vi standardisera så mycket som möjligt för att det inte ska behöva dokumenteras för mycket. Men det är absolut inte så att … Alltså många tror ju att om man jobbar agilt behöver man inte dokumentera, och det är inte en sanning för

man måste dokumentera och många tror att man kommer undan det. Men i alla projekt så lägger vi ungefär 10 % av projekttiden eller estimatet läggs till dokumentation.

Brukar dokumentation vara mest teknisk eller arkitekturspecifik?

Teknisk som i koden, eller så är det en dokumentation kring övergripande varför vi har gjort som vi har gjort och arkitekturen också. Men arkitekturen ser jag ändå som ganska teknisk. Sen så har vi ju, vi utgår från en bas när vi kodar och den använder vi i dem flesta projekt som vi gör. Den tar vi med oss för att effektivisera utvecklingen. Den är dokumenterad var för sig så den arkitekturen är dokumenterad och allt som är specifikt för ett projekt dokumenteras separat. Inget är standard och allting förändras lite så måste man dokumentera det.

Vem är dokumentationen främst till för?

Vi är ju väldigt öppna med koden, kunden äger ju alltid koden vi gör. Vi äger ju aldrig produkten. Så kunden ska kunna gå till vilken aktör som helst och bara flytta sin kod och då ska ju nästa aktör förstå sig på den, det ska inte kräva att vi konsulterar i två år för att dem ska kunna utveckla i koden utan det ska gå att flytta vidare. Vi riktar ju oss mot företag som inte har någon egen IT-avdelning som inte sköter egen utveckling, så små till medelstora företag. Det kan hända att dem växer och helt plötsligt ska dem skaffa en egen IT-avdelning och då måste dem kunna supporta eller hantera det själva och då ska dem kunna ta över.

Tycker du att produktiviteten och flexibiliteten ökar eller minskar på grund av dokumentation?

Med rätt mängd så ökar flexibiliteten för det blir lättare att ta in nya resurser. Det blir lättare att förändra projektets struktur. Men gör man det för mycket blir det plottrigt i koden och det blir ofta förlegat väldigt snabbt och det gör att det blir svårare att jobba med det och det tar tid. Säg att vi skulle lägga 20 % istället på ett projekt på 100 timmar så säg att en 1000 lapp då, det är lite högt räknar, per projekt, 100 000 kronor så lägger du på 10 % då är det 10 000 extra, men om du lägger på 20 - 25 % så är det 20 - 25 000 extra. Så jag menar det är ganska stora skillnader i kostnader och det är ju sällan dem, om man då säger ett projekt på en miljon då har du väldigt stora omkostnader bara för dokumentation. Man ska vara försiktig med dokumentation för det kan vara så att folk tycker att det är alldeles för viktigt men det är viktigt. Men det får inte ta över, det får inte vara en större del i själva utvecklingen.

Tycker du att det blir lättare blir det att genomföra förändringar i systemet när det finns noggrann dokumentation om det?

Ja, ibland. Men som sagt, jag har flera gånger varit med om att man måste göra förändringar och att dokumentationen är förlegad, att den inte är uppdaterad. Eller att den är skriven på ett sådant sätt att det inte går att hålla den uppdaterad hela tiden utan att det blir för mycket arbete. Och då är det bättre att testa sig fram.

Påverkar det på något sätt, när ni underhåller systemet för kund, om man tänker på underhållskostnaden där, gäller samma sak där att det inte spelar någon roll om det finns dokumentation eller inte utan det beror lite på?

Nej, i våra systemunderhållsfall så har vi ju kunskapsspridning som är både genom dokumentation och genom vi när vi lämnar över så tar vi in en resurs som har varit med i projektet som hjälper till att föra över och kanske sitter med den resursen som ska ta över för systemvård, vi har ju en systemvårdsdel. Så överföringen av kunskapen där sker både genom dokumentation, mest när man ska deploya eller publicera eller om man ska hämta hem koden, och få upp en utvecklingsmiljö. Sådana grejer är det oftast, sen står det i koden var i koden saker händer. Men det står inte specifikt på varje metod. Sen är det bra att dokumentera varför vi har gjort applikationen. Vad syftet är med den och vad det är vi försöker uppnå så att det inte blir så att den som ska ta över applikationen inte har en förståelse för affärsnyttan i lösningen. Om man inte förstår den så kan man sälla utveckla mer och skapa bättre nytta åt kunden.

Påverkar dokumentationen på något sätt tiden ni lägger på underhåll av system?

Påverkar absolut underhållet av system, om man har dålig dokumentation eller dålig struktur i koden så kommer det ta längre tid att underhålla. Vi tar ju både över våra andras applikationer och egna applikationer. Och genom att arbeta med andras och även våra egna applikationer har vi lärt oss hur man inte ska göra. Vi är inte felfria utan vi har haft applikationer, historiskt som inte har vart i den kvalitet den bör vara. Man lär sig av sina och andras misstag när man tar över ett projekt.

Hur mycket tid brukar ni lägga på underhåll av system som ni levererar?

Den gyllene regeln är att om det kosta en krona i utveckling så kostar det tre kronor i förvaltning. Så vi brukar alltid utgå från det med kunden från början; att om du lägger 500 000 kronor när du kommer så kommer du på sikt kosta två miljoner totalt innan det är dags för applikationen att försvinna. Och det är minimum liksom. Vi vet inte, vi har inte hunnit testa så många applikationer så att vi vet att det till 100 % stämmer. Det är i alla fall det forskningen säger och det är det vi försöker jobba utifrån.
Mycket är ju att när man skapar ett system så hittar man funktionalitet och man gör exakt det kunden vill att man ska göra men sen är det små fix som gör att projektet tar längre tid än vad man tror. Har man 3000 sådana förändringar över fem år så blir det ganska stora kostnader.

Som jag fattar det så är dokumentation inte det huvudsakliga.

Nej, har du rätt kompetens på systemvården så ska det inte ta mer tid utan det är klart att det tar tid att få upp en utvecklingsmiljö men då snackar vi en två dagar innan individerna är igång på riktigt och utvecklar. Man har fyra till fem timmar på sig att få upp miljön med databaser och allt och några timmar att hitta i koden. Men i och med att vi kör så lik struktur i alla våra projekt så ska det vara bra liksom. Sen kommer ju den strukturen ändras över åren, men då kompetensutvecklar vi vår personal mot det.

Så dokumentation är inte så viktig för den blir förlegad men det är fortfarande viktigt att ha en viss kunskap kvar om koden?
Ja, absolut!

Tycker du att det inom litteraturen är för lite fokus på dokumentation?
Ja, jag tror att uppfattningen om att det skulle vara gratis med Scrum eller i den agila utvecklingen att det inte skulle behövas dokumentation det är en myt som måste bort för man måste lära folk att från början dokumentera. Jag lärde mig den hårda vägen för jag gick tillbaka och läste min kod fyra veckor senare och frågade mig; vad gör den här koden? Eller om man är trött så kan det ta en dag innan man förstår vad koden gör. Så dokumentationen behövs, speciellt vid specialfall om koden inte är läsbar på ett enkelt sätt men då ska man också fundera över sin kod en extra gång.

7.3 Transkribering informant 2

Hur mycket tid läggs per dag på dokumentation under ett projekt?
I projekt som jag jobbar i har vi dedikerade personer som skriver dokumentation, det vill säga användardokumentation. Men just dokumentation av kod gör vi inte mycket av. Helst vill man förstå koden utan att behöva skriva dokumentation kring det.

Är alla era anställda tvungna att dokumentera programvara?
Nej, varje utvecklare måste skriva koden så att andra förstår den.

Har ni några särskilda riktlinjer för hur dokumentation ska vara utformad?
Den är ganska uppstyrd av hur det ska se ut, i vilket format det ska vara, vilka ordval man väljer och sådana saker. Men det finns inte för själva koden.

Vilken typ av dokumentation är mest värdefull; teknisk eller arkitekturspecifik?
Arkitekturspecifik.

Vem är dokumentationen främst till för?
Våra kunder är ju de som använder produkten och då är det ju viktigast att den dokumentationen som kunder använder att den är bra.

Anser ni att flexibiliteten i projektet ökar eller minskar på grund av dokumentation?
Man ska ju lägga till dokumentation där det behövs. Man ska inte hålla på att dokumentera varenda rad för den är tänkt att vara skriven på ett sådant sätt att man ska förstå den. Men om det är komplicerade delar då behöver man ju dokumentera och normalt sätt måste man dokumentera själva koden i sig. Men man kan skriva lite kommentarer på ställen där det är lite extra komplicerat så att det inte ska ta så lång tid att förstå. Håller man på med agil utveckling är det meningen att alla ska kunna all kod, det är inte så att en person har ansvar för en viss del, därför måste man skriva koden tydligt.

Anser ni att produktiviteten i projektet ökar eller minskar på grund av dokumentation?
Om man dokumenterar på rätt sätt ökar produktiviteten. Det är det man försöker göra, man dokumenterar det som behövs för att få högre produktivitet.

Hur mycket lättare blir det att genomföra förändringar i systemet när det finns god dokumentation?
Det är antagligen lättare. Dokumentation har ju också att göra med att andra människor ska förstå det man har gjort. Om det är flera personer som jobbar med kod då behöver man ju dokumentera. Och antagligen är det inte samma person som ska göra alla ändringar och därför behöver man ha det dokumenterat på rätt nivå.

Hur mycket tid lägger ni ungefär på underhåll av system som har levererats till kund?

En tredjedel kanske. Det är en produkt som används. Vi underhåller och bygger ut det hela tiden. Jag vet inte vad som menas med underhåll om det är rättning av buggar eller uppdateringar av features för att i så fall är det 100 %. Vi har ju ett system och det är det som det bygger på.

Hur påverkar god dokumentation underhållskostnaden för ett system?

Kostnaden blir mindre antar jag.

Hur påverkar god dokumentation tiden som måste läggas på underhåll?

Normalt sätt påverkas det inte så mycket för att de system vi har just nu ligger det fräscht i minnet. I början är det ingen mening att göra dokumentation för att då rör det sig otroligt mycket. Det är klart man behöver dokumentera men det kan man inte göra förrän det är ganska stabilt. När man skriver en hundra rader lång funktion i en klass då skriver man givetvis dokumentationen direkt för att man ska förstå vad den gör för någonting. Om produkten är gammal kan det vara bra att ha dokumentation för då har det säkert kommit nya personer som inte har jobbat med koden alls. Då tar det lång tid att förstå den om man inte redan har den dokumenterad.

Tycker ni att man ser för lätt på dokumentation inom agila metoder?

Nej, det tycker jag inte. När vi jobbar agilt behöver vi inte så mycket dokumentation eftersom vi jobbar så nära kunden.

7.4 Transkribering informant 3

Hur mycket tid läggs ungefär per dag på dokumentation under ett projekt?
Det är svårt att säga per dag. Det är mer när saker förändras som kräver dokumentation. Det är då man dokumenterar. När man slutför uppgiften brukar man dokumentera en del som man kan visa för kund. Det blir varannan dag eller liknande. Det kan vara så att man grupperar ett par olika uppgifter och dokumenterar dem tillsammans. I och med att vi jobbar agilt så blir det max en dag per uppgift.

Är alla era anställda tvungna att dokumentera all programvara?
Nej, vi har inga sådana direkta krav. Det är mer generella riktlinjer för projekten.

Har ni några särskilda riktlinjer för hur dokumentation ska vara utformad?
Vi har en viss mall som vi utgår från. Sedan jobbar vi vidare med dokumentationen under projektets gång. Så det finns inte någon direkt riktlinje.

Vilken typ av dokumentation anser ni vara mest värdefull?
All typ av dokumentation är viktig. De tekniska bitarna kanske är viktiga för oss som utvecklare men kunden kanske inte bryr sig om dem överhuvudtaget. De systemtekniska kanske är viktiga för de som driftar systemet, medan den mer generella dokumentationen om hur systemet är tänkt att fungera är viktig för kunden. Olika typer av dokumentation är viktig för olika ändamål.

Vem är dokumentationen främst till för?
Det är både för oss och kunden. Dels för att vi ska ha koll på vad vi har gjort och hur saker fungerar, men även för att kunden ska hänga med på vissa bitar. Men kunden är inte intresserad av den tekniska dokumentationen direkt. Den biten är främst för vår del.

Anser ni att flexibiliteten i projektet minskar eller ökar på grund av dokumentation?
Varken eller. Det ska fungera smidigt, det är en del i processen liksom. Utan dokumentation hade det blivit svårare att driva projektet.

Anser ni att produktiviteten i projektet minskar eller ökar på grund av dokumentation?
På kort sikt minskar det. I och med att man inte kan producera lika mycket när man gör det här. Men på lång sikt så blir det mer effektivt i och med att man kanske inte sätter sig in i områdena på nytt igen utan man har dokumenterat hur saker och ting fungerar.

Hur mycket lättare blir det att genomföra förändringar i systemet när det finns noggrann dokumentation?
Det blir betydligt enklare. Man kan ju spara veckor när man ska göra vissa saker. När man förändrar en viss sak kanske man har glömt att det finns en annan funktion som är beroende av den och då kanske man förstör den om man inte hade vetat om det. Det är sådant man ska kunna läsa sig till innan man börjar ändra saker. På så sätt jobbar vi med att minimera riskerna. Det är viktigt.

Hur mycket tid lägger ni ungefär på underhåll av system som levererats till kund?
Just nu jobbar vi 50 % med underhåll och 50 % med vidareutveckling. Men det beror helt och hållet på system till system. Det jag jobbar med nu är under utvecklingsfasen fortfarande så att vi håller fortfarande på och vidareutvecklar mycket av det så det är egentligen inte så mycket underhåll av det. Medan system som har varit i drift länge kommer såklart ha mer underhåll och mindre vidareutveckling. Det kan nog variera rejält.

Hur påverkar god dokumentation underhållskostnaden för ett system?
Det påverkas positivt. Det minskar kostnader, dels att man inte behöver sätta sig in i vissa saker på nytt men även att man kan förstå hur vissa saker fungerar och underhålla på rätt sätt så att man följer strukturen som finns sedan tidigare. Exempelvis om man får in en ny medarbetare så är det viktigt att den personen kan sätta sig in hur det fungerar.

Hur påverkar god dokumentation tiden som måste läggas på underhåll?
Tiden som krävs för ett visst underhåll minskar i och med att man behöver sätta sig in i saker på nytt. Men hur mycket kan jag inte riktigt svara på.

Är det något mer du vill tillägga?
Det jag skulle vilja trycka på är att olika typer av dokumentation som är till för olika ändamål. Det är ju helt olika personer som ska läsa dokumentationen. Jag kan ju dokumentera koden men det är något som kunden aldrig kommer att se. Utan det är för vår skull. Det finns tre-fyra olika typer av dokumentation. Kod, systemteknisk, övergripande kravdokumentation kring systemet som är mer riktad mot kunden, processbitarna, work prospective till exempel. Det är mer att separera de olika typerna av dokumentation kan vara intressant.

7.5 Transkribering informant 4

Respondent 4 har bett om att transkriberingen av intervjun inte ska bifogas som bilaga. Av forskningsetiska själ finns därför respondent fyras transkribering inte bifogat i detta arbete.

7.6 Transkribering informant 5

Kan du berätta lite kort hur länge ni har arbetat agilt och hur er agila metod ser ut?

Jag vet inte hur länge den har funnits i företaget, men jag tror att det är sedan sex år. Vi använder en blandning av Scrum med lite Kanban. Det är inte alla projekt som kör den, det beror på hur stort projektet är. Om det är en person som gör lite konsultarbete åt en kund är det inte så mycket agilt fokus. Sen har vi en större produkt som vi arbetar med kontinuerligt och då används den agila metoden. Vi har sprintar på tre veckor med en färdig backlog i början på sprinten som då arbetas av i tasks.

Hur mycket tid lägger ni i genomsnitt per dag på dokumentation?

Per dag är svårt att säga. Som jag har suttit så har jag suttit kontinuerligt i stort projekt. Jag skulle säga att det är runt 10 %.

Är ni tvungna att dokumentera all programvara som ni gör under projektet?

Ja, det måste vi göra. Det är ingenting som toppbestäms utan det är vi i teamet som bestämmer och vi har kommit fram till att vi måste göra det.

Har ni några riktlinjer för hur dokumentationen ska vara utformad?

Ja, det har vi. Vi har listor på vårt intranät där vi skriver dokumentation. Vi fyller då i för vilken kund det gäller, vad det är för förändring, i vilket kundprojekt det ligger, och när det gjordes. Det måste ske när man slår ihop koden till ett visst ställe.

Vilken typ av dokumentation är mest värdefull; teknisk eller arkitekturspecifik?

Det viktigaste för oss är att hålla koll på vilka kodändringar som görs. Vi har en produkt som finns hos flera kunder, men kunder har lite olika kodpaket. Det viktigaste är till exempel vilken sammanlagd kodmängd som en viss kund har och om de senaste ändringarna redan finns hos kund.

Har ni då en grundkod som ni utgår ifrån för varje kund? Och sen har ni de specifika sakerna som ni dokumenterar för varje kund?

Ja det är både och. Dels har vi vår grundplattform där vi dokumenterar allt. Varje gång vi gör en ändring i koden dokumenteras det. Att från det här datumet bör finnas med i framtida releaser för alla kunder. Då säger vi att den här koden finns i releasepaket. Men sen har vi också kundspecifika paket. Då dokumenterar vi för vilka kunder som kodändringen gäller. På så sätt kan vi filtrera på hela projektet eller bara på kunder. Vi har inte dokumentation av själva grundprodukten. I nuläget dokumenterar vi bara ändringar i den. Det hade säkert varit bra att ha allt dokumenterat men vi gjorde inte det från början. Det är något som kommit med tiden.

Vad ser du för nackdelar med att inte ha grundprodukten dokumenterad?
Det är jättesvårt att sätta in nya utvecklare i det. För oss är det självklart, men för en ny blir det svårt eftersom det inte står någonstans hur man gör.

Är dokumentation främst till för nyanställda?
Nej. Den är till för oss att hålla reda på vilken kod som finns för kund. Det finns många som sitter i teamen och gör egen dokumentation för vad som görs för att komma ihåg till nästa gång. Men vi har ganska lite dokumentation av hur man gör saker. Det hade jag önskat att vi hade mer av.

Vad hade det varit bra för anser du?
För att kunna sprida kunskap av hur man gör snarare än vilka funktioner som finns.

Anser du att flexibiliteten i projektet minskar eller ökar med dokumentationen?
Varken eller tror jag. Flexibiliteten ligger ju i sprintarna. Man kan snabbt bestämma vad som ska göras. Alltså att det inte behöver vara vattenfallsmetoden, att vi blir klara med det vi gör om tre veckor och inte ett år. Det är där flexibiliteten ligger tycker jag. Och den förhindras inte av dokumentationen tycker jag eller blir inte bättre av det heller.

Tycker du att produktiviteten minskar eller ökar på grund av dokumentationen?
Initialt så minskar den för att man måste lägga ner tid på att komma ihåg den. I längden ökar den för att man inte behöver gå tillbaka och leta så mycket. Så länge den är välgjord kan man hitta saker enklare.

Blir det lättare att genomföra förändringar i ert system när det finns noggrann dokumentation?
Mycket. Det är mycket lättare att buggsöka också. Det blir lättare att hitta vem som gjorde vad och när. Om vi hade haft mer och bättre dokumentation på hur man gör så hade det blivit lättare att införa nya saker.

Hur mycket tid lägger ni på underhåll av system som levererats till kund?
Vi lägger ganska lite i underhåll på fungerande kod. Vi gör ingen omfakturering av kod. Däremot har vi en support som stödjer samma kod som teamet gör. De lägger inte alls mycket tid på det. Det är ju en tid av underhåll att supporten måste fixa saker som går sönder. Men rent underhåll det gör vi väldigt lite av.

Hur påverkar god dokumentation underhållskostnaden för ett system?
Det minskar, eftersom vi delar dokumentation. Mycket av våra underhållskostnader ligger i att supporten måste gå in och fixa saker som har gått sönder. Och om de då kan se exakt när något gått sönder och vilken kod det var som den kunden fick just när det gick sönder så minskar kostnaden eftersom de slipper leta.

Tycker du att tiden minskar för att hitta fel i koden med dokumentation?

Om det är en bugg i originalkoden så hjälper vår nuvarande dokumentation oss inte så mycket. Däremot om det är dokumentation i allting. Förra veckan gjorde vi en uppdatering och innebär det ett problem kan vi se vad som händer.

Tycker ni att man ser för lätt på dokumentation inom agila metoder? Skulle den behöva en mer central roll för agila projekt?

Jag vet inte. Jag tycker att dokumentation är viktigt, man klarar sig inte utan det. Jag har aldrig sett att det skulle vara oviktigt. Däremot beskriver de olika saker. Att den agila metodbeskrivningen beskriver hur man jobbar, hur man planerar, men inte hur man sköter system i längden.

Finns det något du vill tillägga?

I den agila metoden ligger ju att man skriver vad som ska göras och det är ganska svårt att få in i processen att det faktiskt ska dokumenteras. Den biten fattas. Man måste avsätta tid till det också. Det är inte något som sker av sig själv utan man bör aktivt estimera dokumentationen också.

7.7 Transkribering informant 6

Vi gör utveckling i stort med inriktning mot webben och e-business och vi använder oss huvudsakligen av ett CMS-system som heter SiteCoer som är i .NET och utöver det är företag i stort väldigt inriktat på säkerhet, det är väldigt mycket banksäkerhet och inbyggda system i bilar. Så vi har inbyggda system i bilar, säkerhet och e-business.

Hur mycket tid läggs, per dag, på dokumentation under projektets gång?

Det är väldigt varierande, i och med att vi är ett konsultbolag och inte ett produktbolag så är det upp till kunden vad den vill ha. Om kunden vill att vi ska dokumentera varenda funktion som finns i systemet då är det det man gör. Men annars så är det så att projektledaren, om det finns en sådan, från vår sida när systemet är klart skriver en leveransdokumentation eller något likande utifrån systemet. Så jag sitter inte och skriver dokumentation varje dag, så är det verkligen inte. Det vi skriver är ganska så självförklarande och det tar väldigt mycket tid att dokumentera. Om man skriver kod på ett bra sätt så behöver man inte dokumentera så mycket. Skriver du beskrivande kod med beskrivande funktionsnamn så kan du läsa väldigt mycket vad det är som händer i systemet. Så det behövs inte så mycket dokumentation för det tar väldigt, väldigt lång tid och det tar lång tid att ändra. Ändrar du något i systemet så måste du även ändra i dokumentationen och då blir det dubbelt arbete. Vi använder väldigt lite tid för att dokumentera faktiskt.

Så era anställda är inte tvungna att dokumentera allt?

Det beror på kunden. Hade vi varit ett produktbolag så hade vi ju bestämt det själva. Då hade vi själva bestämt om vi vill dokumentera vår egen produkt. Om vi bygger en produkt mot kundens kunder, så är det klart att det ska finnas någon användarmanual för att hantera systemet. Vilken dokumentation är det vi pratar om? Det är ju två olika, man ska dokumentera koden eller om det är en manual för att använda systemet.

Har ni några riktlinjer för hur dokumentationen ska vara utformad?

Nej. Jag vet att vissa andra som sitter i andra projekt så är det styrt utifrån en viss standard. I .NET finns det ett fåtal olika standarder så man kan skriva dokumentationen på ett visst sätt i koden så att den drar ut den och så skapar den ett dokument. En innehållslista där man kan klicka så står blobben med dokumentation till någon referens på ett ställe i koden. Så det finns vissa sådana standarder, det är vissa som använder sig av det. Jag gör inte det i några av projekten som jag sitter i. Men vissa andra gör det och då är det kunden som har bestämt att det är detta som ska användas och då får man ju skriva dokumentation på sin kod.

Vilken typ av dokumentation är det vanligast att kunden vill ha? Ska det vara hur arkitekturen ser ut eller vad koden exakt gör?

Det är också lite varierande. Nu senast när vi har startat upp ett nytt projekt, det handlar ju om att göra det lätt för kunden att förstå vad systemet gör och det är oftare lättare att kolla på en bild av vad systemet gör. Men kunden förstår oftast inte ett UML-diagram, men dem tycker

att det är roligt att ha det ändå för då syns det att man lagt ner tid på det man nu gör eller om den kanske inte förstår sammanhanget så att. Men det är också lite beroende av vilket förhållande man har med kunden. Är det en ny kund så är det klart man får lägga fram allting; såhär har vi tänkt. Men det är ju oftast så att du inte slutar där ändå där för att om du jobbar i någon form av agil utveckling så svänger det ju ofta åt något annat håll än det planerade slutmålet för att folk kommer på att; "Nej det var inte riktigt bra, nu när vi såg hur det faktiskt utvecklades sig." Och så blir det oftast åt nått annat håll.

Vem är dokumentationen främst till för?

Den är till för nästa som ska ta över. För det är jättesvårt att komma in i ett odokumenterat eller svårskrivet projekt. Då tar det jättelång tid att sätta sig in i hur någon annan har tänkt.

Men om ni inte har något tvång eller inte har några riktlinjer för att dokumentera hur blir det då när det kommer in någon ny i företaget?

Man har ju någon form av överlämning, det finns ju dåliga exempel där folk bara slutar såklart. Då är det ju svårt att ha en överlämning av ett projekt, men så går det ju oftast inte till. Om det är så att det är ett inhouse-projekt som någon annan ska ta över för att resurser flyttas runt, det är jätte vanligt, då är det ju ofta att man har en överlämning och sen sitter man ju kvar så om det är frågetecken kan man ju hjälpa till och berätta att "det är så här det är". Om man bara tänker från början på att skriva tydlig kod så är det en stor hjälp på vägen. Dokumenterade funktioner säger ofta nästan ingenting därför att det är svårt att lita på om dem stämmer eller inte.

Men tror du att överlämningen hade varit enklare om ni hade haft dokumentation på allt?

Jag vet inte. Det är nog lättare att hitta själv, kanske. Om det är så att dokumentationen stämmer till 100 % så klart. Men det är ju ofta som att när du sitter och har tagit över ett nytt projekt, då sitter du oftast och stegar dig igenom koden ändå för att se vad som händer. Om du skriver tydlig kod med långa funktionsnamn så är det ofta det som beskriver vad det är som händer, än att du ska sitta och läsa såhär mycket text som faktiskt säger samma sak.
Men jag har kommit in i projekt som är helt obegripliga.

Anser ni att flexibiliteten i projektet minskar eller ökar på grund av dokumentationen?

Jag tror väl inte det påverkar faktiskt. Dokumentationen påverkar ju inte om man ska gå vidare eller inte eller om man kan göra om någonting. Det är klart det blir jobbigare att skriva om någonting för då måste du skriva om dokumentation. Men det blir inte mindre flexibelt.

Påverkar det produktiviteten då?

Det är klart, för du får ju göra om samma sak två gånger egentligen och du måste vara säker på att du har ändrat på alla ställen. Ändra dokumentationen på alla ställen som du nu har ändrat i koden.

Så du menar att det minskar produktiviteten?
Ja, det gör det väl.

Tycker du att det blir lättare att genomföra ändringar i systemet om du tillexempel tar över ett projekt och så finns det dokumentation Är det lättare att genomföra ändringar då?
Nej det tror jag inte. Det är lättare att göra ändringar om det är testdrivet så att du är säker på att när du ändrar någonting så kan du lita på dina test istället. Så att om du kan köra dem, då har du inte förstört resten av systemet.
Så att det är dokumenterat på något ställe pajar inte koden om jag ändrar någon funktion. Det är inte dokumentationen som körs. Det är bara det att man glömmer ändra den, sen är det jätte jobbigt för nästa som kommer och läser dokumentationen och tror att det är såhär systemet fungerar.

Hur mycket tid lägger ni uppskattningsvis på underhåll av system som levererats till kund?
Vi gör ju jättemycket underhåll. Alltså i och med att många av projekten är mer eller mindre i någon form av agil utveckling så man går ju hela tiden framåt i små steg. Och det är hela tiden kunder som vill ha till nya funktioner och då räknas det som någon form av underhåll. Du har ju en första leverans sen efter det är det ju någon form av underhåll tillbygge på det befintliga systemet. Jag sitter väl och gör underhåll varje dag. I någon form. Alltså om det då är för att kunden vill ha dit en ny knapp eller om det är för att något har gått sönder. Det är jättelätt att det introduceras nya buggar i program när du nu ändrar någonting såklart. Därför att omfattningen kan ju vara jättestort och när du ändrar och testar det som du nu vill ha ändrat så går det sönder någon annanstans.

Hur påverkar god dokumentation underhållskostnaden för ett system?
Du får ju lägga mer tid på att dokumentera och kunderna betalar ju per timme så det blir dyrare, så klart. Om det är svar på frågan.

Tycker ni att man ser för lätt på dokumentation inom agila metoder?
Jag tycker väl någonstans att dokumentation, om det fortfarande är dokumentation av kod vi pratar om och inte en beskrivning av systemet för kunden liksom så dem kan bläddra och se vad systemet kan göra. Så tycket jag att dokumentation är gammalmodigt i sådana fall, därför att. Jag tycker väl att dokumentation är onödigt dubbelarbete om man bara skriver koden ordentligt istället, alltså så skulle jag nog vilja säga.

Så inte mer dokumentation, utan mindre än vad ni har?
Mindre dokumentation och sen tydligare kod. Det har ju skiftat lite grann det är en gammal grej att de här kompilatorerna tog längre tid på sig att kompilera koden när du skrev långa funktionsnamn men den problematiken har du inte längre därför datorerna är så kraftiga ändå. Det är liksom inga problem. Mindre scope mindre namn, så att i en lite funktion så kan du ha jätte små variabelnamn, med stora scope som publika metoder kan du skriva jätte långa namn

som faktiskt beskriver vad den gör och så vet du vad hela funktionen gör genom att bara titta på funktionsnamnen. Och att då skriva dokumentation på den som faktiskt säger exakt samma sak som funktionsnamnet det är ju dubbelarbete. Och du ändrar aldrig dokumentationen utan du ändrar funktionsnamnet kanske. För att funktionen ändrar funktion och då ändrar det ju automatiskt överallt om man gör en ordentlig refactor, och då har du egentligen dokumenterat rakt i koden. Och småsaker som if-satser att man istället för att kalla det något så kalla det något med IS framför, för då blir det som att du kan läsa koden som en engelsk text. Så det blir IF - IS - True egentligen. Om du bara läser det så vet du exakt vad det är som händer istället för någonting som du inte kan läsa. Och då blir det bara som att du sitter och läser en lång dokumentation utan att du har dokumenterat någonting. Så att det hänger ihop med det här att dokumentation börjar bli gammalmodigt. Det har bara hängt med gamla utvecklare att "skriv så kort och så lite som möjligt, då blir det snabbt att läsa", men det blir jätte mycket mer svårläst. Därför att sitta och läsa "i + k = j", det är ingen som vet vad det gör.

7.8 Transkribering informant 7

Hur mycket tid lägger ni per dag på dokumentation under ett projekt?
Jag kan tänka mig att det är runt, 4-5 % av den fullständiga tiden som ett projekt pågår. Och då i praktiken så, det senaste projektet här så lade vi väl, och det var på fyra månader… Det beror på vad man menar med dokumentation för i början är det intensivt med kraven att det dokumenteras, och sen är det designen och sen i slutet är det användarmanual och liknande systemdokumentation.

Vi fokuserar på dokumentation av kod.
Av kod. Låt säga att jag tror att vi hade väl projekt på fyra månader med några resurser, på slutet lade vi fem dagar tror jag. Timeboxade fem dagar för att dokumentera upp det vi hade gjort. Och då är det så att man timeboxar, och säger "Vi har fem dagar och inte mer", så kör man så gott man kan.

När ni timeboxar då, kör ni på och ser hur mycket ni hinner dokumentera eller finns det vissa delar av era system som era anställda måste dokumentera?
Det är ju både och, för att i början så gör vi ju en bedömning att "Vi tror vi hinner såhär långt ungefär på fem dagar.", eller vad det nu är. Men det är svårt att säga exakt när vi är klara. Vi har gjort en bedömning innan men då ha vi ju sagt liksom "Då är det såhär vi kör.". För där i slutet, där är det ju mer kritiskt med timmarna för då har man inte så mycket kvar att ta av och kan inte påverka, så då måste man vara mer strikt i slutet av projektet.

Har ni några riktlinjer för hur dokumentation ska vara utformad?
Det är dåligt med det. Vi har mallar. Men det är jättedåligt faktiskt, hur systemdokumentation, specifik dokumentation av kod då, om det var det ni var ute efter, hur det dokumenteras. För en sak är ju att dokumentera design och hur man underhåller det här och generella principer, hur man har tillämpat designen. Men sen ska ju koden egentligen dokumenteras och de gör ju ofta i koden som kommentarer. Och där är det väldigt få riktlinjer, oftast är det så att om det är en arkitekt eller utvecklare som kan det tekniska ramverket, som jag ju ofta inte kan, så tillämpar de en dokumentationsstandard där som de är vana vid. Som till exempel Microsoft har lite standarder för sig man tar till. Och det jag gör då är att när jag skriver projektplanen, så har jag ju arkitekt och så frågar jag arkitekten: "Har du koll på det här, är det någon standard du kommer tillämpa?". Och då säger dem ofta: "Vi använder det där".

Så det är olika i olika projekt?
Ja det är dåligt med det och det är väldigt svårt att styra upp och ja, det skulle kunna göras men vi har lagt energin på annat.

Vilken typ av dokumentation är mest värdefull tycker ni, är det den tekniska eller arkitekturdelen?
Arkitekturdelen.

Vem är dokumentationen främst till för när ni har utformat den, är den till för utvecklarna eller kund, så att dem kan se hur allt fungerar?
Arkitekturen är ju till för arkitekten och utvecklarna i det att om man gör designspecifikationer, det fungerar så här i alla fall, att i bästa fall då när man tillämpar det att man gör en arkitektur och arkitekturdokumentet och sen när det är klart så gör man kanske detaljerade specifikationer hur den ska tillämpas i dem olika leverablerna. Och är det till exempel indier vi har att gör med eller andra, så måste man specifikationer väldigt noga, nästan såhär stegvis, gör såhär, gör såhär, gör såhär. Medan har vi egna interna resurser och ett fåtal resurser så specificerar vi inte alls hur det ska utföras utan då litar vi på att vi kan det själva här och att vi sitter tätt och kommunicerar. Men underhålls- och systemdokumentation är ju främst till för förvaltning och vidareutveckling så det är ju en annan del.

Anser ni att flexibiliteten minskar eller ökar på grund av dokumentation?
Är det här specifikt inom agila projekt?

Ja det är det.
Jag skulle säga rent praktiskt att den ... om man säger godartad flexibilitet så ökar den. Men malign eller dålig flexibilitet vilket innebär att man kan göra hur som helst och där inte är någon plan. Det är ju dåligt, och det har man ju när man inte dokumenterar eller egentligen planerar eller dokumenterar. Så om man säger att den goda flexibiliteten att man vet vilka ramar man rör sig inom så ökar den skulle jag säga, därför man kommer snabbare till beslut och förståelse om vad man kan göra och vilka vägar man kan ta.

Så då ökar produktiviteten lite också kanske.
Jag tycker det.

Hur mycket lättare blir det att genomföra förändringar i systemet när det finns noggrann dokumentation? Vi tänker då på förvaltning och förändringar i systemen när dem måste förbättras.
Ja, det förenklas. Signifikant skulle jag säga. Därför att dels har man ju i arkitekturen ett ramverk som man ska hålla sig inom så man förstår vilka principer man ska tillämpa vid utveckling så det harmonierar med det tidigare utvecklade. Så att man tillämpar dem principerna och inte hittar på något nytt. Det kanske kommer en annan resurs som har lite annat tänk och löser uppgiften på ett annat sätt, efter att projektet är avklarat. Då är det viktigt att den förstår hur tankesättet var, vilka principer som har tillämpats. Så jag skulle säga att det är väldigt mycket effektivare på det sättet och gör det i senare steg lättare att underhålla för då blir det inte olika principer. Sen är det ju alltid viktigt tycker jag man har kodat att man har lite historik mot koden. Vad är ett metadata? Vad är den här modulen till för? Vad gör den här modulen liksom? Och sen så kanske i bästa fall har någon också lagt in lite förändringshantering. “Jag gjorde detta, jag gjorde den här förändringen”.
Och ofta är det då kommentarer i koden. Men då måste det vara på ett strukturerat sätt.

Men är det mer dokumentation om vad som görs och inte hur det görs?
Det är både och. Arkitekturen och riktlinjer är liksom hur ni ska göra mer, medan kommentarer i koden är vad som har gjorts.

Hur mycket tid lägger ni på underhåll av system som har levererats till kund?
Ibland gör vi ju inte alls det, utan levererar ett system sen tar någon annan över underhållet. Men hur mycket tid? … Ganska mycket.

Hur påverkar dokumentation underhållskostnaden för ett system?
Det påverkar positivt, det gör det helt klart. Av de anledningarna jag har sagt att man snabbare vet vad man ska förhålla sig till och snabbare kan hitta. Det finns liksom kringeffekter, är det bra dokumenterat så är det ofta en bra design och tanke bakom som gör att det i sig är lättare att underhålla även om det inte hade funnits dokumentation. Så det liksom hänger ihop lite så också. Sen är det ju en avvägning hur djupt man ska dokumentera. Blir det för mycket dokumentation så blir det för tungrott, och uteslutande skulle jag nog säga uppdateras inte dokumentationen då. Vilket gör att tilliten till den successivt försvinner och då blir den inte använd och då försvinner fördelarna med att det är dokumenterat, så man får inte gå för djupt heller. Det är en vanlig situation, att man har överdokumenterat.

Hur tycker du att god dokumentation påverkar tiden som läggs på underhåll?
Det har en signifikant skillnad, dokumentation är väldigt värdefullt. Vanligt inom ett BI (Business Intelligence) system är att en användare som har bra koll på rapporterna, som kommer ut i andra änden av systemet. Det kommet in data liksom och så tröskas det runt och så kommer det ut en rapport eller något på andra sidan, märker att har är det siffror i rapporten som inte stämmer med det som kommer in. För ofta har dem koll på affärssystemen och så undrar dem: "Kan ni kolla upp det här? Vad beror det på? Är det rätt? Det verkar inte rätt.", och då är det en process som utvecklare måste gå igenom, och vanligt är att det är dåligt dokumenterat och även dåligt designat. Så det är väldigt svårt att hitta var detta sker, och när man väl har hittat det, så har det varit så jobbigt så man bara vill bli av med det och bli klar med det och pyssla vidare med dem grejer man tycker är roliga istället. Men vanligt är att den kommer ofta den frågan. Då blir det att man lär sig successivt "Okej! Jag hittar ungefär där" och så gör man det. Men det är på ett ostrukturerat och ett ad-hoc mässigt sätt. Så är det dokumenterat bra så vet man vad det är som händer i den här processen då. Då vet man vart man ska leta och kan förstår mer vart man ska leta och det går snabbt. För det är en lite långsam process det här att kolla igenom exakt i koden då. Och bara: "Vad är det som händer här nu och vad är det för affärsregler som tillämpas där då?".

Och det kanske blir svårare att se hur alla delar hänger ihop eftersom man måste sätta sig in och förstå det själv?

Det också. Men det hänger också ihop här med att har man dokumenterat på ett bra sätt så finns det oftast en tanke innan var affärsreglerna tillämpas och då kan man isolera mer i sitt sökande och strukturera det. Men vanligast skulle jag nog säga är att det är upp till utvecklarna att komma på var affärsreglerna faktiskt tillämpas i detalj. Och då blir det lite hipp som happ, det blir på lite olika ställen och det är inget bra.

8 Referenslista

Ahimbisibwe, A., Cavana, R. Y., & Daellenbach, U. (2015). A contingency fit model of critical success factors for software development projects: A comparison of agile and traditional plan-based methodologies. Journal of Enterprise Information Management, 28(1), 7-33.

Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., & Merlo, E. (2002). Recovering traceability links between code and documentation. Software Engineering, IEEE Transactions on, 28(10), 970-983.

Avison, D. E., & Fitzgerald, G. (2003). Where now for development methodologies?. Communications of the ACM, 46(1), 78-82.

Awad, M. A. (2005). A comparison between agile and traditional software development methodologies. University of Western Australia

Bassil, Y. (2012). A simulation model for the waterfall software development life cycle. International. Journal of Engineering & Technology.

Basili, V., Boehm, B., Costa, P., Dangle, K., Lindvall, M., Shull, F., Tesoriero, R., Williams, L. & Zelkowitz, M. (2002). *Empirical findings in agile methods.* Springer Berlin Heidelberg.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001): The agile manifesto. http://agilemanifesto.org/ (besökt 2015-04-15)

Berger, H. & Flouri, K. (2010). Agile Development – Scrum Adopted in practice but not in principle. (21), 2-25. University of Wales Institute Cardiff.

Bonner, N., Hardin-Baylor, M. (2010) Predicting leadership success in a agile environments: An inquiring systems approach**.** *Academy of Information and Management Sciences Journal, 2(13)*

Brandt, P. (2005). *Systemförvaltningsmodeller* (Vol. 3). Lund University.

Chasalow, L (2008) What makes agile development different?: A case study of agile in practice. In *proceedings of the Southern Association for Information Systems Conference, Richmond, VA, USA March 13 th -15th*

Cockburn, A. (2002). *Agile Software development.* Boston. Addison-Weasley.

Cusumano, M. A. (2007). Extreme programming compared with Microsoft-style iterative development. *Communications of the ACM*, *50*(10), 15-18.

Ding, W., Liang, P., Tang, A., & Van Vliet, H. (2014). Knowledge-based approaches in software documentation: A systematic literature review. Information and Software Technology, 56(6), 545-567.

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software, 85(6), 1213-1221.

Drury, M., Conboy, K., & Power, K. (2012). Obstacles to decision making in Agile software development teams. Journal of Systems and Software, 85(6), 1239-1254.

Drury-Grogan, M. L., & O'dwyer, O. (2013). An investigation of the decision-making process in agile teams. International Journal of Information Technology & Decision Making, 12(06), 1097-1120.

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

Ebert, C., Abrahamsson, P., & Oza, N. (2012). Lean software development. *IEEE Software*, (5), 22-25.

Estler, H. C., Nordio, M., Furia, C. A., Meyer, B., & Schneider, J. (2014). Agile vs. structured distributed software development: A case study. Empirical Software Engineering, 19(5), 1197-1224.

Fitzgerald, B., Russo, N. L., & Stolterman, E. (2002). *Information systems development: methods in action*. McGraw-Hill Education.

Gandomani, T. J., Zulzalil, H., Ghani, A. A. A., Sultan, A. B. M., & Nafchi, M. Z. (2013). Obstacles in moving to agile software development methods; at a glance. Journal of Computer Science, 9(5), 620.

Garceau, L., & Jancura, E. (1990). Documentation and Training as a Systems Development Tool. The CPA Journal, 60(11), 84.

Garousi, G., Garousi-Yusifoğlu, V., Ruhe, G., Zhi, J., Moussavi, M., & Smith, B. (2015). Usage and usefulness of technical software documentation: An industrial case study. Information and Software Technology, 57, 664-682.

Greer, D., & Hamon, Y. (2011). Agile software development. Software: Practice and Experience, 41(9), 943-944.

Gustavsson, T. (2007). *Agile-konsten att slutföra projekt*. TUK Förlag AB.

Hall, J. (2014). Crowdsourcing Ideas for an Agile Leadership System. Talent Development, 11(68) , 42-46

Holme, I. M., Solvang, B. K. (1997). *Forskningsmetodik: om kvalitativa och kvantitativa metoder*. Studentlitteratur.

Hummel, M., Rosenkranz, C., & Holten, R. (2015). The Role of Social Agile Practices for Direct and Indirect Communication in Information Systems Development Teams. Communications of the Association for Information Systems, 36(1), 15.

Höst, M., Regnell, B. & Runesson, P. (2006). *Att genomföra examensarbete*. Studentlitteratur.

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

Kakar, A. K., Hale, J., & Hale, D. (2012). Social Traps of Agile methods. AMCIS, Association for Information Systems.

Kayes, I., Sarker, M., Chakareski, J (2014) *Product Backlog Rating: A Case Study On Measuring Test Quality In Scrum*. University of South Florida, Tampa, FL, USA, 2-26.

Khan, A. I., Qurashi, R. J., & Khan, U. A. (2011). Comprehensive Study of Commonly Practiced Heavy & Light Weight Software Methodologies. IJCSI International Journal of Computer Science Issues, 441-450.

Layman, L., Williams, L., Damian, D., & Bures, H. (2006). Essential communication practices for Extreme Programming in a global software development team. *Information and software technology*, 48(9), 781-794.

Leau, Y. B., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). Software development life cycle AGILE vs traditional approaches. In International Conference on Information and Network Technology, 37(1), 162-167.

Lee, G., & Xia, W. (2010). Toward agile: an integrated analysis of quantitative and qualitative field data. Management Information Systems Quarterly, 34(1), 7.

Leffingwell, D. (2011). *Agile Software Requirements - Lean requirements practices for teams, programs, and the enterprise.* Boston. Addison-Weasley.

Martin, R.C. (2003). *Agile Software Developement, Principles, Patterns and Practices*. Upper Saddle River, New Yersey: Prentice Hall PTR

Matook, S., & Vidgen, R. (2014). Harmonizing Critical Success Factors in Agile ISD Projects. In AMCIS 2014: 20th Americas Conference on Information Systems. Association for Information Systems Conference.

Moe, N., Barney, S., Aurum, A., Khurum, M., Wohlin, C., Barney, H., Gorschek,T., Winata, M. (2012). *Fostering and sustaining innovation in a fast growing agile company*. 13th international conference on product-focused software process improvement. Springer Berlin Heidelberg.

Munassar, N. M. A., & Govardhan, A. (2010). A comparison between five models of software engineering. IJCSI, 5, 95-101.

Pakdil, F., & Leonard, K. M. (2014). Criteria for a lean organization: development of a lean assessment tool. International Journal of Production Research, 52(15), 4587-4607.

Papadopoulos, G. (2015). Moving from Traditional to Agile Software Development Methodologies Also on Large, Distributed Projects. Procedia-Social and Behavioral Sciences,

175, 455-463.

Parker, D. W., Holesgrove, M., & Pathak, R. (2015). Improving productivity with self-organized teams and agile leadership. International Journal of Productivity and Performance Management, 64(1), 112-128.

Petersen, Kai., Wohlin, C. (2010) The Effect of Moving from a Plan-Driven to an Incremental Software Development Approach with Agile Practices Empirical software engineering: an international journal, 6(15), 654-693

Prause, C. & Durdik, Z. (2012). Architectural Design and Documentation: Waste in Agile Development? *IEEE 978-1-4673-2352-9(12)*

Rao, K. N., Naidu, G. K., & Chakka, P. (2011). A study of the Agile software development methods, applicability and implications in industry. International Journal of Software Engineering and its applications, 5(2), 35-45.

Royce, W. W. (1970, August). Managing the development of large software systems. In proceedings of IEEE WESCON, 26(8), 328-388.

Rubin, E., & Rubin, H. (2011). Supporting agile software development through active documentation. Requirements Engineering, 16(2), 117-132.

van Ruler, B. (2015). Agile public relations planning: The Reflective Communication Scrum. Public Relations Review, 41(2), 187-194.

Ruparelia, N. B. (2010). Software development life cycle models. *ACM SIGSOFT Software Engineering Notes*, 35(3), 8-13.

Rüping, A. (2005). *Agile documentation: a pattern guide to producing lightweight documents for software projects*. John Wiley & Sons.

Santos, V., Goldman, A., & de Souza, C. R. (2014). Fostering effective inter-team knowledge sharing in agile software development. Empirical Software Engineering, 1-46.

Selic, B. (2009). Agile documentation, anyone?. Software, IEEE, 26(6), 11-12.
SouarIssa, S. & Stenlund, P. (2012). Agila systemutvecklingsmetoder vid systemförvaltning. Luleå: Institutionen för system- och rymdteknik.

Stettina, C. J., & Heijstek, W. (2011). Necessary and neglected?: an empirical study of internal documentation in agile software development teams. Proceedings of the 29th ACM international conference on Design of communication, 159-166.

Stettina, C. J., Heijstek, W., & Fægri, T. E. (2012). Documentation work in agile teams: the role of documentation formalism in achieving a sustainable practice. IEEE, 31-40.

Stober, T., Hansmann, S. (2010). *Agile Software Development: Best Practices for Large Software Development Projects.* Springer Berlin Heidelberg

Turk, D., France, R., & Rumpe, B. (2014). Assumptions underlying agile software development processes. Journal of Database Management, 16(4), 62-87.

Vetenskapsrådet. (2002). Forskningsetiska principer inom humanistisk-samhällsvetenskaplig forskning.
http://www.codex.vr.se/texts/HSFR.pdf (besökt 2015-04-15).

Williams, L. (2012). What agile teams think of agile principles. Communications of the ACM, 55(4), 71-76.

Zhang, P., Carey, J., Te'eni, D., & Tremaine, M. (2004). Integrating Human-Computer Interaction Development into SDLC: A Methodology. AMCIS 2004 Proceedings, 4620-4626.

Zhang, X., Hu, T., Dai, H., & Li, X. (2010). Software development methodologies, trends, and implications. Information Technology Journal, 9(8), 173-178.

Zhi, J., Garousi-Yusifoğlu, V., Sun, B., Garousi, G., Shahnewaz, S., & Ruhe, G. (2015). Cost, benefits and quality of software development documentation: A systematic mapping. Journal of Systems and Software, 99, 175-198.