



EKONOMIHÖGSKOLAN
Lunds universitet

Institutionen för informatik

Koordination i stora Scrumprojekt

Kandidatuppsats, 15 högskolepoäng, INFK01 i informatik

Framlagd: Juni, 2009

Författare: Max Martinsson
Joel Olofsson

Handledare: Claus Persson

Examinatorer: Lars Fernebro
Anders Svensson

Abstract

Titel	Koordination i stora Scrumprojekt
Författare	Max Martinsson Joel Olofsson
Utgivare	Institutionen för informatik
Handledare	Claus Persson
Examinatorer	Lars Fernebro Anders Svensson
Publiceringsår	2009
Uppsattstyp	Kandidatuppsats
Språk	Svenska
Nyckelord	Scrum, koordination, integration, beroende, sprintar, teams

Abstract

Scrum är en agil projektledningsmetod som används inom systemutveckling. Metoden i sin ursprungliga form fokuserar på projekt med endast ett team, där arbetet koordineras mellan utvecklare främst genom informell kommunikation på de dagliga mötena. Koordination ser vi i denna uppsats som den aktivitet som hanterar beroenden mellan aktörer, för att möjliggöra den framtida sammansättningen av deras respektive resultat till en hel produkt. När flera team arbetar på samma projekt, uppstår det beroenden mellan teamen, vilket gör att de behöver koordineras. Scrum förespråkar dock ingen tydlig mekanism för att koordinera arbetet mellan dem, då det bara hanterar ett team. För att undersöka hur denna koordination sker, har vi genomfört kvalitativa intervjuer på företag som använder Scrum i stora projekt. Våra resultat från dessa intervjuer tyder bland annat på att arbetet mellan team koordineras med hjälp av informell kommunikation, precis som i Scrums originalutförande. Men det visar sig även att koordinationsmekanismer såsom toppstyrning och standardisering används i kombination med detta. Resultaten tyder också på att det är viktigt hur teamen organiseras, för att minska beroenden mellan team, vilket kan minska behovet av koordination.

Innehåll

1	Inledning	1
1.1	Bakgrund	1
1.2	Problem	1
1.3	Syfte och frågeställning	2
1.4	Avgränsningar	2
1.5	Tillvägagångssätt	3
1.6	Definitioner	3
2	Scrum och koordination	4
2.1	Scrum	4
2.2	Koordination	6
2.3	Stora Scrumprojekt	9
2.4	Teoretiskt ramverk	12
3	Undersökningsmetod	13
3.1	Tillvägagångssätt	13
3.2	Intervjuguider	13
3.3	Urval av respondenter	13
3.4	Metodkritiska överväganden	14
3.5	Validitet & reliabilitet	15
4	Empiriska resultat	16
4.1	Presentation	16
4.2	Scrum-efterlevnad	17
4.3	Teamens organisering	18
4.4	Integration	20
4.5	Koordinationsaktiviteter	21
5	Analys och diskussion	24
5.1	Beroenden	24
5.2	Koordinerande roller	26
5.3	Koordinationsaktiviteter	27
5.4	Scrum	28
6	Slutsatser	29
6.1	Slutord	30
Bilaga A	Intervjuguider	31
A.1	ScrumMaster/Produktägare	31
A.2	Utvecklare	32
Bilaga B	Intervjuprotokoll	33
B.1	ScrumMaster, Företag A	33
B.2	Produktägare, Företag B	38
B.3	Utvecklare, Företag A	45
B.4	Utvecklare, Företag B	49
	Referenser	51

1 Inledning

1.1 Bakgrund

Scrum är en relativt ny metod för att utveckla system. Den tillhör de agila metoderna, som också kallas lättviktsmetoder, men är ingen specialiserad systemutvecklingsmetod utan snarare en projektledningsmetod som används för att styra en systemutvecklingsprocess. Det finns ingen utpräglad projektledarroll i Scrum, den mesta makten ska utgå från utvecklarna själva och självorganisering är en viktig del. För att ge struktur finns det två roller till; ScrumMaster som ser till att metoden följs och röjer undan hinder för utvecklarna och produktägaren (Product owner) som representerar kunden och prioriterar funktionalitet. (Schwaber, 2004)

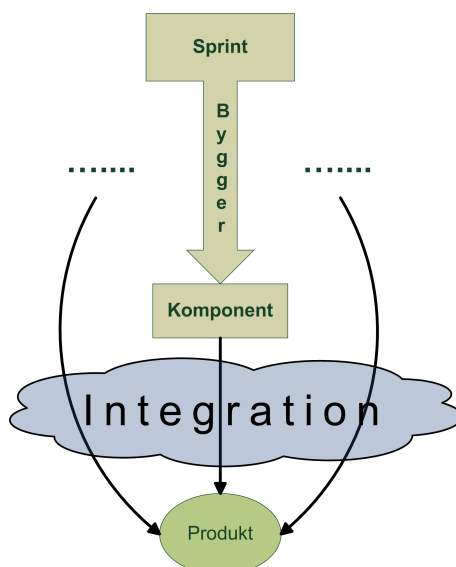
De agila metodernas utgångspunkt är vattenfallsmetodernas problematik med att hantera förändring under projektets gång. Avison och Fitzgerald (2006) påpekar att det är först när användare får se och testa systemet som de kommer lite närmre att förstå vad de vill eller inte vill ha. Detta motiverar de agila metodernas tillvägagångssätt, med snabba iterationer och mycket användarmedverkan gentemot de traditionella metodernas planerande arbetssätt som i värsta fall inte märker användarrelaterade problem förrän i slutet av utvecklingsprocessen (Avison och Fitzgerald, 2006). I Scrums iterativa utveckling används ordet *sprint* för att benämna en tidsbestämd arbetscykel där ett team producerar en körbar version av produkten.

Det finns studier som visar på att Scrum leder till mer framgångsrika projekt (Sutherland, 2005), men då i sammanhang av mindre utvecklingsprojekt. Just små utvecklingsprojekt är vad Scrum i sitt ursprung är avsedd för; ett team, en ScrumMaster och en produktägare. Några storskaliga Scrum-projekt, med mer än ett team, har dokumenterats (Bedoll, 2003; Lyon och Evans, 2008), men nästan uteslutande av Jeff Sutherland, som är en av skaparna av Scrum och VD för företaget Scrum Inc., eller Ken Schwaber, som också är en av skaparna. Metoden har också kritiserats då det hävdats att agiliteten i stora Scrum-projekt minskar på grund av ökat behov av koordination (Turk, France och Rumpe, 2002). Mot denna bakgrund ser vi problem med att hantera skalningen av Scrum.

1.2 Problem

I utveckling som involverar flera aktörer formas slutprodukten genom integration av de delar dessa producerar. Denna integration av komponenter kräver att de är designade för att fungera tillsammans, men om komponenterna tillverkas av olika aktörer måste arbetet på något sätt koordineras mellan dem för att möjliggöra och underlätta den framtida integrationen.

Vårt problemområde som beskrivits i text ovan, illustreras i figur 1.1. Flera sprintar (punkterna i figuren) skapar komponenter som måste sättas samman ("Integration"-molnet) till en komplett produkt. Denna integrering kan underlättas eller minskas på många sätt. Denna uppsats försöker klargöra hur.



Figur 1.1: Denna uppsats problemområde

I små projekt med bara ett team sker denna koordination av arbete genom de dagliga möten utvecklarna har (Schwaber, 2004). I stora projekt med flera team, måste det ske koordination av dessa teams sprintar, av samma anledning som det måste ske hos utvecklarna inom ett team – att en körbar version av produkten måste levereras vid sprintens slut.

Denna koordination mellan team representeras inte av någon aktivitet i den ursprungliga Scrum-metoden, eftersom den bara hanterar ett team. Ett stort problem med att skala Scrum är därför att bibehålla den nära koordinationen av aktiviteter som finns i metodens ursprungliga form. Därför är det intressant att undersöka vad det finns för motsvarighet till Scrums enkla koordination i stora projekt.

1.3 Syfte och frågeställning

Syftet med denna uppsats är att nå en djupare förståelse för hur arbetet koordineras mellan sprintar i stora projekt som använder Scrum. För att nå detta syfte formulerar vi en frågeställning. Koordination i projekt med bara ett team sker som vi sagt via de dagliga mötena. Eftersom vi i denna uppsats vill behålla fokus på hur arbete koordineras, men ändra sammanhanget till stora projekt, så formulerar vi vår undersökningsfråga så här:

Hur koordineras arbetet mellan team i stora Scrumprojekt?

1.4 Avgränsningar

Som nämnts i problembeskrivningen, hanterar vi inte de eventuella projekt där man har ett enda stort team med alla teammedlemmar i. Detta eftersom vi vill titta på koordineringen mellan flera team och inte inom ett team. Alltså är vår definition på stora projekt “projekt bestående av flera team”.

Att team överhuvudtaget koordinerar sitt arbete sinsemellan är inte alls säkert -de kan ha så pass separata arbetsuppgifter att de aldrig behöver kommunicera. Detta innebär dock att de egentligen inte jobbar på samma projekt, utan snarare är egna projekt. Därför är de heller inte intressanta för oss.

1.5 Tillvägagångssätt

För att få svar på vår fråga och nå vårt syfte har vi gjort en undersökning bland företag och organisationer som använder Scrum i stora projekt. Vi genomförde kvalitativa intervjuer med personer som arbetar i dessa projekt. För att få en mer nyanserad bild av respektive företag, intervjuade vi två personer på varje företag. Intervjupersonerna hade olika roller på företagen; utvecklare, ScrumMaster eller produktägare.

För att kunna genomföra undersökningen började vi med att ta till oss teorier och modeller som är relevanta för vårt syfte och frågeställning. Här inkluderas de två stora teoribaserna av uppsatsen; Scrum och koordination. Dessa teoribildningar mynnar ut i en modell – ett ramverk – som vi presenterar i avsnitt 2.4.

Detta ramverk är en grund för resten av arbetet, och är källan till frågorna i de intervju-guider som presenteras i Bilaga A. Därför används också ramverket för analysen av det resultat som intervjuerna leder till. Ramverket fungerar i analysen som en “lins” att se resultaten genom.

1.6 Definitioner

Nedan är en presentation av begrepp som läsaren av uppsatsen kanske är obekanta med. Det finns även begrepp som kan vara tvetydiga och därför definieras tydligare.

Beroende är ett mått på två eller flera teams eller sprintars ömsesidiga beroende. Det hänger tätt samman med begreppet koordination och integration.

Integration är synonymt med “sammansättning”. I denna uppsats menas mer specifikt “Sammansättning av komponenter till en hel produkt”

Komponent är en del, modul eller komponent i ett IT-system

Koordination är den proaktiva aktivitet som sker mellan sprintar, för att möjliggöra och underlätta integrationen mellan de komponenter de producerar.

Multimedlemmar är personer som är med i flera team.

Sprint är en tidsbegränsad arbetscykel där ett team producerar en körbar version av en produkt.

Stora projekt är projekt som har mer än ett team, och därmed flera parallella sprintar.

Team är en engelsk term för en arbetsgrupp.

Utvecklare är en person som jobbar i ett eller flera team.

2 Scrum och koordination

De teorier som uppsatsen baseras på presenteras här och därför innehåller detta kapitel först en beskrivning av metoden Scrum, sedan presenteras forskningsresultat om koordination. Dels generell sådan men även med avseende på organisationer, vilket gör att en del organisationsteori också tas upp. Eftersom koordination hör tätt samman med beroenden och integration kommer dessa begrepp också att presenteras. Koordinationsteorin kopplas sedan till Scrum och till stora Scrumprojekt, vilket slutligen resulterar i vårt ramverk.

2.1 Scrum

Scrum är en agil projektledningsmetod och eftersträvar kontinuerlig anpassning till föränderliga krav (Abrahamsson, Salo, Ronkainen och Warsta, 2002), genom den iterativa utvecklingen. Timeboxing är grunden till att arbeta iterativt. Innebörden är att man låser tidsramen för en del av utvecklingen, och låter istället funktionaliteten vara variabel. Detta är tänkt att uppmuntra prioriteringen av den viktigaste funktionaliteten så den blir gjord först. Genom den iterativa utvecklingen som blir resultatet av denna timeboxing ska man kunna hantera förändringar lättare än om man planerar allt från början. Scrum förespråkar även självorganisering av team, där toppstyrning undviks så mycket som möjligt och Scrumteamet tillåts bestämma sina åtaganden och hur de ska genomföras. (Schwaber, 2004)

2.1.1 Scrum's faser

Den första av Scrum's tre faser är Pre-gamefasen som innehåller två delfaser; planering och arkitektur. I planeringen skapas en så kallad produktbacklogg (eng. Product backlog), som innehåller alla för tillfället kända krav som kan komma från till exempel kunden, marknadsavdelningen eller utvecklarna. Varje krav får en prioritering och en estimering på hur lång tid det tar att genomföra. I planeringsfasen ingår också att bestämma vilka verktyg och resurser som ska användas, riskhantering och andra traditionella planeringsaktiviteter. I arkitekturfasen används kraven från produktbackloggen för att skapa en högnivåbild av systemets funktionalitet. Upptäcks ändringar revideras berörda krav i backloggen. (Abrahamsson et al., 2002)

Utvecklingsfasen är det Scrum är mest känt för och det som egentligen gör att den passar in i det agila facket. Utvecklingsfasen består av att produktbackloggen delas upp i så kallade sprintar som är korta utvecklingscykler som skall producera en körbar produkt. Varje sprint får en egen backlog. En viktig princip är att man inte är rädd för att hantera oförutsedda händelser. Förutsättningarna och kraven för utvecklingsarbetet (exempelvis tidsram, kvalitet, resurser, tekniker och verktyg) tillåts förändras och välkomnas av utvecklingsprocessen. (Abrahamsson et al., 2002)

Post-game är avslutningsfasen i projektet. Detta infaller när det inte finns fler saker att implementera i backloggen. Inga saker eller problem kan heller tillföras projektet. Systemet är redo att levereras, inklusive aktiviteter såsom integrering (i traditionell mening), systemtestning och dokumentation. (Abrahamsson et al., 2002)

2.1.2 Roller i Scrum

ScrumMastern är en unik roll för Scrum, som kan liknas vid projektledare fast utan detaljstyrning. ScrumMastern ska se till att kommunikationen mellan utvecklarna inom teamet sker smidigt och rollen är även till för att hjälpa till snarare än att kontrollera. Det ingår i ScrumMasterns roll att lära organisationen hur man ska arbeta med Scrum. (Schwaber, 2004)

Produktägaren är den person som är officiellt ansvarig för projektet och dess produktbacklogg. Han eller hon representerar kundens önskemål och deltar i planerandet av utvecklingsarbetet och omvandlingen av delar i produktbackloggen till saker att utveckla. (Abrahamsson et al., 2002)

Scrumteamet är en mindre enhet, på 5-9 personer, som är självbestämmande över vad som behöver arbetas på för att slutföra målen i sprinten och för hur det ska gå till. Teamets medarbetare ska ha olika kompletterande roller. Det kan innebära att programmerare får ingå i team med designerns, testare och användare för att bilda en så blandad grupp som möjligt. Teamet har som uppgift att delta i planeringsarbetet för Sprintbackloggen, göra tidsuppskattning för sitt arbete, utvärdera produktbackloggen och berätta om hinder som behövs undanröjas för att arbetet ska fortskrida. (Abrahamsson et al., 2002)

2.1.3 Artefakter

Produktbackloggen är en prioriterad lista på allt arbete som behöver göras för att produkten ska bli klar. Det är viktigt att tillägga att detta baseras på den nuvarande kunskapen om problemområdet och att det kan ändras under processens gång. Produktbackloggen kan innehålla saker som funktionalitet, bugggrättningar och önskvärda förbättringar. Uppskattning av arbetsinsatsen för de olika uppgifterna i produktbackloggen sker iterativt och uppdateras när det finns mer information om respektive uppgift. De som är ansvariga för detta är produktägaren tillsammans med Scrumteamet. (Abrahamsson et al., 2002)

Sprintar är en central del av metoden, och är en iteration i utvecklingen av förbestämd längd, vanligtvis 1-4 veckor. Här realiserar ett Scrumteam de uppgifter som finns i Sprintbackloggen till funktionalitet som går att visa upp på nästa Sprint Review Meeting. Teamet organiserar sig själv och sitt arbete för att producera en uppdaterad fungerande version av systemet. En viktig princip är att det är Scrumteamet som gemensamt är ansvarigt för utvecklingen. (Schwaber, 2004)

En sprints längd hålls under 30 dagar av två skäl. Det första skälet är att utvecklingsarbetet kan utföras ostört, utan att man behöver producera extra programkod eller dokumentation, vilket ska hjälpa utvecklarna att fokusera på arbetet. Det andra skälet är att intressenterna hålls uppdaterade. Genom att ha korta perioder mellan presentationsmötena kan intressenter lättare avgöra om det som teamet producerar är i linje med deras önskemål. (Schwaber, 2004)

Sprintbackloggen är en samling uppgifter som ska utföras under sprinten. Dessa uppgifter tas från produktbackloggen på Sprint Planning Meeting, och specificeras och delas

upp ytterliggare innan de läggs in i Sprintbackloggen. Det är produktägaren som bestämmer vilka uppgifter som ska tas in från produktbackloggen. Scrumteamet gör sedan en uppskattning av hur stor arbetsinsats de kan göra under sprinten och det läggs till i Sprintbackloggen. (Schwaber, 2004) När Sprintbackloggen är bestämd ändrar man inte den under sprintens gång. Det går däremot att bryta en sprint innan den är slut om den upplevs inaktuell eller inne på fel spår. (Schwaber, 2004)

2.1.4 Möten

Daily Scrum är en kontrollprocess i form av ett dagligt möte där Scrumteamet och ScrumMastern möts för att checka av var man är någonstans. Enligt Schwaber (2004) frågar ScrumMastern varje teammedlem tre frågor;

- Vad gjorde du igår?
- Vad ska du göra idag?
- Har du några hinder eller problem?

Detta fungerar också som ett informellt planeringsmöte då alla får reda på vad alla andra kommer att jobba med (Schwaber, 2004). Mötet ska idealt vara cirka 15 minuter långt och för att hålla det kort rekommenderas att man står upp (Schwaber, 2004). Andra än ovan nämnda personer får vara med på mötet men då som observatörer. De får inte prata eller på något sätt påverka mötesdeltagarna. (Schwaber, 2004)

Inför varje sprint möts teamet i ett så kallat Sprint Planning Meeting, vilket är en tudelad process. I första mötet deltar kunderna, användarna, produktägaren och Scrumteamet för att komma fram till önskade mål och funktionalitet för nästa sprint. I andra mötet väljer teamet, med vägledning av produktägaren, ut de element ur produktbackloggen som ska tas med i sprintens egen backlog, baserat på önskemålen från det första mötet. Båda mötena leds av ScrumMastern. (Schwaber, 2004)

Efter varje sprint presenterar Scrumteamet och ScrumMastern resultatet av senaste sprinten, vilket helst ska vara en fungerande produkt eller delprodukt. Medverkande på mötet är kunder, användare, produktägaren och ledningen. Man utvärderar framstegen i sprinten och bestämmer fortsatt färdriktning för projektet. Detta möte kan leda till nya uppgifter i produktbackloggen och även helt ändra inriktning för systemets fortsatta utveckling. (Schwaber, 2004)

2.2 Koordination

Mintzberg (1989) hävdar att all organiserad mänsklig aktivitet ger upphov till två motstridande krav; *arbetsfördelning* till delaktiviteter och *koordination* av dessa delaktiviteter. Arbetsfördelning handlar om att dela upp aktiviteten i mindre, hanterbara delar, medan

koordination av dessa delaktiviteter kräver att man har ett helhetsperspektiv för att garantera att de fungerar tillsammans. Motstridigheten ligger i att man vill ha ett helhetsperspektiv samtidigt som anledningen till att man vill dela upp aktiviteten är för att slippa se helheten.

Detta resonemang liknar delar av systemteori, som delvis bygger på att helheten är mer än endast dess delar, om delarna är ömsesidigt beroende (Avison och Fitzgerald, 2006; Bertalanffy, 1950). I det fall då delarna är oberoende är helheten endast en summering av dem (Bertalanffy, 1950). Cheng (1983) gör också kopplingen mellan systemteori och organisationer när han hävdar att i en organisation med låg integration¹ är helheten endast en summering av aktörernas arbetsbidrag, medan vid hög integration blir det mer än så. Mintzbergs (1989) tidigare nämnda motstridighet är också en form av detta holistiska synsätt.

Malone och Crowston (1994) visar på generella och gemensamma teman av koordination inom skilda områden, från biologi till IT-system. Det som dessa system har gemensamt är att de innefattar aktiviteter som beror på och påverkar varandra. Deras generella definition av koordination blir därför "hanteringen av beroenden mellan aktiviteter":

Coordination is managing dependencies between activities. (Malone och Crowston, 1994)

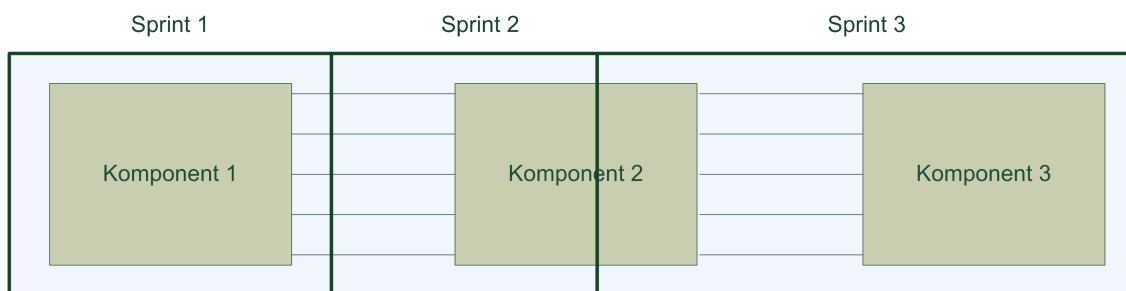
Med beroenden mellan aktiviteter menas snarast avhängighet, det vill säga till vilken grad aktiviteten i fråga påverkas av hur eller när andra aktiviteter utförs. Denna uppsats behandlar fallet då dessa aktiviteter utförs av olika aktörer och mer specifikt när dessa aktörer grupperas till team. Beroenden mellan aktiviteter överförs då till aktörerna och deras respektive gruppering.

2.2.1 Beroenden och integration

Att beroenden mellan team av aktörer påverkas av beroenden mellan de komponenter de ska producera (deras aktivitet) har brett stöd (Kraut och Streeter, 1995; Sutherland, 2005; Cheng, 1983). Särskilt Sutherland (2005) beskriver hur beroenden mellan komponenter i ett projekt krävde liknande beroenden mellan teamen som utvecklade de respektive komponenterna. Kraut och Streeter (1995) tar också upp att den precisa integrationen mellan komponenter är ett av de stora problemen när det gäller att utveckla system, och att bristfällig koordination mellan team som utvecklar komponenterna kan leda till att det inte går att integrera dem till en hel produkt. Relaterat är att Cheng (1983) också i en undersökning av 33 organisationer visar på att ju mer beroende (eng. interdependent) teamen är, desto mer koordination sker mellan dem.

Eftersom integrationen är ett stort problem (Kraut och Streeter, 1995), är det önskvärt att sträva efter att uppnå så lågt beroende mellan team som möjligt. Att minska beroenden mellan grupper av aktiviteter kan åstadkommas genom att gruppera ihop de aktiviteter som har högst beroenden (Malone och Crowston, 1994). Ifall utvecklare kan likställas med de aktiviteter de utför, motiverar detta att skapa team där utvecklarna som ingår arbetar på väldigt nära relaterade saker, kanske till och med samma komponent eller del av

¹Med integration menar Cheng (1983) samma sak som "beroende" i denna uppsats



Figur 2.1: Tre komponenter och tre sprintars arbetsfördelning av dessa.

systemet. Anledningen till att detta minskar beroenden mellan team är för att alla som jobbar på samma komponent kommer att finnas i samma team. Det är dock viktigt att poängtera att detta inte är samma sak som att gruppera personer baserat på deras yrkesroller, utan teamen bör bestå av personer med blandade roller som Scrum också föreskriver (Schwaber, 2004).

Att gruppera utvecklare för att minska beroenden kan också kopplas till begreppen kohesion och koppling inom objektorienterad analys. Att låta ett team dela ansvar för en komponent med ett annat team, som Sprint 2 och 3 i figur 2.1 gör, leder till att sprintarnas beroenden sinsemellan blir lika svåra som de beroenden som bildats ifall komponenten delats upp på riktigt. Detta är problematiskt eftersom försök att dela upp en komponent med hög kohesion leder till en hög grad av koppling (Mathiassen, Munk-Madsen, Nielsen och Stage, 2001). Alltså bildas det hög koppling (beroenden) mellan sprintar som delar på utvecklingen av en komponent. Detta ger ytterligare stöd för att gruppera utvecklare med mycket beroenden.

2.2.2 Koordinationsmekanismer

För att koordinera arbete mellan aktörer (de som utför arbetet), beskriver Mintzberg (1989) olika koordinationsmekanismer. Dessa mekanismer är ömsesidigt anpassande, toppstyrning och standardisering av arbete, resultat, kunskap respektive normer.

- *Ömsesidigt anpassande* är den enklaste formen av koordination och består av informell kommunikation mellan medarbetare eller grupper. Mintzberg (1989) påpekar att ömsesidigt anpassande är den mekanism som främst används vid enkla organisationer, men paradoxalt också i de mest komplexa.
- *Toppstyrning* innebär att koordination sker genom att en person ger order till andra, vilkas arbete är beroende av vartannat, om hur de ska jobba.
- *Standardiseringsmekanismer* har gemensamt att de i förväg bestämmer vissa krav på arbetet, produkten, kunskap respektive normer, hos aktörerna som måste uppfyllas. Då alla aktörer känner till vilka krav som de andra ska uppfylla, kan de anpassa sitt arbete utan eller med mindre kommunikation.

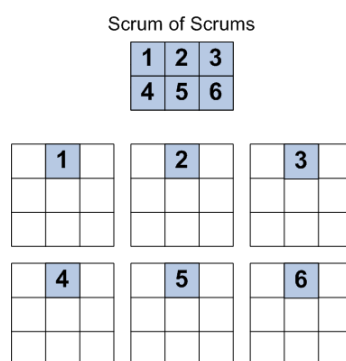
2.2.3 Kommunikationslinjer

Brooks (1995) beskriver svårigheter med att öka personalstyrkan på projekt som orsakas av ökande kommunikation. I projekt där alla medarbetare måste kommunicera med alla andra medarbetare ökar antalet kommunikationslinjer mellan dem kvadratisk. Att alla pratar med alla kan exempelvis ses i Scrum på de dagliga mötena, där alla teammedlemmar träffas för att koordinera sitt arbete sinsemellan. Det blir på grund av det ohanterligt att bara öka storleken på sitt team när projektet blir större, utan motiverar att man måste dela upp projektet i flera team. Därmed stoppas den explosionsartade ökningen av kommunikationslinjer effektivt.

2.3 Stora Scrumprojekt

Scrum som beskrivits ovan är avsett för projekt bestående av endast ett Scrumteam. Eftersom denna uppsats behandlar projekt bestående av flera team, är det intressant att se vilka olika metoder det finns för att skala upp Scrum till större projekt.

Den metod som förespråkas av Scrums grundare kallas Scrum-of-scrums (Schwaber, 2004). Den ämnar till att koordinera arbetet mellan team inom ett projekt, på samma sätt som Daily Scrum koordinerar arbetet mellan utvecklare inom ett team. En Scrum-of-scrums-struktur är alltså ett Daily Scrummöte där varje deltagare är en representant från varje team i projektet. Denna representant brukar vara ScrumMastern från respektive team. (Schwaber, 2004). Denna skalningsmetod illustreras i figur 2.2 med sex team där en representant från varje ingår i ett övergripande Scrumteam med sex medlemmar.



Figur 2.2: Scrum-of-scrums (anpassad från Cohn, 2009)

Hur Scrum-of-scrums implementeras i detalj är olika och ingen klart definierad strategi finns att tillgå, men Schwaber (2004) tar upp några strategier för att handskas med stora projekt och Scrum. Han menar att en infrastruktur för skalning måste vara på plats innan ett projekt kan börja skalas upp. Staging kallar han processen att sätta upp denna infrastruktur som tar plats före den första Sprinten och görs under en dag. Det är där planeringen för hur uppskalningen av projektet ska gå till sker. Kraven som representerar staging prioriteras högt och blir därför tillsammans med de högst prioriterade funktionella delarna av produktbackloggen det första som hanteras i de första sprintarna. Så länge krav i backloggen för skalningsinfrastrukturen återstår används enbart ett team. Det är först när alla krav uppfyllts som fler team läggs till. För varje team som läggs till förflyttas en teammedlem ur ursprungsteamet till det nya teamet. (Schwaber, 2004)

Schwaber (2004) tar själv upp kritik som riktades mot honom, angående att Scrum-of-scrams skulle vara en motsägelse mot Scrums grundprinciper om självorganiserade och självstyrande team. Kritikens poäng var att det inte var teamen själva som organiserade och styrde arbetet fullt ut, utan hierarkiska strukturer blev lagda över dem. Schwabers svar på denna kritik är att det finns olika sätt att handskas med stora Scrumprojekt beroende på hur komplext projektet är. Grundtanken ska vara att man förlitar sig på självorganisering av och i teamen, men om den inte sker tillräckligt snabbt, måste man gå in med regler och struktur och hjälpa organiseringen på vägen. (Schwaber, 2004)

2.3.1 Organisation och beroenden i stora projekt

När ett Scrumprojekt innefattar flera team, har dessa team egna sprintar och det är egentligen dessa sprintar som behöver koordineras. De olika teamens sprintar kan vara organiserade tidsmässigt på olika sätt. Detta är inget som beskrivs i det ursprungliga Scrum, och antas därför inte vara tidsjusterade på något speciellt sätt. De kan även vara justerade så att de går under samma tidsperiod, vilket gör att de levererar sina respektive delar samtidigt, vilket skapar tydligare punkter för integration (Lyon och Evans, 2008).

Eftersom Scrum dikterar att teamens medlemmar ska ha blandade roller, borde rimligtvis en organisation som följer Scrum dela in utvecklare i team där alla roller för att utveckla funktionaliteten i fråga finns. Det är något som dokumenterats särskilt av Microsoft och kallas där "feature crew"-modellen (Miller och Carter, 2007). Detta alternativ stöds också av Malone och Crowston (1994) som motiverade att dela in utvecklare som jobbar med samma funktionalitet i team för att minska beroenden mellan dem. Med funktionalitet menar Miller och Carter (2007) användarfunktionalitet som levereras av ett team på mellan fem till tio veckor. Efter att en sådan funktionalitet färdigställts, hade man först principen att teamet löses upp och att dess medlemmar formade nya "feature crews". (Miller och Carter, 2007) Man valde dock att låta teamen fortsätta bestå av samma medlemmar även mellan utveckling av olika funktionalitet. Fördelen med detta är att de kan bli mer effektiva över tid eftersom de slipper omformas beroende på uppgift. Nackdelen är att arbetsbördan blir svår att få jämn. Att teamen arbetar på en del av funktionalitet av systemet kallar vi i denna uppsats för att vara *funktionalitetsfokuserad*.

Teamen på Microsoft var både funktionalitetsfokuserade och *rollblandade* (Miller och Carter, 2007), vilket Scrum dikterar (Schwaber, 2004). De blandade rollerna innebär att designers, programmerare och andra ingår i samma team – alla roller som behövs för att utföra arbetet finns i samma team. Genom att applicera Malone och Crowstons (1994) teorier ser vi att eftersom utvecklare då inte behöver gå utanför teamets ramar lika ofta för att söka information, leder detta till minskade beroenden mellan team, givet att alla övriga parametrar är oförändrade.

Det har visat sig betydelsefullt att utvecklare befinner sig på samma plats. Samplacering (eng. co-location) betyder i vårt fall att teamen sitter på samma plats, och alltså inte är distribuerade geografiskt. Siemens upplevde att samplacering underlättade samarbetet mycket mellan team (Moore och Spens, 2008). Microsofts feature crews var också samplacerade (Miller och Carter, 2007) och Miller (2008) rekommenderar samplacering av team i så stor utsträckning som möjligt. Ifall delar av projektet distribueras geografiskt ska man se till att samplacera och gruppera de team som arbetar på samma funktionalitet (Miller, 2008), vilket anknyter till föregående avsnitt.

2.3.2 Integration i stora projekt

Hur och när integration sker i projekt är mycket varierande. En taktik som ofta används i samband med agila metoder är så kallad Continuous Integration (CI) – på svenska kontinuerlig integration. Begreppet innefattar många olika tekniker som sammantaget ska leda till att utvecklarnas integration av arbete mot "huvudkoden" sker flera gånger dagligen (Abrahamsson et al., 2002). Att implementera kontinuerlig integration i en utvecklingsorganisation har visat sig svårt (Miller, 2008; Moore och Spens, 2008) även om det kan leda till stora vinster (Miller, 2008). När Siemens skulle implementera kontinuerlig integration misslyckades det och man blev tvungen att tillsätta ett "integration team" (Moore och Spens, 2008).

Ett integrationsteam är ett team som är ansvarigt för att få de andra teamens komponenter att fungera tillsammans (Kähkönen, 2004; Moore och Spens, 2008). Medlemmarna i detta team kan antingen vara med i bara integrationsteamet eller endast vara där på deltid medan de jobbar i ett vanligt utvecklingsteam i övrigt. Att låta någon annan än teamen själva sköta integrationen är det flera som dokumenterat (Moore och Spens, 2008; Miller och Carter, 2007; Kähkönen, 2004). Miller och Carter (2007) har exempelvis en så kallad "build engineer" och en "release manager" som sköter detta.

2.3.3 Koordination i stora projekt

När team ska koordinera sitt arbete finns det flera mekanismer som kan användas. Vi tog upp dessa i avsnitt 2.2.2. Här tar vi upp mer specifika sätt att koordinera som praktiker dokumenterat.

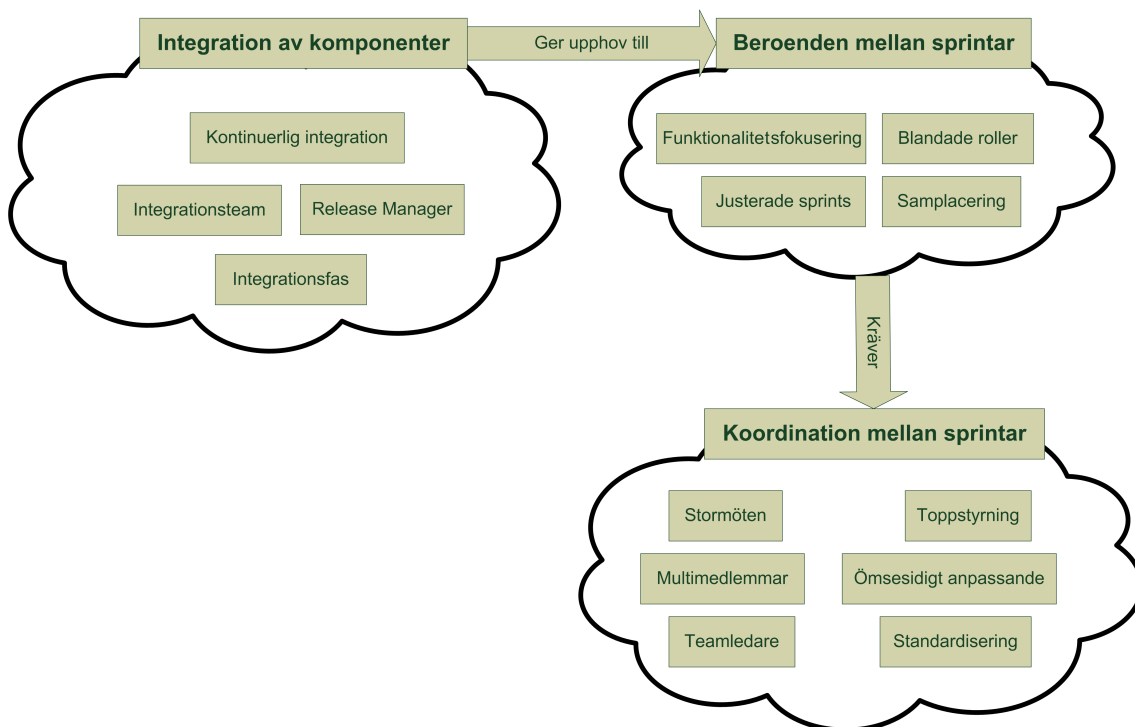
Microsoft hade personer som var "leads" för två-tre team var (Miller och Carter, 2007). Teamen planerade sitt eget arbete, men var tvungna att få planeringen godkänd av ledningen via sin lead. På det sättet hade ledningen en möjlighet att sköta arbetsfördelningen i varje iteration. Leads var dock framförallt ansvariga för att spåra arbete som hade påverkan utanför teamet (Miller och Carter, 2007), alltså en form av att koordinera arbetet.

Att ha personer som jobbar i flera team kan vara ett sätt att förmedla information mellan teamen. Det kan också vara bra att det är en specifik person som sköter detta eftersom det kan vara mödosamt att ändra fokus från det egna teamet till hela projektet. Detta var lea- dens uppgift hos Microsoft och Siemens tänkte sig en version med par som jobbade över teamgränser med en person i varje team (Moore och Spens, 2008). Vi kallar medlemmar som är med i flera team för multimedlemmar.

Eftersom koordination handlar om att se till att flera aktörer arbetar tillsammans och inte divergerar i sitt arbete, är det viktigt att alla har någon uppfattning om vad andra håller på med. British Broadcasting Corporation (BBC) införde därför "End of Sprint"-demos, där alla team samlades och demonstrerade vad de hade åstadkommit den senaste sprinten. Det gjorde att det skapades gemenskap och att projektmedlemmarna kände sig delaktiga i projektet, vilket ledde till att motsättningarna mellan teamen minskades. (Lyon och Evans, 2008)

2.4 Teoretiskt ramverk

Den teori vi gått igenom behandlar olika men relaterade delar av uppsatsens syfte. Det saknas fortfarande en tydlig helhetsbild som kopplar samman de olika teoridelarna. Den avsaknaden kommer vi nu att fylla. Vi presenterar därför ett ramverk och illustrerar det i figur 2.3. Ramverket består av tre delar; integration, beroenden och koordination. Dessa tre punkter är starkt relaterade och dess inbördes påverkan kan ses i illustrationen: behovet av att integrera komponenter gör att de respektive teamen får ett högre beroende mellan sig och koordination är den aktivitet som hanterar dessa beroenden. Tillsammans med de tre punkterna illustreras också de faktorer som teorin sägs påverka respektive punkt.



Figur 2.3: Vårt teoretiska ramverk

Integration är att sätta samman komponenterna till en helhet, som vi såg i figur 1.1. Detta kan göras på flera sätt och vi har tagit upp integrationsmekanismer såsom *kontinuerlig integration*, att låta någon annan ta hand om det (*integrationsteam* och *release manager*) eller att ha en efterföljande *integrationsfas*.

Mängden beroenden mellan team kan påverkas av *funktionalitetsfokusering*, *blandade roller*, *justerade sprintar* och *samplacering*. Alla dessa bidrar enligt vår teori till att minska mängden beroenden mellan team. Detta i sin tur minskar enligt teorin behovet av koordination, eftersom koordination är hanteringen av beroenden.

Koordination sker på många nivåer, men denna uppsats behandlar främst koordinationen mellan team. Olika mekanismer för koordination finns, och mycket forskning har gjorts. Mintzbergs (1989) koordinationsmekanismer *Ömsesidigt anpassande*, *Toppstyrning* och *Standardisering* är tre, men från praktiska studier om stora agila projekt har vi också hittat *stormöten*, *multimedlemmar* och *teamedare*.

3 Undersökningsmetod

Här presenteras hur arbetet med denna undersökning har gått till. Vi tar upp våra teoretiska ståndpunkter och hur de påverkar vårt arbete och insamlingen av data. Syftet med att beskriva sin metod är replikation och evaluering (Backman, 1998). Därför försöker vi här beskriva vårt tillvägagångssätt så noga som möjligt och motivationen till varför vi valt att göra så.

3.1 Tillvägagångssätt

I denna uppsatsen har det utförts en *kvalitativ undersökning*, baserad på *intervjuer* med personer som arbetar med *Scrum* i stora projekt. Vi började arbetet med att sätta oss in i teorier i ämnet. En *litteraturstudie* gjordes där relevanta begrepp och teorier gick igenom. Grundad på våra kunskaper i ämnet tog vi sedan fram en *undersökningsfråga*. Därefter utvecklade vi ett *teoretiskt ramverk* som har använts för att ta fram intervjufrågor. För att belysa undersökningsfrågan från två perspektiv, valde vi att göra två intervjuer på varje företag, där intervjupersonerna från varje företag inte innehar samma roll. Alla intervjuerna, förutom epostintervjun, spelades in med en diktafon, detta med respondenternas samtycke.

Det gjordes en fullständig transkription av samtliga muntliga intervjuer. Dessa återfinns i Bilaga B där även epostintervjun finns med. Vi gjorde en löpande analys av intervjuerna, det vill säga att analysarbetet av intervjuerna påbörjades innan alla intervjuer var genomförda. Detta för att kunna bibehålla tankegångar och idéer som uppkom under intervjuerna (Patel och Davidson, 2003).

3.2 Intervjuguiden

I denna studien har det utförts fyra intervjuer av personer som arbetar med Scrum. Vi valde att utföra semi-strukturerade intervjuer, med öppna frågor, eftersom denna intervjuform ger en möjlighet till spontana och levande iakttagelser av världen vi ämnar undersöka (Kvale, 1997). Med epost kan man inte göra en semi-strukturerad intervju och därför innehåller den bara svar på precis de frågor vi ställt.

Vi utformade två intervjuguiden; en anpassad för ScrumMasters och produktägare respektive en anpassad för utvecklare.

3.3 Urval av respondenter

Vi har gjort intervjuer på två företag, som i den här uppsatsen döpts om till Företag A och Företag B i enlighet med intervjuetik (se avsnitt 3.4.1 nedan). Valet föll på Företag A och B eftersom de uppfyllde våra krav på projekt som använde Scrum med fler än ett team i.

Kontakten med Företag A togs via en person vi kände på företaget som både gav oss upplysningen om att de använde Scrum och att detta gjordes i större projekt. Via denna person förmedlades våra intentioner och vi fick kontaktinformation till en ScrumMaster som ville ställa upp, och via epost kunde vi boka en intervju. Det utfördes också en intervju med kontakten som uppfyllde kravet av att vara en teammedlem.

Företag B kontaktades via en mer omständlig kedja av kontakters kontakter. En väns pappas arbetskamrat rekommenderade Företag B som ledde oss till en av deras projektledare som jobbat med ett stort Scrum-projekt och som blev vår första intervjuperson där. Via denne fick vi sedan även kontakt med en utvecklare som vi intervjuade via epost.

3.4 Metodkritiska överväganden

3.4.1 Intervjuetik

Vi har följt rekommendationer från Kvale (1997) i så hög utsträckning som möjligt för vår intervjuetik. Det införskaffades ett samtycke till intervjun, där informanten informerades om syftet med undersökningen och dess upplägg. Vi skickade respektive intervjuguide till informanten i förväg för att denne skulle kunna förbereda sig för intervjun. Vid första intervjun (ScrumMaster, Företag A) skedde detta inte, vilket är beklagligt. Informanten fick också ta del av våra åtaganden för konfidentialitet. Vi valde att hålla deltagande företag anonyma då deras identitet inte var relevanta för vår undersökning. Vidare har även våra informanter fått förbli anonyma, så inga namn använda i uppsatsen är överensstämmande med de riktiga. Valet av en striktare konfidentialitet är i linje med Israel och Hay (2006) för att skydda informanten från att oönskade uppgifter blir offentliga.

3.4.2 Insamling, sammanställning och analys av data

De muntliga intervjuerna spelades in, vilket underlättar sammanställningen samt bidrar till att man inte missar några viktiga detaljer. Vid överföringen av intervjuerna från inspelningarna till skrift, transkriberingen, finns det en del ställningstaganden att göra. Skillnaden mellan talspråk och skriftspråk är en viktig aspekt att ta upp här. Enligt Kvale (1997) är översättningen mellan talspråk och skriftspråk som att översätta från ett språk till ett annat, detta på grund av de retoriska skillnaderna i språket. Transkriberingen kan därför ses som en tolkning av det inspelade intervjuerna. (Kvale, 1997)

Vi har valt att vara sparsamma med omskrivningar och tolkningar för att inte riskera att något försvinner i processen, men trots detta finns inte pauser och känslouttryck med i utskriften eftersom vi ansåg att de skulle bli onödigt långa. För att i så stor utsträckning som möjligt fånga fullständiga meningar har stakanden strukits. I epostintervjun har inga av ovanstående omskrivningar gjorts. För presentation i empiridelen har intervjumaterialet meningskoncentrerats. Meningskoncentrerings är ett förlopp i flera steg där intervjumaterialet på ett lämpligt sätt koncentreras för att bli mer kortfattat och är i relation till syfte och frågeställningar. (Kvale, 1997) I analysen har vi genom vårt ramverk belyst det som lyftes fram i empiridelen och diskuterat detta.

3.4.3 Källkritik

För att kunna bedöma sanningshalten i den litteratur man läser är det viktigt att inta en kritisk ställning till det man läser. Innebörden av källkritik är att man tar reda på var, när, och varför ett dokument producerats. Det är också viktigt att förstå författarens syfte och under vilka omständigheter dokumentet har framlagts. Det kan dessutom vara bra att veta vem författaren är och vilken bakgrund han har. (Patel och Davidson, 2003) Vi har tagit hänsyn till detta när vi letat källor för denna uppsatsen. Valet av källor för beskriva Scrum är i stor utsträckning hämtad från upphovsmakarna. Detta medför både att en tillförlitlig beskrivning av metoden kan göras men också att en brist gällande utebliven kritik riktad mot densamma föreligger.

Skalning av Scrum och andra agila metoder är ett nytt ämne, och det finns därför inte så många vetenskapliga studier i ämnet. En del empiriskt orienterade artiklar har skrivits om hur organisationer gått till väga för att lösa stora Scrumprojekt och deras erfarenheter av detta. På grund av det är även våra källor i ämnet väldigt nyttkomna. En annan konsekvens av källornas typ är att det inte finns någon konsensus om hur man ska gå till väga vid skalning, vilket leder till att det kan finnas motstridigheter i hur de olika artiklarna presenterar sina förslag.

3.5 Validitet & reliabilitet

För att tydligare motivera att undersökningen representerar vårt syfte och frågeställning, har vi utarbetat ett ramverk som används genom hela uppsatsen. Ramverket är skapat för att tydliggöra och sammanfatta hur koordination sker enligt teorin. Eftersom ramverket används genomgående vid både design av intervjuguide och analys av resultat, kan man tydligare se att analysen representerar det som efterfrågats i intervjuerna.

Vi har också använt oss av respondentvalidering (Kvale, 1997) också kallad kommunikativ validitet (Patel och Davidson, 2003), då vi låtit våra intervjupersoner för de muntliga intervjuerna ta del av transkriberingen av intervjun för att kunna identifiera feltolkningar från vår sida eller låta dem ändra uttalanden som de gjort av misstag. Detta ska bidra till att vår transkribering stämmer mer överens med intervjupersonernas meningar.

Reliabilitet utgör ett viktigt kriterium för bedömning av kvalitativa studier. Reliabilitet kan ses ur två perspektiv, extern och intern reliabilitet (Seale, 1999). Den externa reliabiliteten syftar till den mån i vilken studien kan replikeras, och det beror på hur väl man har beskrivit vilka teorier och metoder man använt i forskningsprocessen. Eftersom vår studie är kvalitativ är det svårt att hålla hög extern reliabilitet då varje intervjutillfälle i den är unikt och inte går att replikera. (Seale, 1999) Med intern reliabilitet menas enligt Kvale (1997) hur tolkning av information som samlas in görs. För att öka den interna reliabiliteten lät vi som sagt informanterna ta del av våra intervjutranskriberingar för att se till att de överensstämde med deras uppfattning. Intervjuerna transkriberades också direkt efter intervjutillfället för att säkerställa hög reliabilitet. (Kvale, 1997)

4 Empiriska resultat

I detta kapitel finns en presentation av intervjuerna vi genomfört på de båda företagen. Det är intervjuguidernas frågor som ligger till grund för uppdelningen av resultaten. Vi har dessutom gjort en gruppering av närbesläktade frågor för att göra presentationen mer överskådlig. Däremot så rubricerar vi inte svaren från de olika företagen under respektive del; det följer ordningen Företag A följt av Företag B.

Respondenterna på Företag A har vi valt att döpa om till Utvecklare A och ScrumMaster A och på Företag B till Produktägare B och Utvecklare B, i enlighet med våra åtaganden för anonymitet.

4.1 Presentation

4.1.1 Företag A

Företag A är ett stort företag som tillverkar konsumentprodukter. Vi gjorde båda intervjuerna på samma avdelning, och därför kommer "Företag A" i fortsättningen syfta på denna utvecklingsavdelnings Scrumprojekt, som arbetar med att förbättra utvecklingsverktygen för resten av organisationen. Utvecklare A:s team på denna avdelning arbetar med att anpassa utvecklingsverktyget Eclipse i ett samarbetsprojekt med ett annat företag. Just Utvecklare A jobbar mer specifikt med utökningar (plugins) för Eclipse.

4.1.2 Företag B

Företag B fick i uppdrag att utveckla en hemsida med webbshop åt ett konsumentproduktföretag. Projektet bestod totalt av 150 personer och utvecklingsarbetet höll på i knappt 2 år. Mjukvaruutvecklingen sköttes av en avdelning med 70-80 personer och eftersom det är denna utvecklingsavdelning vi är intresserade av i denna uppsats är det den som termen "Företag B" kommer att syfta till. Projektets produkt är nu i drift men det finns fortfarande aktivitet kring det. Projektet slutade använda Scrum som utvecklingsmetod efter 70

Produktägare B var formellt sett "lead system architect", vilket i praktiken innebar projektledare men med en fokus på det tekniska. I Scrum-termer hade han rollen som assisterande produktägare. Ansvarsområdet var uppdelat så att de tekniska frågorna ställdes till vår respondent medan frågor rörande kundnytta var den ordinarie produktägarens ansvarsområde.

4.2 Scrum-efterlevnad

Som man kan se i Tabell 4.1, används Scrum på båda företagen vi undersökte om än i olika utsträckning. På Företag A använder man metoden endast inom varje team medan man använder mer traditionell projektledning på nivån ovanför teamen. Företag B använder också Scrum inom teamen, men de hade dessutom en Scrum-of-scrums-struktur som täckte hela utvecklingsprojektet och hade som syfte att koordinera arbetet mellan teamen.

Tabell 4.1. *Huruvida Scrum används både inom och ovanför teamen.*

ScrumMaster A	Produktägare B
Alla teamen använder Scrum. Ovanför teamen är det bara två personer som arbetar, och de följer ingenting, utan de jobbar linjejobb.	Ja, förutom att teamen använder Scrum internt träffas varje ScrumMaster i en Scrum-of-scrums två gånger i veckan.

Förutom att undersöka på vilka nivåer Scrum används är det också intressant att se hur väl metoden följs. Eftersom vi endast undersöker Scrum-projekt är det viktigt att se hur väl metoden följs i organisationerna. En hög efterlevnad gör resultaten mer relevanta för oss. Tabell 4.2 visar hur informanterna ser på denna sak.

Tabell 4.2. *Hur väl Scrum följs.*

ScrumMaster A	Utvecklare A	Produktägare B	Utvecklare B
De har inget renodlat Scrum. De har inte Scrum-of-scrums, de har teamleads och de planerar Sprintarna väldigt tydligt.	De tycker de följer Scrum ganska väl. Det finns en person i teamet som tycker de ska följa det mycket medan ScrumMastern tycker de ska anpassa.	De följde Scrum och hade ScrumMasters, Scrum-of-scrums, Produktbacklogg, Sprintbacklogg och stort delegerat ansvar.	Inom teamen, ganska väl, dock gör kundens preferenser att det är svårt att implementera Scrum fullt ut.

4.3 Teamens organisering

När man vill undersöka hur teamen är organiserade framgår det av intervjuerna att områden såsom arbetsfördelning, beroenden, teamens sammansättning av medlemmar och hur teamen är placerade är intressant att titta på. När vi undersökt arbetsfördelning på företagen har vi fått information om hur arbetet är fördelat på teamen, ifall de anser att teamen är funktionalitetsfokuserade och hur fördelningen har skett. Informanternas svar på den första frågan presenteras i Tabell 4.3, och man kan se att teamen i allmänhet har separata arbetsuppgifter.

Tabell 4.3. *Vad teamen har för arbetsuppgifter.*

ScrumMaster A	Utvecklare A	Produktägare B	Utvecklare B
Nästan alla team äger ett eget verktyg. Vilket innebär att de är ansvariga för testning och underhåll av sin egen kod.	De jobbar med att utveckla Eclipse plugins för att förbättra utvecklingsmiljön för resten av företaget.	Teamen är uppdelade på funktionalitet. De gör en mobil webbsida, en vanlig webbsida, betalningslösning, statistik och rapportering samt grafisk design.	De jobbar med utveckling av den mobila webbsidan.

En relaterad fråga är ifall teamen är funktionalitetsfokuserade. Svaren på den här frågan, som presenteras i Tabell 4.4, har en tydlig likhet med svaren på den föregående frågan. Utifrån detta ser vi alltså att teamen har egna delar av systemet som de arbetar på. Produktägare B beskrev ingående hur teamen på Företag B var uppdelade på horisontella skikt där varje team arbetar med ett mjukvaruskikt i systemet; exempelvis affärslogik, databas, användargränssnitt och så vidare.

Tabell 4.4. *Teamens eventuella funktionalitetsfokusering.*

ScrumMaster A	Utvecklare A	Produktägare B	Utvecklare B
Teamen är uppdelade i funktionsgrupper som innebär att varje team jobbar med en del av koden i form av ett verktyg.	Målet är att göra utvecklingsmiljön bättre för resten av organisationen.	Generellt sett var de uppdelade på olika funktionalitet i systemet.	Ja, de är fokuserade på den mobila front-end applikationen.

Trots att vi nu har fått svar på hur arbetet är fördelat på teamen, saknas det fortfarande information om varför fördelningen gjorts på detta sätt. Det framgick av intervjuerna att företagen har använt olika principer för att sköta uppdelningen av arbete på sina team och informanternas svar på detta presenteras i Tabell 4.5.

Som vi sa tidigare kan teamens arbetsfördelning leda till att teamen får mer eller mindre beroenden sinsemellan. Därför har vi försökt ta reda på hur informanterna upplever beroenden mellan teamen, vilket presenteras i Tabell 4.6. Man kan utifrån detta se att informanterna beskriver mycket problem när de pratar om beroenden. I kontrast till detta beskriver Produktägare B statistik- och rapporteringsteamet på Företag B. De ses som

Tabell 4.5. *Ifall arbetsfördelning gjorts för att medvetet minska beroenden.*

ScrumMaster A	Produktägare B
Nej, det är ju helt enkelt uppdelat på verktyg. Visserligen kan det hända att teamen gör samma sak, fast uppdelat på sina respektive verktyg.	Skaparna av Scrum hävdar att teamen ska vara självorganiserande, och det var de också så det blev samma teammedlemmar som alltid jobbat tillsammans.

ganska isolerade och ställde små krav på resten av systemet, vilket ansågs bidra till att de presterade mycket bättre än de andra teamen.

Tabell 4.6. *Hur informanterna upplever beroenden mellan team.*

ScrumMaster A	Utvecklare A	Produktägare B	Utvecklare B
Oftast är teamen skilda åt men vissa team har stora beroenden sinsemellan. Exempelvis kan ett teams arbete baseras på ett annat teams och då måste de jobba väldigt tätt tillsammans.	Det finns ganska mycket beroenden mot andra team. Dessa beroenden kan leda till att man både gör samma sak eller går åt helt olika håll.	De horisontella skikten ledde till att det skapades enorma beroenden mellan team. Dessa beroenden gav upphov till ohanterliga problem när teamens olika byggen skulle integreras.	Utvecklaren beskriver hur teamet har beroenden mot andra team, särskilt mot det teamet som jobbar med så kallade backend-systemet. Beroendena leder till att alla team oftast behöver göra någon implementation för att få ny funktionalitet på plats.

En annan dimension när det gäller teamens organisering är medlemmarna i teamen. I vår undersökning ställer vi frågor om två sätt som struktureringen av medlemmar kan påverka beroenden mellan team; multimedlemmar och blandade roller. Informanternas svar på huruvida det finns medlemmar som har roller i fler än ett team kan ses i Tabell 4.7. Där kan man se att i båda företagen är det enstaka personer som är medlemmarna i flera team. I båda fallen handlar det om specialister och på inget av företagen har det i syfte att koordinera.

Tabell 4.7. *Multimedlemmar i teamen.*

ScrumMaster A	Utvecklare A	Produktägare B	Utvecklare B
Det finns i enstaka fall men syftet är inte att koordinera arbetet.	Det finns i vissa team och i de fallen handlar det om arkitekter.	Det finns enstaka personer som är medlemmar i flera team och det gäller framförallt specialister, men det upplevdes inte som koordinerande.	Det finns inga multimedlemmar.

Blandade roller är något vi ser som ett sätt att organisera medlemmar för att minska beroenden mellan team. Som man kan se i Tabell 4.8 framgick det av vår undersökning att

ingen av informanterna tyckte att de hade blandade roller i sina team. Trots detta säger ScrumMaster A att utvecklarna var ansvariga att testa och integrera sin egen kod, vilket kan ses som att de besitter flera roller. Även om Företag B sa att de inte använde blandade roller är det viktigt att notera de försökte införa denna typ av organisering.

Tabell 4.8. *Blandade roller i teamen.*

ScrumMaster A	Utvecklare A	Produktägare B	Utvecklare B
Det finns bara utvecklare i teamen men ansvarsområdet för dessa är även att testa sin kod. Teamen integrerar även sin egen kod.	Det finns bara utvecklare i teamen men de har självklart olika specialkompetens.	De hade inte blandade roller men försökte införa det, dock utan framgång.	Alla i teamet har i stort sett samma kompetens.

Den sista aspekten avseende teamens organisering är hur teamen är placerade. Teamens samplacering på företagen kan ses i Tabell 4.9. Här kan man se att på Företag B finns alla teamen på samma ställe, medan Företag A har mer utspridda team, även om de som finns på samma ort är samplacerade.

Tabell 4.9. *Teamens samplacering*

ScrumMaster A	Utvecklare A	Produktägare B	Utvecklare B
Det sitter fyra team i Lund, ett i Karlstad och fem i Brasilien.	De teamen som finns i Lund sitter tillsammans.	Teamen sitter tillsammans på ett helt våningsplan.	Alla team sitter ganska nära varandra på samma våningsplan.

4.4 Integration

När vi undersökt hur företagen hanterar integrationen av respektive teams resultat har vi ställt frågor om kontinuerlig integration och om rutiner kring integration.

Företagens svar på hur kontinuerlig integration används presenteras i Tabell 4.10 nedanför. I Företag B användes det redan från start av projektet medan i Företag A fanns det ett pågående arbete med att införa det.

Tabell 4.10. *Hur kontinuerlig integration används.*

ScrumMaster A	Utvecklare A	Produktägare B	Utvecklare B
Kontinuerlig integration används inte fullt ut än.	Teamet har inte börjat med det än men planerar att göra det snart.	Det ställdes krav på att kontinuerlig integration skulle användas redan från början.	Det används.

I Tabell 4.11 presenteras i mer detalj hur integrationen går till på respektive företag. Man kan se att på Företag A var utvecklarna själva ansvariga för att integrera sin kod i sprinten. På Företag B kom detta ansvaret att ligga på ett enskilt team, kallat integrationsteam, som upprättades i ett senare stadie.

Tabell 4.11. *Hur integrationen sker i företagen.*

ScrumMaster A	Produktägare B	Utvecklare B
Det finns ingen integrationsfas utan varje sprint är ansvarig för att integrera sitt eget resultat.	Det skapades ett integrationsteam som var ansvariga för att integrera de andra teamens resultat.	Det finns resurser för att bygga och paketera systemet.

4.5 Koordinationsaktiviteter

För att hantera beroenden mellan team krävs det koordination av deras arbete. I vår undersökning har vi försökt ta reda på vilka olika aktiviteter företagen använder för att koordinera arbetet mellan team. Vi har dels ställt frågor om specifika koordinationsaktiviteter som stöds av våra teoretiska utgångspunkter, men också försökt finna andra aktiviteter genom att ställa öppna frågor om hur koordination och kommunikation sker.

Det förekom stormöten på båda företagen i olika form och omfattning. Företag B hade stormöten ungefär en gång i månaden. Där samlades alla utvecklarna för att berätta hur deras respektive arbete fortskred. Tabell 4.12 visar vad informanterna har sagt om stormöten. Av denna tabell framgår också att det fanns stormöten i Företag A.

Tabell 4.12. *Stormöten i projekten.*

ScrumMaster A	Utvecklare A	Produktägare B	Utvecklare B
Det finns stormöten där man diskuterar vad som händer i sprintarna.	Avdelningsmöte en gång i veckan, ibland visas demos på någon viktig funktionalitet. Teamen berättar för varandra vad de arbetar med.	Olika former av stormöten har funnits på en oregelbunden basis.	Nej, det fanns inga stormöten.

En av de specifika koordinationsaktiviteter som vi undersökt är ifall företagen haft justerade sprintar, och resultaten från detta visas i Tabell 4.13. Utifrån den kan man utläsa att Företag A inte hade tidsjusterade sprintar, medan Företag B hade det på alla team förutom ett.

Tabell 4.13. *Justerade sprintar.*

ScrumMaster A	Utvecklare A	Produktägare B	Utvecklare B
På grund av att Scrummasters har flera team var hade de inte hunnit med att sprintarna startade och slutade samtidigt.	Det finns inte stora beroenden mellan teamen och därför behöver inte deras sprintar vara justerade.	Sprintarna är tidsjusterade.	Alla team utom ett hade samma tidslinje på sina sprintar. Det teamet är speciellt då det inte har beroenden mot övriga team på samma sätt.

4.5.1 Kommunikation mellan team

I våra intervjuer har vi ställt frågor om hur kommunikationen mellan team förs. Intervjувaren angående detta har vi kategoriserat i följande kategorier; allmänt, förmedling av dokumentation, löpande arbetsfördelning och kommunikationsproblem.

Från Tabell 4.14 ser vi att det förekommer mycket kommunikation mellan team på båda företagen. Det framgår också från intervjuerna att den mesta kommunikationen sker informellt utan bestämda rutiner. Utvecklarna kan alltså kommunicera med varandra över teamens gränser spontant. Utifrån intervjuerna framgick det också att tre av fyra informanter nämnde beroende i samband med sin förklaring till varför de kommunicerade.

Tabell 4.14. *Informanternas syn på kommunikation mellan team.*

ScrumMaster A	Utvecklare A	Produktägare B	Utvecklare B
Utvecklarna kommunicerar självklart med varandra mellan team för det finns ett beroende ändå.	Teamen är absolut beroende av varandra och detta löser man genom att ha kontinuerlig dialog och prata mycket.	Mål och vision saknas i Scrum och bristande kommunikation gjorde att teamen jobbade åt olika mål.	Ganska mycket tid går åt till att kommunicera eftersom de olika teamen har beroenden mot varandra och måste synkronisera sitt arbete. Detta kan ske via e-mail, messaging eller ansikte mot ansikte.

En specifik form av kommunikation sker när ett team ska använda ett annat teams kod för sitt eget utvecklingsarbete. Här krävs det att det förmedlas hur denna kod fungerar för att teamet i fråga ska kunna använda den; det vi kallar förmedling av dokumentation. Tabell 4.15 visar hur de tillfrågade utvecklarna beskriver hur denna förmedling går till. Ett tydligt mönster som syns här är att båda utvecklarna i första hand använder skriftlig dokumentation och om det inte finns prata med den ansvarige utvecklaren.

Tabell 4.15. *Förmedling av dokumentation.*

Utvecklare A	Utvecklare B
Det brukar finnas ganska mycket dokumentation, men finns det inte så kan man ta reda på det genom att fråga utvecklaren.	Det fanns en wiki där viss del av dokumentationen lades ut, men mestadels skedde det via informell kommunikation.

Något annat som framgick under intervjuerna och som anknyter till kommunikationen mellan team är hur de hanterar förmedling av krav. Detta sker när ett team måste utveckla en viss funktionalitet innan ett annat team kan fortsätta sitt arbete. I Tabell 4.16 presenteras hur denna process fungerar i båda företagen.

Här saknas ett tydligt mönster i hur informanterna svarat, det beskrivs både planerande och informella metoder för hur förmedling av krav går till. Det framgår också av intervjuerna på båda företagen att informanterna ansåg att eftersom teamen hade väl definierade ansvarsområden så var det ofta lätt att avgöra vem som skulle göra vad.

Tabell 4.16. *Förmedling av krav mellan team.*

ScrumMaster A	Utvecklare A	Produktägare B	Utvecklare B
Det planeras ingående vad som ska göras i varje sprint. Även leveranser av funktionalitet mellan sprintar planeras i detalj.	Ifall ett team vill ha något levererat från ett annat team kan man först gå direkt till utvecklarna och kontrollera rimligheten. Sedan kan man ta det den officiella vägen via sprintbackloggen eller Scrummastern.	Det var produktägarens roll att lyfta fram det som hade högst prioritet vid sprintplaneringsmötena.	Eftersom teamen är fokuserade på olika delar är det ofta rätt tydligt vilka team som ska göra vad. Ofta måste alla team vara inblandade både i diskussion och implementation.

Den fjärde och sista kategorin för kommunikation mellan team behandlar de problem som informanterna upplevt. Dessa presenteras i Tabell 4.17 där man kan se att det råder delade uppfattningar på respektive företag om vilka problem som finns.

Tabell 4.17. *Kommunikationsproblem mellan team.*

ScrumMaster A	Utvecklare A	Produktägare B	Utvecklare B
Det har säkert hänt, men har inget konkret exempel.	Det uppstår helt klart kommunikationsproblem, exempelvis om ett annat team ändrar på något som vårt verktyg använder utan att vi får reda på det.	Det fanns samarbetsproblem där teamen bevakade sina egna intressen gentemot varandra.	Har inte märkt något.

5 Analys och diskussion

I detta kapitel förs en diskussion av de empiriska resultat som presenterades i förra kapitlet. Diskussionens fokus ligger på att anknyta resultaten till vårt teoretiska ramverk. Tillvägagångssättet för detta är att utskönja skillnader och likheter mellan resultaten och teorin samt att diskutera anledningarna bakom detta.

5.1 Beroenden

Det framgår av resultaten att båda företagen upplever att det finns beroenden mellan team i deras organisationer. De upplever vidare att dessa beroenden leder till problem i utvecklingsarbetet och behov av att teamen kommunicerar med varandra. De teorier om beroenden vi presenterat förutsäger att aktörer som har höga beroenden sinsemellan har större behov av att koordinera sitt arbete. Det är vår tolkning att den kommunikation som informanterna anser att det finns behov av, har som syfte vara just koordinerande och utifrån detta ser vi dessa resultat i linje med rådande teorier.

Hur teamen är organiserade kan behandlas på två nivåer i vår undersökning, dels hur teamen fungerar sinsemellan och dels hur teamen fungerar internt. Oavsett på vilken nivå analysen förs, handlar den om beroenden mellan aktörer. Därför kommer detta avsnitt att ha ett tydligt tema – nämligen hur organiseringen av team påverkar dess beroenden mot andra team. Att veta mer om hur beroenden uppstår är relevant, då de teorier om koordination som presenterats visar på att en minskning av beroenden mellan aktörer även innebär en minskning av deras behov av att koordinera sitt arbete.

5.1.1 Teamens organisering

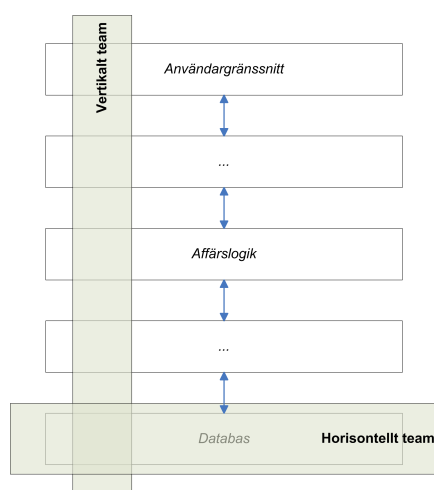
I intervjuerna med informanterna på Företag A framgår det att varje team ansvarar för ett verktyg var och att de sällan delar ansvarsområden. Detta är ett angreppssätt som liknar "feature crew"-modellen (Miller och Carter, 2007), vilket går hand i hand med den gruppering av aktiviteter som Malone och Crowston (1994) presenterar som metod att minska beroenden. Det får även stöd i den Scrum litteratur vi presenterat, där teamen själva ska vara kompletta nog att kunna producera en hel bit av användarfunktionalitet (Schwaber, 2004). Denna form av organisering borde minska beroenden mellan team, och utifrån resultaten så har Företag A också låga beroenden mellan sina team. Mot detta är det intressant att ScrumMaster A berättar att hur uppdelningen av utvecklare i team skedde inte hade något uttryckt syfte att minska mängden beroenden. Denne uttrycker dock att det finns många andra parametrar att ta hänsyn till än beroenden mellan team.

Företag B lät utvecklarna självorganisera sig till team, då självorganisering av teamen och att ge mer ansvar åt utvecklarna ansågs vara något som Scrum förespråkar. Denna självorganisering ledde till att teamen bestod av utvecklare med samma kompetens och roller; databasteam, grafiskt team, backendteam med flera, alltså en struktur som står i kontrast till den tvärfunktionella arbetsgrupp som Scrum beskriver och baseras på. Från denna följd av händelser kan det verka som om Scrum innehåller en motsägelse; självorganiseringen kan leda till strukturer som strider mot Scrums andra principer.

En återgång till den litteratur som definierar Scrum kan hjälpa till att reda ut begreppen. Självorganisering och utvecklaransvar är i linje med vad både Schwaber (2004) och Sutherland (2005) anser att Scrum bör förespråka, men med två skillnader. Dels beskrivs självorganiseringen snarare som ett självstyre inom teamet än en verklig organisationsförändring, och dels skriver Schwaber (2004) om hur man bör gå in och detaljstyra när självorganiseringen inte uppfyller de krav som ställs. Dessa två punkter verkar vara skillnader mellan hur Företag B tillämpat Scrum och hur metoden beskrivits.

Teamen på Företag B var alltså fokuserade på varsin del av systemet, men enligt vår tolkning är dessa team inte funktionalitetsfokuserade. Anledningen till detta är att de är fokuserade på en *funktion* och inte en *funktionalitet*. Denna skillnad formulerades av Produktägare B på ett annat sätt; genom att kalla det sätt som teamen var organiserade på för horisontella team, i kontrast till vertikala team. Vår tolkning av denna formulering illustreras i figur 5.1, där mjukvarusystemet i fråga visas bestående av horisontella skikt och de horisontella respektive vertikala teamen.

Den första observationen vi kan göra från en sådan struktur är att varje skikt är byggt på skiktet under, vilket leder till att det finns höga beroenden mellan dessa. En andra observation är att för att utveckla en bit användarfunktionalitet måste man involvera alla mjukvaruskipten. Dessa två observationer motiverar också var för sig teamens funktionalitetsfokusering.



Figur 5.1. Vertikala och horisontella team

På Företag B hade man team som var grupperade på roller och inte blandade roller som Scrum föreskriver. Problemen med en sådan organisering visar sig speciellt vid varje ny funktionalitet som ska utvecklas då flera team måste vara inblandade och göra olika delar, istället för att varje team kan ta eget ansvar för sitt arbete. Kompetensen i ett team räckte alltså inte för att bygga en funktion.

5.1.2 Blandade roller

En analys av huruvida teamen hade blandade roller eller inte beror på vilken definition man använder. Både Företag A och Företag B hade endast utvecklare i sina team, men alla

utvecklarna hade egna spetskunskaper. På Företag A hade teamen även ansvar för att testa sin egen kod. Då testare och utvecklare kan ses som separata roller kan även teamet ses som rollblandat. För att reda ut begreppen erinrar vi oss syftet med rollblandning; att kompetensen för att producera en hel bit av användarfunktionalitet ska finnas inom teamet. På grund av detta ser vi definitionen av rollblandning som helt beroende på arbetsuppgiften i fråga. Om det bara behövs utvecklare med snarlik kompetens, eller en blandning mellan utvecklare och grafiska designers, är helt olika från projekt till projekt.

Eftersom vi nu vidgat begreppet rollblandning till att inte alls kräva flera roller, anser vi att begreppet inte har något med sitt namn att göra längre. Det har i själva verket samma syfte som begreppet vertikala team – att all kompetens för att utveckla funktionalitet ska finnas inom teamet. Detta är dessutom samma definition som funktionalitetsfokusering. Därför bör dessa tre begrepp ses som ett och samma.

5.1.3 *Samplacering*

Vi såg samplacering som starkt bidragande för att underlätta koordination mellan team, men Företag A verkade hantera sina utspridda team på ett effektivt sätt genom att ha väldigt nära kommunikation med dem på flera sätt. På Företag B beskrivs det hur utvecklarna spontant går och pratar med varandra vid behov, därför kan man se att samplacering underlättar för den informella kommunikationen. Samplacering och genom det effektivare kommunikation är också något som Scrum förespråkar. (Avison och Fitzgerald, 2006)

Dessa exempel verkar inte ge något enhetligt tecken på hur samplacering påverkar beroenden mellan team. Det ena företaget var helt samplacerade medan de andra inte var det, men utifrån detta kan man inte avgöra huruvida samplacering används som en mekanism för att minska beroenden eller om det bara är en organisatorisk fråga.

5.1.4 *Justerade sprintar*

Då informanterna beskrivit varför de har eller inte har justerade sprintar, har de också nämnt beroenden. På Företag B var alla teams sprintar förutom ett tidsjusterade och det teamet beskrevs som att ha låga beroenden mot de andra teamen. Företag A hade inte tidsjusterade sprintar alls, och en av anledningarna var att teamen inte hade så mycket beroenden sinsemellan. Dessa beskrivningar antyder att team har tidsjusterade sprintar endast om de är beroende av varandra. Syftet med tidsjusterade sprintar är att skapa gemensamma tidpunkter för integration (Lyon och Evans, 2008). Om sprintarna inte har så mycket beroenden mellan sig, blir det därför inte längre lika viktigt med tidsjustering.

5.2 **Koordinerande roller**

Vi har presenterat begreppet teamledare som ansvariga för att spåra arbete som hade påverkan utanför teamet. Företag A hade en roll i varje team som kallades för teamlead. De förklaras som ansvariga både inåt teamet, för att bestämma vem som ska göra vad, och

utåt, för att kommunicera resultat. Denna yttre kommunikation passar mot våra teoretiska teamledares uppgift. De var till exempel ansvariga för att visa upp det resultat som producerats i sprinten vid avdelningsmötena. Teamledare enligt vår teori används alltså i Företag A för att koordinera arbete.

I vår teori och vårt ramverk fyller multimedlemmar en koordinerande funktion. Då de är medlemmar i mer än ett team har de möjligheten att agera som informationsbärare mellan team. Båda företagen hade olika former av multimedlemmar, men det visade sig vara i mycket liten utsträckning och att syftet inte var koordinerande utan det var snarare bara en ren personalfråga.

5.3 Koordinationsaktiviteter

5.3.1 Stormöten

Båda företagen är positiva till hur deras respektive stormöten fått utvecklarna att förstå vad de andra håller på med för tillfället. Ibland har det också förekommit demos där ett team har visat upp vad de gjort under senaste sprinten. Detta stämmer väl med den teori vi tagit upp angående stormöten och “End of sprint”-demos, som bidrar till en gemenskap, med projektet i fokus snarare än teamet självt (Lyon och Evans, 2008).

Det är anmärkningsvärt att Utvecklare B säger att de inte har stormöten, medan Produktägare B påstår motsatsen. Det kan tyda på att antingen så har Utvecklare B helt enkelt inte varit närvarande på de stormöten som varit, eller så finns det skilda åsikter om vad ett stormöte är för någonting. Vi väljer i vilket fall att lita på Produktägare B och antar att de faktiskt haft stora möten. Detta tyder på att stormöten är en koordinationsaktivitet som används av båda företagen.

5.3.2 Detaljerad planering

Företag A använde en detaljerad veckovis planering för att hantera överlämningar mellan team mitt i sprintar. Detta är en form av koordination som vi tycker främst kan kopplas till koordinationsmekanismen *standardisering av resultat* (Mintzberg, 1989). Man standardiserar i förväg vad som ska levereras och när, vilket ger någonting konkret att koordinera arbetet kring. Förutom detta går det även att göra liknelser med Mintzbergs (1989) andra mekanismer. Det är på sätt och vis *toppstyrande* för att det bestäms i förväg, även om utvecklarna gör det själva tillsammans med ScrumMastern och produktägaren. Man skulle här kunna se utvecklarna som sina egna chefer, vilket kanske snarare skulle föra tankarna till *ömsesidigt anpassande*, men det sker inte kontinuerligt under arbetets gång, utan före, vilket gör att inte heller detta passar helt in.

5.3.3 Ömsesidigt anpassande

Enligt Mintzberg (1989) är ömsesidigt anpassande den koordinationsmekanism som används både i väldigt enkla och i väldigt komplicerade organisationer. Den bedrivs genom

att aktörer kommunicerar informellt mellan sig för att anpassa sitt arbete på ett koordinerande vis. Eftersom denna informella kommunikation är den form som också det ursprungliga Scrum förespråkar, kan man förvänta sig att ömsesidigt anpassande spelar en stor roll även i stora Scrumprojekt. Detta är också just vad vi funnit. Den form av koordination och kommunikation som beskrivits som den vanligast förekommande i våra intervjuer, är just den informella kommunikationen mellan utvecklare.

Det är främst två fall där den informella kommunikationen framträder. Dels är det när ett team önskar att ett annat team ska utveckla en viss funktionalitet för att man behöver det i sitt framtida arbete. Dels är det när man vill använda ett annat teams redan utvecklade funktionalitet och behöver få veta hur det fungerar.

I det första fallet har det beskrivits att utvecklaren börjar med att ta kontakt med en utvecklare i det andra teamet för att kontrollera rimligheten i önskemålet. Ifall man får ett godkännande kan man sedan föra fram önskemålet på ett mer formellt sätt. Alltså börjar man med informell kontakt, för att sedan göra en formell förfrågan.

Det andra fallet, när man vill använda redan utvecklad funktionalitet, beskrivs som att ifall det finns dokumentation för funktionaliteten använder man den och annars frågar man det ansvariga teamet. Detta tillvägagångssätt börjar alltså med en formell eftersökning av dokumentation, och i andra hand en informell kontakt. Då det ofta saknades formell dokumentation blev den informella kontakten den vanligaste formen.

Dessa två fall handlar båda om att anpassa sitt arbete mot andra team och i båda fallen är den informella kommunikationen starkt representerad. Detta tyder på att ömsesidigt anpassande är en koordinationsmekanism som används av båda företagen.

5.4 Scrum

Utifrån den diskussion som förts i detta kapitel, ser vi ett ytterligare mönster. De principer för kommunikation och organisering av team som förespråkas av Scrum återfinns både i teorin och empirin. Kommunikation sker till stor del informellt inom team i ursprungsmetoden och det är även vad som sker till stor del mellan team i undersökningen. Organiseringen av vertikala team stöds också både av metoden, koordinationsteorin och empirin. Med detta som underlag ser vi att metoden har flera kvaliteter som är applicerbara för att minska beroenden och underlätta koordination mellan team i stora projekt.

6 Slutsatser

I denna uppsats har vi haft som mål att besvara frågeställningen:

Hur koordineras arbetet mellan team i stora Scrumprojekt?

Genom våra intervjuer och analys av dessa har vi nått fram vid en uppsättning slutsatser som ska besvara denna frågeställning.

En första övergripande slutsats är att båda företagen vi undersökt har beskrivit att de upplever att deras team har beroenden mellan sig, och att koordination är problem som de tycker är relevant för systemutveckling. De uttrycker också åsikter att koordination behövs mellan team som är beroende av varandra. Det tyder på att vi faktiskt har undersökt ett fenomen som upplevs som viktigt i dessa utvecklingsorganisationer.

Principer i Scrummetoden kan understödja de strukturer och mekanismer som främjar koordination och minskning av beroenden i stora projekt. Organiseringen av vertikala team och den informella kommunikationen är begrepp som återkommer i såväl ursprungsmetoden, som koordinationsteorin och det empiriska underlaget.

Beroenden

Om man jämför organisationen av team i Företag A mot Företag B och tar hänsyn till vad Produktägare B ansåg om vertikala och horisontella team, verkar det som om vertikala team bidrar till minskade beroenden mellan team. Detta stämmer dessutom överrens med vårt ramverk. Våra resultat tyder därmed också på att det blev höga beroenden när man organiserade team i horisontella skikt. Det krävdes insatser och samarbete mellan flera team för varje bit funktionalitet som skulle byggas.

Scrum som metod föreskriver att team ska vara självförmedlande på så vis att de själva som en grupp kan producera funktionalitet (Schwaber, 2004). Detta har samma syfte som den vertikala organiseringen av team har. Därför drar vi slutsatsen att Scrums föreskrifter i detta fall går hand i hand med en *vertikal organisering av team* och därigenom en minskning av beroenden mellan team.

Beroenden mellan team påverkar enligt vår tolkning även ifall deras sprintar är *tidsjusterade*. Det verkar som om de team vars sprintar är tidsjusterade mot varandra, är de team som har höga beroenden sinsemellan, medan team som har lägre beroenden har mer självständig tidssättning av sprintar. Detta verkar rimligt då de team som är beroende av varandra har mest behov av att synkronisera både arbete och leverans.

Koordinationsmekanismer

Som i vanliga Scrum används mycket informell kommunikation för att koordinera mellan teamen. Detta anser vi är en form av ömsesidigt anpassande. Det innebär att utvecklarna tar eget ansvar för att koordinera sitt arbete, även mellan teamen. Men ömsesidigt anpassande är inte alltid tillräckligt. Om ett team vill ha en viss funktionalitet av ett annat team, kan informell kontakt för att kontrollera rimligheten i förslaget vara en början för utvecklarna. Men för att förslaget ska gå igenom krävs att man går den officiella vägen via

ScrumMastern. Det tyder på att koordination kan ske som en kombination av *toppstyrning* och *ömsesidigt anpassande*.

Standardisering av resultat återfinns också som en koordinerande mekanism, som vi sett, i den detaljerade planering som Företag A gör. Det minskar behovet av löpande koordination under arbetets gång, eftersom det görs en planering av arbetet innan sprinten påbörjas. Med detta görs överenskommelser om hur, när och vad som ska levereras mellan team i den kommande sprinten.

Båda företagen hade och var positiva till *stormöten*. Det medförde att teamen fick koll på vad de andra höll på med. Detta stämmer överens med våra teoretiska utgångspunkter och bör därför ses som en koordinationsmekanism som används av företagen.

Teamledarna i Företag A fungerade lite som koordinatörer eftersom de ansvarade för att presentera resultat mot andra team. De stämde dock inte helt överens med de teamledare vi presenterade i vårt teorikapitel, men får fortfarande anses ha en viss koordinerande roll.

6.1 Slutord

Det finns resultat som inte stämmer med vår teori. Det verkar exempelvis som om de multimedlemmar vi tog upp i teorin inte kunde återfinnas i något av företagen. De verkar alltså inte ha haft en koordinerande roll i något av projekten. Inte heller från företagets samplacering kunde vi göra en tydlig koppling till vår teori, även om det ena företaget satt på samma geografiska plats.

På grund av tidsbrist utfördes den sista intervjun via epost, vilket gör att svaren inte innehåller samma djup av uttryck som en muntlig intervju ger. Det verkar också som om vi fått vissa motstridiga uttalanden, särskilt på Företag B, exempelvis vad gäller stormöten. Där har vi valt att lita mer på produktägaren än utvecklaren, då utvecklaren kanske helt enkelt inte var närvarande eller medveten om dem.

Det kan också kritiseras att vi endast undersökt två företag. Anledningen var att vi ville få mer djup än bredd i undersökningen, och exempelvis upptäcka ifall intervjupersonerna uppfattar saker på olika sätt. Att ha intervjuat både en utvecklare och en annan roll tycker vi har bidragit till detta djup. Det hade varit intressant att göra samma undersökning fast i annan form, både att gå ännu mer på djupet i ett företag och göra en fallstudie, eller att göra en kvantitativ studie med många företag.

Andra saker som hade varit intressanta att fokusera på, som vi först efter uppsatsens genomförande upptäckt, är exempelvis att undersöka mer ingående hur teamens struktur påverkar beroenden och koordination. Exempelvis kan det finnas många andra konfigurationer än de vertikala och horisontella vi funnit. Det hade också varit spännande att se hur organisationer hanterar och inför kontinuerlig integration, eftersom båda företagen vi undersökt verkade ha problem med det.

Bilaga A Intervjuguider

A.1 ScrumMaster/Produktägare

A.1.1 Inledande frågor

- Hur många team har ni i ert projekt?
- Hur stora är teamen?
- Vad gör de olika teamen? Arbetsuppgifter!
- Finns det beroenden mellan några speciella team?
- Använder alla Scrum?
- Använder ni bara Scrum inom teamen eller även mellan/ovanför?
- Hur väl anser du att ni följer Scrum? (Inom och mellan team)
- Hur är ScrumMasters och produktägare fördelade på teamen?
- Är teamen som du har mycket mer integrerade än andra team?

A.1.2 Koordineringsfrågor

- Har ni medvetet delat upp arbetet mellan teamen för att minska deras integration?
 - Vem gjorde det? Utvecklare / Toppstyrt?
- Har ni stormöten med alla utvecklare?
- Använder in kontinuerlig integration? Detaljera!
- Har ni multimedlemmar?
- Har teamen blandade roller?
- Är teamen fokuserade på en specifik feature?

A.1.3 Överlappande arbete

- Händer det att team “trampar varandra på fötterna” – att de gör samma sak?
- Händer det att team har gjort saker som hindrat andra team?
- Vad har produktägaren för roll i det?
- Har team hindrats av att andra team inte blivit klara i tid?

A.1.4 Påverkande arbete

- Hur sker arbetsfördelning? Om en stor ny uppgift kommer in, vem bestämmer vilket team som ska göra vad av det? Att två team måste komma överens hur man ska göra en gemensam uppgift?
- Två team måste komma överens hur man ska göra en gemensam uppgift?
- Händer det? Hur?

A.2 Utvecklare

A.2.1 Inledande frågor

- Hur stort är ditt team?
- Hur många team finns det?
- Vad gör ditt team? Arbetsuppgifter.
- Hur väl anser du att ni följer Scrum? (Inom teamen och mellan teamen)
- Hur är ScrumMasters och produktägarefördelade på teamen?
- Hur sitter ni i era lokaler?

A.2.2 Koordineringsfrågor

- Är ert team fokuserat på en specifik feature?
- Hur mycket tid lägger du på att kommunicera med utvecklare i andra team?
- Om ni är beroende av något annat team, på vilket sätt vet ni hur man använder deras produkt? Informellt, dokumentation, annat? Händer det alls?
- Har ni stormöten med alla utvecklare där ni får reda på vad de andra håller på med?
 - Funkar det bra? Hjälper det?
- Använder ni kontinuerlig integration? Hur? Detaljera!
- Har ni multimedlemmar?
- Har ert team blandade roller?

A.2.3 Överlappande arbete

- Finns det något annat team som har med er att göra / som ni har tydligt beroende av?
- Händer det att team “trampar varandra på fötterna” – att de gör samma sak?
- Händer det att team har gjort saker som hindrat andra team?
- Har team hindrats av att andra team inte blivit klara i tid?

A.2.4 Påverkande arbete

- Hur sker arbetsfördelning? Om en stor ny uppgift kommer in, vem bestämmer vilket team som ska göra vad av det? Att två team måste komma överens hur man ska göra en gemensam uppgift?
- Händer det? Hur?

Bilaga B Intervjuprotokoll

B.1 ScrumMaster, Företag A

Genomfördes 2008-11-20

MM Max Martinsson

SA ScrumMaster A

MM Vi kan ju börja med att du beskriver vad du jobbar med.

SA[1]: Ja, jag jobbar som FGL - Funktionsgruppsledare. Här på mjukvarusidan av Företag A är vi uppdelade i funktionsgrupper. Det innebär att man sysslar med en del av mjukvaran. På det andra stället [i närheten] där man jobbar med själva produkterna är det mer att man jobbar i ett projekt och följer det och jobbar med kanske mekanik eller annan teknik. Här så jobbar vi i funktionsgrupper för att som jag förstår det att låt oss säga att du jobbar på en specifik del.. mycket återanvänds ju, hittar du ett fel i ett av mjukvaruspåren så ska du ju göra rättningar i alla. Istället är man grupperade och jobbar i en linje och rapporterar till projekten, så man är en linjefunktion med en chef och en avdelning och jobbar mot projekten. Men inte inne i projekten, man jobbar mot allihopa samtidigt.

MM Och projekten är olika produkter?

SA[2]: Ja de är olika mjukvaruversioner kan man säga, som körs på olika produkter. Vi delar in dem i olika heart-beats. Så vi jobbar mot allihopa hela tiden. Så mitt jobb som FGL. Det traditionella jobbet som FGL är att styra de här linjeavdelningen, och hjälpa dem att synka mellan vad de gör och projekten. För de passerar olika milstoppar och det är vårt jobb att hålla reda på det. Utvecklare... ja vissa har koll på det, men de flesta skärmar av sig lite och jobba på med funktionaliteten och sen får jag skynda på dem "nu är vi snart inne i den här milestonen, då måste detta vara klart". Och i och med att man då levererar till så många olika projekt, så är det många datum och milestones att passera. Och det ska in en massa dokumentation och det är sånt som vi gör. Men eftersom att vi är inte trad. funktionsgrupp som andra som har hand om produktkod utan vi har Development Environment – utvecklingsmiljö. Allt från emulatorer, som då ska emulera produkter för att du ska kunna programmera mot en emulator som är likadan som produkten. Och sen är det själva byggsystemet, CME...

MM Och CME är ..

SA[3]: Ja, det är samma sak som SourceSafe och dem. Det är versionshantering av koden. Att checka in och ut. Och det är ju ett monsterbygge i sån här produkt, för alltihopa byggs ihop till en enda klump. Så, vi äger det. Och så lite olika kvalitetsverktyg och sådär för att man ska förbättra koden. Vad jag skulle säga var att de andra de jobbar lite mer som vattenfallsmetoden, men i och med att vi är så, våra miljö måste vara färdig innan de andra kan börja koda. Och det kan snabbt dyka upp krav, som t.ex. att en viss produkt ska stödja X eller en viss feature. Och detta kan ske rätt snabbt och då måste vi snabbt ändra fokus. Så därför passar det oss väldigt bra att jobba i Scrum. Då kan vi ha korta iterationer, snabbt kunna ändra fokus. Men också att, ja vi brukar köra en-månaders-iterationer för att hela tiden kunna ändra fokus, men ändå ge teamet en stund att koppla bort omvärlden. Och så är det så många olika saker så vi har många olika team, och då passar det rätt bra att iterera. Så mitt jobb är att vara ScrumMaster, och då måste man ha den här kontakten med projekten för det är ju de som kommer med kraven. Jag och min kollega Henrik är ju koordinatörer, kan man verkligen kalla oss.

MM Och Henrik jobbar som ...

SA[4]: Också som FGL, vi har samma tjänst liksom.

MM Hur många team finns det?

SA[5]: I vår avdelning så finns det 4 i Lund, 1 i Karlstad och 5 i Brasilien.

MM Som jobbar med utvecklings..

SA[6]: Ja.. Vi har delat upp det lite olika. Jag har just nu fyra team, och Henrik har två, två som Olof har, och två som en annan avdelning faktiskt har för tillfället.

MM Så du är också ScrumMaster för dem i Brasilien?

SA[7]: Ja

MM Hur fungerar det med kommunikationen?

SA[8]: Det funkar bra. Vi har lagt upp det så att de har en lokal ScrumMaster i Brasilien, så egentligen vill jag nog inte kalla mig ScrumMaster för dem. Men övergripande liksom, jag är med på deras demos och sen så prioriterar vi produktbackloggen tillsammans, och sen gör de själva planeringen själv och det är något team som vi är med och kör hela planeringen över produkten. Men det funkar faktiskt väldigt bra, för det första har vi bra, eller nja kanske inte bra, men vi har ett system som heter ScrumWorks, och det är ingen höjdare kan jag lova. Dessutom har vi gratisversionen, så det är inte världens bästa. Men det är väldigt bra att ha det i ett system. Jag vet att många tycker att man ska ha en tavla, men det går ju inte om de sitter i Brasilien - jag kan ju inte titta på deras tavla. Men där [via ScrumWorks] har jag ju full koll och varje dag kan jag se hur deras sprintar går, vad är det som har hänt och sådär. Och sen så, produktägaren sitter här i Sverige, så jag är ju egentligen en..., min viktigaste uppgift i det samarbetet är att synka med produktägaren och ScrumMastern i Brasilien. Jag är den länken. Produktägaren vill ju egentligen bara komma in och gå ut, titta på demon och säga vad man vill ha gjort tills nästa gång. Men att se till att allt passar med tiderna och sätta upp inför demona och så, men också checka läget inför nästa sprint - vad är det som är viktigt? För att ofta är det dessutom så att produktägaren här, han har koll men den detaljerade kollen har ju teamet och han lägger över mycket av ansvaret på teamet själva, och då är det min uppgift att informera teamet om vad som kommer. Att i nästa projekt, eller om några månader, måste ni vara klara med det här. "men då vill vi börja på det nu. -okej" och sen pratar jag med produktägaren. Så det här är inte renodlat Scrum egentligen, så som det ska vara enligt reglerna, men det funkar väldigt väldigt bra tycker jag. Så det är väl egentligen min roll. Och sen har vi jag varje vecka avstämningsmöte med ScrumMastern nere i Brasilien, att de skickar en rapport till mig och sen så pratar vi igenom det och vad som har hänt och om de har några "impediments" som jag kan hjälpa till med. Ska de prioritera detta eller detta, om det är något som dyker upp. Hjälper dem i det pågående arbetet hela tiden, men inte den dagliga kontakten för det sköter de själva. Vi försöker lägga över det ansvaret på dem. Och det ska ju vara så också, att teamet får sköta sig själva.

MM Hur stora är teamen?

SA[9]: Det är några team som bara är tre-fyra personer och sen är där vissa som är 7-8.

MM Det kanske är någon sorts smärtgräns?

SA[10]: Jag tror att de skulle kunna vara typ två till, men sen så går det nog inte. Men de brukar nog säga 7+-2 personer.

MM Sen skulle vi vilja veta vad de olika teamen har för arbetsuppgifter.

SA[11]: Nästan allihopa "äger" ett verktyg, en emulator eller simulator eller någon annan grej. Så det är både nyutveckling, men det är också maintenance, för de har olika projekt och det är olika faser. I en maintenance-fas är det ju mycket buggar som dyker upp, det testas och testas och även att det kommer in kanske något fälttest. Så det är väldigt brett. Man kan väl säga att maintenance är också en utveckling för det är ju något som inte funkar som man får koda om liksom. Och vi har inget test hos oss, det sitter inga testare. Teamen själva får testa sin kod. Och sen kan det ju dyka upp buggar under projektets gång och då kommer det tillbaka till oss, så får man rätta det.

MM Är det bara utvecklare som finns i teamen?

SA[12]: Ja. Ja, där är team leads, som egentligen inte följer Scrum-modellen, men det följer Företag A:s standard liksom att man har team leads. I Sverige jobbar de 15% med team lead och 85% med kodning, så de är ju väldigt mycket kodare, det är bara det att det är dem som, enligt Företag A:s definition är det de som bestämmer vem som gör vad. Men samtidigt så bestämmer man ju själv det i Scrum, men jag tycker det brukar falla sig väldigt naturligt, för att ofta äger man varsin del av funktionalitet. Men det är officiellt team leaden som tar ansvar för fördelningen. I Brasilien är det så att vi jobbar med ett institut som i sin tur jobbar nära universitet, så där kommer många ny-utexaminerade och oerfarna, men duktiga utvecklare. Team leaden har ofta en längre bakgrund, och har jobbat ett tag. Så team leaden jobbar inte så mycket med utveckling, utan den hjälper till hela tiden. Så det verkar som ett drömläge för en nyutexad att börja där för där har man verkligen den stöttningen hela tiden av den här team leaden, som då är insatt i allt, vilket känns väldigt trygghet för när man ringer dem och frågar hur funkar detta, så har alltid team leaden benkoll för han eller hon har varit med i varenda steg som har tagits. Så ja, han eller hon håller också på med utveckling, men inte på ett eget område på samma sätt oftast.

MM Jag tänkte gå tillbaka till det här med arbetsuppgifter så vi klargör det. De olika teamen, är deras arbetsuppgifter relaterade? Så att man kan säga att de påverkar varandra?

SA[13]: Ja, det gör de ju i vissa delar. Oftast är de ju väldigt skilda åt. Men t.ex. ett verktyg i Brasilien bygger man på emulatorn som finns här i Lund så de har ju ett jätteberoende av varandra, och ibland när verktyget kraschar så kraschar emulatorn också, så de behöver ju jobba väldigt tätt tillsammans.

Och likadant ett annat verktyg där nere som byggs på den andra emulatore, det är samma grej där. Och likadant, funkar inte byggsystemet så funkar inte mycket av det andra. Så ja, de är ju beroende av varandra. Och ibland så händer det ju att någon måste bli klar med en viss feature för att någon annan ska kunna - att det är ett beroende i kedjan. Och då är det ju mitt jobb att se till att de planerar så.... T.ex. i de här heartbeat-projekten så når vi snart en milstolpe på ett av dem, och då måste en av emulatorerna vara färdiga för att de andra är beroende av den, de bygger på den. Inte alla, men några. Och då är det ju mitt jobb att de är klara i tid, och så fort vi har en label vi kan bygga på så måste de bli klara och sen så planerar man tiden så de kan fortsätta, men ändå göra detta i en iterativ så... och veta att Nu ska ni köra igång, och räkna på det när man planerar sprinten. Så det är beroenden mellan de olika teamen. Inte alltid, men då och då.

MM Kan man göra det även under Sprintarna? Sker kommunikationen mellan team som är beroende bara precis mellan Sprintarna eller..

SA[14]: Nej nej, det har vi räknat med. Så när vi räknade på den Sprinten så hade vi räknat att den och den veckan så kommer ni att jobba med det. Sen så när vi skulle planera den här andra Sprinten så säger vi att den och den veckan kommer de att bli färdiga och sen behöver ni ju så många timmar för att bli klara då räknar vi där. Ja, hur många timmar har ni innan då? Ja då gör vi så och så. På så sätt är det inte det här renodlade Scrum, där man mer eller mindre bara kan sätta allt på en tavla så kan man gå och ta vad som helst. "Åh nu vill jag hämta en ny item att göra", så går man och tar det från väggen. Utan det är hela tiden en vattenfall.... vi jobbar ju mot vattenfallsprojekt och det påverkar oss. Men vi jobbar ändå i iterationer för vi kan ändra vårt scope från sprint till sprint. Och sen tycker jag att det är ett väldigt bra sätt att mäta, vad har vi gjort den här månaden.

MM Och alla teamen använder Scrum?

SA[15]: Jäpp

MM Om vi säger ovanför eller mellan teamen, då använder ni inte Scrum där?

SA[16]: Ja, ovanför teamen, det är ju egentligen jag och Henrik då, och vi följer inte någonting, utan vi bara jobbar linjejobb. Varje dag är densamma. Och sen ovanför oss, de vi rapporterar till, de kör ju PROPS eller liknande.

MM Men det är ändå på nåt sätt ovanför er?

SA[17]: Ja, det blir ju det eftersom det är de vi levererar till. Så det är inte den här med massa Scrumteam som är i ett annat Scrumteam. Nej, det är inte den här pyramiden som de gärna..., som jag tror inte funkar i ett företag. I ett litet företag ja, då skulle det kanske funka. Säg att du har 50 personer eller nåt sånt där. Då kanske det skulle funka. Men det funkar inte i ett såhär stort företag, som dessutom håller på med produktion. Det är min teori iallafall!

MM Du sa att du har några team, och Henrik har några och Olof... Är det så att de teamen som du har, har större beroenden på varandra än alla sinsemellan?

SA[18]: Nej, så har vi nog inte. Ja lite kanske. Nä vi har bara delat upp dem lite inom vilka områden, det har varit lite slumpen också. Men jag tror att vi kommer nog ändra lite på det till våren. Och då tror jag nog att vi försöker hålla lite mer... Men det är så mycket som spelar in, det är både beroendena där, och sen så är det typen av verktyg, eller kanske man ska ha båda emulatorerna t.ex. Ja, det är många saker som spelar in.

MM Har ni medvetet delat upp arbetet mellan teamen för att minska beroenden?

SA[19]: Njaj, alltså allt arbete har ju med just det verktyget att göra. Teamen är uppdelade på verktyg helt enkelt. Så att antingen håller man på med den ena emulatore eller den andra liksom. Och då, javisst, många av de sakerna får de göra samma sak på båda men det är två helt olika teknologier så att de har liksom inte med varandra att göra egentligen, men de gör egentligen samma sak - nästan. Och likadant alla de här testverktygen, de är också - de bygger på en viss sorts teknologi då lägger vi dem i ett team, ja och så ibland kan det passa bättre där och där.

MM Händer det någonsin att ni har stormöten där alla träffas?

SA[20]: Nej, inte ur projektsynpunkt. Däremot har vi avdelningsmöten varje vecka, som vilken linjeorganisation som helst, men då talar vi inte så mycket om vad man gör i sprintarna. Vi brukar ha sån här round-the-table typ där team leaden berättar vad som händer i sprinten. "-Det här jobbar vi med just nu. -Okej." Men annars pratar vi om linjeregjer, omorganisationer, nya chefer, och Företag A-tjafs liksom. Men annars har vi inget sånt nej.

MM Använder ni sån här Continuous Integration?

SA[21]: Jäpp!

MM Så ni checkar in kod hela tiden? Så har vi förstått det i alla fall.

SA[22]: Tja, dit är vi på väg, hela företaget. Men det är ett verktyg vi äger, ett av teamen är Continuous Integration, alltså supporterar att rulla ut det i organisationen. Att man checkar in, skapar sina nattliga byggen och sånt där, för att kolla fel.

MM Har ni haft problem med att team "trampar varandra på fötterna"? Att de råkar göra samma sak?

SA[23]: .. [paus] Det har säkert hänt, men jag kan inte komma på något på rak arm, men det har säkert varit någon kollision, men då har väl förhoppningsvis jag eller Henrik räddat den eller kommit på att de jobbar med samma saker. Framförallt i det här verktyget som bygger på emulatorens, så är det lite svårt att säga om det är fel på emulatorens eller verktyget. Vem ska göra vad? Men då får de gå och snacka med varandra. Och ibland om det har varit någon stor grej, har vi brutit båda sprintarna och sagt, "okej, nu måste ni sätta er tillsammans och lösa det här problemet".

MM Så det att utvecklarna pratar direkt med varandra?

SA[24]: Ja, absolut, absolut.

MM Finns det några fler fall när utvecklarna från två team pratar med varandra?

SA[25]: Ja, det gör de ju nästan dagligen, vi sitter ju i samma öppna landskap, och ibland är det ju så att "jamen du, hur funkar det där?" och "hur gör man med det där?". Sen ibland har de ju haft i sina sprintar att få det till att bygga på det här CI. "Så ska det byggas varje natt och se till att det funkar", och då får de ju jobba ihop där, och då ingår det i deras [CI-teamets] sprint att leverera något till dem egentligen, för att kunna bygga den biten.

MM Men det är på något sätt något som de sköter?

SA[26]: Ja. Och annars som sagt, frågor däremellan har de ju säkert hela tiden, hur olika saker funkar. För där är ju ett beroende ändå, eftersom allt ska matas in i byggmodellen.

MM Om det kommer en förfrågan om funktionalitet som täcker två team, vem delar upp den då? Vem tar emot det?

SA[27]: Det gör ju jag och Henrik, ser till att det hamnar i rätt produkt-backlog

MM Och ni delar upp det också? Eller det händer inte alls?

SA[28]: mm, nja. Ja det gör vi väl på något sätt. Eller det gör väl egentligen teamen tillsammans.

MM Det kanske inte har...

SA[29]: Nä, inte på det sättet, utan de har kommit med direkta önskemål på självaste verktyget. Och då får det läggas in i produkt-backloggen som externa krav och sen är det produktägaren som lyfter in det i själva produkt-backloggen. Läger det under en speciell flik, externa krav. Så det är upp till produktägaren att lägga in det.

MM Produktbackloggen, har ni en? Eller en för varje team?

SA[30]: En för varje team. Eller en för varje verktyg faktiskt, för att vi i två av fallen så är det där två team. På en av emulatorerna så är där ett team i Lund och ett team i Brasilien, och samma sak på det andra faktiskt. Så då är det en produktbacklog, men de delar själv upp det sen i två - vem som gör vad.

MM ..det gör teamen?

SA[31]: Mmm, alltså, för det är så. Här[1] är en produktägare för Lundteamet, och sen är det team leaden från Lundteamet som är produktägare för det outsourceade i Brasilien[2]. För att de[2] är dessutom inte helt uppe på banan, de[1] här har ju jobbat med de här i rätt många år, så de[2] behöver läras upp. Och det är bara en person som kan bestämma vad han vill använda de här resurserna till, och det är team leaden här. Så att visst, den här översta produktägaren, det är ändå han som styr skutan lite - varåt vill vi. Men sen är det upp till honom att säga vad de[2] ska göra. Så det blir som ett ...träd, eller en kedja.

MM Och ni har inga s.k. cross-team members - personer som är med i flera team?

SA[32]: Nä... ja, lite har vi haft en person faktiskt. Han har varit med både i ett team här i Lund samtidigt som han har varit lite av en CM för de i Brasilien, men då har jag planerat in alla hans timmar i sitt team, så jag planerar inte in honom i två sprintar, utan hans timmar ligger i en sprint, och där är det liksom - det han ska göra på sitt team, och en task som är CM typ, och de sakerna han ska göra där. Det blir för rörigt att ha honom på två ställen.

MM Vad var anledningen?

SA[33]: Vi behövde en som tog hand om leveranserna från Brasilien

MM Så det var inte på något sätt för att koordinera....

SA[34]: Nä nä, det var bara för att vi behövde honom där att hjälpa dem, men det var ingen heltidstjänst utan det är bara lite då och då när det behövs. Så därför hamnade han ju... så ja egentligen var han ju bara med i team kanske. Vi har faktiskt en team lead i Brasilien som är team lead för två team, som är 50% i varje. Två små team på 3 personer.

MM Är det mest för att de är små så de behöver inte så mycket...

SA[35]: Ja, ja. Det blir för mycket overhead om vi hade... Vi kunde lagt dem i samma team men där tyckte vi det var lämpligt att dela upp dem, för det är vitt skilda saker de håller på med.

MM Då har vi inga fler frågor, utan känner att vi har fått en bra bild av hur ni jobbar.

Kompletterande frågor:

MM Har ni justerade sprintar, så att teamens sprintar börjar och slutar samtidigt? För alla team, vissa team eller inte alls? Och i så fall, hjälper det er att koordinera mellan de team som har justerade sprintar?

SA[36]: Nej, vi har inte justerade sprintar. Och varje sprint är inte lika lång. Detta då det är omöjligt att få sprintarna att vara lika långa; både för att jag och Gustav inte alltid är tillgängliga på regelbunden basis, men också för att det inträffar saker som stör cyklerna, tex helgdagar, konferenser, kurser och annat. Vi skulle inte kunna ha sprintar som startar och slutar samtidigt då jag inte kan hålla demos och sprintplaneringar för två eller flera team samtidigt. Jag föredrar att ha en start/slu per vecka, så att jag hinner med mitt vardagliga arbete emellan. Jag har haft veckor då jag har haft tre team som startat/slutat en sprint, och det funkar inte. Jag blir låst i demos och planeringar fulltid och halkar efter i annat arbete.

MM Finns det något som man kan säga är en integrationsfas, där allt byggs samman till en produkt? När sker isåfall den och vad görs då? Vem tar hand om det?

SA[37]: Nej, varje sprint innehåller leverans och integration till huvud-branch. Dvs, när man har utvecklat ngt så ingår det att leverera det till huvudprojektet för att det ska anses vara 'done'. Varje projekt gör sin egen leverans på vår avdelning, då våra verktyg oftast inte är beroende av varandra. Då de har ett beroende meddelas detta till CM för huvudprojektet vid leverans.

B.2 Produktägare, Företag B

Genomfördes 2008-11-27

MM Max Martinsson

PB Produktägare B

JO Joel Olofsson

MM Vi kan ju börja med att säga att du kommer få ta del av transkriberingen och godkänna den, och du kommer få vara anonym och företaget också.

PB[1]: Okej

MM Vad är det du jobbar med?

PB[2]: Jag jobbar primärt som projektledare i mjukvaruprojekt och det har jag gjort i 6-8 år ungefär, med bakgrund som C++-programmerare. Och sen som systemarkitekt, teknisk projektledare/projektledare. Det är vad jag gör. Så jag programmerar inte nu, mer än för husbehov.

MM Okej, och vad är det för projekt du leder?

PB[3]: Just nu är jag inte det för något projekt. Men jag har nyligen varit med i ett projekt som jag kan referera till som passar bra in på det jag tyckte jag hörde ni ville prata om. Scrum i stora projekt?

MM Precis.

PB[4]: I det projektet var jag formellt sett lead system architect, så formellt sett var jag ansvarig för design och arkitektur. I praktiken jobbade jag som projektledare, teknisk projektledare. Så jag var ansvarig för... I själva verket gjorde jag allt utom budget, kan man säga, som projektledarna gör. Jag gjorde planen och följde upp planen, och så vidare.

MM Okej

PB[5]: Det fanns en projektledare som hette projektledare som gjorde budget. Så det var min roll i det projektet.

MM Okej

PB[6]: Projektet var för ett företag i regionen som tillverkar konsumentprodukter. Som skulle bygga en tjänst som komplement till sina produkter. Så det var en marknadsföring, försäljning och image / varumärkesbyggnad-webbsajt. Det var vad projektet gick ut på.

MM Okej. Hur många teams fanns i det här projektet?

PB[7]: Det beror lite på hur man räknar, antalet mjukvaruutvecklingsteams...

MM Ja, det är väl det vi är mest intresserade av.

PB[8]: Projektet på sin topp, när det var som störst, sysselsatte ungefär 150 man. Varav 70-80 inom mjukvaruutveckling. Resten jobbade med sånt som service, customer service, infrastruktur och såna här grejer. Mjukvarumässigt, ungefär, fyra till sex team. Det kan ha varit så många som sju team vid något tillfälle.

MM Så det varierade mycket?

PB[9]: Ja, och projektet skalade upp. Både antalet personer och antalet team.

Det var ett långt projekt. Projektet varade, beroende lite på hur man räknar, min första kontakt med projektet var faktiskt så tidigt som i september 2006. Och själva mjukvaruprojektet som var konsekvensen av det arbetet drog igång någon gång i april-maj 2007. Och håller i någon mån fortfarande på men, gick i produktion, live, färdigt någon gång i slutet av augusti i år.

MM Okej, så hur många team fanns det från början?

PB[10]: Allra första början var de tre team.

MM Okej, och de här olika teamen, om vi bara tar dem i mjukvaruprojektet, vad hade de för olika arbetsuppgifter?

PB[11]: Generellt sett var det ju uppdelat på olika funktionalitet i systemet. Så ett team byggde en mobil webbsida. Ett team byggde en vanlig webbsida, ett team byggde en betalningslösning bland annat, ett team byggde en statistik och betalningslösning. Ett team jobbade med grafisk design och informationsarkitektur.

MM Okej, hur många medlemmar var det per team?

PB[12]: Mellan fyra och åtta. Det minsta teamet var fyra man och det största teamet var strax under tio man. Och sen när det skalade upp, när det blev riktigt hektiskt, så de som byggde webb-fronten var under någon period upp mot tjugo stycken.

MM I samma team?

PB[13]: Ja, men det var en ganska begränsad insats för att komma ikapp.

MM Okej, på vilket sätt använde ni Scrum i det här?

PB[14]: Tanken var ju att mjukvaruutvecklingen skulle ske lite som i Scrum, det vill säga med sprintar, med en ScrumMaster, med Product Backlog och med stort delegerat ansvar. Det var grundtanken, det var så det var uppsatt. Alltså riktig Scrum på den nivån, vad det gällde mjukvaruutveckling.

MM Mm. Du säger att det var tänkt att vara så?

PB[15]: Ja, men det funkar ju inte i praktiken.

MM Okej

PB[16]: Det fanns hela tiden Scrumteam och det fanns ScrumMasters och det fanns ett Scrum-of-scrums och det fanns som sagt en Product Backlog och det fanns Sprint Backloggar. Det fanns! men det funkade inte.

MM Okej, kan man göra någon övergripande analys av det?

PB[17]: Det kan man, man kan göra många olika analyser. Beroende på vad man vill peka på. En viktig sak som jag upptäckte var att teamet inte förstod vad vi höll på med.

MM Okej

PB[18]: Och ingen förstod egentligen i början vad vi höll på med. Jag var en av dem som fick frågan innan vi satt igång och programmera, vad kommer det här kosta? Ungefär så här mycket sa jag. Det var först senare jag förstod att det här är mycket mycket större.

Det som för mig började som Malmös bästa webbshop - fair enough - växte och det som egentligen beställdes från början var världens bästa webbshop. Det tog lång tid för mig att fatta, det tog lång tid för de flesta att fatta och därför underskattades vissa funktioner i projektet gravt. Dels rent programmeringsmässigt, hur mycket kod måste vi faktiskt knacka ihop för att lyckas med det här. Och dels hur mycket tid och pengar och resurser måste vi faktiskt lägga på att jobba med vår Product Backlog.

MM Mm.

PB[19]: Där kan man säga problemet var, förväntans-hantering - ingen visste vad de gjorde - och då gör man definitionsmässigt fel. Och sen en annan faktor som spelar in, tycker jag väldigt mycket, var ju att väldigt få av utvecklarna hade den erfarenhet som krävs för att köra Scrum.

MM Okej.

PB[20]: Det krävs väldigt mycket av individen på lägsta nivån, för att klara av de ansvaret som krävs, för att vara så självgående, för att kunna ta det ansvaret som krävs. Vi såg i Sprint efter Sprint, i början, hur de konsekvent inte levde upp till målen. Vi kom till Sprint Review och det var inte färdigt. Och de stod själva och sa: "– Vi är inte färdiga. – Och vad ska ni göra åt det då? – Ah, det får vi göra bättre nästa gång." Och så blev det fel igen och igen.

MM Okej.

PB[21]: Jag hade förmodligen fel angreppssätt. Min roll var ju liksom rätt komplicerad. Jag lyckades inte få de att jobba bättre i alla fall. Och de själva sa, vi vill köra Scrum. Men då måste ni ju ha produktionsfärdig kod här. "Done". Jag kände inte själv till den stringenta definitionen inom Scrum, done. Alltså hur viktigt det är att man definierar done. Det är en lärdom jag har lärt också. Man måste vara väldigt tydliga med - vad är done? När är vi klara? För jag gick bara in och sa "Ni får leverera hur lite ni vill, men det måste vara produktionskvalité." Men då får man ändå prototyp efter prototyp efter prototyp levererad och då blir man till slut...

MM Kan man fortfarande säga att ni följde Scrum?

PB[22]: Visst gjorde vi det. Vi hade Product Backlog, vi hade Sprintar, vi lät teamet välja. Å då stödjer man på ett annat problem med en Product Owner som ska stötta sex team. Gärna på samma tider. Alltså det går inte, det är gravt underskattat att tro att man kan göra det. Jag menar kravställningen på ett sånt system består ju av.. säg jag tror vi hade 150 övergripande krav. Och då pratar vi övergripande här, typ man ska kunna köpa.

MM Funktionella krav alltså?

PB[23]: Ja, jag menar på feature nivå, alltså att den här ska innehålla de här fem typerna. När man sätter sig ner och ska detaljera ner detta och inte har utvecklare som är självgående utan de kommer verkligen

och frågar “–Ska den knappen vara där eller ska den vara fem pixlar till höger?” Då blir man överlastad med frågor.

MM Det är klart. Men ni använde det här Scrum-of-scrums tillvägagångssättet?

PB[24]: Mm, teamen gjorde det. Själv stod jag lite utanför det. Men det var tänkt som en koordinerad mellan teamen.

MM Och då var det en representant från varje team?

PB[25]: Ja, en ScrumMaster för respektive team träffades två gånger i veckan och snackade igenom saker.

MM Okej, så där var en ScrumMaster för varje team?

PB[26]: Mm.

MM Och en enda Product Owner för hela projektet?

PB[27]: Mm.

MM Okej, och det förändrades inte under hela projektet?

PB[28]: Nä, sen jobbade Product Ownern och jag nära ihop. Så de mer tekniska frågor delegerade han till mig. Så kom de till mig och fråga teknik och så kom de till honom och fråga kundnytta.

MM Okej.

PB[29]: Men vi skulle behövt vara ett team på kanske fem personer för att kunna reda ut det.

MM Om man försöker säga någonting om hur de olika teamen är beroende av varandra? Kan man se några sådana tydliga kopplingar mellan vissa team, att vissa team var mer beroende av ett annat teams resultat?

PB[30]: Absolut. Det team vi startade upp sist var statistik och rapporteringsteamet och de var ju ganska isolerade. De ställde vissa ganska små krav på resten av systemet att skjuta vissa triggers eller mata in viss data men sen kunde de jobba och bygga. Och det teamet levererade från dag ett. Klockrent. Och de hade en jättebra ScrumMaster som tänkte själv och tog ansvar. Han kom till mig och sa: “– Någon har gjort det här, är det okej? – Klart det är okej, lös uppgifterna.” Det teamet funkade väldigt bra, väldigt scrummit och tog stort eget ansvar. Det lilla teamet, hade fyra medlemmar inklusive ScrumMaster, trivdes bra och hade kul och gjorde väldigt bra grejer. Sen de andra teamen hade större beroenden och fast vi upptäckte väldigt tidigt att vi gör fel, att vi bygger skikt, horisontella skikt. Här har vi databasen, här har vi affärslogiken och här har vi GUI:t typ. Så blev det enorma beroende. Och ingenting funkade i integration. De kunde inte sätta ihop det här.

MM Okej.

PB[31]: Ingenting spela ihop första halvåret - nio månader. Ingenting. I systemet ingick en jättestor databas i terabyte storlek. Och helt plötsligt på en Sprint Review står ett team och visar en databas och ett annat team visar en annan. “– Wake up and smell the coffee. Det här har ni nog inte beställt. Det är absolut inte okej. – Ja, men vi sätter ihop det. – Ja, men det är inte produktionsfärdig kod.”

PB[32]: Vi highlighta det tidigt och jag sa det tidigt att nu måste vi bygga vertikala team. Vi tar en GUI, en business logic, en databas och så får ni bygga en funktion. Men då gick inte det av diverse politiska skäl och personkemiskäl och olika mognadsgrad inom de olika teamen och olika förståelse för Scrum. Så vi fortsatte alldeles för länge med horisontella team - fram till en viss punkt som är väldigt väl definierad - då allting gick käpprätt åt helvete och från den punkten ska jag säga styrde vi om till traditionell projektledning.

MM Okej.

PB[33]: För att rädda slutprodukten. Det glömde jag nämna innan. Alltså vi slutade med Scrum ungefär efter 70% av tiden.

MM Slutade helt alltså?

PB[34]: Ja, då förbjöd vi det rakt upp och ner. Vi tog in en riktig projektledare för utvecklingsteamet. Och avkrävde helt och hållet tidplan och följande commitments. Vi tog ett ryck med kraven. Jag satt en lördag då i den här krisen och hjälpte våra underleverantörer och förstå det här måste ni commita till det här datumet och nu är det färdig scrummat. Då hade det ju verkligen gått åt helvete.

PB[35]: Men integration mellan team är en viktig lärdom. Och bygger du inte vertikala team är du i princip körd för du hamnar i en integrationsfas som inte finns i Scrum. För varje Sprint ska leverera något som funkar, det är en grundprincip. Om du då levererar en Sprint bara för att leverera in det till en integration, då är det inte Scrum. Då tappar du en väldigt stor del av kärnan. Det är bättre att bygga det uppifrån och ner. Bygga enkla saker som funkar hela vägen.

MM Scrum dikterar väl också att man ska ha, vad de kallar för cross-functional teams och det är just att de ska vara vertikala.

PB[36]: Jo, jag kallar de vertikala team. Ni förstår väl?

MM Ja

PB[37]: Men det är lätt att säga men svårt att göra. Vi hyrde in riktigt bra killar för att bygga en riktigt fet databas. Å det här är inte vilken konsult som helst, som du kan göra vad som helst med. Det är en kille som skrivit Oracle databaser i femton år, som vet exakt vad han gör. Å tar inte skit från någon. Att sätta honom i ett team där han ska samarbeta. Det ska vara socialism. Å han ska dela och parprogrammera. Nä, han tar sina tjocka glasögon och sätter sig framför datorn och gör. Det är lätt att säga men svårt att göra.

MM Mm, det är spännande det här. När ni delade upp arbetet mellan teamen. Var den någon medveten tanke med hur man delade upp det?

PB[38]: Teamen ska ju vara självförmedlande. Det är ju det de hävdar, Sutherland och grabbarna. Slänger du 100 människor i en stor container så kommer de organisera sig. Och de gör de mycket riktigt också, för jag känner dig, och du och jag är också bra kompisar och vi har jobbat ihop i fyra år så då gör vi det här. Så självorganiserande blev det på det sättet, det var samma team som alltid hade jobbat ihop, som kände varandra. När vi bytte projektkokal, vilket vi gjorde vid ett antal tillfällen, så flöt de här ihop ut i hörnen.

MM Ni satt öppet eller?

PB[39]: Ja, det var en typ av öppna landskap. Men med så mycket folk blir det ju hela våningsplan vi pratar om. När vi försökte tvinga de att blanda, man gick in och så flyttade man ett skrivbord rent fysiskt. Men sen när vi var tvungna att bytta lokal, på grund att vi inte kunde ha lokalen, så vips - i en ny lokal - här är en stor yta, den tar vi, här är en stor yta, den tar vi. Det där funkade faktiskt bättre och bättre över tid, ju mer de lärde känna varandra. Kärnan i de här gänget, femton programmerare som var med i princip hela vägen, de blir bättre på att samarbeta.

PB[40]: Jag hävdar till skillnad från Sutherland och grabbarna att människor inte är självorganiserande. Myror är självorganiserande. Tar du ett glas med myror och släpper ut dem i skogen. Några timmar senare har du en myrstack. Å det är varje gång reperterbart. Gör du så med människor så är det bara att titta på robinsson, flugornas herre. Det blir maktkamp. Någon vill sko sig på någon annans bekostnad. Status är viktigt. Makt är viktigt. De människor som inser att de inte får någon makt, de slackar. Det finns inte bland myror och bland bin. Det finns liksom ett gäng drönare som bara sitter utanför kupan och dricker bärs hela dagarna. Men det gör människor.

MM Ja. Jag fundera på, var det några stormöten där alla träffades?

PB[41]: Ja, där hade vi lite olika varianter. Inte precis i början, men efterhand som projektet växte så tillsatte först vår underleverantör ett så kallat all hands meeting. Där de försökte samla alla utvecklarna och berätta statusen. Det var ganska sporadiskt om jag minns rätt, någon gång då och då. På månadsbasis.

PB[42]: Halvvägs igenom var jag bland de som tog initiativ till att samla hela projektet på Savoy här i malmö. Alla 150 man, och lite gäster hade genomgång med seminarier där vi berättar om olika saker. Efter den stora smällen sen, bestämde projektledaren, varannan vecka, på tisdag, två timmar, i den här aulan. Så då var det mer traditionellt. Det kallades för all hands meeting, som utvecklingsteamet höll i. Det var ingenting jag drog igenom, det var något de själva kom på att de behövde.

En jättebrist i Scrum är den här kortsiktigheten. Man ser inte skogen för alla träd.

Och det är kanske någonting vi borde jobbat hårdare på kanske så skulle vi varit mer framgångsrika.

MM Tror du att stormötena gav någonting?

PB[43]: Vi fick alltid bra feedback från dem. Utvecklarna kände att “– Ah nu fattar vi lite bättre. Nu känner vi att vi är med, vi förstår helheten.” Och just det att alla får se att vi är 150 man som jobbar direkt med det här och kanske 200 man till som är indirekt berörda av det här projektet mer eller mindre direkt då alltså.

MM Då får man lite mer av en helhetsbild.

PB[44]: Ja, jag saknar det i Scrum teorin, hur jobbar vi med mål och vision. Någonting som är väldigt tydligt i traditionella projektledningsmodeller. Där mål och vision är det enda viktiga. Steps and stones, the big pictures, hit ska vi. Personligen tycker jag inte Scrum tar den problemställningen på allvar. Det finns inget verktyg med i de traditionella böckerna för att svara på de här frågorna.

MM Jo. Jag funderar på en helt annan sak: har ni använt så kallad Continuous Integration?

PB[45]: För det första kan man säga så här. Vi ställde som krav från absolut början av den här scrumifieringen var continuous Integration. Vi förstår om vi inte lyckas denna veckan, men börja någonstans. Vi måste börja integrera. Tillbaks till då, vi hade fruktansvärda leveransproblem. Sen instiftade jag någonting som hette “integration team”. Som var en grupp som var cross-functional. Så började de självorganisera sig. Sen hade jag lite oflyt då det var folk som började och slutade. Jag valde en chef

för gruppen som rapporterade direkt för mig, men som slutade på företaget efter 3 månader, så jag fick byta. Något som var viktigt med det här integration teamet var att det var hela vägen från utveckling till systemet i drift, så jag hade även kallat in folk från den organisation som skulle delta i driften av systemet. Både den mekaniska driften, servrar, nätverk och diskar men även “application management”, konfigurera, lägga till och buggfixar. Jag bad alla i teamet sätta ett test med utvecklingservice allihop. Det var en synnerligen god idé. Men tyvärr skiftade fokus i den här gruppen väldigt mycket. Det var en kille i gruppen som var “configuration manager” och han tyckte vi skulle ha struktur och dokumentation. Sen kom det en annan kille som var väldigt inne på automatiserade verktyg för att dokumentera arkitektur och mjukvara. Så det var väldigt svårt och de levererade halvdåligt. Vi byggde ett verktyg, efter att någon fick en snilleblixt, som vi plöjde ner manmånader av jobb i men som sen dog på grund av att kille som hade gjort det tröttnade. Faktum är att de fick inte till en riktig byggmiljö förrän efter projektet kraschat och vi börjat om igen.

MM Så deras resultat skulle vara att allting funkade tillsammans?

PB[46]: Ja. De hade satt upp det så här. Jag har tre saker ni ska göra. Ett, när jag trycker på knappen till saken är avklarad ska ta mindre än sju minuter. De skulle ha seamless handover till application management team, till drift. Det var ett mål för gruppen att se till att rätt person var inblandad vid rätt tid. Man utvecklar, man har go live, man har stabiliseringsfas, man har underhåll. Och det ska inte märkas när man byter fas. Det var tanken. Jag ville inte ha en massa möten. Jag vill kunna ringa min chef på måndag morgon och höra har ni tagit emot? Ja, vi har tagit emot. Det var deras mål. Tredje målet kommer jag inte ihåg var det var.

Det berodde lite grann vem som ledde det teamet. Om det var klart på sju minuter, eller om det var “seamless handover” eller om det var någonting annat.

PB[47]: Det var på grund av det, att jag återigen underskattade hur mycket jobb som skulle krävas. Å när jag väl fick resurser, så hade det gått väldigt lång tid. Jag upptäckte resursbehovet väldigt tidigt och eskade resurser men fick inga resurser. Sen när jag väl fick resurser så var det för sent. Men jag sa till ganska tidigt att det här kommer inte att gå. Här listas saker jag måste göra. Men jag är bara en människa. Jag behöver ett team. Så det var det integration team vi satte upp, som var folk som egentligen jobbade för andra team men som skulle träffas ibland och prioritera vissa krav och annat.

MM Så det var inget helt eget..

PB[48]: Nä det var inget eget Scrum. Utan det var folk som träffades utanför det. Namngivna personer, i viss mån handplockade av mig. “Han är väldigt duktig på infrastruktur, jag vill att han sitter i det här.” Men det gick ju inte heller som det skulle. För att människor är inte självorganiserande. De gör inte saker om man inte pekar på dem.

MM Nä, vi funderar på bland de andra teamen, fanns där personer som var medlem i flera team samtidigt.

PB[49]: Japp. Det gällde speciellt specialisterna. Någon som är väldigt duktig på databaser jobba i två team, tre team. Det hände inte speciellt mycket men lite.

MM Tror du att det hjälpte?

PB[50]: Nä, det var egentligen en konsekvens av att man inte bygger... om man ska bygga vertikalt, om man har något väldigt specialiserat någonstans i de här rollerna så blir det ju någon person som är väldigt behövd i alla läger. Och eftersom människor är som de är så går du inte in i team och lär upp de andra. Utan de går in och visar sig på styva linan. Samlar credd och gör sig ännu viktigare. Och sen hoppar han till nästa team och är specialist där, och blir ofta deras drivkraft. De har inget som helst intresse att göra andra lika bra. De är inte den typen av människor.

MM Nä, då funderar jag på överlappande arbete. Om vi säger att det kommer en begäran för en viss funktionalitet som påverkar två team. Vem tar hand om det? Vem delar upp den uppgiften? Och säger vem som ska göra vad? Om det händer ens.

PB[51]: Det hände det hela tiden, eftersom har man horisontella teams så allting man ska implementera drabbar ju flera teams. Det börjar ju då med att de träffas på en Sprint Planning Meeting, och man har sju saker man ska göra. Säg då att det var tre team inblandade. Så kommer de alltid välja olika saker som förmodligen inte kommer bli klara. Tre saker i Product Backloggen går bort för de hamnar längst ner i prioriteringen, för längst ner blir aldrig gjort, det är bara att konstatera faktum. Då har man bara fyra kvar. Då sitter de och kommer överens - Nä men vi gör de i ungefär den här ordningen. Men sen går ungefär 75% teamens Sprintar och då kommer de på att det inte är i fas och sen kraschar skiten för de inte hinner implementera. Och så är det bara. När de väl satt sig i sina team, så bevakar de sin grupp mot andra grupper. Min grupp är åtminstone bättre än den gruppen. Vi måste prioritera våra grejer, vi måste lyckas. Och där fostrar man ju de till att tänka team, små team inte stora team, inte kollektiv. Vilket gör att de finns inget naturligt samarbete om man inte tvingar de. Och det får man inte. Har man väl lämnat Sprint Planning Meeting så har man och då ska de sköta det här själva. Scrum-of-scrams ska lösa det,

vilket funkar i teorin men inte i praktiken. Det är bara att konstatera, horisontella teams är dödsdömt. Ingenting talar för att de ska gå. Vertikala teams är mycket mycket svårare och ställer mycket mycket större krav på att programmerarna är mer generalister. Man måste kunna lite bredare.

MM Mm, Jag funderar på de här Sprint Planning Meetings. Då är alla team med?

PB[52]: Vi hade väl generellt sett ett möte med varje team och sen fick jag då som assisterande produkt owner och product ownern propagera att vi verkligen behöver den här funktionen nu. Vi gjorde också tidigt en Sprint Road Map, där vi spikade upp, på ett ungefär, i den här ordningen ska allt utvecklas. Så det är som jag sa, den här statistik och rapportering kommer sist. Det var en medveten strategi. Bygga det stora datalagret tidigt, för det var förrenat med stor risk.

PB[53]: Det kallas ju i traditionell projektledning för rolling wave planning. Efterhand du rullar framåt upptäcker du nya saker. Det är ju lite grann det Scrum bygger på fast de glömmer att planning finns där. Återigen med mål och vision. Det fanns stöd i någon litteratur för att göra Sprint Road Maps. Det var det som gick att sälja till de här hardcore Scrumkillarna att vi faktiskt var tvungna att göra en plan.

MM Har ni haft några konsulter som varit här och fört in Scrum?

PB[54]: Scrum kom från leverantören. De sa till oss. Jag är ju konsult på Företag B, jag är utyrd till det här tredje företag som köpte tjänsterna. Det tredje företag kom och sa när vi skulle börja - Vi vill köra Scrum. Go Big. Men sen visade det sig att de saknade kompetens. Men det fanns vissa individer i det här tredje företaget som hade goda erfarenheter och kunde teorin och visste vad de höll på med. Vi pin-pointade ett fåtal konsulter som vi skickade in i utvecklingsorganisationen som vi hade förtroende för som kunde det här. Men det visade sig ganska snart att det inte var i närheten av den erfarenhet som behövdes på utvecklare, ingen kunde metodologin.

MM Ja, jag funderade på gemensamma uppgifter...

PB[55]: Det funkar inte, inte om man inte lägger in en integrationsfas efteråt.

MM Men jag tänker även om man har vertikala team kan man tänka sig att det finns funktionalitet som berör två team?

PB[56]: Nä inte om det är vertikalt. Bygger du perfekta vertikala team. Då sätter du ett team som gör exakt denna funktionaliteten. Oberoende.

MM Och levererar färdiga saker.

PB[57]: Ja, och levererar "done". Och så bestämmer du vad done är. Efter min tid som programmerare. Så sa jag det kompilarar - det funkar. Okej, men då är man rätt skicklig. Normalt sätt måste du testa, integrera system, testa produktion. Bestämna gränsen vad är done.

Produktionsfärdig betyder ingenting. Därför bygga vertikala team där det är någon som tar ansvar för versionerna.

MM Jag funderar på det, om man då skulle ha vertikala team. Då är inte varje team fokuserade på en viss sak.

PB[58]: Nä precis, utan man måste kunna lite av allt.

MM Så inför varje unik Sprint så skulle man komma på det efterhand?

PB[59]: Om vi generaliserar världen bygger du ett system som består av sju olika lager. Så gör du team med sju personer i varje. Då skulle varje team kunna ha en specialist inom teamet. Man behöver inte bryta upp teamen varje gång det här sker. Då kan man fokusera på teamen och team känslan och där du kan få en sund tävlan mellan teamen. - Vi har en den här hastigheten. - Och vi har den här hastigheten. Vad beror det på? Att ni estimerar dåligt eller att vi programmerar snabbare? Man kan byta om det är någon personligen inte passar in så kan man byta mellan teamen. Men har man horisontella team är man specialist och det är dem. Och det går inte nog att understryka vad i min mening är en success faktor för det här att man satsar mer på generalister. Vilket blir jättesvårt när man jobbar med specialiserad teknik. Alltså att skriva en databas som defakto klarar av att hantera 70 terabyte data det är inget skämt. Det finns sådana. Tänk om man ska ha fem team som allihop ska jobba ihop om det här. Då måste man då skriva ihop massa guidelines och regler eller sätta in en databaspolis som pekar med hela handen. Men då är du återigen där och tafsar på teamet och deras självkänsla. Det är jättesvårt. Men i teorin funkar det. Och det finns ställen det funkar på. Scrum ska man bara använda där det finns förutsättningar. Ni har sett diagram, är det kaos eller ordning, är det osäkert eller säkert. Försöka mappa in sig där och jag tror att det bara funkar i små projekt. Jeff Sutherland säger att det funkar i hur stora som helst. Han påstår ju att Google kör Scrum och Nokia kör Scrum, och det gör dem på sina ställen. Men det är inte så att hela dessa organisationerna scrummar loss i Scrum-of-scrums. Det är bara skitsnack. Det är inte sant.

MM Ja, det var nog alla frågor vi hade faktiskt, så vi tackar så mycket.

Kompletterande frågor

JO Har ni justerade sprintar så alla börjar och slutar samtidigt?

PB[60]: Så var det.

JO Kan du förtydliga hur relationen var mellan företagen?

PB[61]: Jag är anställd av Företag B som projektledare.

Jag var uthyrd till företag X. Företag X byggde systemet. Företag X hyrde in företag Y för att göra utvecklingsarbete.

I den roll jag hade på företag X jobbade jag alltså mycket med företag Y.

B.3 Utvecklare, Företag A

Genomfördes 2008-11-28

JO Joel Olofsson

UA Utvecklare A

JO Vi tänkte börja med lite inledande frågor. Hur stort är ditt team?

UA[1]: Alltså, mitt team är 6 personer. Sen, jag kollade lite på de här frågorna.. det beror på hur man menar också. Vi har en massa team på vår avdelning, men sen finns det ju massor av avdelningar och det är ju över hela världen också. Men, så det är viktigt att börja med vad man menar med team. Men det som vi kallar för team, då är det 6 personer. Och då, hur många team finns det. Ja, på vår avdelning finns det 5 team kan man säga. Sen har de teamen delar av sig på andra platser också, ett brasilien-team. Men det är väl 5 stycken kan man säga. Är det det ni är ute efter?

JO Ja, vi vill ju inte bara ha en siffra, utan mer en utläggning. Så det är bara bra.

UA[2]: Ja, för sen finns det ju hur många som helst på Företag A, men i det lilla så är det 5 team. Vad vi gör.. Jag är med i ett Eclipse-team, och det är ett samarbetsprojekt med företag X ([systerföretag]) att göra Eclipse-miljön bättre för utveckling i både Företag A och X. Så vi har uppgifter som kommer från Företag A och som kommer från X, så ska vi ha alla dem i samma produktbacklog, och så får vi då väga dem mot varandra. Vi har en styrgrupp med folk från både Företag A och X, för att väga, vilket som är viktigast. Då blir det att man jobbar lite med Företag A-saker, och då jobbar ju alla med det, och sen jobbar man lite med bara deras saker.

JO Du sa att där var en styrgrupp, hur fungerar de om man tänker på produktägaren i Scrum?

UA[3]: det är riktigt klurigt med det där för grejen är ju att vi vill ha en person. Om någon kommer utifrån och säger “-Gör det här.” så vill vi säga “-Prata med vår produktägare.”, men vår produktägare är en grupp personer som många av dem är höga chefer som har väldigt fyllda kalendrar. Så det är väldigt svårt att ha sammanträde så fort man vill ha in något i produkt backlogen. Så min chef, Bengt, han är liksom satt som produktägare, så att vi har en person. Men det är ändå styrgruppen som är produktägare. Medan sprintarna håller på, och vi ska föra in saker, så är det ändå här. Men är det oklarheter var det ska in, så får han prata med styrgruppen.

JO Så de kommunicerar rätt mycket med varandra?

UA[4]: Ja, det är ju tanken i alla fall. Så som vi har det nu så finns det styrgruppsmöten som är uppsatta, som är varje månad ungefär om inte oftare. Det är svårt att få ihop allt folk.

Så vår arbetsuppgift är allting som kan göra Eclipse-miljön bättre för utvecklare. Så då har vi ju samlat ihop krav från folk, “-Vad stör ni er på?” “-Vad vill ni ha för funktionalitet?”

JO Det är intressant att det är mellan de företagen också.

UA[5]: Mm, det är lite speciellt. Det ger ju komplikationer såklart. Men det känns som om det funkar ganska bra än så länge.

JO Sen kommer hur väl du tycker att ni följer Scrum.

UA[6]: Tja, halvt. Vi har en person i teamet som vill göra det precis som det står. Och sen vår Scrum-Master vill väl att man ska anpassa det, man märker att någon grej funkar och så kan man göra det så. Där hamnar vi väl lite i diskussionen hur vi ska göra. Men jag tycker att vi följer det ganska väl, det finns väl en del kluriga saker, som t.ex. estimat och story points, hur man ska göra det. Där har vi gått lite fram och tillbaka. Vi ville ju göra det helt i story points, så man säger “-Det här är så jobbigt”, och så kommer man fram till att vi klarar det *här*, och så i varje Sprint klarar vi så härr så många – vi har en velocity på 10 per

JO Men snackar du om alla team eller bara...

UA[7]: Nej, jag snackar om mitt team. Jag gissar på att vi försöker följa det mer än de andra teamen. De andra, ja de jobbar inte med story points, utan de jobbar med allt i tid, som väl man brukar säga att man inte ska göra i Scrum. Så jag tror att vi försöker följa det lite mer, vi använder t.ex. planning poker. Jag vet inte om det hör till Scrum från början eller om det bara är någon som har hittat på det, som är någon Scrumfantast.

JO Mm, för om ni använder olika system, kan ni förstå varandra?

UA[8]: Ja, våra kanske är svårare att förstå eftersom deras är ju mätta i tid. Det är ju inte så svårt att förstå. Men när vi säger att "vi tror att det här är 7 på vår skala" så är det ju svårare. Men i sprintarna så har vi planerat i tid. Men jag vet inte, det håller fortfarande på att sätta sig. Det är inte riktigt klart, nu har vi testat att planera sprintarna i tid, så får vi se hur det går.

UA[9]: Mellan teamen vet jag inte riktigt. Jag har inte så mycket koll, det är väl mest ScrumMasterns roll att vara medlare mellan. Men när vi går mellan teamen, är det snarare att vi behöver någonting från ett team. Då snackar man ju dels med teamet, "-Kan ni göra det". Men sen för att det faktiskt ska hamna i deras produkt backlog, så kan man gå via ScrumMastern, produktägare och säga att "-Vi tycker att de ska göra det här. -Vi tycker det är viktigt".

JO Finns det några bestämda rutiner för det?

UA[10]: Vi har produktbackloggarna i olika Excelark, så folk kan komma åt dem. Och sen har vi en flik i det excelarket där vem som helst kan lägga in "-Jag tycker att ni ska göra så här." och sen så får teamet, produktägaren och ScrumMastern för det teamet då sätta sig med den här och så ser de att någon tycker att man ska göra *det här*, och sen lyfta in det i produktbackloggen eller komma fram till att det var inget bra.

JO Du sa också att ni kanske går och pratar med dem.

UA[11]: Ja. Det är ju inga rutiner, men om man kommer på att man behöver någonting från någon, så är det lättaste sättet antagligen att bara snacka med dem och fråga om det låter vettigt eller svårt, skulle det ta lång tid?

JO Så det är bara att gå dit.. på eget bevåg?

UA[12]: Ja vi sitter ju ganska nära varandra. Framförallt bara för att få den första kontakten, "ja det kan vi ju göra", då kanske man gör det mer formellt, att man skriver in det. Så man vet att det inte tar år att göra. För då är det dumt att sätta det kravet.

JO Sen undrar vi hur ScrumMasters och produktägare är fördelade på teamen.

UA[13]: Vi har två stycken ScrumMasters på vår avdelning. Det är SA som ni har träffat, och sen är det Henrik. Och de är ju ScrumMasters båda, även för team på andra sajter också.

UA[14]: Produktägare är lite olika. För de flesta teamens produkter så är det min chef som är produktägare. För några, så gör vi det på den här avdelningen men produktägaren sitter någon annanstans. För att vara en bra produktägare, kan man inte vara det för hur mycket som helst. Han är produktägare, men han gör ju väldigt mycket andra saker också. Så det finns team som har en produktägare på ett helt annat ställe och styr därifrån. Men teamen jobbar på vår avdelning ändå.

JO Så det finns färre produktägare än team?

UA[15]: Ja, det finns det. Produktägaren är ofta produktägare för flera.

JO Du sa att ni satt ...

UA[16]: Ja, vår avdelning har en sådan här länga. Sedan utgörs vi av en sektor som är två andra avdelningar också. Vår avdelning sitter ganska samlat.

JO Och er avdelning är då ..

UA[17]: Den är cirka 20 personer.

JO Så det är 2-3 team eller?

UA[18]: Nja, det är fem team. Men i några av de teamen så sitter det knappt någon här, utan de är på andra ställen.

JO Men alla team som sitter här sitter tillsammans?

UA[19]: Ja precis.

JO Mm. Ni har någon specifik feature?

UA[20]: Ja, alltså vårt mål är ju att göra Eclipse-miljön bättre. Det är ju något kontinuerligt.

JO Det finns ingen annan som jobbar på samma sak?

UA[21]: På just det här, nä det finns det nog inte. Företag X har ju motsvarande team egentligen, håller på med Eclipse. Så det är lite samma. Men där är tanken att vi ska hjälpa till, vad ska man säga.. Det var mycket de [Företag X] inte hann göra, så när vi skulle skapa ett Eclipse-team skickade de med folk, så kunde vi ta en del av de sakerna som de skulle göra. Men på Företag A är det ingen annan som gör samma sak och det ska ju inte vara någon som gör precis samma sak, då är det ju något koordineringsfel.

JO Hur mycket tid lägger du på att kommunicera med utvecklare i andra team?

UA[22]: Ja,...

JO Det var ju lite med det du sa innan, att du kunde gå direkt till andra team och be om saker.

- UA[23]: Ja, man går ju inte dit och säger “-Vi vill ha det, gör det” till den direkte utvecklaren. Utan då är det snarare så att man kommer på att en sak de hade kunnat göra. Då kan man ju börja att ta en konversation med dem, om det är svårt att göra, går det att göra, är det ni som ska göra det? Och kommer man då fram till att det är de som ska göra det och de kan göra det så går man den formella vägen. Egentligen att ta in saker så från sidan, är ju inte så bra. Man vill ju att det ska synas i Sprintsen att man tar in det.
- JO Händer det att ni är beroende av vad någon annan gör så att ni måste ta reda på hur någonting fungerar som någon annan har gjort?
- UA[24]: Ja, så är det ju helt klart. Till exempel om det finns ett verktyg och man kommer på att man vill ha en Eclipse-plugin som tar hand om det här. Då behöver vi ju veta vad verktyget gör, om vi ska kunna ’wrappa’ det i en plugin. Antingen så är det ju om de har dokumentation, annars så går man och snackar med dem och tar reda på hur.. “-Kan du visa mig hur det funkar”.
- JO Brukar det finnas dokumentation?
- UA[25]: Det brukar väl finnas ganska mycket dokumentation. Finns det inte det så går du och tar reda på det av dem som har gjort det. Jag kan inte svara helt ja eller nej, för det skiljer ganska mycket.
- JO Har ni stormöten...
- UA[26]: Vi har avdelningsmöte en gång i veckan. Och då går man igenom.. dels är det nyheter som alla behöver, men sen brukar vi ha statusgenomgång också så man säger “-Ja, vi håller på med det här”, och sen går man vidare till nästa team och “-De håller på med det här” osv. Så får man en liten update om vad folk håller på med
- JO Har ni demos också, där ni visar vad ni gör?
- UA[27]: Vi har ju sprintdemos, men det är ju teamets produktägare och ScrumMaster. Man är inte så mycket på andras sprintdemos. Men sen ibland om det är något stort man har gjort som är väldigt bra så kan man visa det på avdelningsmötet också. Så vi har oftast, inte alltid men ibland, att någon visar vad de håller på med som alla borde få se.
- JO Ja, använder ni continuous Integration?
- UA[28]: Vi i vårt team har inte kommit igång med det än, men våra intentioner är ju att vi ska. Jag satt ju i ett continuous Integration-team tidigare, som handlade om continuous Integration för hela Företag A. Så det används ju men vi har inte kommit igång med det än. Dels så ska vi sätta upp lite byggmiljö så vi kan bygga headless, och efter det ska vi kolla på det. Så om någon månad eller något hoppas jag att vi kan ha det uppe och körande.
- JO Har ni cross-team members, att..
- UA[29]: Om det är folk som är med i flera team. Nej det har vi väl egentligen inte. De flesta avdelningar har ju några roller som arkitekter, kan ju vara att de är för avdelning, här och där och hjälper till. “-Hur ska vi strukturera det här.” så är arkitekterna med. Men vi har ingen som är med i flera team på så sätt.
- JO Har ni någon så här ledare, som är ansvarig för teamet?
- UA[30]: Ja, vi har en team lead. Mm, ScrumMastern kan ju ta lite den rollen också, att vara.. team lead ska vara lite ansiktet utåt, så kan ju ScrumMastern vara det också. Det är ju ScrumMasterns uppgift att vara interface, så om folk vill ha något ska de gå via ScrumMastern. Så det delas väl lite där, men vi har en team lead på varje.
- JO Blandad kompetens...
- UA[31]: Ja, rent naturligt har man väl blandad kompetens, eftersom folk har sysslat med olika saker innan.
- JO Vi tänkte nog mer, på deras branch..
- UA[32]: Ja, förra teamet jag satt kändes det som om det var mer så. Om det var en sån *här* uppgift, den går till *den* personen. Man vill väl alltid göra såhär att sprida kunskapen så alla kan allting. Det är ju en sån sak man oftast strävar efter, men det är ganska svårt eftersom.. ja vi vill göra det här snabbt, ja men han kan det! Men det känns som i det teamet jag är nu, att vem som helst kan ta en uppgift. Det känns som det är lite mer så i det här teamet. Så det tycker jag är rätt positivt. Men det är inte alltid så, som nu när vi håller på att porta pluginer till att vara baserade på ett annat ramverk. Då är det jag som har gjort pluginerna från början, och då jobbade jag med den ena av de här pluginerna med en person, och sen med den andra med en annan person. Då sprider man kunskapen. Men vi har väl blandad kompetens, men det är inte uttalat så riktigt. Men det blir ändå lite så, att okej det handlar om det här, så det är du duktig på, så. Då går det snabbare. Så man avväger det, göra saker snabbt, dvs. att folk gör det de kan, mot att försöka sprida det.
- JO Men om man säger.. alla är ändå utvecklare?

- UA[33]: Alla är utvecklare. Så är det helt klart.
- JO Nu tänkte vi lite på överlappande arbete. Händer det att ni har något gemensamt, som gör att ni måste samarbeta.
- UA[34]: Ja så är det ju helt klart. Vi har ganska mycket gränssnitt, eller vad man ska säga, med andra team.. “-Ja vi ska göra det här, då behöver vi det här från det här helt andra teamet”. Så vi har helt klart saker som vi kanske snart kommer att behöva samarbeta med andra team om. Och det är andra team på helt andra avdelningar också och inom avdelningen. Så det är massa saker i produkt-backloggen som är att man behöver information från *dem*, och möte med *dem* hur vi ska lösa detta. Det är vi som ska göra det, men vi behöver kanske någon funktionalitet från *dem*, eller så behöver vi bara veta hur man gör. Så att så är det helt klart för det här teamet. Det är väldigt olika mellan teamen hur mycket sånt de har. Vissa är ju lägre ner, sitter med lite lågnivå, och har kanske inte så mycket beroende. Vi som sitter i Eclipse-teamet är ju väldigt högt, så vi behöver vi ju saker från andra, byggsystem, SDK o så.
- JO Hur löser man det?
- UA[35]: Hur löser man det, ja man får ha möten och kontinuerlig dialog med *dem*.
- JO För att undvika att ni gör samma saker....
- UA[36]: Ja, samma saker, eller går åt helt olika håll. “-Nu är vi här -Jaha, men det blev ju jättesvårt, nu måste vi göra om allting”
- JO Ja, eller andra får vänta kanske för att..
- UA[37]: Ja precis, så det får man ju lösa genom att snacka. Snacka mycket.
- JO Händer det att ni får problem med det här med hinder. Att ni gör saker som hindrar andra.
- UA[38]: Ja, det är ju helt klart så att kommunikationsproblem uppstår ju. Nått exempel är väl att vi använder oss av någon funktionalitet, något annat team hade ändrat på sina saker, utan att riktigt veta att vi använde oss av *dem*. Och plötsligt fungerade det inte för oss. Men det kan man ju lösa genom att prata mycket om det. Så att har man en bra kommunikation så.., men har man inte det, om man inte diskuterar med de här teamen som man har angränsningsytor mot så kan det ju hända att de går åt olika håll och då funkar ingenting längre. Så det har ju hänt.
- JO Ser du att ScrumMaster och kanske produktägaren har nått ansvar i att försöka samordna?
- UA[39]: Ja, det känns som det är helt hur man vill sätta upp det, men ScrumMastern ska ju vara ansiktet utåt från teamet och vara ett interface mot andra. Så där kan han ju helt klart ha en roll tycker jag. Produktägare vet jag inte, om de behöver vara så inblandade i det här, men ScrumMaster tycker jag helt klart. Men då måste de ju känna till det här, så då får man informera sin ScrumMaster att “-Vi använder oss av det här som det här teamet gör”.
- JO ...så ScrumMastern kan ta upp det om han leder *dem* också..?
- UA[40]: Ja, och det är inte alls säkert att det är så. Exemplet jag pratade om var med ett team som satt på en annan avdelning.
- JO Då är det ScrumMaster som..
- UA[41]: Då kan han kommunicera med deras ScrumMaster. Men jag känner också att i våra fall här, så är ju teamet helt klart mer inblandat än vad ScrumMastern är, men jag tror ändå att ScrumMastern kan hjälpa till.
- JO Om det kommer in en stor uppgift som
- UA[42]: Ja, vem bestämmer vilket team som ska göra det. Oftast så sköts det ju så att teamen har ansvarsområden. Kraven kommer oftast in till teamen. Oftast är det ju så att “-Vi vill att det här verktyget ska fungera så” och då är det klart och tydligt vem som ska göra det. Så vad det här skulle röra sig var om det kom in ett mer allmänt stort krav, om att “-Vi vill ha en utvecklingsmiljö som gör det här, eller den här biten av utvecklingsmiljön” när det är något som inte alls finns på plats. Men.. om det kommer in till min chef, så antingen är det ju lätt att förstå vem det är som ska ha det, då tar han bara upp det med team leaden och ScrumMastern sen. Är det inte det så.. jag vet inte om jag har något exempel på det, men då får man väl lösa det genom att diskutera mellan teamet och med chefen. Men oftast så är det att kraven kommer in på en specifik del och då vet man ju vem det är som har det som ansvarsområde.
- JO Ja då har vi nog fått svar på det mesta. Tack.

Kompletterande frågor

- JO Har ni justerade sprintar så alla börjar och slutar samtidigt?

UA[43]: Nej, vi har inte alignade sprintar. Teamen på avdelningen har ganska väl avgränsade områden och har oftast inte så stora beroenden till varandra, så vi har inte sett något behov att samordna start och slut av sprintar. Vi kör "interna" sprintar om 2 veckor för att kunna vara rörliga och märka snabbt ifall något går snett. Sen har vi 4-veckorssprintar utåt sett, mot vår styrgrupp eftersom de inte har tid att samlas varannan vecka. Detta innebär att vi varje 2 veckor antingen har ett internt sprintdemo med bara produktägaren, scrummaster och teamet eller så har vi det riktiga sprintdemot med styrgruppen också. Sprintplanering/lessons learned har vi oavsett.

B.4 Utvecklare, Företag B

Genomfördes 2009-01-02

JO Joel Olofsson

UB Utvecklare B

JO Vi skulle först vilja ha svar på lite inledande frågor om projektet och det team där du ingick.

JO Hur stort är ditt team?

UB[1]: Vi var / är 8 personer i "mitt" team.

JO Hur många team finns det i projektet?

UB[2]: 5 team

JO Hur väl anser du att ni följer Scrum? (Inom ditt team och mellan teamen)

UB[3]: Inom teamen ganska väl (estimering / uppföljning i ScrumWorks, dagliga möten). Dock gör kundens preferenser att det är svårt att implementera Scrum fullt ut.

JO Hur är ScrumMasters och Product Owners fördelade på teamen?

UB[4]: En ScrumMaster / team, en "produktägare" för hela projektet.

JO Hur sitter ni i era lokaler?

UB[5]: Man har försökt samla hela projektet på ett ställe, dock har det växt och blivit för stort för att få plats. Vi sitter uppdelade per team, och alla team sitter hyggligt nära varandra på samma våningsplan.

JO Sedan skulle vi vilja veta hur arbetet koordineras hos er

JO Läger du mycket tid på att kommunicera med utvecklare i andra team? Vad är den främsta anledningen till detta och vilka team gäller det? På vilket sätt kommunicerar ni då?

UB[6]: Ganska mycket tid går åt till att kommunicera med andra team. Eftersom de olika teamen har beroenden mot varandra måste arbetet synkroniseras och det blir en hel del integrationsarbete. Kommunikation sker främst via mail, instant messaging eller genom att man helt enkelt letar upp personen som har informationen man behöver.

JO Om ni är beroende av något annat team, på vilket sätt vet ni hur man använder deras produkt? Informell kommunikation, via dokumentation, annat? Händer det alls?

UB[7]: Viss dokumentation finns på projektets Wiki, men mestadels via informell kommunikation.

JO Har ni stormöten med alla utvecklare där ni får reda på vad de andra håller på med? Vad tycker du i så fall om det?

UB[8]: Nej

JO Har ni personer som är med i flera team? Vilka och fyller det någon speciell funktion?

UB[9]: Nej, inte vad jag kan komma på just nu

JO Är era sprintar justerade tidsmässigt, så att de börjar och slutar samtidigt som de andra teamens sprintar? Finns det någon anledning till det?

UB[10]: Ja, alla team (utom ett) följer samma tidslinje. Det team som kör egna sprintar är lite speciellt på så sätt att det inte har beroenden mot övriga team på samma sätt som oss andra.

JO Angående beroenden mellan team:

JO Finns det något annat team som har med er att göra / som ni har tydligt beroende av?

- UB[11]: Vi har främst beroenden mot teamet som jobbar med backend-implementation
- JO Händer det att team “trampar varandra på fötterna” – att de gör samma sak?
- UB[12]: Nej, inte vad jag upplevt i alla fall
- JO Händer det att team har gjort saker som hindrat andra team?
- UB[13]: Nej
- JO Arbetsuppgifter:
- JO Vad jobbar ditt team med? Era arbetsuppgifter?
- UB[14]: Javautveckling
- JO Är dessa arbetsuppgifter fokuserat på en specifik funktionalitet i systemet? Går denna funktionalitet över flera “skikt” i systemet? Har ert team då blandad kompetens eller är alla i ert team lika när det gäller kompetens?
- UB[15]: Ja, vi är fokuserade på den mobila frontend-applikationen. Alla i teamet har i stort sett samma kompetens
- JO Hur sker arbetsfördelning i hela projektet? Om en stor ny uppgift kommer in, vem bestämmer vilket team som ska göra vad? Händer det att två team måste komma överrens hur man ska göra en gemensam uppgift?
- UB[16]: Eftersom teamen är fokuserade på olika delar av systemet (t.ex mobil frontend, webb-frontend, backend, betalningssystem) är det oftast ganska “självklart” vilket team som behöver göra vad om en stor uppgift ska lösas. Oftast blir resultatet att alla team behöver göra någon implementation för att få funktionaliteten på plats. Detta innebär ju även att alla team måste vara inblandade i diskussionen om hur saker och ting skall implementeras
- JO Integration:
- JO Använder ni Continuous Integration?
- UB[17]: Ja
- JO Finns det något som man kan säga är en integrationsfas, där allt byggs samman till en produkt? När sker isåfall den och vad görs då?
- UB[18]: Till projektet finns det kopplat CM-resurser, som är ansvariga för att bygga och paketera det som levereras till kund. Detta sker även regelbundet (1 -2 ggr / vecka) för deploy på våra interna testsystem. Före dessa byggen måste naturligtvis all kod vara incheckad och bygga i Hudson.
- JO När och hur sker annars integrationen av sprintarnas resultat, ifall det görs?
- UB[19]: Se föregående svar

Referenser

- Abrahamsson, P., Salo, O., Ronkainen, J. och Warsta, J. (2002), *Agile software development methods, review and analysis*, Technical Research Centre of Finland, VTT Publications, Espoo, Finland.
- Avison, D. och Fitzgerald, G. (2006), *Information systems development : methodologies, techniques and tools*, 4. ed. edn, McGraw-Hill, London.
- Backman, J. (1998), *Rapporter och uppsatser*, Studentlitteratur, Lund.
- Bedoll, R. (2003), A tale of two projects: How agile methods succeeded after traditional methods had failed in a critical system-development project, in F. Maurer och D. Wells, eds, 'Lecture Notes in Computer Science', Proceedings of the Third XP and Second Agile Universe Conference, Springer-Verlag, New Orleans, Louisiana, USA.
- Bertalanffy, von, L. (1950), 'An outline of general system theory', *British Journal for the Philosophy of Science* **1**(2), 134–165.
- Brooks, F. P. (1995), *The Mythical Man Month: Essays on Software Engineering*, Addison-Wesley. Anniversary Edition.
- Cheng, J. L. C. (1983), 'Interdependence and coordination in organizations: A role-system analysis', *Academy of Management Journal* **26**(1), 156–163.
- Cohn, M. (2009), 'Scrum team', <http://www.mountaingoatsoftware.com/scrum-team>. Hämtad 10 juni 2009. Bild återgiven med tillåtelse från författaren.
- Israel, M. och Hay, I. (2006), *Research ethics for social scientists : between ethical conduct and regulatory compliance*, Sage, London.
- Kähkönen, T. (2004), 'Agile methods for large organizations - building communities of practice', *Agile Development Conference, 2004* pp. 2–10.
- Kraut, R. E. och Streeter, L. A. (1995), 'Coordination in software development', *Association for Computing Machinery. Communications of the ACM* **38**(3), 69–82.
- Kvale, Steinarand Torhell, S.-E. (1997), *Den kvalitativa forskningsintervjun*, Studentlitteratur, Lund.
- Lyon, R. och Evans, M. (2008), 'Scaling up pushing scrum out of its comfort zone', *Agile 2008 Conference* pp. 395–400.
- Malone, T. W. och Crowston, K. (1994), 'The interdisciplinary study of coordination', *ACM Computing Surveys* **26**(1), 87–120.
- Mathiassen, L., Munk-Madsen, A., Nielsen, P. A. och Stage, J. (2001), *Objektorienterad analys och design*, 2. uppl. edn, Studentlitteratur, Lund.
- Miller, A. (2008), 'A hundred days of continuous integration', *Agile 2008 Conference* pp. 289–293.
- Miller, A. och Carter, E. (2007), 'Agility and the inconceivably large', *AGILE 2007 (AGILE 2007)* pp. 304–308.

- Mintzberg, H. (1989), *Mintzberg on management : inside our strange world of organizations*, Free Press, New York.
- Moore, E. och Spens, J. (2008), 'Scaling agile: Finding your agile tribe', *Agile 2008 Conference* pp. 121–124.
- Patel, R. och Davidson, B. (2003), *Forskningsmetodikens grunder : att planera, genomföra och rapportera en undersökning*, 3., [uppdaterade] uppl. edn, Studentlitteratur, Lund.
- Schwaber, K. (2004), *Agile project management with Scrum*, Microsoft Press, Redmond, Wash.
- Seale, C. (1999), *The quality of qualitative research*, SAGE, London.
- Sutherland, J. (2005), 'Future of scrum: parallel pipelining of sprints in complex projects', *Agile Conference, 2005. Proceedings* pp. 90–99.
- Turk, D., France, R. och Rumpe, B. (2002), Limitations of agile software processes, *in* 'In Proceedings of the Third International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2002)', Springer-Verlag, pp. 43–46.