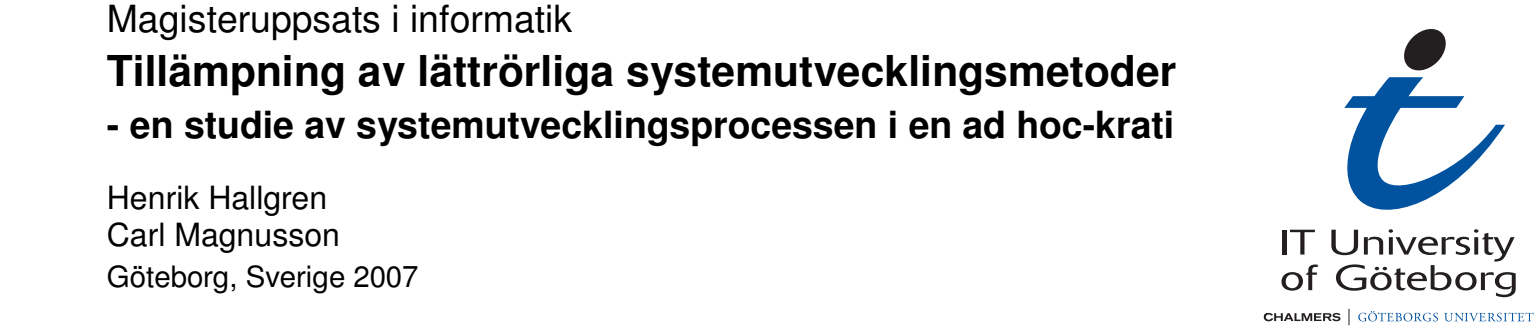
Magisteruppsats i informatik

Tillämpning av lättrörliga systemutvecklingsmetoder
- en studie av systemutvecklingsprocessen i en ad hoc-krati

Henrik Hallgren
Carl Magnusson

Göteborg, Sverige 2007



Institutionen för tillämpad informationsteknologi

REPORT NO. 2007:55

Tillämpning av lättrörliga systemutvecklingsmetoder

- en studie av systemutvecklingsprocessen i en ad hoc-krati

Henrik Hallgren
Carl Magnusson

Department of Applied Information Technology
IT UNIVERSITY OF GÖTEBORG
GÖTEBORG UNIVERSITY AND CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2007

Tillämpning av lättrörliga systemutvecklingsmetoder
- en studie av systemutvecklingsprocessen i en ad hoc-krati

Henrik Hallgren
Carl Magnusson

Rapportnummer: 2007:55
ISSN: 1651-4769
Institutionen för tillämpad informationsteknologi
IT-universitetet
Göteborgs universitet och Chalmers tekniska högskola
Box 8718
402 75 GÖTEBORG
031-772 4895

Göteborg, Sverige 2007

Applying agile methods
- a case study of the systemdevelopmentprocess in an adhocracy

Henrik Hallgren, Carl Magnusson
Department of Applied Information Technology
IT University of Göteborg
Göteborg University and Chalmers University of Technology

ABSTRACT
Due to fast growing and rapid changes in markets today, most of the organizations that are trying keep up with these changes have a high demand for software that are of high quality and comes with a low price. To meet these goals software companies have to implement new methods to develop software that are suited for rapid changes. The concept of agile methods can fulfill this need. Our purpose with this thesis was to investigate how well agile methods could be applied in organizations that could be identified as adhocracies. A case study was conducted in an organization which we could identify as an adhocracy. During the project a construction was developed with extreme programming methods as a starting point. The conclusion is that agile methods core values are similar to those in adhocracies in a number of ways and the method that was constructed appears to fit very well in organizations that could be identified as adhocracies.

The report is written in Swedish.

Förord

Arbetet med den här magisteruppsatsen har bedrivits under vårterminen 2007 genom en fallstudie på företaget Asivo. Arbetet med uppsatsen har pågått under 20 veckors tid. Under den tiden bedrevs också ett systemutvecklingsprojekt i 8 veckor. Vi vill tacka alla medarbetare på Asivo och ett speciellt tack till Jonas Lindblom som har varit en bra handledare på företaget och kommit med många bra synpunkter och idéer under projektets gång. Vi vill vidare tacka vår handledare Kjell Engberg som har varit ett mycket bra bollplank vilket har givit oss många bra uppslag. Vi vill även rikta ett stort tack till alla andra som har inspirerat och motiverat oss…

Göteborg 2007-05-24

Henrik Hallgren & Carl Magnusson

Författare: Henrik Hallgren, Carl Magnusson
Handledare: Kjell Engberg
Examinator: Faramarz Agahi
IA7400 Informatik: Magisteruppsats 20 poäng

Innehållsförteckning

Figur- och Tabellförteckning

1. Inledning

Plandriven systemutveckling där systemutvecklingsprocessen sker i ett sekventiellt flöde har tidigare varit tongivande som metod för systemutveckling. Under senare tid (1990-talet) har ett nytt ramverk för systemutvecklingsmetoder växt fram som ett svar på omvärldens allt snabbare förändringstakt (Lindvall et al., 2002). Kraven på att snabbt kunna leverera system med hög kvalitet till en låg kostnad har ökat och traditionella systemutvecklingsprocesser hanterar inte den problematiken på ett lika effektivt sätt (Everette, 2002). Enligt Everette kan kostnaden för att i efterhand hantera problem som uppstår med ett levererat system, vara 60-100 gånger högre än om problemen lösts under projekttiden då systemet utvecklades.

Lättrörliga metoder som är samlingsnamnet för den nya typen av tankegångar kring systemutvecklingsprocessen myntades 2001. Det som ett svar på att använda sig av nya systemutvecklingsmetoder i dagens samhälle, vilket är en förutsättning för att lyckas med sina systemutvecklingsprojekt (www.agilealliance.org). Lättrörliga metoder är inte en enskild metod för systemutveckling utan samlar ett tiotal systemutvecklingsmetoder med samma värdegrund och principer (Cockburn, 2001). Metoderna karaktäriseras av att de är iterativa och inkrementella vilket i sig inte är något nytt då Zurcher och Randell (1968) i slutet av sextiotalet lade fram tankar om att utveckla system genom att arbeta på ett iterativt sätt snarare än ett sekvensbaserat. Det som adderats under senare tid är den sociala aspekten såsom interaktion och kommunikation mellan de samverkande delarna i systemutvecklingsprocessen. Genom att placera personer fysiskt närmare varandra, använda ”whiteboards” och vanliga samtal för att ersätta dokument. Genom det kan kostnaden för att transportera information och att gå från beslut till handling minskas vilket i synnerhet också påverkar kostnaderna i en positiv riktning (Cockburn & Highsmith, 2001).

Även Lindvall et al. (2002) identifierar de sociala faktorerna som viktiga och uppger tre faktorer som direkt inverkar på hur slutresultatet i ett projekt kommer att bli, kulturen i organisationen, människorna och kommunikationen. De kulturella aspekterna är av stor betydelse för att få acceptans för det förändringsarbete som det innebär för en organisation att implementera en systemutvecklingsmetod i sina utvecklingsprojekt (Koch, 2004).

I flera artiklar (Cohn & Ford, 2003; Everette, 2002) fokuseras det på transitionen mellan traditionella processer kontra lättrörliga systemutvecklingsprocesser och framförallt fördelen med införandet av lättrörliga metoder i sin systemutvecklingsprocess. Lindvall et al. (2002) konstaterar att lättrörliga metoder bäst lämpar sig för mindre organisationer och att det finns stor erfarenhet samlat kring implementering av lättrörliga metoder där utgångspunkten har varit en traditionell systemutvecklingsmetod. Även Grennning (2001) har genomfört etnografiska undersökningar, för att se vad det blir för resultat genom att implementera en lättrörlig metod i en organisation som driver sina projekt med stark anknytning till traditionella plandrivna systemutvecklingsmetoder. Till skillnad från Grenning (2001) som fokuserar på att utgå ifrån en etablerad och strukturerad systemutvecklingsprocess till en annan, så syftar vår uppsats på att utgå ifrån organisationer som kan identifieras som ad hoc-kratier och som saknar förankring av traditionella systemutvecklingsmetoder. I denna typ av organisation är strukturen

organiskt, informell och arbetet anpassas efter kundens behov (Mintzberg, 1981). Vi går från en ostrukturerad systemutvecklingsprocess till att implementera en lättrörlig metod i ett systemutvecklingsprojekt. Det som vi ämnar undersöka är:

Hur väl fungerar det att implementera en lättrörlig metod som systemutvecklingsmetod i en ad hoc-krati utan förankring av traditionella systemutvecklingsmetoder?

Pikkarainen, Salo och Still (2005) presenterar i sin artikel ett ramverk för implementering av lättrörliga metoder i en organisation. Pikkarainen et al. (2005) utgår ifrån QIP[1] och anpassar det för lättrörliga metoder. För att kunna genomföra en kontrollerad anpassning av en lättrörlig metod i systemutvecklingsprojektet, ämnar vi tillämpa Pikkarainen, Salo och Stills framtagna ramverk för detta ändamål.

1.2 Disposition

Dispositionen av uppsatsen ser ut på följande vis. Avsnitt 2 beskriver och definierar den metod som vi använder oss av för att erhålla de resultat, vilka vi grundar våra slutsatser på. Avsnitt 3 presenterar det teoretiska ramverk som används i studien för att implementera och anpassa den lättrörliga metoden. I detta avsnitt beskrivs också den typ av organisation som kunde identifieras under fallstudien. Vidare beskrivs också här de typer av lättrörliga metoder som är aktuella som metod för den organisation där vi ämnar implementera en lättrörlig metod. I avsnitt 4 återfinns det resultat från fallstudien som framkommit under systemutvecklingsprojektet. Avsnitt 5 innehåller den diskussion där vi diskuterar om vad vi kommit fram till under fallstudien. Avsnitt 6 presenterar den slutsats som är ett direkt svar på frågeställningen som ställdes i inledningen av uppsatsen. Avsnitt 7 innehåller de referenser som det refereras till i den löpande texten. Avsnitt 8 består av bilagor till uppsatsen såsom insamlat material under fallstudien som har utgjort grunden för det resultat som presenterats.

[1] Quality Improvement Paradigm. En metod i tre steg för att förbättra utvecklingsprocessen (http://herkules.oulu.fi/isbn9514265084/html/x287.html)

2. Metod

2.1 Val av metod

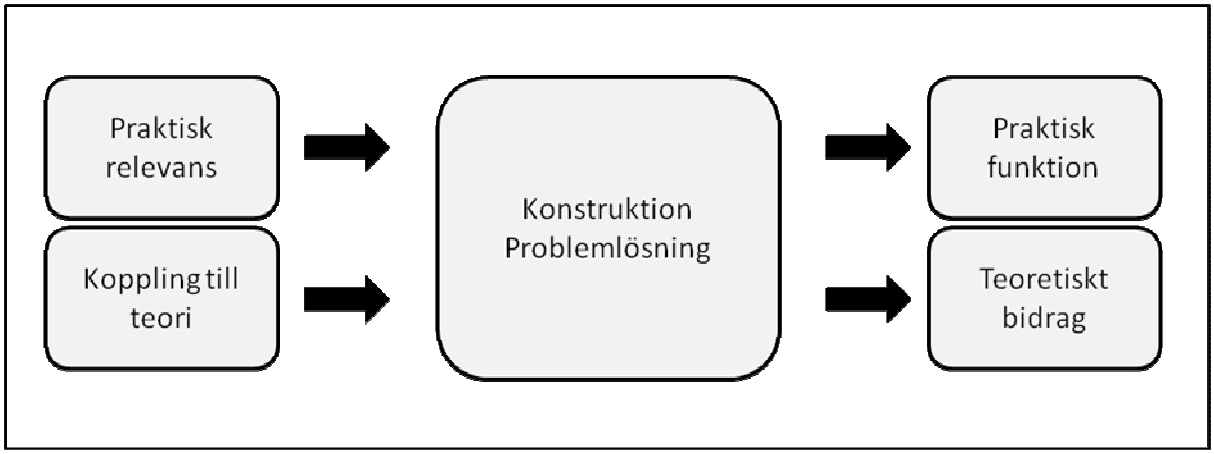
Uppsatsens frågeställning är det som styr valet av metod. I den mening att den metod som skall väljas också är den metod som förväntas ge mest tillförlitlig kvalitativ data som möjligt (Easterby-Smith, Thorpe & Lowe, 2002). Metodvalet baseras också på den access som ges forskaren. En bra metod som svarar bra mot frågeställningen kan bli en mindre bra metod vid insamling av data om forskaren enbart ges partiell access till ett område, där hela området hade varit relevant för att erhålla material som ger möjligheter att svara på frågeställningen med god validitet. Metodvalet avspeglar sig också i traditionella värderingar som forskaren har och också den kontext som hon rör sig i. Vilket innebär att om det vetenskapliga synsätt som hon har kan identifieras som positivistiskt, relativistiskt eller social konstruktivistiskt också avspeglas i vilken metod hon använder sig av (Easterby-Smith et al., 2002).

Vi ämnar genomföra vår undesökning genom ett praktiskt tillvägagångssätt och använda oss av ett ramverk, konstruktiv forskningsansats som presenteras av Kasanen, Lukka och Siitonen (1993). Med hjälp av ovannämda metod ämnar vi konstruera en prototyp och kring den konstruktionen samla in kvalitativa data för att föra en diskussion och komma fram till en slutsats. Utgångspunkten tas i etablerade lättrörliga systemutvecklingsmetoder och en av dem implementeras i ett systemutvecklingsprojekt och en prototyp konstrueras i form av en lättrörlig systemutvecklingsmetod som är anpassad för den typen av verksamhet som vi undersöker.

2.2 Konstruktivistisk metod

Konstruktiv forskning som arbetssätt och metod för att angripa ett problem har varit ett etablerat tillvägagångssätt inom naturvetenskapen. Enligt Lukka (refererad i Lukka, 2000) är ett praktiskt orienterat förhållningssätt en förutsättning för att möjliggöra konstruktioner av artefakter, och det resultat som är en produkt av konstruktionen. I och med att konstruktionen fungerar utgör det ett bevis på ett korrekt antagande. Kasanen et al. (1993) presenterar sitt ramverk konstruktiv forskningsansats som ett sätt att använda sig av tillämpad forskning även inom områden där det handlar om ett konstruktivistiskt tillvägagångssätt, men inte nödvändigtsvis en konstruktion av en teknisk artefakt. Kasanen et al. (1993) definierar konstruktiv forskningsansats som en metod för att konstruera modeller, planer och diagram för att lösa problem som uppstår i styrning av verksamheter. Målsättningen med konstruktiv forskningsansats är av dubbel natur. Den ger möjlighet till en praktisk lösning men bidrar också med ett teoretisk bidrag. Den praktiska lösningens existens är möjlig enbart om konstruktionen varit framgångsrik. Det som på förhand fungerar i form av tekniska termer av hur en prototyp ska implementeras säger ingenting om hur det fungerar i verkligheten (Kasanen et al., 1993). Det som är huvudkärnan i konstruktiv forskningsansats är konstruktion och

problemlösning (se figur 1). Eftersom det är inom den ramen som innovationen sker. Den konstruktion som skapas fylller två funktioner. Det ena är att konstruktionen fyller en praktiskt funktion för organisationen och det andra är att resultatet av konstruktionen ger ett teoretiskt bidrag. I slutändan innebär det att om ingen ny lösning påträffas finns heller ingen anledning att fortsätta med studien (Kasanen et al., 1993).



Figur 1 Flödet i konstruktiv forskningsansats (Kasanen et al., 1993).

Kasanen et al. (1993) definierar konstruktiv forskningsansats som ett sekventiellt flöde i sex steg. Lukka (2000) adderar ett sjunde steg till metoden och påvisar vikten av att som forskare vara närvarande i organisationen under en längre tid. Då man som forskare i en organisation ska arbeta efter konstruktiv forsknings ansats och konstruera en prototyp ute i en organisation menar Lukka (2000) att det är av stor vikt att etablera ett långsiktigt forskningssamarbete under en längre tid med vald organisation. I den här uppsatsen har vi haft en begränsad tidsplan för att konstruera en prototyp. Med anledning av detta väljer vi att inte ha med Lukkas tillägg till den befintliga modellen.

2.2.1 Hitta ett praktiskt relevant problemområde med forskningspotential.

Kärnan i konstruktiv forskningsansats är det praktiska problem som får sin lösning genom att använda metoden för att konstruera en lösning. Avgörande för detta är beslutsfattaren i organisationen som direkt har en påverkan och koppling till problemet och dess lösning. Emellertid ska problemet också ha en vetenskaplig anknytning vilket står utanför organisationens kontroll vilket också distanserar materialet ifrån att vara en konsultrapport (Kasanen et al., 1993). Det medför att materialet blir publicerat till skillnad från konsultrapporter som enbart publiceras internt och enbart innehåller en lösning på problemet. I konsultrapporten kan det också förekomma uppgifter som är direkt olämpliga för företaget att publicera för omvärlden ur konkurrenssynpunkt(Kasanen et al., 1993). För att problemområdet ska ha en vetenskaplig förankring gäller att inga tidigare lösningar på problemet finns publicerade. Vilket innebär att man inte ska kunna påvisa en lösning genom enbart granskning av litteratur (Lukka, 2000). Enligt Labro och Toumela (2003) anger inte

flertalet av publicerade artiklar, som använt sig av konstruktiv forskningsansats som metod, hur forskare hittar organisationer som presenterar ett relevant problemområde med en vetenskaplig grund. I denna uppsats gav sig organisationen själv tillkänna genom en presentation på IT-universitetet och därmed kunde en diskussion om problemområde och relevant frågeställning företas med organisationen i ett tidigt skede av uppsatsperioden. För att säkerställa att det vetenskapliga resultat man erhåller är tillräckligt intressant bidrag till forskningen och att konstruktionen är möjlig att skapa. Det är viktigt att göra den bedömningen i inledningsskedet, så att projektet inte måste avbrytas på grund av svårigheter med att färdigställa konstruktionen (Labro & Toumela, 2003).

2.2.2 Inhämta generell och omfattande kunskap inom problemområdet

Det andra steget i konstruktiv forskningsansats följer med under hela projektets gång då ny kunskap adderas allteftersom konstruktionen skapas, då teoretiska kopplingar väntas uppkomma efter att slutsatser kan dras utifrån konstruktionen. Enligt Lukka (refererad i Lukka, 2000) särskiljer sig inte detta ifrån andra typer av metoder, ifråga om tillvägagångssätt för att tillskansa sig kunskap om problemområdet. Där Lukka nämner intervjuer och observationer som exempel. I uppsatsen har vi genom intervjuer i inledningsskedet och också genom observationer kunnat identifiera vilket typ av verksamhet som man bedriver och hur organisationen är uppbyggd. Vi har också kunnat identifiera vilken typ av organisationsstruktur som organisationen arbetar utefter. För att kunna verka i så bra forskningsmiljö som möjligt menar Labro och Toumela (2003) att redan inledningsvis lägga ner mycket tid på att göra sig bekant med arbetsplatsen i form att aktivt bekanta sig med personalen. En bra integration ger större förutsättningar att förstå beteenden bland personalen vilket i sin tur påverkar konstruktionen. I uppsatsen har vi genom litteraturstudier inhämtat kunskap om problemområdet i stort och fått en mer generell bild av problemet. Vilket har givit oss möjlighet tillsammans med identifiering av hur verksamheten i organisationen bedrivs, att göra ett selektivt urval av en lättrörlig metod som används som utgångspunkt i systemutvecklingsprojektet.

2.2.3 Konstruera en lösning på problemet

I den här fasen råder stor kreativitet och det som särskiljer konstruktiv forskningsansats från aktionsforskning är sökandet efter både en praktiskt och en teoretisk lösning. Konstruktiv forskningsansats är inte applicerbar där en teoretisk modell utgör bas och att konstruktion sker efter den teoretiska modellen även om den praktiska lösningen är ny för organisationen i fråga (Labro & Toumela, 2003).
I uppsatsen använder vi oss av ett teoretiskt ramverk som är framtaget för implementering och anpassning av lättrörliga metoder i verksamheter som bedriver systemutvecklingsprojekt. Konstruktionen består i att skapa en lättrörlig systemutvecklingsmetod som passar en organisation som kan identifieras som en ad hoc-krati med utgångspunkt från den lättrörliga utvecklingsmetoden Extrem programmering (XP). Där XP som metod implementeras och anpassas i systemutvecklingsprojektet med hjälp av det teoretiska ramverket. I varje iteration

adderas nya slutsatser till konstruktionen som passar in på typen av organisation och sättet som arbetet bedrivs på.

För att kunna konstruera en lösning på problemet och dra slutsatser så använder vi oss av två datainsamlingsmetoder. Den ena är dagboksanteckningar som har skrivits under varje iteration och den andra är intervjuer som genomförts i början av projektperioden och i slutet av perioden.

2.2.3.1 Dagbok som metod för insamling av data

Att använda dagbok som metod för att samla in data har använts ända sedan andra världskriget, där det gjordes massobservationer på befolkningen i Storbritannien. Där en stor mängd vanliga människor ombads att skriva dagbok om allt som de var med under varje dag som studien gjordes (Easterby-Smith et al., 2002). Studien gjordes för att se hur den brittiska befolkningen reagerade på olika händelser under kriget. En stor fördel med att använda dagbok som insamlingsmetod är att informationen som ges inte blir påverkad av forskarens egna preferenser. Dagboksförfattaren måste själv formulera sig och kan inte ta hjälp av forskaren, vilket kan vara fallet när bara intervju används som forskningsmetod. Det är också lätt att byta miljö och organisation för forskaren om den så önskar. En nackdel med att använda metoden är att den inte tillåter den interaktion som en intervju eller observation kan göra, där det finns större möjligheter med följdfrågor och uppföljning om så behövs. För att komplettera den data som har samlats in så går det även att använda intervjuer som datainsamlingsmetod eller observationer (Easterby-Smith et al., 2002).

Enligt Richard Thorpe (Easterby-Smith et al., 2002) som har använt dagboksmetoden i många av sina studier så finns det några viktiga lärdomar att tänka på. Det första är att välja deltagare som har förmågan att uttrycka sig i skrift. Det är viktigt att skapa någon form av struktur på dagboken vilket skapas genom att lista några punkter som ska vara med i dagboken varje dag. En annan viktig lärdom är att vara uppmuntrande och stödjande till dagboksskrivarna under hela studien, annars kan det leda till att alla inte fortsätter och skriva dagbok under hela studietiden. Vidare är det väldigt viktigt att anonymitet och integritet används vid den typen av studier eftersom det kan kännas utlämnande att skriva om sina personliga åsikter.

När dagbok används som metod för att samla in data så kan de vara antingen kvalitativa eller kvantitativa beroende på vilken information som de ska samla in. Att använda den kvantitativa modellen som insamlings metod gör att det går att dra statistiska slutsatser och härledningar genom att hitta mönster av aktiviteter. När dagboken innehåller mycket egna funderingar och slutsatser eller andra former av konstruktiva idéer så går det mer åt den kvalitativa dagboksmetoden. Genom att använda den kvalitativa dagboksmetoden så finns det möjlighet att få fram en rik bild av olika motiv och perspektiv för forskaren (Easterby-Smith et al., 2002).

2.2.3.2 Intervju som datainsamlingsmetod

Intervjuer som insamlingsmetod i forskningsvärlden kan delas upp i flera olika typer av intervjuer, där den allra enklaste intervjuformen är enkätintervjuer som ofta består av flervalsfrågor vilket kan användas om det är lite enklare frågor som ska besvaras. Denna metod är väldigt statisk det vill säga att det inte finns utrymme för flexibilitet och följa upp intressanta svar. Om forskningen kräver lite mer flexibilitet i datainsamlingen så är kvalitativa intervjuer att föredra. Den kvalitativa intervjumodellen går att dela upp i strukturerade och semistrukturerad intervjuer, där strukturerade intervjuer följer en i förväg framtagen frågestruktur och frågorna ställs exakt likadant och i samma ordning (Cornford & Smithson, 2006). Att använda strukturerade frågor gör att det blir hög jämförbarhet mellan intervjuerna, vilket gör att denna typ av frågor lämpar sig till statistiska jämförelsemodeller. Den semistrukturerade modellen ökar flexibiliteten hos intervjuaren där intervjun följer en intervjuguide som ska fungera som milstolpelista där viktiga punkter finns noterade så att intervjun belyser de områden som den är tänkt att göra. Frågorna som ställs följer inte någon speciell ordning utan vilka frågor som ställs beror på vad den intervjuade har svarat. Så frågorna ställs i en ordning som beror på vad svaret blir vilket ska göra att intervjun ska flyta på smidigare och flödet ska kännas naturligt. Det finns dock en risk med den här metoden, jämförelsen mellan olika intervjuer kan bli svårare och följden kan bli att det blir väldigt svårt att tolka den insamlade datan.

Två andra intervjutyper än de ovan nämnda är observationsintervjuer och gruppintervjuer, där gruppintervjuer samlar en grupp människor som samlas och diskuterar ett givet ämne som intervjuaren har gett dem. Den ostrukturerade intervju formen har inga i förväg bestämda frågor utan intervjun påminner om ett vanligt samtal, men där intervjuaren vet i förväg vilken typ av information som är av intresse. Den här formen intervjuer ställer stora krav på intervjuaren som måste se till att rätt typ av information kommer fram. Ett problem med intervjuer i allmänhet är att det kan vara svårt att komma fram och intervjua rätt personer, en del kanske inte tycker att de har tid. Så för att öka möjligheten att nå fram med sina intervjufrågor så finns det lite olika möjligheter att genomföra intervjun på ”face to face”, e-post, telefon, videosamtal eller med webbformulär. Där de olika forumen har olika möjligheter och potential, ett webbformulär ger i princip ingen möjlighet till återkoppling och om intervjuaren skulle vilja ställa följdfrågor så är den möjligheten begränsad. Medan ”face to face”, telefon och videosamtal har goda möjligheter att göra uppföljningar på varje svar om så önskas (M.Bergquist, 2006).

2.2.4 Demonstrera att lösningen fungerar

Att kunna visa att konstruktionen fungerar i praktiken är av dubbel betydelse. Det betyder dels att forskningsprocessen har varit lyckad och dels att det man faktiskt byggt är klart, och är redo för att testas i praktiken (Lukka, 2000). I uppsatsen sker testning och utveckling av konstruktionen samtidigt. Vilket innebär att ett skarpt utvecklingsprojekt bedrivs med en lättrörlig metod som anpassas för organisationen i iterativa steg. Under systemutvecklingsprojektet omarbetas metoden för att anpassas

till verksamheten och testning huruvida det fungerar eller inte görs kontinuerligt allt eftersom projektet och iterationerna framskrider. Om något sätt att arbeta på inte fungerar tas det bort eller omarbetas för att testas av i nästa iteration. Enligt Labro och Tuomela (2003) är en av de viktigaste aspekterna i detta steg att vid implementering av konstruktionen fortsätta med ett nära samarbete och att inte forskaren slutar sitt arbete i verksamheten. Om så är fallet får forskaren inte vetskap om varför konstruktionen inte implementerades. Eftersom vi testar konstruktionen samtidigt som vi anpassar den under projekttiden har vi stora möjligheter att närvara och erhålla viktigt återkoppling från projektledningen.

2.2.5 Undersök inom vilket kontext som lösningen är implementerbar

När konstruktionen är färdig och den har testats i praktiken där det fastställts att den fungerar tillfredställande bör forskaren i det här steget diskutera vad som kan implementeras i ett vidare perspektiv. Vad kan överföras till andra organisationer från de resultat som erhållits i forskningsprocessen? Vilket samtidigt har givit ett positivt resultat i testningsförfarandet (Labro & Tuomela, 2003). Även om inte implementering av konstruktionen lyckas menar Lukka (2000) att även de slutsatser kan adopteras i ett vidare perspektiv för att bidra med erfarenheter för att undvika de problem som uppstått. Kasanen et al. (1993) presenterar tre nivåer på hur konstruktioner kan implementeras i ett vidare perspektiv.

- ”Weak market test”

  Det är det första steget som anses uppfyllt om personalen är positivt inställd till konstruktionen och faktiskt använder sig av den i den egna organisationen.
- ”Semi-strong market test”

  I det här fallet har konstruktionen blivit implementerad i flera företag på marknaden.
- ”Strong market test”

  Om konstruktionen passerar strong market test har den blivit såpass allmänt accepterad att det är som ”bröd och vatten”. Konstruktionen är ett måste för verksamheten för utan den produceras inte lika bra resultat (Labro & Toumela, 2003).

I uppsatsen når vi enbart upp till ”weak market test”. Vilket enligt Lukka (2000) är det enda möjliga steget att uppnå under tiden konstruktiv forsknings ansats används som metod för att konstruera en lösning i en organisation.

2.2.6 Visa de teoretiska kopplingarna och lösningens bidrag till forskningen

De teoretiska kopplingarna som kan visas är ett led i att konstruktionen kan transfereras till andra organisationer utanför den organisation där problemområdet uppstod och konstruktionen skapades (Kasanen et al., 1993). Enligt Lukka och Kasanen (1995) innebär det att en konstruktion som skapats och fungerar i en organisation, också har förutsättningar att implementeras i andra organisationer med samma organisationsstruktur. Det är dock en sak att säga att förutsättningarna finns

och en annan att verkligen implementera konstruktionen i andra organisationer vilket Lukka och Kasanen (1995) påpekar.

3. Teori

3.1 Organisationer

Att beskriva organisationer på ett övergripande sätt kan bli svårt eftersom det kan skilja mycket mellan olika organisationer, men det som alla organisationer har gemensamt är att de ska lösa en eller flera uppgifter (Jacobsen & Thorsvik, 2002). Arbetsuppgifter och arbetsrutiner mellan ett företag som gör service på bilar och en psykiatrisk mottagning skiljer sig mycket åt vilket gör det svårt att beskriva organisationer på ett generellt sätt. Eftersom det kan skilja mycket mellan olika arbetsuppgifter inom olika organisationer så går det också att förvänta att organisationerna är organiserade på olika sätt (Jacobsen & Thorsvik, 2002). För att kunna förstå hur det kan skilja så mycket mellan olika organisationer så måste typologierna varuproducerande och tjänsteproducerande företag klarläggas. Den mest uppenbara skillnaden mellan varu- och tjänsteproducerande företag är att varuproducerande företag tillverkar fysiska ting, som går att rent fysiskt greppa och ta på.

Tjänster är lite mer svår definierat, när en tjänst produceras så konsumeras den samtidigt av kunden. En tjänst går inte att lagra eller lagerhålla, en vara kan lagras och tillverkas på ett ställe men konsumeras på en annan plats. En tjänst utbyts ofta ansikte mot ansikte, där en människa skapar tjänsten varje gång som den efterfrågas. Det är människor som skapar tjänsten vilket innebär att tjänsteproduktion är mer arbetsintensiv, medan produktion av varor ofta sker med hjälp av maskiner. En viktig skillnad mellan varu- och tjänsteproduktion är att tjänsteproducerande företag är mer beroende av humankapital och varuproducerande företag är mer kapitalintensivt. Hur de olika typerna av organisationer ska struktureras och ledas varierar stort. I tjänsteföretag så är det viktigt med motiverade och serviceinriktade människor, därför är det av största vikt att sådana organisationer skapar en kultur som främjar den typen av värderingar. Att skapa engagemang hos medarbetarna borgar för en kvalitetssäkring(Mullins, Walker, Boyd & Larréché, 2005).

Enligt Mintzberg (1981) är en annan viktig faktor som måste undersökas för att kunna förstå hur en organisation fungerar och är strukturerad, är att se på vilka möjligheter det finns för rutinisering av arbetsuppgifter inom organisationen. Organisationsstruktur syftar på hur en organisation använder styrsystem och fördelning av auktoritet, arbetsdelning, samordning och kontroll av arbetsuppgifter. Arbetsuppgifter som är liknande och utförs på samma sätt varje dag kan lätt standardiseras. Arbetsuppgifter som varierar från gång till gång blir svårt att standardisera vilket ställer högre krav på den som utför arbetsuppgiften. En arbetsrutin vid en maskin på en fabrik kan standardiseras och utföras på samma sätt varje dag, en resebyrå som servar olika kunder från dag till dag är inte lika

standardiserat även om många av arbetsuppgifterna är lika, så kommer önskemål och andra krav skilja mycket från kund till kund vilket gör att personalen som servar kunderna måste ha stor förmåga att kunna anpassa sig till olika situationer.

3.2 Organisationsformer

Vilken organisationsform som ett företag har beror på vilka uppgifter, teknik och storlek organisationen har. Detta har lett till att det under vissa förutsättningar finns vissa organisationsformer som är mer ändamålsenliga än andra (Jacobsen & Thorsvik, 2002). Enligt Mintzberg (1981) så består en organisation i varierande grad av fem huvuddelar: 1, en operativkärna som utför det arbete som är förknippat med själva produktionen i företaget, det kan vara försäljning, inköp och produktion. 2, mellancheferna är de som har till uppgift att övervaka och leda produktionen, de ska fungera som en länk mellan den operativa kärnan och ledningen. 3, strategisk ledning är de som har det högsta ansvaret för organisationen här hittar man VD, avdelningschefer och annan ledande administrativ personal. 4, teknostrukturen består av olika stabsfunktioner som kan tänkas behövas inom organisationen. Den gruppen av människor som ingår i teknostrukturen ingår inte i själva produktionen utan har t ex som uppgift att bedriva internutbildningar och sköta ekonomikontroll. 5, Servicestrukturen är också en grupp människor inom organisationen som inte är inblandade i själva produktionen utan kan vara t ex vaktmästare, städare och receptionister. Mintzberg (1981) karaktäriserar organisationer efter vilken typ de är och hur de bedriver sin verksamhet och delar in dem i fem olika kategorier: entreprenörsorganisationen, maskinbyråkratin, divisionaliserad organisation, professionell byråkrati och ad hoc-krati eller den innovativa organisationen.

3.2.1 Ad hoc-kratin eller den innovativa organisationen

Den här typen av organisationer som benämns som innovativa eller Ad hoc-kratier kallas även för organiska organisationer. Vilket kännetecknas av följande egenskaper: 1, Nätverksstruktur för auktoritet, kontroll och kommunikation. 2, Arbetsuppgifterna omdefinieras ständigt och anpassas efter behov. 3, Den enskildes roll är generellt definierad. 4, Kommunikationen är både vertikal och horisontell. Allt efter behov (Jacobsen & Thorsvik, 2002). Den största styrkan med här typen av organisationer är att de främjar kreativitet och innovationer. Det kan ofta vara svårt att se någon struktur, om det finns någon struktur så är den ofta väldigt oklar och flytande. Mintzberg (1981) gör skillnad på två olika typer av innovativa organisationer: operativa och administrativa ad hoc-kratier. Där den operativa ad hoc-kratin ofta ska lösa problem åt andra t ex konsultfirmor.

Men administrativa ad hoc-kratier skapas för att främja innovation inom den egna verksamheten. Genom att använda den innovativa organisationsformen så finns det en risk för att det kan bli oklara auktoritetsförhållanden vilket kan leda till frustration och osäkerhet (Jacobsen & Thorsvik, 2002). Genom att använda den här företagsstrukturen så skapar det förutsättningar för att ta tillvara personalens kompetens, det skapar även goda förutsättningar för flexibilitet och god samordning.

3.3 Lättrörliga metoder

3.3.1 Extrem programmering

Metoden extrem programmering som brukar kallas för XP - utveckling skiljer sig från traditionella utvecklingsmetoder genom att XP - utveckling förespråkar en rik kontakt och ett nära samarbete med kunden genom direkt kommunikation. Fokus ska ligga på personligt samarbete och användningen av olika analys- och designdokument ska begränsas (Beck, 2000). Genom att XP - utveckling förespråkar ett nära samarbete mellan kund och utvecklare så ska det öka flexibiliteten, det vill säga om det tillkommer nya företeelser som måste beaktas så ska det var lätt att införa dessa förändringar. En grundsten inom XP-programmering är att det inte finns någon sekvensorienterat arbetsflöde där olika sekvenser ska följas utan att arbetet ska utföras iterativt.

I XP så ingår fyra huvudkategorier (www.extremeprogramming.org) under varje kategori så varierar det vilka underpunkter som ingår beroende på vilken XP-metodik som används. Tanken med detta är att organisationer som väljer att implementera XP ska kunna välja ut de delar som passar likt ett smörgåsbord:

- Planering
- Kodning
- Design
- Testning

”Face to Face”, Dagliga möten

Punkten ”face to face” eller dagliga möten benämns lite olika beroende på vilken XP metodik som man arbetar efter, vi väljer att använda begreppet ”face to face”. Men båda benämningarna syftar på samma sak, att varje dag ska starta med ett möte med alla deltagare i projektet. Det dagliga mötet syftar till att belysa och lösa problem som kan uppstå, det dagliga mötet ska också se till att projektgruppen behåller fokus på projektet (http://www.extremeprogramming.org).

Planeringsmöten

Att ha regelbundna planeringsmöten med kunden och planera hur projektet ska utvecklas och bena ut vad som är viktiga prioriteringar och vad som inte är viktigt. Planeringsmöten strävar efter att svara på två frågor: vad ska åstadkommas till ett specifikt datum och vad som ska göras efter det. Planeringsmöten ska hållas på veckobasis där nya direktiv ges och uppföljning av uträttat arbete sker. Utvecklarna

bygger program under korta tidsperioder och visar sedan upp det för kund vilka ger respons direkt och åtgärder för att realisera direktiven sätts igång direkt.

Testning
Utvecklingen sker genom kontinuerlig testning av systemet, där användare och kund ger konstant återkoppling. Metoder för att testa programkoden ska konstrueras före utvecklingen av källkod för att veta vad metoder och funktioner ska uppfylla. Testning av koden och demonstration för kunden ska ske inom korta tidscykler för att på ett tidigt stadium upptäcka fel. Kunder får också hela tiden möjlighet att komma med synpunkter och förslag till ändringar. Eftersom testningen sker inom korta tidsperioder så ökar möjligheten till flexibilitet.

Parprogrammering
Mjukvaruutvecklingen ska ske i par om två stycken där den ena kodar och den andra tänker ut hur problem och funktioner ska lösas. De olika rollerna i parprogrammering är inte beständiga utan utvecklarna ske växla mellan de två rollerna. Tanken med parprogrammering är att kodningen sker sida vid sida och att utvecklingen sker på en och samma dator. Genom att använda parprogrammering så ska det bli bättre kvalitet på koden, bättre testning och bättre design. Forskning om parprogrammering visar att bättre kod produceras och mindre felaktig kod genereras (Koch, 2004). Två individer bidrar med olika kunskap och den totala kunskapsbasen blir större.

Återvinning av kod
Återvinning av kod handlar om att hitta mönster i koden som återkommer, där funktioner och andra beteenden är snarlika varandra. Genom att göra funktioner som är lika mer generella så är tanken att de ska gå att använda till att lösa uppgifter i framtiden som har liknande problematik. Det gör att koden går att återanvända mer vilket ska leda till att man sparar mer tid och arbetsresurser. Antal rader kod som produceras ska minskas och det leder även till bättre kvalitet på koden.

Enkel design
Användning av enkel design ska användas genom hela projektet från start till mål och det innefattar både programtestning och andra design förbättringar. I en iterativ utvecklings metod som extrem programmering så är bra design viktig eftersom det skapar förutsättningar att gå vidare med utvecklingen.

Gemensamt ägande av kod
Med extrem programmering som metod så ska varje programmeringspar kunna ta del av andra programmeringspars kod och kunna göra förbättringar och säkerställa att koden är skriven på ett korrekt sätt. Vilket borgar för ökad kvalitet på koden och minskar risken för att felaktig kod används. Genom gemensamt ägande av koden så blir det enklare att ändra andras kod och kommat tillrätta med upprepningar som kan uppstå i koden annars.

Kontinuerlig integrering
Kontinuerlig integrering sker så fort en del är klar så integreras den med det redan färdiga systemet. Vilket i extrem programmering kan ske flera gånger om dagen eller på veckobasis. Genom att frekvent integrera koden så ska integrerings problem minskas eller upptäckas på ett tidigt stadium.

Kund på plats
Kunden är med under hela utvecklingen av programmet/systemet för att kunna svara på frågor. Kunden ska finnas tillgänglig när som helst för att snabbt kunna bidra med sin kunskap. Kund och utvecklare ska sitta under samma tak och lokal för att hela tiden finnas på plats för att främja samarbete.

Små lanseringar
En lansering ska släppas redan på ett tidigt stadium, så kunden redan från början har möjlighet att påverka och styra så att systemet/programmet kommer att uppfylla de krav som finns. Små lanseringar ska kontinuerlig ske igenom hela projektet.

40-timmarsvecka
Extreme programmering strävar efter att arbeta 40 - timmars veckor, men är inte bundna vid det utan behövs det jobbas mer under visa veckor så ska detta göras. Det ska strävs efter att inte arbeta mer än två veckors övertid i en sammanhängande period. För filosofin inom XP är att en utvilad programmerare jobbar bättre och gör mindre fel (Beck & Fowler, 2000) .

Kodstandard
Alla utvecklare i teamet ska följa samma kodstandard, all kod ska mer eller mindre se ut som det är skrivet av en och samma person. Det viktiga är inte vilken standard som följs utan att alla följer samma kodstandard och mönster, vilket gör att all kod alltid går att känna igen. Det skapar bra förutsättningar för kollektivt samarbete mellan olika programmeringspar.

Systemmetafor
Utvecklarna ska konstruera en vision för hur programmet/systemet ska fungera och vad det ska hantera vilket i XP kallas för metafor. Metafor är en bildlig beskrivning av något som är mer eller mindre likt, så utvecklarna ska komma på en bra metafor som beskriver hur systemet är tänkt att fungera. Om det är ett klient server system som ska beskrivas så går det att använda liknelsen: En båt som trafikerar mellan två olika destinationer där de olika hamnarna är olika klienter och själva båten är servern som utför arbetet och klienterna fungerar som ett gränssnitt.

3.3.2 RAD Rapid Application Development

James Martin utvecklade metoden rapid application development på 1980 - talet (http://www.blueink.biz/RapidApplicationDevelopment.aspx). Metoden främjar en iterativ utvecklings process, och i metoden ingår delarna: prototyper, iterativ utvecklingsprocess, tidshantering, grupp medlemmar, projektstyrning och RAD verktyg. RAD brukar också delas in fyra olika livscykler; behovs analys, design analys, konstruktion och implementering. Metoden är inte lämplig att använda i alla systemutvecklings projekt eftersom den vänder sig till projekt som inte är för omfattande och där projektet lätt kan brytas ner till mindre delar. Projektgruppen som ska utveckla systemet/mjukvaran ska bestå av två till åtta medlemmar och gruppens medlemmar ska ha den erfarenheten som krävs, dessutom vara bekant med den teknik som ska användas (http://www.blueink.biz/RapidApplicationDevelopment.aspx).

Projektet ska ha så få beslutsfattare som möjligt för att beslut ska kunna fattas snabbt och inte fastna i någon byråkratisk linda där teamet får sitta och vänta. Som namnet antyder så är meningen med rapid application development att snabbt kunna komma fram med ett resultat. Tanken är också att kunna leverera ett system som har hög kvalitet och uppfyller kundens behov och önskemål, samt minska risken för eventuella buggar som kan uppstå. Detta uppnås genom att kunden hela tiden får ta del av små releaser och har möjlighet att hela tiden testa systemet skarpt.

Prototyper

Redan på ett tidigt stadium så ska en prototyp konstrueras så kunden från början kan komma med önskemål och kritik. Vilket borgar för att produkten i slutänden kommer att motsvara de förväntningar och krav som kunden har på systemet. En prototyp blir något mer konkret för både kund och utvecklare att prata om och det blir lättare att komma med konstruktiv återkoppling.

Iterativ utvecklingsprocess

Genom att släppa nya små releaser inom korta intervall ökar möjligheten för en bättre dialog mellan kund och utvecklare. Det ska vara ett kontinuerligt flöde av releaser till systemet har uppnått den funktionalitet som kunden förväntar sig (http://www.blueink.biz/RapidApplicationDevelopment.aspx). Eftersom det blir en kort livscykel mellan releaserna så kommer kund och utvecklare träffas ofta vilket ska ge upphov till ett bra samarbete som leder till en bra dialog i gruppen.

Tidshantering ”Time boxing”
Extra finesser som ska vara med i systemet ska skjutas på framtiden om de inte hinner bli klara till utsatt datum. Tid och kraft ska läggas på att kunna presentera en så bra version på kortast tid. Strikt tidsplanering är en viktig del i RAD metoden, om det blir för lång tid mellan presentationen av olika versioner så finns det en risk att utvecklingsprocessen glider åt en mer vattenfalls liknande utvecklingsmetod (http://www.blueink.biz/RapidApplicationDevelopment.aspx). Där samarbetet mellan kund och utvecklare inte blir optimalt, för sker utvecklingen utan att kunden har möjlighet att på ett tidigt stadium komma med synpunkter så finns det en risk att det blir väldigt krävande att göra de ändringar som kunden vill.

Grupp medlemmar
Utvecklingsprojekt som använder RAD ska jobba i små grupper som består av erfaren-, mångsidig- och motiverad personal som kan uppträda i många olika roller. I RAD utveckling så spelar kunden än viktig roll och då är det viktigt att kunden finns tillgänglig under hela utvecklings cykeln.

Ledarskapsrollen
För att undvika att utvecklings cykeln blir förlängd så ska en motiverad och engagerad projektledare användas. Projektledaren måste vara både konsekvent och ståndaktig i sina beslut för att kunna använda rapid application development som metod. För att det ska vara möjligt att hålla sig till strikt tidsplanering så måste projektledaren grundligt välja vilka gruppmedlemmar som ska vara delaktiga i projektet. Det gäller för projektledaren att försöka förutse olika byråkratiska och politiska spörsmål som kan uppstå vid val av gruppmedlemmar.

Utvecklings verktyg, RAD - verktyg
RAD förespråkar att använda den senaste tekniken av olika utvecklingsverktyg som är aktuella. För att kunna producera kod på ett effektivare sätt och säkerställa att kvaliteten blir bättre. Sedan James Martin utvecklade denna metod på 80-talet så har det kommit till en mängd olika verktyg att tillgå för systemutveckling vilket gör att RAD inte pekar på något specifikt verktyg. RAD förespråkar att den senaste tekniken ska användas för att underlätta kodningen (http://www.blueink.biz/RapidApplicationDevelopment.aspx).

3.3.3 Scrum

Scrum som utvecklingsmetod antar att mjukvaruutveckling är både komplicerad och outsägbar. Den behandlar mjukvaruutvecklingen som det inte är en fullt utvecklad process till skillnad från andra modeller som vattenfallsmodellen och spiralmodeller som ser mjukvaruutvecklingen som en färdig process som ska följas (Schwaber, 1995). Används vattenfallsmodellen som utvecklings teori så sker utvecklingen efter stegen analys, design, implementering och testning där en ny del inte får påbörjas förrän den föregående är klar. Scrum jobbar inte efter någon sådan modell utan där kan varje aktivitet börjas utan att föregående aktivitet är klar eller så kan en aktivitet ingå i flera olika faser. De olika delarna i Scrum måste inte utföras i någon speciell sekvens utan det sker iterativt. Vilket ska öka flexibiliteten och produktiviteten i utvecklingen. Scrum föreslår ingen speciell programmerings teknik utan fungerar som ett projektstyrningsverktyg. Om utvecklingen sker på gränsen till kaos men ändå under kontroll desto mer konkurrenskraftigt och bättre kommer det slutgiltiga resultatet bli (Schwaber, 1995).

Förberedande stadiet

Fasen förberedande innefattar två under kategorier: planering och arkitektur/ högnivå design. Planerings fasen innefattar att säkerställa vad som ska vara med i systemet, vilka olika krav som kund/köpare har på systemet. En krav lista skapas över alla kända krav som en kund har av systemet. Kraven rankas och listas efter vad som är viktigt och vad som är mindre viktigt. En uppskattning av vilka resurser som krävs för att kunna implementera en lösning av problemet upprättas också (Schwaber, 1995).

Kravlistan uppdateras kontinuerligt under hela utvecklingsprocessen i form av att nya krav kan komma till eller mer detaljerad beskrivning för befintliga önskemål tillkommer. Prioriteringsordningen kan ändras under hela processen. Det är kunden som styr vilka olika prioriteringar som ska göras. Under planerings fasen ingår att utse vilka som ska ingå i projektgruppen och vilka olika verktyg som ska användas under utvecklingen. Även bestämma vilka resurser som krävs för att projektet ska lyckas vilket kan innefatta riskanalys, kontroll verktyg och utbildning av gruppmedlemmar.

Medlemmar som är involverade i ett projekt som följer Scrum teorin har olika roller som fyller olika syften och ska lösa olika uppgifter. Där den ena rollen är Scrummaster som har till uppgift att kontrollera att gruppens medlemmar jobbar efter de riktlinjer som finns listade enligt kravlistan. Där Scrummaster samverkar med både med projektteam, kund och är med under själva projektstyrnings framtagandet. Scrummaster är ansvarig för att lösa de problem och förändringar som kan uppstå och se till att gruppmedlemmarna är produktiva och motiverade (Schwaber, 1995). Rollen produktägare är den som officiellt är ansvarig för hela projektet och utses av Scrummaster och kunden under planeringsfasen. Den som har rollen som ägare av produkten har den slutgiltiga rätten att bestämma vilka olika uppgifter som ska finnas med på kravlistan och vilka resurser som det krävs för att kunna lösa problemet,

alltså vilka resurser som ett problem ska få sig tilldelat. Gruppmedlemmarna har möjlighet att bestämma hur varje sprint (se beskrivning nedan) ska läggas upp och hur problemen i sprinten ska lösas för att kunna uppnå målen för varje sprint. De kan också komma med förslag på vilka olika punkter som ska tas bort från kravlistan som är kopplad till en sprint. Det gäller att hitta vilka olika krav som är orimliga och utesluta dem från listan. Kundens roll är att komma med vilka krav som ska finns med kravlistan och komma med ändringar under utvecklingens roll, de ska även ta del av de ändringar som ändras på produktlistan.

Utvecklings fasen
I Scrum så är utvecklingsfasen den del som kan kategoriseras som lättrörlig, fasen ska behandlas som en ”black box” där det oväntade är att vänta (Abrahamsson, Salo, Ronkainen & Warsta, 2002). Utvecklingen planeras efter olika ”sprints” där varje sprint har definierade riktlinjer för vad som ska vara klart för innan nästa sprint får påbörjas. Varje sprint består av de klassiska utvecklingsstegen analys, design, utveckling och implementering. En sprint ska vara mellan en vecka till två månader (Abrahamsson et al., 2002).

Avslutnings fasen
Det sista stadiet innebär att projektet avslutas, vilket innebär att det inte finns några behov som inte har blivit uppfyllda och inga nya förändringar kan ske. När detta är uppfyllt så kommer implementering och integrering av systemet där även testning och dokumentering av systemet ingår.

Utse projekt grupp
Det första som sker innan utvecklings processen börjar är att skapa ett team som ska fungera som vilket lag som helst där det är viktigt med enskilda prestationer och att alla i laget jobbar mot samma mål. Scrum som utvecklingsmetod förespråkar grupper om 6-7 stycken personer. Varje grupp ska ha en Scrummaster vilken ska fungera som projektledare (Schwaber, 1995).

Projekt segmentering
Hela projektet delas upp i olika perioder där varje period kallas för en sprint och där varje sprint får en kravlista som ska följas och realiseras under varje sprint, en sprint ska max vara fyra veckor lång.

Scrum möte
Under varje sprint så ska det hållas dagliga möten som ska vara på samma plats och samma tid varje dag. Varje möte ska inte överstiga en längd på 30 min. Under det dagliga mötet ska Scrummaster ställa tre frågor till varje gruppmedlem: 1, Vad har du gjort sedan det senaste mötet? 2, Har något hindrat dit arbete? 3, Vad planerar du att göra mellan dagens möte och nästa möte? Det är Scrummaster som är ansvarig för att det fattas beslut omgående om det behövs. Det är meningen att de dagliga mötena ska fastställa varje gruppmedlems status (Beedle, Devos, Sharon, Schwaber & Sutherland, 2000).

Sprint
En sprint är en del sträcka i projektet där varje sprint inte ska vara längre än fyra veckor. Under sprinten så sker arbetet med att utveckla systemet/produkten. Utvecklingen sker inte i någon sekvens utan efter ett iterativt arbetssätt. En sprint kan delas in följande underaktiviteter: utveckling, förpackning, granskning och justering. Vilka aktiviteter som ingår beror helt på vad som finns beskrivet i backlogen för sprinten. Så för en aktivitet kanske det bara är nödvändigt att granska och tjustera, medan andra aktiviteter kräver att alla underaktiviteter är inblandade. Varje sprint följs upp av en ”sprint review” där den föregående sprinten granskas och där nya punkter kan tillkomma till den ”backlog”, vilken fungerar som en kravlista, som finns för nästa sprint. Under sprintens granskning så kan utvecklarna, beslutsfattare, kunder, säljare och marknadsförare vara närvarande för att komma med synpunkter.

3.4 Ramverk för implementering av en lättrörlig metod
Det finns ett flertal software process improvement modeller som beskriver ett cykliskt flöde för systemutvecklingsprocesser. Syftet med modellerna är att ge underlag för att optimera systemutvecklingsprocessen genom att göra utvecklingen effektivare och öka möjligheterna till att producera mjukvara med färre fel. Enligt Kinnula (2001) är modellerna relativt generella på grund av att de beskriver ett övergripande sätt för hur en systemutvecklingsprocess ska bedrivas. Det ges därför stort utrymme för att göra en specialanpassning så att de går att applicera på ett specifikt problem som återfinns i en specifik typ av organisation.

3.4.1 Quality improvement paradigm
QIP skapades genom ett samarbete mellan NASA flight division och universitetet i Maryland. Arbetet presenterades 1976 och syftet var att visa på ett sätt att arbeta genom att använda sig av QIP som ett verktyg för att utveckla nya sätt att arbeta på. QIP fungerar som en guide för hur man ska gå tillväga för att nå en optimal utveckling i organisationer som står för förändring och modellen skapar ett underlag för att i en experimentell miljö skapa eller förändra en arbetsmetod, som senare kan implementeras i andra kontext inom eller utanför organisationen (Kinnula, 2001). QIP cykeln består i huvudsak av två stycken cykler en huvudcykel och en delcykel där utvecklingen av metoden sker. Huvudcykeln även kallad organisationscykeln bidrar med kunskap i ett organisatoriskt perspektiv medan delcykeln även kallad

projektcykeln bidrar med iterativ projektkunskap. Enligt Kinnula (2001) kan QIP cykeln användas som en modell för att både erhålla mer kunskap utav en redan existerande systemutvecklingsmetod eller som en modell för att skapa en helt ny utvecklingsmetod. Om det senare är syftet med projektet så behövs fler iterationer i projektcykeln för att i varje iteration erhålla kunskap om vad som fungerar och vad som inte gör det. Det ger upphov till vad som krävs för att det skall fungera vid ingången när nästa iteration påbörjas (Kinnula, 2001).

3.4.2 Vidareutveckling av QIP

Som tidigare nämnts är QIP cykeln generellt utformad, för att göra det möjligt att använda den i samband med implementering av en lättrörlig metod i en organisation argumenterar Pikkarainen, Salo och Still (2005) för en vidareutveckling av QIP. Där förfinar man stegen i cykeln för att anpassa modellen till att hantera utveckling och bearbetning av lättrörliga metoder. Pikkarainen et al. (2005) menar att organisationer behöver ha lättrörliga riktlinjer för att hjälpa dem i valet av lättrörlig metod samt vid anpassning och implementering av vald lättrörlig systemutvecklingsmetod. I det nya ramverket för implementering av lättrörliga metoder har man skalat ner antalet steg i cykeln från sex till fyra.

3.4.3 Selektera lättrörlig metod

Innan ett projekt ska starta bör organisationen ha genomfört en analys om vilka mål med projektet som man har. Däri ingår vilka resurser man har till förfogande såsom lokaler, personal och utrustning. Då detta är fastställt görs ett val av lättrörlig metod som passar typen av organisationsstruktur. Enligt Pikkarainen et al. (2005) är de sätt som existerar för att välja en lättrörlig metod ostrukturerade. En lättrörlig metod kan väljas genom att studera litteratur inom området. Genom egna studier kan organisationen själv bilda sig en uppfattning om gällande lättrörliga metoder och vilka som kan vara aktuella. Ett annat sätt är att undersöka vad liknande organisationer använt för typ av metod och om de fungerat tillfredställande kan slutsats dras om att det också kan implementeras i den egna organisationen (Pikkarainen et al., 2005).

Enligt Pikkarainen och Passoja (2005) kan enbart ett fåtal organisationer erhålla goda resultat med att enbart ta en redan specificerad lättrörlig metod såsom Scrum eller RAD. Det är då bättre att först undersöka på vilket sätt man själv i organisationen arbetar efter i ett systemutvecklingsförfarande. För på så sätt erhålla mer kunskap om vilken uppsättning av lättrörliga arbetssätt som skulle ligga till grund för att tillföra en bättre systemutvecklingsmodell än tidigare använda modeller (Pikkarainen & Passoja, 2005). En diskussion förs om vilka metoder som kan vara aktuella i organisationen och vilka möjligheter de skulle ge organisationen. Syftet med den initiala utvärderingen av lättrörliga metoder är att utröna hur systemutvecklingsprojektet kan förbättras med hjälp införandet av en lättrörlig metod och vilken metod som skulle passa organisationen och det aktuella projektet bäst. Enligt Pikkarainen och Passoja (2005) är tanken med valet av en lättrörlig metod att det inte skall ske genom ett komplext förfarande såsom utvärdering med hjälp av

CMMI[2]. Grundtanken är att bedömningen av vilken metod som kan vara lämplig också ska ske enligt lättrörliga principer såsom att bedöma vilken metod som är mest lämpad genom ”face to face” kommunikation, snabb återkoppling och tydliga enkla dokument (Pikkarainen & Passoja, 2005). Det finns också enligt Pikkarainen och Passoja (2005) anledning till att genomföra litteraturstudier och analysera vilka lättrörliga metoder som kan anses vara lämpliga då det saknas erfarenhet hur väl implementering av en lättrörlig metod varit.

3.4.4 Planera implementering

För att genomföra en övergripande förändring i en organisation såsom införande av ett nytt sätt att arbeta på eller ett nytt system så agerar organisationer på olika sätt. Oavsett vilken typ av organisation man är, och på vilket sätt man inför nya sätt att arbeta på är det vikigt att i initialskedet bestämma på vilket sätt återkoppling skall ske under systemutvecklingsprojektets gång (Pikkarainen et al., 2005). Anledningen är att erhålla en konkret och viktig information som i ligger till grund för hur man kan förbättra och skräddarsy den lättrörliga metoden som från början valts som systemutvecklingsmetod för projektet (Pikkarainen et al., 2005).
Under denna fas bör också förberedelser göras innan projektstart. Det inbegriper ett flertal punkter såsom val av lokal, inredning för att passa lättrörliga värderingar. Här sker även installation och förberedelser av de utvecklings och testverktyg som är aktuella under projektets gång. De förberedelser som vidtas i denna fas är högst baserat på vilken typ av situationsanpassad metod man valt eftersom olika metoder ställer krav på olika typer av förberedelser där XP moment är en av de metoder som kräver större förberedelser med fokus på en öppen arbetsmiljö utan dörrar mellan projektmedlemmarna.

3.4.5 Implementering av lättrörlig metod

I den här fasen skapas förutsättningar för att utforma och förbättra den lättrörliga metod som organisationen valt. För att optimera processen med att skräddarsy den lättrörliga metoden sker utvecklingen av metoden i tre steg som följer projektets iterationer. Befintlig lättrörlig metod används i den första iterationen. Mellan iterationerna genomför man vad Pikkarainen et al. (2005) definierar som PIW[3]. En iterationscykel baseras på tre steg: 1. Genomför implementering , 2. Iterativt förbättra metoden, paketera och analysera återkoppling och 3. Ge återkoppling till organisationen.

Enligt Pikkarainen et al. (2005) så bidrar PIW med olika aspekter beroende på om man befinner sig på projektnivå eller på ett övre organisatoriskt plan. Syftet med PIW är att i projektet bidra med verktyg för projektmedlemmarna att kunna genomföra förändringar på den lättrörliga metoden under projektets livslängd och i med detta möjliggöra förändringar på ett kontrollerat och korrekt sätt. PIW kan också användas

[2] Capability Maturity Model Integration. En komplex metod för att ge organisationer möjlighet till att effektivisera sina processer(http://www.sei.cmu.edu/cmmi)
[3] PIW Post Iteration Workshop.

för att bidra med återkoppling till organisationen om hur projektet fortlöper i form av de förbättringar av den lättrörliga metoden man gjort (Pikkarainen et al., 2005). Det som ligger till grund för de data som samlas in baseras på projektmedlemmarnas egna erfarenheter under systemutvecklingen med den lättrörliga metoden. För att genomföra en systematisk analys i varje PIW presenterar Salo (2005) ett tillvägagångssätt för att strukturera de positiva och negativa erfarenheter från de moment som ingår i den lättrörliga metoden under systemutvecklingsfasen steg 1.

Händelseutvecklingslistan specificerar de moment som ingår i den lättrörliga systemutvecklingsmetoden och genom att anteckna erfarenheter som erhållits under iterationen kan slutsatser dras om hur uppkomna problem ska lösas. Under varje moment ingår fem steg som ska fyllas i (Salo, 2005).

Tabell 1 Händelseutvecklingslista (Salo, 2005)

Förändrings moment:	
1. Problem identifiering	Producera detaljerad information om problemet som ska lösas.
2. Åtgärd	Klargör vad som ska åtgärdas till nästa iteration.
3. Berörd	Vilka inom projektet som är berörda.
4. Plan för resultat	Plan för genomförande av förändring.
5. Resultat	Kvalitativ data från systemutvecklarna.

I punkt 1 antecknas det problem som uppkommit kring det moment där problemet uppstått under den iteration som varit. Det är projektmedlemmarnas egna uppfattningar om problemområden som nedtecknas (Salo, 2005). Under PIW genomförs sedan ett möte som där de åtgärder diskuteras om hur man skall lösa problemet, som förs in under punkt 2. Åtgärden för att lösa problemet implementeras under nästa iteration. Under punkt 3 anges vem som är ansvarig för att lösa problemet. Syftet är att säkerställa att åtgärden vidtas under nästkommande iteration. Under punkt 4 anges den plan för hur resultatet ska valideras vilket innebär att åtgärderna för att lösa uppkomna problem vidtas först under nästkommande iterationsperiod. Det medför i sin tur att valideringen av resultatet kommer efter att åtgärder har vidtagits. I den sista punkten anges resultatet av den åtgärd som vidtagits och som validerats under punkt 4 . Därefter uppdateras nästa iteration med aktuell förändring. Efter varje iteration ges möjligheter för återkoppling till organisationen med förändringslistan som underlag.

3.4.6 Analysera, förbättra och paketera

När projektet är slutfört och inga iterationer återstår ligger fokus på att samla och analysera de erfarenheter som erhållits under projektets gång. Syftet är att tillföra organisationen kunskap om hur och vilka anpassningar som gjorts (Pikkarainen et al., 2005). De anpassningar som gjorts och de erfarenheter som dragits samlas ihop och sparas tills nästa systemutvecklingsprojekt. I denna fas skapas också möjligheter att med insamlade erfarenheter som grund att använda sig av olika moment separat ute i organisationen. Vidare görs en bedömning om den lättrörliga systemutvecklingsmetoden är lämplig och passande för organisationen att använda vid senare systemutvecklingsprojekt (Pikkarainen et al., 2005).

4. Fallstudie

4.1 Asivo som organisation

Företaget Asivo Solutions AB är en mindre organisation som bildades 2003. Företaget har fyra stycken anställda som specifikt arbetar med verksamhetsstyrning gentemot andra företag och organisationer. Företaget bedriver sin verksamhet genom att sälja och specialanpassa affärssystemet Ataio till kunder. Ataio fungerar som en utvecklingsplattform som erbjuder en standardlösning men anpassas efter kundens behov. Förutom att sälja produkten Ataio så bedriver företaget också konsultverksamhet där man säljer konsult timmar. Den utveckling som bedrivs i företaget är när olika kundanpassningar görs antingen i Ataio eller som en fristående applikation som kopplas ihop med affärssystemet.

Företaget Asivo kan kategoriseras som en ad hoc-krati, vilket känns igen på att organisationen inte i någon större utsträckning arbetar med strukturerad dokumentation. Den kommunikation som sker inom företaget sker både på ett vertikalt och horisontellt plan, det vill säga att det finns inga fördefinierade kommunikationsvägar(Jacobsen & Thorsvik, 2002). Mycket av kommunikationen sker genom ”face to face” möten och genom spontanitet. Asivo arbetar med olika kundanpassningar och konsultverksamhet vilket leder till att arbetsuppgifterna ständigt omdefinieras vilket ställer höga krav på flexibilitet gentemot kunden. Personalresurserna på företaget har kompetens inom området IT och verksamhetsstyrning. Mintzberg (1981) skiljer på två typer av ad hoc-organisationer, operativ och den administrativa ad hoc-organisationen där Asivo kan kategoriseras i den operativa ad hoc-kratin. Det som är signifikant för den operativ ad hoc-organisation är att den primära uppgiften ligger i att lösa problem åt andra organisationer.

4.2 Ramverk för implementering av lättrörlig metod

4.2.1 Selektera lättrörlig metod

Under projektet valde vi att använda extrem programmering som utvecklingsmetod. Vi kom fram till att använda XP genom att bedriva litteraturstudier och läsa in oss på olika lättrörliga metoder. Vi genomförde även intervjuer med utvecklingsansvarig på Asivo för att redogöra på vilket sätt de bedriver sitt arbete i dagens läge och vilka olika metoder de har erfarenhet av. Genom intervjun som vi gjorde med utvecklingsansvarig i organisationen så kom det fram relativt fort att de inte arbetar utefter någon utvecklingsmetod i den bemärkelsen att de benämner eller klassificerar den enligt gällande teorier. När vi frågade utvecklingsansvarig vilken/vilka utvecklingsmetoder de arbetar efter så blev svaret:

> *”Ska jag vara ärlig så har jag väl inte jobbat särskilt mycket med utvecklingsmetoder överhuvudtaget sen hur det låter... men på något sätt så är man ju metodisk”*

Så genom intervjun försökte vi att utröna vilken utvecklingsmetod som skulle passa Asivo bäst och som vi kunde arbeta med och ha som utgångspunkt under systemutvecklingsprojektet. När vi frågade utvecklingsansvarig om han hört talas om lättrörliga utvecklingsmetoder så blev svaret:

> *”Ja det har jag gjort och då framför allt det som kom då var ju det här med XP ”*

Genom litteraturstudier så började vi göra jämförelser mellan tre olika lättrörliga utvecklingsmetoder som kunde vara aktuella att använda som utvecklingsmetod under projektet, de tre olika metoderna är XP, RAD och Scrum. Efter att vi hade gjort intervjun med utvecklingsansvarig så kom vi fram till att Scrum som utvecklingsmetod inte skulle passa Asivo speciellt bra. Bland annat så förespråkar Scrum grupper om 6-7st personer. Vilket inte passar Asivo i dagsläget då företaget inte arbetar i så stora projekt. Scrum som utvecklingsmetod arbetar efter olika former av ”backlog” som representerar olika former av dokumentation, vilket Asivo arbetar begränsat med internt, som också framkom under intervjun med utvecklingsansvarig:

> *”Men när det gäller våran interna kommunikation eller då metod så eller vad vi nu, så vill man ju svar direkt. Då sätter ju inte jag mig och skriver massor av papper som vi går igenom utan då frågar man ju istället.”*

Det som skulle kunna passa Asivo med Scrum är att utvecklingen delas upp i olika ”sprintar” där det finns en prioriteringsordning vad som ska göras först och vad som kan vänta. När Asivo bedriver sina utvecklingsprojekt så sköts det ofta av en person utan någon uttalad utvecklingsmetod, men under intervjun går det att tolka att företaget jobbar med små releaser och testning mot kund kontinuerligt. Genom att deras affärssystem, som de saluför, ger dem möjligheten att redan dag ett komma med en tidig version och därefter jobba med kontinuerlig utveckling. Så fort en ny del är klar, ber man kunden testa och utvärdera om det var det här man ville. Vilket går att koppla mot de två begreppen ”Små releaser och testning” i XP. Även om Asivo själva inte benämner deras utveckling som XP-relaterad så går det ändå att identifiera de här stegen. Detta leder till kontinuerlig kontakt med kund som också ingår som en punkt i XP utveckling. En annan viktig del i XP är ”face to face” -mötena som ska ske på daglig basis, vilket Asivo inte arbetar med i den bemärkelsen att de identifierat och namngett sina möten till ”face to face”. Det finns inte kopplingar till varje punkt som finns i XP till Asivos sätt att utveckla på, men det går att hitta fragment som stämmer in med XP som utvecklingsmodell.

Det går vidare att hitta fragment som stämmer in på att använda RAD som utvecklingsmetod också. En av punkterna som stämmer in med RAD som utvecklingsmetod är den iterativa utvecklingsprocessen som ingår metoden. Det innebär att arbeta med prototyper gentemot kund och att släppa många små releaser samt föra en bra dialog med kunden precis som i XP. En del i RAD som utvecklingsmetod är att arbeta med tidshantering, där extra funktionalitet ska skjutas på framtiden. Det kan vara svårt för Asivo att göra detta gentemot kund som ofta

kräver att det ska fungera direkt. Det som också talar för RAD som utvecklingsmetod är att Asivo använder professionella utvecklingsverktyg, vilket framgår genom intervjun när vi frågar om de arbetar med olika utvecklingsverktyg.

> ”Det är Visual Studio uteslutande skulle jag vilja påstå och Notepad”

I RAD som utvecklingsmetod förespråkas det att en uttalad projektledare används under utvecklingen. Det skulle inte passa Asivo då de ofta arbetar i såpass små grupper att det inte behövs och de har heller inte arbetat så innan. Utan när deras projekt bedrivs så sker de ofta enskilt men ibland om det är lite större projekt så kan det innebära att de är två till fyra stycken inblandade i samma projekt.

Vi har tittat på vad som kan passa organisationen bra och vilka mål som finns samt den erfarenhet som projektmedlemmarna har av olika utvecklingsmetoder. I det här fallet så är det vi som författare som är projektmedlemmar och ska genomföra själva utvecklingen. Vi konstaterar att vår erfarenhet av systemutveckling består av olika projekt som vi har bedrivit under utbildningen och i många av de utvecklingsprojekten har vi jobbat i grupper om två. Det innebär att vi känner oss hemma att arbeta med parprogrammering som används i XP. Vi har även under utbildningen fått lära oss att använda olika designmönster och att göra koden så generell som möjligt för att kunna återanvända den vilket också är en stor del i XP. Det är flera moment i XP som vi inte har någon erfarenhet av, till exempel att arbeta med små lanseringar och kontinuerlig integrering, där vi presenterat hela system för kursansvarig i slutet på kursen snarare än kontinuerlig presentation.

Utvecklingsmodellen RAD är den metod som passar in minst baserat på vår erfarenhet. I RAD så arbetar man med att visa upp prototyper vilket kan likna små lanseringar vilket vi inte arbetat med. RAD förespråkar vidare att professionella utvecklingsverktyg ska användas vilket vi nästan har obefintlig erfarenhet av under utbildningen där vi hela tiden har kodat ifrån grunden utan hjälp av smarta funktioner i utvecklingsverktyg. Vi känner oss inte bekanta med att arbeta efter ”tidshantering” där avancerad funktionalitet läggs på framtiden utan vi har arbetat tvärt emot där de avancerade funktionerna lösts först för att sedan implementera den lätta funktionaliteten. Det moment som passar oss i RAD är att modellen förespråkar små projektgrupper där projektmedlemmarna ska kunna uppträda i många olika roller. Vilket vi har fått göra många gånger under olika utvecklingsprojekt där vi har haft rollen både som kravställare, producerat designen och sedan som programmerare.

Den sista utvecklingsmetoden som vi har tittat närmare på är Scrum. I den går det att hitta moment som vi kan basera på egna erfarenheter från tidigare utvecklingsprojekt. Momentet ”bildning av grupp” som är det första som sker innan utvecklingsprocessen börjar, känner vi igen från skolans värld, där det gäller att bilda en projektgrupp som ska jobba mot samma mål och kunna bidra med olika prestationer. Den här delen är ofta något som vi studenter inte har behövt befatta oss med utan det har kursansvarig ansvarat för, men momentet känns väldigt bekant. Ett

annat moment som känns igen från Scrum som utvecklingsmetod är ”product backlog” som fungerar som kravlista för vad som ska göras och uppnås under projektet. Under utbildningen har vi kallat det för laborationsbeskrivning. Momentet ”release backlog” vilket ska prioritera vilka olika moment som är viktiga och mindre viktiga är inget som vi är bekanta med utan för oss har alla punkter på en laborationsbeskrivning varit lika viktiga. Vi har vidare inte jobbat med att dela in utvecklingsprojekten i olika ”sprintar” där varje sprint är en del sträcka i det hela projektet. Genom att läsa in oss på de olika utvecklingsmetoderna och försöka att hitta några likheter med våra egna erfarenheter av utvecklingsprojekt så faller valet på XP som utvecklingsmetod vilket vi tycker passar bäst in även när man tittar på våra egna erfarenheter.

4.2.2 Planera implementering

Vi blev tilldelade varsitt kontor för att bedriva den analytiska delen av projektet. Därför innebar valet av lättrörlig systemutvecklingsmetod att vi behövde förbereda och iordningställa arbetsmiljön innan vi kunde sätta igång med utvecklingsarbetet. Det första som företogs var att flytta ihop i ett rum. Ett skrivbord användes med plats för två bärbara datorer. Den ena arbetsstationen fungerade som utvecklingsdator och den andra arbetsstationens syfte var att alltid vara tillgänglig för informationssökning och var också den som inte programmerade i parprogrammeringens arbetsstation för att snabbt kunna söka efter information på Internet om de problem som uppstod under systemutvecklingen. Installation av relevant programvara gjordes också på de två arbetsstationerna vilket tog lång tid i anspråk då omfattande installationer måste göras på båda arbetsstationerna. Det utvecklingsverktyg som nästan uteslutande används inom företaget är Visual Studio och verktyg som tillhör SQL servern 2005-sviten. De installerades för att erhålla en optimal utvecklingsmiljö i linje med företagets egen utveckling. Även webbservern IIS 7 installeras lokalt på arbetsstationerna för att ge möjlighet att testa av applikationen på den webbserver som var aktuell för applikationen i ett skarpt läge. Då utvecklingen skulle ske i ASP.net och Ajax hade vi en lägre kompetensnivå än vad som vad önskvärt i projektet. I denna fas fick en del tid avsättas för instudering på ovanstående tekniker för att främst få klart för sig vilka möjligheter som fanns med tekniken innan utvecklingen tog fart. Kontoret hade en större glasruta som vette ut mot korridoren vilket gav upphov till en naturlig mötesplats, där också glasrutan kunde användas som ”whiteboard”. I detta skede skapades också en projektplan i samråd med utvecklingsansvarig som klistrades upp i möteshörnan för att på ett tydligt sätt visa iterationscyklerna och projektets längd. I möteshörnan sattes också den initiala kravspecifikationen upp i punktbaserad form vilket gav generella riktlinjer och stöd för ”face to face” -kommunikationen. Som en typ av mötesprotokoll från ”face to face”-kommunikationen fastställdes det att post-it lappar skulle användas med en dag per lapp som också klistrades upp i möteshörnan så att ett dag för dag flöde om vad som åstadkommits och vad som skall göras lätt kunde urskiljas.

Återkoppling från iterationscyklerna skedde genom att vi under tiden som utveckling bedrevs bidrog med kvalitativa data i form av dagboksanteckningar som bygger på de erfarenheter som dragits under iterationen och som i varje PIW antecknades ned i händelseutvecklingslistan och analyseras (Salo, 2005). Eftersom iterationerna bestämts till tvåveckors intervaller sker detta varannan vecka och underlaget bidrar till beslut om förändring av utvecklingsmetoden av projektmedlemmarna innan nästa iteration sätter igång.

4.2.3 Implementering av lättrörlig metod

Utvecklingsprojektet delades in i fyra olika iterationer där varje iteration bestod av två veckor, vilket innebar att utvecklingen bedrevs under sammanlagt 8 veckor. Under varje iteration så avslutades veckan med ett möte med uppdragsgivaren där vi presenterade och visade upp vad vi åstadkommit från föregående möte. Uppdragsgivaren kommer med synpunkter och förbättringar som är önskvärda eller om det har tillkommit några nya önskemål. Efter varje iteration är slut så genomförs en PIW där vi går igenom förändringsmomentlistan och anpassar eller lägger till nya steg i XP-modellen och gör en plan för hur de ska genomföras.

4.2.3.1 Första iterationen

Efter att ha genomfört den första utvecklingsiterationen så kunde vi identifiera tre olika moment som vi inte var helt nöjda med. Det första var parprogrammeringen som vi valde att plocka bort till nästa iteration och detsamma gäller momentet kund på plats som XP förespråkar. Vi valde bort det här momentet och bestämde istället att prova med om kunden kunde vara tillänglig via Skype[4] där det finns möjlighet att kommunicera via både ljud och bild. Det fanns också en möjlighet att använda delat skrivbord på datorn vilket gjorde det enkelt att prata kring en gemensam bild eller applikation. Vi valde att arbeta utefter den här nya metoden från och med nästkommande iterationscykel. Det sista momentet som vi har med som förändringsmoment är expertkunskap som inte är specificerad som en separat punkt i XP men det beskrivs som att det är av stor vikt att medlemmarna i utvecklingsprojektet är experter inom området (Boehm, 2002; Koch, 2004). Vi valde att behandla även detta som ett moment. Vilket ledde till att vi måste läsa in oss bättre på de programmeringsspråk och tekniker som används inom ramen för projektet. Instuderingen sker parallellt med utvecklingen under nästa utvecklingsfas.

[4] Skype är ett företag som erbjuder en tjänst som gör det möjligt att ringa via Internet.

Tabell 2 Händelseutveckling Parprogrammering iteration 1

Förändrings moment: Parprogrammering, Första iterationen	
1. Problemidentifiering	Ineffektivt, passivitet, informationsökning. Resursslöseri med humant och ekonomiskt kapital.
2. Åtgärd	Använda par programmering endast när det är lite svårare uppgifter som ska lösas, när det kan vara av stor vikt att ha någon att resonera med. Parprogrammering när det behövs pseudokod och när genomgång sker vid ”whiteboard”.
3. Berörd	Alla projekt medlemmar.
4. Plan för resultat	Kontroll hur utfallet har blivit kommer att ske vid nästa iterationsmöte.
5. Resultat	Resultatet från mötet blev att det fungerar bra att använda parprogrammering vid behov och att använda parprogrammering vid genomgång av pseudokod. Vilket har lett till att vi har blivit mer effektiva i kodandet. Det vill säga att utvecklingen går snabbare.

Tabell 3 Händelseutveckling Kund på plats iteration 1

Förändrings moment: Kund på plats, Första iterationen	
1. Problemidentifiering	Ineffektivt att ha kund på plats i alla lägen. Tillför inget till programmeringen.
2. Åtgärd	Prova att föra en bra dialog med kunden genom Skype - telefoni och använda olika former av remote -kontroll[5] av skrivbords miljö för att kunna visa och föra en dialog kring applikationen.
3. Berörd	Projektmedlemmar och kunden.
4. Plan för resultat	Kontroll hur utfallet har blivit kommer att ske i nästa iterationsmöte.
5. Resultat	Möte med kund har skett genom Skype med möjlighet att använda samma skrivbord. Vilket har lett till att dialogen går att föra på ett tillfredställande sätt utan ”face to face” med kunden.

[5] Remotekontroll gör det möjligt att dela databild och även att kunna styra en annan dator på distans.

Tabell 4 Händelseutveckling Expertkunskap iteration 1

Förändrings moment: Expertkunskap, Första iterationen	
1. Problemidentifiering	Expertkunskap saknas hos projektmedlemmarna.
2. Åtgärd	Projektmedlemmarna måste genomföra extra inläsning på området AJAX, C# och ASP.NET.
3. Berörd	Projektmedlemmar
4. Plan för resultat	Kontroll sker på nästa iterationsmöte.
5. Resultat	Tiden har inte räckt till att bli expert så den här punkten fortsätter en iteration till.

4.2.3.2 Andra iterationen

Under andra iterationen så behandlar vi tre olika moment där det första momentet är återvinning av kod vilket ska leda till att det går att återanvända koden. Vi väljer att inte använda återvinning av kod på befintliga bibliotek utan kommer endast att tillämpa det på egen producerad kod. Eftersom vi fortfarande inte har hunnit bli experter ännu så fortsätter inläsningen på de olika programmeringsspråken även under nästa iteration. Men här väljer vi även att implementera ett moment från RAD - tidshantering. Vilket innebär att svåra och extra funktioner läggs på framtiden och all kraft ska läggas på implementera det som är absolut nödvändigast.

Tabell 5 Händelseutveckling Återvinning av kod iteration 2

Förändrings moment: Återvinning av kod, Andra iterationen	
1. Problemidentifiering	Färdigt Bibliotek, tidsmässiga problem.
2. Åtgärd	Använder inte återvinning av kod på färdiga bibliotek, utan enbart på nyproducerad kod.
3. Berörd	Projektmedlemmar
4. Plan för resultat	Kontroll sker kontinuerligt att projektmedlemmarna följer det.
5. Resultat	Att bara använda återvinning av kod på nyproducerad kod och inte på befintliga lösningar.

Tabell 6 Händelseutveckling Tidshantering iteration 2

Förändrings moment: RAD Tidshantering, Andra iterationen	
1. Problemidentifiering	Extra funktionalitet tar lång tid att implementera.
2. Åtgärd	Extra funktioner läggs på framtiden, i enlighet med tids-hantering i RAD (Timeboxing).
3. Berörd	Projektmedlemmar
4. Plan för resultat	Den här punkten läggs till i nästa iteration.
5. Resultat	Mer tid läggs på funktionalitet som ska användas.

Tabell 7 Händelseutveckling Expertkunskap iteration 2

Förändrings moment: Expertkunskap, Andra iterationen	
1. Problemidentifiering	Expertkunskapen saknas hos projektmedlemmarna.
2. Åtgärd	Projektmedlemmarna fortsätter och studera programmeringsspråken.
3. Berörd	Projektmedlemmarna
4. Plan för resultat	Kontroll sker på nästa iterationsmöte.
5. Resultat	Det är fortsatt inläsning som gäller för projektmedlemmarna.

4.2.3.3 Tredje iterationen

Under den tredje iterationen så uppkom inga nya moment som vi ville förändra eller lägga till utan inläsning av de olika programmeringsspåren fortsätter som vanligt under nästa iteration.

Tabell 8 Händelseutveckling Expertkunskap iteration 3

Förändrings moment: Expertkunskap, Tredje iterationen	
1. Problemidentifiering	Expertkunskapen saknas hos projektmedlemmarna.
2. Åtgärd	Projektmedlemmarna fortsätter och studera programmeringsspråken.
3. Berörd	Projektmedlemmarna
4. Plan för resultat	Inläsning till nästa iteration.
5. Resultat	Projektmedlemmarna är fortfarande inte experter på området.

4.2.3.4 Fjärde iterationen

I den sista iterationen kan vi konstatera att vi fortfarande inte har de kunskaper som krävs i programmeringsspråket för att kalla oss experter.

Tabell 9 Händelseutveckling Expertkunskap iteration 4

Förändrings moment: Expertkunskap, Fjärde iterationen	
1. Problemidentifiering	Expertkunskapen saknas hos projektmedlemmarna.
2. Åtgärd	Projektmedlemmarna fortsätter och studera programmeringsspråken.
3. Berörd	Projektmedlemmarna
4. Plan för resultat	Finns ingen plan för införandet. Men problemet tas upp i analys stadiet efter sista iterationen
5. Resultat	Konstaterar att vi inte har tillräckligt med kunskap för att kalla oss experter.

4.2.4 Analysera förbättra och paketera

Efter den fjärde iterationen genomfördes ett sista möte där projektmedlemmarna förde en diskussion om de erfarenheter som erhållits under de fyra iterationerna. Mötet varade ca en halvdag vilket är den tidsrymd som förespråkas av Dingsöyr, Moe

och Nytrö (2001) i projekt med mindre än tio projektmedlemmar, då möten ur ett lättrörligt perspektiv inte ska vara formella och ta lång tid. Under mötet gick varje iterations förändringsplan igenom och diskuterades. Även dagboksanteckningar från respektive period analyserades om det skulle komma fram något mer i respektive iterationsomgång. Det som kom fram under mötet var flertalet definitionsfrågor där delar av XP-metodens delmoment kunde identifieras och samlas under ett och samma namn. Ett avslutande möte företogs där vi presenterade vår anpassade metod för företagsledningen. Alla delmoment gicks igenom och vi gav återkoppling på de moment där vi hade gjort förändringar under iterationerna och under vårat projektmöte. Utvecklingsansvarig hade synpunkter på flera av momenten och identifierade på vilket sätt de arbetade idag och hur mycket av metoden som de redan idag kunde använda i deras utvecklingsprojekt. I flera av delmomenten i systemutvecklingsmetoden argumenterade utvecklingsansvarig för att etablera ett alternativt arbetssätt sida vid sida med det sätt som vi presenterat. Utvecklingsansvarig hade också positiva synpunkter och tillägg till flera av momenten som gjorde att vår bild, av hur arbetet med momentet ska bedrivas, förstärktes.

4.3 Resultat

Nedanstående delmoment är det som utgör den lättrörliga systemutvecklingsmetod som vi under projektets gång kommit fram till vara den bästa för den här typen av organisationer och projekt. De tre första delmomenten designmönster, kund på plats och tidshantering har vi förändrat under iterationerna under projektets gång. De övriga delmoment som inte anpassats under iterationerna har använts som de är beskrivna under XP.

4.3.1 Designmönster

Vi väljer att lägga till ett samlingsnamn för momenten: enkeldesign, återvinning av kod, gemensamt ägande av kod och kodstandarder. För under hela utvecklings projektet så arbetade vi efter att följa dessa moment. Men eftersom vi bara var två stycken under projektet så var det naturligt med gemensamt ägande av kod genom att båda följer samma kodstandard. Vi valde att under iteration två modifiera momentet återvinning av kod att bara innefatta ny producerad kod. Tanken är att koden ska bli så generell som möjligt vilket ska göra det möjligt att återanvända koden. Vi väljer att lägga momentet återvinning av kod under samlingsnamnet designmönster också för att de olika momenten tenderar att flyta samman till viss del. Då vi på ett tidigt stadium fick klart för oss att organisationen Asivo inte i någon större utsträckning arbetar efter olika designmönster känns det naturligt att lägga in alla under ett och samma namn. Detta för att samlingsnamnet ska räcka som association för alla moment som ligger väldigt nära varandra och det räcker inom organisationen Asvio där de olika momenten inte är starkt prioriterade.

Utvecklingsansvarig på Asivo kunde identifiera nyttan och vinsten med att arbeta med designmönster och menade att verksamheten som idag på utvecklingssidan består av två utvecklare inte hade de behoven för designmönster då man bara var ett

fåtal utvecklare och att man med lätthet kunde följa med i varandras kod utan bekymmer.

> ”Vi är ju bara två utvecklare här och jag kollar på X och känner igen mig i hur han kodar.”

Enligt utvecklingsansvarig bedriver man på så sätt ett arbete med designmönster med gemensamma kodstandarder utan att i förväg kommit överens om hur det skall bedrivas. Då organisationen befinner sig i ett expansivt skede ser han större möjligheter med designmönster framöver när ett par utvecklare till anställts. Då skulle det vara aktuellt för att kunna använda redan utvecklad kod i andra projekt av samma karaktär.

> ”Vi kommer att bli fler det innebär att det ställer det mer krav att jobba på det sättet.”

Utvecklingsansvarig kunde se fördelarna att arbeta med designmönster då man i nuläget arbetar på egen hand i vissa projekt och att man idag har erfarenheter av återanvändning av kod i den webbaserade produkt som man säljer. Där den utveckling som görs av produkten för att passa en viss kund också kommer nästa kund till gagn då den tidigare utvecklade funktionen nu finns med som standardkomponent i produkten.

4.3.2 Kund på plats

Då det inte varit praktisk möjligt för uppdragsgivaren att deltaga i samma rum där projektets medlemmar bedrivit utveckling så har uppdragsgivaren befunnit sig på plats men i annat rum i samma lokaler. Uppdragsgivaren har under tiden som projektet bedrivits också varit inblandad i ytterliggare två andra projekt vilket har medfört att uppdragsgivaren under delar av tiden inte befunnit sig på plats. Då fann vi att användning av voice-klienten Skype var ett mycket användbart verktyg i kontakt med uppdragsgivaren. Det var ett normalt sätt att arbeta på då uppdragsgivaren i sina affärskontakter använt sig mycket av Skype i det dagliga arbetet. Då en kontakt etablerades via Skype fanns uppdragsgivaren tillgänglig hela tiden fast på ett virtuellt plan vilket medförde de fördelar som är syftet med det ursprungliga momentet i XP. Det gavs även möjligheter för oss att ansluta till dennes fjärrskrivbord för att på ett enkelt sätt få inputs via applikationer som uppdragsgivaren demonstrerade och de funktioner som det önskemål av att få implementerade i det pågående projektet.

Då man i organisationen idag har ett väletablerat arbetssätt att använda Skype som kanal mot både kunder, leverantör och internt, fanns det en positiv attityd till att etablera en kund på plats via Skype eller motsvarande på tillgänglig applikation. Kunderna finns spridda över hela landet och det kan vara flera projekt som bedrivs samtidigt och Skype utgör basen för samtal med kunderna vilket är en fördel då kunden ibland befinner sig på ett stort geografiskt avstånd.

”Det är ju integrationsdelarna som vi pratar om där vi gör någon form av utveckling, då kommer det momentet av sig själv för då kan de logga in och köra och så använder man Skype som kommunikation ”

Vid en implementering av den webbaserade produkten ber man i första läget alltid kunden om att installera Skype. Därefter används Skype för att bedriva möten och även ge tillgång till dator i form av fjärrskrivbord. Detta är möjligt genom att kunden själv kan sitta och köra skarpt i den webbaserade produkten och via Skype komma med inputs, och ändringar kan ske i realtid vilket medför att kunden ser resultatet direkt. Utvecklingsansvarig påtalar att samtal via Skype ibland skapar problem då det är många beslut som skall fattas och kunden kommer med mycket inputs om hur saker skall implementeras.

”Det händer ofta saker väldigt fort och beslut fattas väldigt snabbt man kan ändra sig och så vidare då”

”mail måste skickas och konfirmera det som man kommit överrens om, annars blir det lätt oklart när man pratar på skype och för krav på att göra ny funktionalitet”

Ett annat problem gäller tillgängligheten där kunden lätt kan få en uppfattning att bara för att man är online så är man också tillgänglig och att kunden genom att inte vara närvarande inte ser vad som pågår med utvecklingsarbetet.

”så blir det så att man säger att vi tar det på mötet på torsdag men så ser de att man [kund] är på Skype så frågar man direkt så tänker man svarar han så svarar han”

4.3.3 Tidshantering

Då det finns starkt intresse från kundens sida att få med så mycket funktionalitet som möjligt till en första prototyp av applikationen, har stor vikt i projektets början lagts vid att fundera ut hur en eventuell implementering av funktionaliteten skulle gå till. Då projektets medlemmar hade en erkänd avsaknad av expertkunskap inom problemområdet. Blev det extra tydligt för att hålla tidsramen och kunna utveckla en fungerande applikation med basfunktionalitet, att restriktioner var tvungna att införas kring utökad funktionalitet vars användningsområde var föremål för diskussion. Därmed infördes i iteration två ett moment hämtat från RAD -tidshantering för att på förekommen anledning stävja problematiken med utveckling av finesser och funktionalitet, som inte var helt säkert att det skulle komma att användas, i ett tidigt skede av projektet.

”utrymme för att skjuta på saker som är av sekundär betydelse”

Utvecklingsansvarig påpekade att i de projekt som de deltar i, finns det en önskan att finna möjligheter att säga till kunden vad som är av primär betydelse och som kan lösas direkt och vad som är av sekundär betydelse och som därmed får vänta tills ett senare skede.

> *”vill göra så här och jobbar också så här i viss mån”*

Det finns en önskan från utvecklingsansvarig att arbeta mer på detta sätt i fortsättningen och framförallt planera för det innan man sätter igång ett projekt. Utvecklingsansvarig menar dock att det finns skillnader i utveckling och implementering av ett problem mellan säljansvarig som säljer in projektet till kund och som för diskussionen med kund.

> *”det finns kanske en oenighet mellan säljare och de som ska göra projektet ”*

Det finns en skillnad mellan säljare och utvecklare som främst utgörs av tekniska barriärer om vad som är möjligt att leverera och inom vilken tidsplan. Enligt utvecklingsansvarig kan det uppstå meningsskiljaktigheter mellan utvecklaren och säljaren där säljaren i sin roll, säljer en produkt som ska tillfredställa ett behov. Vilket innebär att säljaren diskuterar i termer av att lösa ett problem för kunden på en mer abstrakt nivå medan utvecklaren på ett detaljerat plan ska lösa dessa problem. Det finns svårigheter med att på förhand identifiera tidsaspekten av en viss funktionalitet utan att känna till den problematik som måste lösas för att implementera funktionaliteten.

> ”ofta är det så när man ska sätta tider på en sak så är det svårt för när det handlar om utveckling så är det ofta saker som du inte har gjort innan”

Utvecklingsansvarig ser stora fördelar med att arbeta på det sättet. För att säkerställa att den funktionalitet som inte säkert kommer att användas eller att det kommer blir problematiskt att implementera att den skjuts på framtiden efter att den grundläggande funktionaliteten har gjorts.

> *”Jag tror att metoden som sådan främjar till att du kommer till ett alltså, mer lyckat projekt”*

4.3.4 ”Face to face”

Utvecklingsansvarig har förståelse för hur man bedriver ”face to face” -möten och kan också identifiera att man arbetar med ett ”stand up meeting” när man har utvecklingsprojekt som involverar flera projektmedlemmar. Även om de inte sker dagligen finns momentet i verksamheten.

> *”för det är ju det som är, för det är ju så vi gör”*

”Och så länge man har det på det planet att det är den interna diskussionen som sker på det viset då. Så är det klockrent då för det är ju precis så man vill ha det.”

Utvecklingsansvarig kunde se fördelar i möten med ”whiteboard” som redskap internt inom organisationen och att använda sig av direkt kommunikation istället för att arbeta med att producera och skicka runt dokument. I organisationen arbetar de på detta sätt idag och med förhoppning av att kunna arbeta mer så i framtiden. Dock har man enbart erfarenhet av att arbeta med ”face to face” -möten internt och utvecklingsansvarig påtalade ett osäkerhetsmoment av att använda sig det gentemot kund. Om det blev för mycket av att man bestämde en sak för att sen inte ha täckning för vad som sagts.

4.3.5 40 - timmars arbetsvecka

Vi har inte förändrat något i detta delmoment som tagits direkt ur systemutvecklingsmetoden XP. Under projektets gång har vi arbetat efter att genomföra åtta timmar per dag och har också nått det målet med undantag för någon dag. Det har till och med varit så att vi ett par dagar under projektets gång att vi enbart arbetat sex timmar. Utvecklingsansvarig kunde identifiera de problem som fanns sammankopplade med att inte har någon formulerad tidsram.

”Ja, för då är det något som är fel om det är 80 timmars veckor liksom en längre period”

”då har man ju planerat fel o då i och för sig så sitter man ju i skiten.”

Det är det som är 40-timmars arbetsveckas styrka att i ett tidigt stadium identifiera de problem som uppstår med projektet som är av tids eller resursmässig karaktär. Utvecklingsansvarig samtycker i frågan och kan se momentets uppenbara fördelar med att vara uppmärksam på detta gör att man kan fatta ett beslut i ett tidigare skede.

”Men å andra sidan om man har följt den här metoden så borde man ha upptäckt det tidigare så därför så blir det inte sådana veckor då”

”Nu har jag gjort 40 timmar nu måste jag gå hem.”

Enligt utvecklingsansvarig måste det finnas flexibilitet i arbetstiden vilket också återfinns i 40-timmars arbetsvecka där det ges utrymme för övertid för att lösa uppkomna problem men inte konstant övertid under längre perioder.

4.3.6 Planeringsmöten, Testning, Kontinuerlig integrering, Små lanseringar

Vi har inte förändrat något av dessa steg utan använt oss av momenten direkt så som de beskrivs i grundutförandet för systemutvecklingsmetoden XP. Utvecklingsansvarig i organisationen menar att det finns utrymme för flexibilitet med att genomföra små lanseringar eller möjlighet för kunden att testa implementerad

funktionalitet då den plattform som man utvecklar på medger för kunden att testa via ett webbaserat gränssnitt.

4.3.7 Systemmetafor

Vi har i projektet internt använt oss av systemmetaforer i form av att vi ofta i samband med ”face to face” -mötena förklarat för varandra vad vi menar om hur olika problem ska lösas. Metaforerna har till och med varit i form av scharader där en person med kroppen och rummet som arena försökt förklara för den andre hur ett problem uppstår och en eventuell lösning på problemet. Inga förändringar av momentet har gjorts utan det har tagits direkt från XP.

Utvecklingsansvarig medger för egen del att finns det svårigheter att på rak arm formulera metaforer men menar att det blir avsevärt enklare om man gjort en viss del integrering hos en kund och man vill förmedla en förståelse hos kunderna. Utvecklingsansvarig menar också att det är aktuellt när situationen kräver då vissa kunder med låg teknisk kompetens har ett större behov för att få en metaforuppbyggd bild, medan andra kunder med mycket hög teknisk kompetens inte har något större behov för det.

5. Diskussion

5.1 Anpassning av lättrörlig metod på Asivo

5.1.1 Designmönster

I projektet så jobbade vi till en början helt enligt XP-filosofin med de olika stegen: gemensamt ägande av kod, kodstandarder, återvinning av kod och enkel design. XPs fyra hörnstenar är planering, kodning, design och testning (www.extremeprogramming.org). Under rubriken planering ingår momenten återvinning av kod och enkelhet, under rubriken kodning så ingår momenten kodstandarder och gemensamt ägande av kod vilket vi valde att samla under samlingsnamnet designmönster. Genom att använda designmönster så bildas en gemensam standard för organisationen att jobba efter och det ökar även återanvändbarheten av redan skriven kod. Det viktiga är inte vilken standard som företaget väljer att jobba efter, utan det viktiga är att det finns en standard att jobba efter som alla kan känna igen sig i. Att använda sig av XP-begreppen, gemensamt ägande av kod och kodstandarder, som vi inledningsvis använde oss av, fungerar också, men då begreppen går in i varandra väljer vi att paketera de under samma namn. Vi menar även att begreppen återvinning av kod och enkel design går att kategorisera under detta namn också, då det har med själva kodandet att göra. I organisationen Asivo där det inte finns några tydliga drag av olika designmönster och kodstandarder så borde detta begreppet räcka. På en direkt fråga om företaget använder olika designmönster eller andra former av kodstandarder till utvecklingsansvarig så blir svaret:

> *”Det kan jag säga att vi inte gör”*

Men det finns inom företaget en medvetenhet om olika designmönster även om de har svårigheter med att sätta namn på det. Men vid närmare eftertanke av utvecklingsansvarig så framkommer det att företaget ändå i någon form jobbar efter designmönster och kodstandarder.

> *”Vad ska man säga, vi jobbar med Microsofts delar de har naturligtvis också designmönster det är klart att det där har jag gått in tittat på hur jag ska bygga mina dataobjekt för att det här ska bli smidigt då. Okej, då har jag tittat på det, jag har naturligtvis byggt sådant efter de regler som gäller då”*

Det vi vill påvisa här är att det blir för detaljerat för företaget att jobba efter de specifika punkter som XP delar upp designmönster i. Eftersom företaget inte har någon större kännedom om de olika designmönster som ingår i XPs filosofi. Detta gör att det känns naturligt att baka in dem i ett samlingsbegrepp och det samlingsbegreppet ska öka medvetenheten för designmönster och andra former av kodstandarder. Det går att göra en jämförelse med att generellt beskriva hur ett hus är uppbyggt vilket ofta inte blir så komplext, men beskrivs husbyggandet i detalj så ökar det genast graden av komplexitet. Genom att beskriva komplexa situationer på

en generell nivå så ökar det överblickbarheten (Rescher, 1998). Vilket vi tillämpar genom att lägga det under ett gemensamt samlingsnamn.

5.1.2 Kund på plats

I XP så förespråkas det att kunden ska närvara under hela utvecklingsprojektet vilket vi under projektets inledning följde till viss del. Det vill säga att kunden inte var närvarande i samma rum där utvecklingen ägde rum utan kunden satt i samma lokaler men några rum bort. Något annat var inte möjligt i denna studie. Det fungerade bra men vi märkte här att det inte var nödvändigt att kunden fysiskt var på plats utan vi började med att prova att ha möten över Skype. Vilket även utökades till att använda olika tekniska verktyg som fjärstyrning[6] av skrivbordet. Det gjorde det möjligt att på distans och över telefon sitta och demonstrera applikationen för kunden. Det resulterade i att kunden genom detta direkt kunde komma med synpunkter eller rent av prova applikationen utan att själv fysiskt närvara. Koch (2004) beskriver också hur samarbete med kunder och andra projektmedlemmar kan ske genom att använda moderna kommunikationshjälpmedel. Genom att använda moderna kommunikations hjälpmedel går det att överbrygga den förlust av rik information som annars kan ske. Detta passar organisationen eftersom de har erfarenhet av att använda tjänsten Skype och deras kunder oftast inte har resurser att närvara under hela projektet. Även på grund av att företagets kunder har en stor geografisk spridning så används Skype som bärare av rik information. Under intervjun så frågar vi utvecklingsansvarig om detta är något som skulle passa Asivo att jobba med så blir svaret:

> *”det tycker jag nog att vi gör redan idag, det är ju egentligen så vi säger till kunderna att installera det här (Skype). Det här med remotestyrning har vi upptäckt är skitbra om man applicerar det framför allt när man sitter och utvecklar och de vill smälla upp miljön någon annanstans.”*

Att jobba på det här sättet passar Asivo bättre än att ha kunden på plats vilket rent av kanske skulle vara omöjligt då företaget jobbar med många projekt på samma gång. Det som även talar för den här formen av kontakt med kund är det affärssystem som Asivo jobbar med att integrera med kunders andra system är webb-baserat. Vilket gör att det finns goda möjligheter att arbeta med kunden på distans då allt kunden behöver för att kunna köra och testa applikationen är en Internet-uppkoppling och en webbläsare. Det som kan tala emot den här typen av kontakt med kunder är de olika tekniska problem som kan uppstå, vilket kan vara trögt bredband som medför att både ljud och bild kan hacka. Det finns vidare även problem med dokumentation när mycket av kommunikationen sker muntligen över Skype, påpekar utvecklingsansvarig. Då det kan vara svårt att bevisa vad som egentligen är sagt.

[6] Fjärstyrning gör det möjligt att dela databild och även att kunna styra en annan dator på distans.

”Sen finns det en nackdel till då , nu finns det i och för sig en historik i chatt-delen men just när man pratar med varann(host,host), det är både bra och dåligt för det händer ofta saker väldigt fort och beslut fattas väldigt snabbt och man kan ändra sig och så vidare då, men det är ju väldigt svårt att hitta när någon säger varför gjorde vi så? Varför sa du så, nej det har jag aldrig sagt, det är ett problem då.”

Detta är inte bara ett problem när man använder Skype eller liknande nya forum för kommunikation. Det gäller även när kommunikationen sker på det traditionella sättet över vanlig telefon eller mobiltelefon. Sker kommunikationen genom att kunden är på plats så finns fortfarande problematiken med att muntlig kommunikation inte alltid dokumenteras. Men om kunden är på plats hela tiden så minskar risken för kommunikationsmissuppfattningar då de kan upptäckas på ett tidigt stadium (www.extremeprogramming.org). Det är inte alltid som det finns möjlighet för kunden att hela tiden finnas på plats och det kan vara av ekonomisk anledning, då det blir väldigt dyrt för ett företag att avsätta en resurs som ska vara tillgänglig hela tiden.

5.1.3 Tidshantering

Momentet tidshantering som inte var ett moment från början utan som kom till under fallstudien, är ett sätt att begränsa applikationen från funktionalitet som inte är nödvändig för att den ska fungera. Detta moment är hämtat från RAD som utvecklingsmetod kändes nödvändig att jobba efter då kontakten med kunden sker regelbundet. När kontakten med kunden sker på regelbunden basis så finns det en tendens att kunden kommer med många nya önskemål. Detta är bra eftersom det är en grundidé i lättrörliga metoder att kunden ska kunna komma med nya krav. Men här har vi märkt att det finns en tendens att kunden kommer med så mycket nya önskemål att det blir svårt att genomföra allt under utsatt projekttid. Vilket gör att det känns naturligt att lägga till detta moment i utvecklingsmetoden där fokuseringen ska ligga på att få klart de absolut nödvändigaste funktionerna istället för att lägga energi på extra funktioner.

5.1.4 ”Face to Face”

Varje utvecklingsdag började med ett ”face to face” -möte där vi informellt började med att kontrollera vad vi bestämt från föregående dag och hur det hade gått. Vi bestämde också vad som skulle göras till nästa ”face to face” möte helt enligt XP-filosofin(Beck, 2000). Att genomföra ”face to face” -möten så ofta medförde att vi fick en väldigt bra uppfattning om hur vi låg till tidsmässigt. Detta skulle vi förvisso haft i det här projektet ändå, eftersom vi endast var två utvecklare och vi jobbade till en början med parprogrammering. Det gjorde att vi båda utvecklare hade bra kontroll på hur projektet framskred och vi hade indirekta ”face to face” -möten hela tiden under parprogrammeringen. Består projektet av fler än två utvecklare så kan ”face to face” -möten fungera som ett kontrollverktyg över att projektet är på väg åt rätt håll. Mötena kan även öka möjligheten till att använda redan skriven kod genom att alla vet vad alla gör och kan då lätt ta del av redan producerad kod. På företaget Asivo som är en

relativt liten organisation så jobbar de inte med en omfattande skriftlig dokumentation internt. Vilket gör att de jobbar mycket med personliga möten och förklarar för varandra på ”whiteboard” eller bara muntligt. Det kan finnas en problematik när den här typen av direkt kommunikation sker mot externa intressenter, det blir väldigt svårt att styrka att kunden har sagt på ett visst sätt. Så till viss del måste det ske någon form av dokumentation i några fall där det finns misstanke om att det kan uppstå meningsskiljaktigheter. Vilket ofta kan uppstå i samband när kunden vill ha utökad funktionalitet under projektets gång.

5.1.5 40-timmars arbetsvecka

Punkten 40 - timmars arbetsvecka kan vara lite missledande, det betyder inte att utvecklarna aldrig ska jobba mer än 40 timmar utan snarare att om varje utvecklingsvecka är längre än 40 timmar så är projektet fel ute. Under detta utvecklingsprojekt så var inte arbetstiden något problem utan det gick att följa devisen 40 - timmars arbetsvecka bra. För organisationen Asivo så möttes denna punkt av skepsis då det blir väldigt svårt för en liten organisation att bara lägga ner sitt arbete bara för att man har jobbat mer än 40 timmar. Vilket utvecklingsansvarig påvisar på en fråga om detta moment:

> *”Det rimmar inte riktigt med hur våran organisation jobbar utan eftersom det beror på projekt, ena veckan kanske du jobbar 60 h och nästa vecka 20 h så det är liksom helt beroende på hur du vill styra det och hur intensivt det är i projektet just då.”*

Det finns här lite missförstånd bakom denna tanke, det är inte alls så att varje vecka ska vara 40 timmar utan tanken är just att kunna vara flexibel. Att ena veckan arbeta 80 timmar och nästa vecka jobba 20 timmar. Vilket faktiskt precis är det som organisationen gör utan att klassificera det enligt någon teori. Under punkten 40 - timmars arbetsvecka så beskrivs det att en utvilad programmerare är en bättre programmerare och att den gör mindre fel . En högre kvalité uppnås också (Beck, 2000). Det gäller inte bara för utvecklare utan också för de flesta yrkeskategorier. Vilket även utvecklingsansvarig påpekar under intervjun.

5.1.6 Planeringsmöten

Utvecklingsprojektet bedrevs under fyra iterationer där varje iteration startade med ett planeringsmöte. Varje planeringsmöte började med att fastställa vad som har hänt från föregående möte och planera vilka uppgifter ska lösas till nästa möte. Detta fenomen med planeringsmöten är ingen nytt för organisationen utan är något som de använder vid uppstart av ett nytt projekt men deras benämning är ”workshop” där själva kravspecifikationen sätts och iterativt kan förändras till nästa möte.

5.1.7 Testning, Kontinuerlig integrering, Små lanseringar

Testning och kontinuerlig integrering har ägt rum på daglig basis där vi som utvecklare själva har utfört testning och integrering. Under varje planeringsmöte för varje iteration så har kunden haft möjlighet att testa och komma med synpunkter.

Kontinuerlig integrering mot det befintliga affärssystemet har också var en del i det dagliga arbetet. Arbetsmetoden med kontinuerlig integrering och små lanseringar passar Asivo som organisation där denna arbetsmetodik tillämpas redan, även om de själva inte namnger detta arbetssätt enligt XP. Eftersom de jobbar med ett webb-baserat affärssystem som gör det möjligt för kunden att prova ny funktionalitet hela tiden genom att bara logga in på applikationen genom en webbläsare. Vilket utvecklingsansvarig berättar under intervjun när vi frågar om det är små lanseringar som de jobbar med:

> *”Ja precis, så försöker vi jobba med Ataio på det sättet, för där kan ju användaren från dag ett vara inne i systemet.”*

Arbetssättet är inget nytt för organisationen utan passar in i ett redan befintligt mönster. Det kan finnas en liten risk med att ha många små lanseringar menar utvecklingsansvarig, den del som kunden testar kanske inte symboliserar hela applikationen, vilket kunden kan tro och de blir missnöjda för att applikationen inte fungerar som de har tänkt. Det är viktigt att göra klart för kunden vad den ska testa och att det inte är den färdiga applikationen vilket utvecklingsansvarig berättar under intervjun:

> *”… och det får man också har klart för sig då så att inte kunden tror att allt ska funka när kunden ska testa det.”*

Att jobba med små lanseringar är en viktig del i XP och det är väldigt viktig att det inte går för lång tid mellan lanseringarna. Om det blir en lång tid mellan lanseringarna, desto mindre tid finns det för att lösa de problem som uppstår (www.extremeprogramming.org). Att ha många små lanseringar gör också att det finns något konkret att diskutera kring mellan kund och utvecklare (http://xprogramming.com). Under utvecklingsprojektet har vi haft stor nytta av att hela tiden ha små lanseringar. Det har medfört att vi har fått mycket bra synpunkter och andra invändningar från kund. Genom en tät kontakt har kunden kunnat vara med och påverka utvecklingen direkt. Kommer kunden med synpunkter och önskemål på ett tidigt stadium så har vi ofta på ett enkelt sätt kunnat implementera dem utan några problem. Hade projektet bedrivits enligt traditionella utvecklingsmetoder, såsom vattenfallsmodellen, hade kunden inte kunnat testa applikationen under projektets gång, vilket medför att synpunkter först kan komma när applikationen är klar. I lättrörliga metoder har kunden möjlighet att komma med synpunkter i varje iteration.

5.1.8 Systemmetafor

Att kunna arbeta med olika former av systemmetaforer kräver en del av utvecklarna för att komma på passande beskrivningar, som på ett bildligt sätt förmedlar budskapet. Vilket kan hjälpa till att överbrygga den klyfta som annars kan uppstå mellan utvecklare och kunder (Koch, 2004). Vi har under projektet oftast inte behövt använda metaforer oss systemutvecklare i mellan. Det fanns några moment där vi

behövde förtydliga hur data skickades i mellan klient och server. Vi använde oss då av olika metaforer för att beskriva det. Vilket var ett bra sätt för att klarlägga och tydliggöra vad som händer. Vi märkte att det fanns en tydlig skillnad i användandet av metaforer mellan utvecklare och säljare på företaget. Där säljaren är den som har mest kontakt med kund vilket också märks eftersom den är mycket mer van att använda systemmetaforer för att få kunden att förstå de tekniska delarna. Det framgår också klart under intervjun med utvecklingsansvarig:

> *”det blir ju när situationen kräver det på något sätt. Sedan tror jag att man är mer eller mindre duktig på det då, jag är nog inte så duktig på det då. Men som säljare till exempelvis så tror jag att du kanske måste ha den förmågan att på något sätt.”*

5.2 Implementering av lättrörliga metoder på Asivo

På Asivo arbetar man utifrån två grundkoncept. I det ena fallet arbetar man utefter konsulttid där ett arvode tas ut per timme. Detta sker främst gentemot befintliga kunder som behöver ytterliggare anpassningar eller utveckling av ny funktionalitet till den produkt man köpt. I det andra fallet arbetar man med nyförsäljning av produkten som ofta sker till en fast kostnad som baseras på vad kunden vill ha för funktionalitet utöver det som levereras i standardsystemet. Beroende på vad för typ av funktionalitet som ska implementeras för kunden, skapas i de fall där en nyförsäljning sker, en tidsplan i form av hur många timmar som projektet beräknas ta utefter den funktionalitet som kunden önskar. Asivo gör ingen skillnad mellan projekt eller konsulttid, då ett projekt där en grundläggande funktionalitet ska utvecklas i princip skulle kunna ta kortare tid att genomföra än ett konsultarbete för en kund som vill ha utökad avancerad funktionalitet. Sättet som arbetet bedrivs på är signifikant för operativa ad hoc-organisationer. Enligt Mintzberg (1981) är det vanligt att planering och design flyter ihop med implementering i projektet. På Asivo arbetar personalen informellt i projekten och formella titlar saknar betydelse, exempelvis deltar VD i projekten för att tillföra kompetens för att lösa uppgiften och deltar inte i projekt i egenskap av VD.

Utvecklarna på Asivo arbetar inte efter en specifik systemutvecklingsmetod när de utvecklar funktionalitet för kunder i projekten. Systemutvecklingen i projekten sker i en ad hoc-anda där man efter ett inledande möte med kunden sätter igång och utvecklar och skapar den funktionalitet som behövs och problem med buggar löses efterhand som de uppstår. Sättet att genomföra utvecklingsprojekten på, baseras på individuell prestation, kunskap och motivation (www.ctg.albany.edu). Utan en strukturerad systemutvecklingsmetod tenderar kunskapen som erhålls från ett projekt att hållas kvar på individnivå snarare än på organisationsnivå. Det uppstår då ett problem när organisationen vill upprepa ett resultat och genomföra ett projekt som är snarlikt ett tidigare. Det finns då inga belägg för att resultatet ska bli detsamma som tidigare om det inte är samma personer som arbetar med det nya projektet som också deltog i det gamla. Det är inte organisationen som har kontroll över processen

och därmed kommer inte förbättringar av den organisationen till gagn (Paulk, Weber, Garcia, Chrissis & Bush, 1993). Genom att Asivo brutit ny mark och gett sig in på en ny marknad medför det att man utvecklar applikationer som är delvis fristående från den plattform man utvecklat på tidigare. Det har gett upphov till längre projekt med systemutveckling från grunden fram till en färdig applikation. Det finns därmed en anledning att frångå ad hoc-utvecklingen som man bedrivit tidigare. Det hämmar organisationens möjligheter att vara produktiv och att genomföra kvalitetsförbättringar över tid (Paulk et al.,1993).

På Asivo arbetar man inte efter en på förhand bestämd specifik lättrörlig metod i projekten, men sättet man bedriver sin verksamhet på är starkt knutet till de värderingar som har sin hemvist i lättrörliga metoder (www.agilemanifesto.org). Lättrörliga metoders fyra värderingsgrunder kan identifieras i Asivos sätt att arbeta på, i den nuvarande verksamheten.

5.2.1 Individer och samspel framför metoder, processer och verktyg

I och med att Asivo är en liten organisation och att de har mindre än nio anställda kan verksamheten bedrivas och koordineras genom personliga möten. Mötena sker både spontant och planerat. De planerade interna mötena äger rum varannan vecka och följer en stående agenda men har förutom de stående punkterna stora möjligheter till att vara flexibla.

> *”men vi försöker ha planeringsmöte varannan vecka då vi går igenom ekonomi och annat sånt här, och i samband med det här så slänger vi in om det är några problem med några kunder[problem som ska lösas åt kund]. Man visar vad man gjort nu och vad man har för idéer”*

För att lyckas med projekten är det enligt Martin (2001) ytterst viktigt att ha medarbetare som är goda på att kommunicera och samverka i grupp. Enligt Martin behöver de inte vara bäst på att programmera kod utan de kan vara genomsnittliga programmerare så länge de kan interagera i gruppen. Då Asivo endast har fyra anställda som arbetar med den delen verksamheten finns redan en etablerad och sammansvetsad grupp. Där medlemmarna samverkar och interagerar med varandra för att lösa den uppgift som man ställts inför. Gruppen kommunicerar med varandra inbördes och gruppmedlemmarna finns tillgängliga för varandra genom applikationer som Skype. Även när man arbetar i separata projekt hjälper man varandra med lösningar på problem. Enligt Boehm (2002) är det av stor vikt att tillföra personer till projektet med en högre kunskapsnivå. Det beror inte på utvecklingen med lättrörliga metoder i sig kräver en större kunskap utan det beror på att till skillnad från traditionella projekt saknas den dokumentation i form av kunskapsplaner. Det är den individuella kunskapen som är essentiell vid utveckling med lättrörliga metoder. Om inte kompetensen finns hos deltagarna i projektet så är det dömt att misslyckas (Cockburn & Highsmith, 2001). Att samverka i gruppen och lära sig av varandra medför att utvecklarna höjer sin kunskapsnivå än vad som varit fallet om man utvecklat individuellt(Cockburn & Highsmith, 2001). Det pekar på att det är i projekt

med lättrörliga metoder är utav betydelse att man tillför expertkunskap till gruppen. Gruppens storlek är också av betydelse för att lyckas med projektet. Enligt Highsmith och Cockburn (2001) är det även om större projekt har lyckats, essentiellt att genomföra projekt i mindre storlek och det genomsnittliga projektet som använder sig av lättrörliga metoder har nio projektmedlemmar. Vilket enligt Highsmith och Cockburn är en bra siffra för att ge möjlighet att använda sig av de flesta lättrörliga moment som står till buds. I Asivo finns förutsättningar att lyckas med projekt som bygger på lättrörliga metoder då man är en mindre organisation.

På Asivo använder man sig inte av några verktyg såsom CASE[7] verktyg som hjälpmedel i utvecklingen utan man använder sig bara av utvecklingsverktyget Visual Studio. Enligt Martin (2001) kan det vara en fördel att enbart använda sig av verktyg som man kan hantera snarare än att ha verktyg som är överproportionerade för verksamheten och som man kan ha en större tilltro till än vad som egentligen är fallet.

5.2.2 Körbar programvara framför omfattande dokumentation

När man på Asivo har sålt in ett system hos en kund har man ett första möte och i samband med det genomförs ofta en workshop där man sitter med kunden och arbetar i det befintliga standardsystemet. Då brukar diskussioner föras kring implementering av ytterligare funktionalitet som kunden har behov av. Asivo arbetar inte i ett sekventiellt flöde som tar sin början med ett analysdokument som baseras på en fastställd kravspecifikation och därefter ta fram ett designdokument som bygger på analysdokumentet. Man arbetar på ett mer flexibelt sätt mycket på grund av att produkten i befintligt skick är en plattform som utökad funktionalitet adderas på. Enligt Martin (2001) är det värre att producera för mycket dokumentation än att producera för lite. Framförallt är det viktigt att producera dokument om det tänkta systemet på en högre abstraktare nivå som täcker grundbultarna för systemets existens. Detta dokument kan från månad till månad revideras för att vara så aktuellt som möjligt (Martin, 2001). På Asivo hålls kontakt med kunden hela tiden under projektets gång och kunden kan under projekttiden komma med önskemål om utökad funktionalitet och man tar hand om de problem som uppstår när de uppstår. Enligt Martin (2001) ska fokus ligga på systemutvecklingen och inte på dokumentationen, vilket ligger i linje med hur man bedriver projekten på Asivo.

> *”Att kravspecen liksom den har på något sätt, betydelsen av den försvinner lite tycker vi då när vi jobbar med de här grejerna.”*

Enligt Martin (2001) blir det extra viktigt med gruppsamverkan och framför allt kommunikationen mellan medlemmarna i projektet då dokumentationen hålls på ett minimum. Det som då finns tillhands i form av dokument inom projektet är den kod som producerats, och den konsensus man kommit fram till för hur ett system ska utvecklas och i vilken riktning det ska gå. Det är viktigt att den kunskap man har,

[7] Computer- Aided Software Engineering. En samling verktyg för att underlätta utveckling och underhåll av mjukvara.

överförs till nya medlemmar som kommer in i projektet, att man lär varandra och har ett tätt samarbete (Martin, 2001). I Asivo genomförs ”face to face” -möten mellan projektmedlemmarna på en daglig basis, där problem som uppstått i projektet dryftas och man använder sig av ”whiteboards” eller A4 ark för att kommunicera inom gruppen för att tydliggöra hur olika problem ska få en lösning. Även om utvecklarna inte deltar i samma projekt så hjälper man varandra och bidrar med kunskap för att hitta en så optimal lösning som möjligt.

5.2.3 Kundsamarbete framför kontraktsförhandlingar

Enligt Martin (2001) visar de projekt med negativ utgång och med en dålig kvalité att man försökt utveckla system genom att först specificera vad som ska göras och sen bestämt vad det får kosta. Enligt Martin (2001) kan inte mjukvara beställas som vilken vara som helst. För att erhålla ett framgångsrikt projekt måste kunden vara med från början till slut. Kunden måste komma med återkoppling om funktionalitet under hela projekttiden. För att inte hamna i en situation där omförhandlingar sker på grund av ett på förhand väldetaljerat kontrakt om vad som skall göras och till vilken kostnad, är det mer lämpligt att ha kontraktet på systemet som utgångspunkt istället för som barriär mellan parterna. Att ha det som utgångspunkt innebär att ett inte lika detaljerat dokument skapas vid projektstart och det innebär att kunden istället är med under projektets gång och genomför acceptanstest för vad som är godtagbart. Det möjliggör också för kunden att komma med förändringar under projektets gång (Koch, 2004). Asivo säljer en produkt som anpassas efter kunden. Där kunden och Asivo kommer överrens om vad för sorts typ av lösningar som ska implementeras. Det ligger i Asivos intresse att behålla kontakten med kunden under projektets gång och även efter projektet. Asivos verksamhet baseras till stor del på konsultering på befintliga kunder som vill vidareutveckla sitt system och sin verksamhet. Det innebär att man från Asivos sida har en hög inställning av att samarbeta och ha en god kommunikation med kunden under projekttiden. Eftersom man är ett relativt litet företag finns också en vilja av att bygga upp en bas med goda kundreferenser vilket borgar för att man som organisation är mån om kundens tillfredställelse.

5.2.4 Anpassning till förändring framför att följa en statisk plan

Asivo som liten organisation har stora möjligheter att ställa om och svara mot förändringar. Förmågan att svara och vara villig till förändring avgör ofta i systemutvecklingsprojekt om projektet kommer att lyckas eller inte (Martin, 2001). För Asivo som liten organisation ligger viljan till förändringsbenägenheten att vara kunderna till lags och svara upp mot deras önskemål.

Enligt utvecklingsansvarig på Asivo görs det mesta av utvecklingen ofta för första gången. Vilket medför att det kan vara svårt att förutse de problem som kan uppstå senare i ett projekt och veta hur mycket tid det kan ta i anspråk. Enligt Martin (2001) är en bra utgångspunkt när man arbetar i ett projekt att ha de närmaste veckorna relativt detaljerade över hur man ska arbeta och vad man ska göra och ju längre bort i tiden som projektet löper desto mer vagt ska man planera. Det ger en ökad

möjlighet för att vara öppen för förändringar längre fram. Dessutom låser man inte in sig och planerar saker och ting längre fram i tiden och när man då kommer dit så existerar inte den funktionaliteten i verkligheten, för det visade sig för länge sedan att den inte var möjlig att implementera (Everette, 2002). Under projekten har kunderna möjlighet att ha kontakt med utvecklarna om de önskar en utökad funktionalitet eller om de är redan befintliga kunder som får problem med sina köpta produkter och behöver uppdatera med ny funktionalitet.

Asivo är mån om sina kunder och som tidigare nämnts vill man ha goda referenser för kommande affärer vilket innebär att de också är högst tillgängliga för kunden, i och med att man som kund har Asivos utvecklares namn i sin skypekontaktlista. Det medför att kunden när som helst kan komma med inputs i ett projekt. Detta innebär en risk för störningsmoment, att hela tiden vara tillgänglig då man sitter och utvecklar en viss typ av funktionalitet för kunden och så hör kunden av sig med andra frågeställningar. Samtidigt vill man tillfredställa kundens behov.

För att effektivisera utvecklingsarbetet och färdigställa applikationer inom given tidsplan som ett krav från en marknad som kräver snabba resultat, och som också står för snabba svängningar(Jacobson, 2002), finns det anledning för Asivo att använda sig av en lättrörlig systemutvecklingsmetod som kan ge utvecklarna de verktyg som de behöver. Detta för att kunna leverera funktionalitet inom en fastställd tidsram och till så bra kvalité som möjligt. Då Asivo som organisation förväntas växa och även nyanställa fler utvecklare finns det anledning att frångå ad hoc-utveckling. Asivo behöver strukturera systemutvecklingsprocessen och uppnår det genom att använda en specificerad metod. Asivos nuvarande sätt att arbeta på har stora drag av det synsätt som står bakom lättrörliga metoder, vilket innebär att steget från dagens arbetssätt till att införa en lättrörlig metod i systemutvecklingsprojekten inte är särskilt stort. Implementering av en lättrörlig metod i Asivos verksamhet möjliggör att man kan bli mer effektivare i utvecklingsarbetet. Det uppnår man genom att återanvända kod och att arbeta strukturerat. Det är också viktigt att arbeta i samverkan för att utveckla mer generella lösningar vilka kan implementeras i flera olika projekt. Genom att använda sig av lättrörliga systemutvecklingsmetoder hålls organisationen inte ute från utvecklarnas kompetens, som är risken i ad hoc-utveckling. Genom att ta till vara på den kod som produceras och arbeta metodiskt med fokus på interaktion och samverkan mellan projektmedlemmarna tillvaratas den kunskap som kommer ifrån projektet (Paulk et al., 1993).

5.3 Implementering av lättrörliga metoder i ad hoc-kratier

De organisationer som kan identifieras som ad hoc-kratier har ett sätt att bedriva verksamheten på som ligger mer i linje med lättrörliga metoder än med traditionella systemutvecklingsmetoder. Enligt Koch (2004) är det svårare för en strukturell och hierarkisk organisation och det tar avsevärt längre tid att anpassa en sådan typ av organisation till ett nytt sätt att arbeta på. Hierarkiska organisationer har fördefinierade kommunikationskanaler och beslutsvägar. Det blir svårare att

implementera lättrörliga systemutvecklingsmetoder i den här typen av organisationer då lättrörliga metoder förespråkar en mer flexibel och en mer direkt kommunikation (Martin, 2001). Vid arbete med lättrörliga metoder är det inte viktigt med titlar och positioner då fokus ligger i att lösa uppgiften och alla deltagare i projektet ska kunna samverka och kommunicera så fritt som möjligt. Enligt Koch (2004) ska man inte underskatta de kulturella krafterna i en organisation när man ska förändra verksamheten. I ad hoc-kratier finns kulturen med att lägga titlar åt sidan och med gemensamt engagemang lösa uppgiften redan etablerad, vilket skulle medföra att införandet av en lättrörlig metod sker mer smidigt. Enligt Koch (2004) är det en förutsättning för att lyckas implementera en lättrörlig metod, att organisationen är van vid att arbeta flexibelt och att de är anpassningsbara. Lättrörliga metoder förespråkar mer generella kravspecifikationer för att möjliggöra för kunden att komma med input under iterationerna i projektet. I en ad hoc-krati arbetar man utefter en större flexibilitet och anpassar sig för att tillfredställa kundens behov i högre grad än i hierarkiska organisationer (Koch, 2004). Enligt Mintzberg (1981) är organisationens storlek en faktor om den kan bedriva verksamheten i form av en ad hoc-krati eller utifrån ett strukturerat arbetssätt. Det som sätter begränsningen på organisationens storlek är möjligheterna till direkt kommunikation. Samordningsproblem uppstår när en organisation blir så stor så att det inte går att bedriva en daglig ”face to face”-kommunikation för att driva utvecklingen framåt i projektet. Enligt Koch (2004) så ska inte projektgrupperna vara större än 10-15 personer för då uppstår kommunikationsproblem vilket i sin tur får konsekvenser för utgången av projektet. Förmågan att kommunicera inom gruppen och interaktivt samverka med gruppmedlemmarna är nyckelord i lättrörliga metoder (Martin, 2001). Synen på det sätt vilket man bedriver verksamheten i ad hoc-kratier harmoniserar med de värderingar och principer som lättrörliga metoder står för (www.agilemanifesto.org). Detta medför att implementering av en lättrörlig metod i en ad hoc-krati kan ske på ett smidigt sätt, då man kan dra fördelar av den kultur man har i organisation till skillnad mot andra organisationer som har en mer strukturerad och hierarkisk verksamhet.

6. Slutsats

Syftet med den här uppsatsen har varit att undersöka hur väl lättrörliga metoder passar i ad hoc-kratier som inte tidigare arbetat utefter traditionella utvecklingsmetoder. Det sätt som verksamheten bedrivs på hos Asivo, såsom direkt kommunikation, spontana möten, personalen innehar flera roller, titlar inte viktigt, starkt fokus på kundens behov och begränsad intern dokumentation, alla dessa faktorer indikerar på att Asivo är en ad hoc-krati. De tankegångar, som genomsyrar organisationen, ligger nära de värderingsgrunder som lättrörliga metoder bygger på. Med extrem programmering som utgångspunkt fann vi att det inte gick att implementera alla moment direkt i verksamheten. Ramverket för anpassning av lättrörliga metoder användes för att utröna vilka förändringar som behövdes göras för att erhålla en lättrörlig metod anpassad för organisationen. De förändringar av XP momenten som har gjorts baseras främst på att organisationen är liten och har flera projekt som pågår samtidigt. Storleken på organisationen är signifikant för ad hoc-kratier och användning av lättrörliga metoder är lämplig i mindre organisationer och grupper.

Då Asivo inte har någon förankring i ett traditionellt plandrivet sätt att arbeta fanns möjligheten att utan ett stort organisatoriskt förändringsarbete implementera en lättrörlig metod. Organisationens storlek var en bidragande orsak till att implementeringen slog väl ut. En god gruppsamverkan och interaktion skapade förutsättningar att lyckas med utvecklingsprojektet då gruppmedlemmarna som saknade expertkunskap kunde dra lärdom av varandra.

XPs möjligheter att likt ett smörgåsbord ge ett stort urval av moment som kan väljas ut passar väl in på ad hoc-kratier. Ad hoc-kratier som organisationstyper förändras snabbt och anpassar sig själva för motsvara yttre krav från uppdragsgivare och har därmed ett behov av metoder som i sig själva kan förändras och anpassas till verksamheten. De värderingsgrunder som återfinns i lättrörliga metoder kan också identifieras i ad hoc-kratier. Det innebär därmed att det finns förutsättningar för den konstruktion vi skapat i en Ad hoc-krati också skulle vara möjlig att implementera i en annan organisation som kan identifieras som en ad hoc-krati.

Vidare undersökning skulle behöva göras huruvida konstruktionen är implementerbar i andra organisationer som kan identifieras som ad hoc-kratier i sin helhet. Även om det finns indikationer på att så är fallet finns inga belägg för detta förrän konstruktionen implementeras i en annan organisation.

7. Referenser

Böcker och artiklar

Abrahamsson, P., Warsta, J., Siponen, M. , Ronkainen, J. (2003). New directions on agile methods: a comparative analysis. *Proceedings of the 25th international conference on software engineering 2003,* 244-254

Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J. (2002). Agile Software Development Methods: Review and Analysis. *VTT Publications 478*.

Beck, K. (2000). *Extreme programming explained*. Boston: Addison-Wesley

Beck, K. Fowler, M. (2000). *Planning extreme programming*. Boston: Addison-Wesley

Beedle, M. Devos, M. Sharon, Y. Schwaber, K. Sutherland, J. (2000) SCRUM: An extension pattern language for hyperproductive software development. *Pattern Languages of Program Design 4, N. Harrison, B. Foote, and H. Rohnert, eds., Addison-Wesley, Reading, Mass., 2000*, 637-651

Boehm, B. (2002). Get ready for agile methods, with care. *Computer, volume: 35*, issue: 1, 64-69

Cockburn, A., Highsmith, J. (2001). Agile Software Development: The People Factor. *Computer, volume: 34,* issue: 11*,* 131-133

Cockburn, A. (2001). *Agile Software Development*. Boston: Addison-Wesley

Cornford, T., Smithson, S. (2006). *Project Research in Information Systems*. New York: Palgrave Mcmillan

Cohn, M., Ford, D. (2003). Introducing an agile process to an Organization. *Computer, volume: 36*, issue: 6, 74-78

Dingsøyr, T. , Moe, N., Nytrø, Ø. (2001). Augmenting Experience Reports with Lightweight Postmortem Reviews. *International Conference on Product Focused Software Process,* 167-181

Easterby-Smith, M. , Thorpe, R., Lowe, A. (2002). *Management Research – an Introduction.* Trowbridge, Wiltshire: The Cromwell Press Ltd

Grenning, J. (2001). Launching Extreme programming at a process intensive company. *IEEE Software, volume: 18*, issue: 6, 27-33

Highsmith, J. (2000). Retiring Lifecycle Dinosaurs. *Software Test and Quality Engineering Magazine (STQE), volume: 2*, issue: 4

Jacobson, I. (2002). A resounding Yes to agile processes – but also to more*. Cutter IT Journal, volume: 15,* issue:1

Kasanen, E., Lukka, K., Siitonen, A. (1993). The constructive approach in management accounting*. Journal of Management Accounting Research, volume: 5*, 243-264

Koch, A. (2004*). Agile software development: Evaluating the methods for your organization*. London: Artech house

Labro, E., Toumela,T-S. (2003). On bringing more action into management accounting research: process considerations based on two constructive case studies. *European Accounting Review, volume: 12,* issue: 3, 409-442

Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., Tesoriero, R., Williams, L., Zelkowitz, M. (2002). Empirical findings in agile methods. *In proceedings of Extreme programming and Agile methods – XP/Agile Universe,* 197-207

Lukka, K. (2000). The key issues of applying the constructive approach to field research*. Publications of Turku School of Economics and Business Administration, Series A-1:2000*, 113-28

Lukka, K., Kasanen, E. (1995). The problem of generalizability: anecdotes and evidence in accounting research*. Accounting, Auditing & Accountability Journal, volume: 8,* issue: 5, 71-90

Mintzberg, H. (1981). Organization design: fashion or fit? *Harvard Business Review, volume: 59*, issue: 1, 103-116

Mullins, W.J., Orville, C.W., Boyd, W.H., Larréche, J-C. (2005). *Marketing Management: A strategic decision-making approach*. New York: Mcgraw-Hill

Pikkarainen, M., Salo, O., Still, J. (2005). Deploying Agile Practices in Organizations: A Case Study. *EuroSPI,* 16-27

Pikkarainen, M., Passoja, U. (2005). An approach for assessing suitability of Agile Solutions: A case study. *Extreme Programming and Agile Processes in Software Engineering, 6th International Conference*, 171-179

Rescher, N. (1998). *Complexity, A Philosophical Overview*. New Jersey: Transaction Publishers

Salo, O., Kolehmainen, K., Kyllönen, P., Löthman, J., Salmijärvi, S., Abrahamsson, P. (2004). Self-Adaptability of Agile Software Processes: A Case Study on Post-Iteration Workshops. *Proceedings of the 5th International Conference on Extreme Programming and Agile Processes in Software Engineering, Springer,* 184 - 193

Salo, O. (2005). Systematical Validation of Learning in Agile Software Development Environment. *Wissensmanagement,* 106-110

Schwaber, K. (1995). The SCRUM development process. *OOPSLA '95 Business Object Design and Implementation Workshop*

Zurcher, F.W. , Randell, B. (1968). Iterative Multi-Level modeling: A methodology for computer system design, *In Proceedings of the IFIP Congress, volume: 2*, 867-871

Webb-dokument

Agile alliance
http://www.agilealliance.org
[2007-04-15]

A Survey of System Development Process Models
http://www.ctg.albany.edu/publications/reports/survey_of_sysdev/survey_of_sysdev.pdf [2007-05-22]

Everette, R. Keith. (2002) Agile Software Development Processes: A Different Approach to Software Design—
http://www.agilealliance.org/system/article/file/1099/file.pdf
[2007-05-22]

Extreme Programming
http://extremeprogramming.org
[2007-05-22]

Kinnula, A. (2001). Software process engineering systems: models and industry cases. University of Oulu, http://herkules.oulu.fi/isbn9514265084/
[2007-05-22]

Martin, R. (2001). Agile Processes.
http://www.objectmentor.com/resources/articles/agileProcess.pdf
[2007-05-22]

Manifesto for Agile Software Development.
http://agilemanifesto.org
[2007-05-22]

METHODS & TOOLS
http://www.methodsandtools.com/PDF/dmt0402.pdf
[2007-05-22]

Paulk, M., Weber, C., Garcia, S., Chrissis, M., Bush M. (1993). Key Practices of the Capability Maturity ModelSM, Version 1.1.
http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr25.93.pdf
[2007-05-22]

Rapid Application Development – RAD
http://www.blueink.biz/RapidApplicationDevelopment.aspx
[2007-05-22]

Xprogramming
http://xprogramming.com/
[2007-05-22]

Personliga

Berquist,M. (2006). Föreläsning kurs, Informatik som vetenskap 5p. Institutionen för tillämpad informatik, IT-universitetet

8. Bilagor

Bilaga 1: Dagboksanteckningar från projektmedlemmar (Materialet finns hos författarna på grund av sekretess skäl)

Bilaga 2: Intervju frågor och intervju material (Materialet finns hos författarna på grund av sekretess skäl)