Handelshögskolan
VID GÖTEBORGS UNIVERSITET
Institutionen för informatik

2006-06-05

Ett förslag till hur Volvo IT kan optimera sina processer för systemutveckling

Abstrakt

Utveckling sker överallt i samhället idag och processer för systemutveckling utgör inget undantag. Denna studie undersökte de positiva och negativa företeelser som upplevdes vid användandet av Rational Unified Process respektive Agile Software Development. Processerna hade tillämpats i vardera två systemutvecklingsprojekt utförda på Volvo IT. De processområden som fokuserades på var projektplanering, kravhantering, arkitektur och testhantering. Studien genomfördes genom att intervjua medlemmar från de båda projekten. Grundat på den information som samlades in, analyserades förslag fram på hur Volvo IT skulle kunna optimera sina processer för systemutveckling i framtiden. Resultatet visar på att det behövs tydliga checklistor från projektstyrningsmodellen och en mer lättrörlig metod, än den RUP-version som används idag. Det är också viktigt att förankra systemutvecklingsprocessen i organisationen, vilken process som än används. Vidare leder ett aktivt kunddeltagande i processen oftast till ett system som mer motsvarar kundens förväntningar. Individen har en fundamental roll i systemutvecklingen, det räcker inte med endast en bra utvecklingsprocess.

Nyckelord: Systemutvecklingsprocesser, Rational Unified Process, Agile Software Development, CMMI

Författare: Jenny Elison och Malin Strömblad
Handledare: Kjell Engberg
Magisteruppsats, 20 poäng

Tack!

Vi vill börja med att rikta ett varmt tack till Anders Jonsson, vår handledare på Volvo IT, för hans enorma hjälpsamhet och tålamod genom hela den här studien.

Vi vill även tacka våra respondenter för att de så vänligt ställt upp och deltagit i vår undersökning.

Självklart vill vi tacka vår eminente handledare på informatik, Kjell Engberg.

Vårt största tack riktar vi ändå till stöttepelarna där hemma, Jennys Robert och Malins Kalle, Tilde och Albin.

Sist men inte minst vill vi tacka varandra för en arbetsam men framförallt rolig tid.

Maj 2006

Jenny Elison & Malin Strömblad

Inledning

I det inledande kapitlet har vi för avsikt att introducera läsaren till problemområdet och ge en bakgrund till den valda problemställningen. Vi redogör för frågeställningen samt de avgränsningar som ansetts lämpliga för uppsatsen. Vi avslutar med att presentera vår målgrupp.

I mångt och mycket finns en drivkraft att hela tiden vilja nå längre, en strävan att bli bättre. Utveckling sker oupphörligt runt omkring oss på ett eller annat sätt. Processer för systemutveckling utgör inget undantag. Vi har sett, genom systemutvecklingens förhållandevis korta historia allt från trial-and-error, vattenfallsmetoden till iterativa metoder och nu Agile Software Development[1]. Vi påstår att syftet med programvaruintensiv systemutveckling är att stödja och underlätta för effektiv utveckling av kvalitativa system i tid och till budget. Problemet med att inte lyckas leverera kvalitet i tid verkar dock alltjämt kvarstå. ComputerSweden uppmärksammar problemet i en artikel med att så mycket som 72 procent av de IT-projekt från svenska, privata och offentliga verksamheter under året 2005, var att betrakta som misslyckade. (Haldén, 2006; www.projketplatsen.se)

I var och varannan populärvetenskaplig tidskrift kan man läsa om varierande former av metoder för systemutveckling och ämnet verkar alltid lika aktuellt och diskussionsbenäget (Haldén, 2006; Ljadas, 2006; Tallungs, 2006; Boehm och Turner 2003). Olika metoder har just sina hängivna anhängare som övertygat hävdar att den metod de förespråkar är den bästa. Flertalet undersökningar påvisar ökad popularitet av lättrörliga metoder i en värld där de mer traditionella metoderna, exempelvis plandrivna, fortfarande står sig. (Nerur, Mahapatra & Mangalaraj, 2005; Boehm, 2002). I en djungel av metoder är Rational Unified Process (RUP) ett välkänt och ofta etablerat koncept, vilket vanligen klassificeras som en så kallad tungvikts process. Plandrivna metoder beskrivs som definierade processer med tillvägagångssätt som involverar arbetsuppgifter, planering av milstolpar och strategier som involverar bland annat kravhantering, design och arkitektonisk planering (Boehm, 2002). RUP innehåller samtliga dessa begrepp.

En allt snabbare förändringstakt av IT och en avpersonifierad effekt av den detaljerade plandrivna utvecklingen, skriver Boehm (2002) är några anledningar till att den nya generationen utvecklare framhåller behovet av att återuppväcka utvecklingsprocesser som lyfter fram det mest väsentliga i en utvecklingsprocess. Alltmedan *”traditionalister förespråkar omfattande planeringar, fastställda processer, och ett rigoröst återanvändande för att göra utvecklingen till en effektiv och förutseende aktivitet som successivt utvecklas åt perfektion.”* (Boehm, 2002)

Boehm och Turner (2004) menar att lättrörliga och plandrivna metoder generellt har setts som motpoler till varandra. De menar dock att båda metoderna har sina för- och nackdelar och att man aktsamt kan kombinera de två. Boehm (2002) anser att RUP, och andra risk- och spiraldrivna

[1] Agile Software Development, kan på ett enkelt sätt beskrivas som ett samlingsnamn för mer eller mindre lättrörliga metoder som är anpassade för förändringar från omvärlden. (www.agilesweden.org) Hädanefter benämner vi begreppet Agile Software Development för lättrörlig metod eller Agile.

metoder, kan förse riktlinjer för hur man hittar en god balans av disciplin och flexibilitet, mellan plandrivna och lättrörliga metoder.

Martin (2003) menar att Agile är ett modeord i moderna systemutvecklingsprocesser, men att många inte riktigt vet vad det innebär som begrepp i sammanhanget. Detta resulterar i att ordet missbrukas och fördelen tas av den positiva laddningen innebörden fått. Det är en fara i det, menar Martin (2003), då Agile har börjat användas som prefix till mycket, vilket kan göra begreppet Agile meningslöst och konceptet kan få helt nya innebörder. Skribenten menar också att de verksamma inom branschen skall vara vaksamma för nya koncept som bär prefixet Agile. (Martin, 2003)

Det finns de som menar att för att vara ett konkurrenskraftigt företag i branschen, måste förståelse fås för att processer och tekniker endast är ett verktyg för att nå fram till en produkt. Det som har störst inverkan på ett lyckosamt projekt är individerna bakom, det vill säga de som nyttjar dessa processer och tekniker. (Martin, 2003) Martin menar vidare att ett lyckosamt projekt bygger på förmågan att skapa samarbetsvilliga och självorganiserade team. Om ett företag kan uppmuntra till en sådan kultur ökar förutsättningarna att få ett konkurrenskraftigt övertag.

Det är inte bara olika metoder för systemutveckling som har framställts genom åren. Även olika kvalitetsramverk för att förbättra affärsverksamheten har konstruerats (Anthes, 2004; Hoffman, 2005; Danielsson, 2004). Efter några år av stagnation börjar det nu åter bli viktigt att sätta en standard för kvalitet på processer och program (Wallström, 2005). För systemutveckling har bland andra *Capability Maturity Model Integrations*, CMMI, skapats för att utvärdera förmågan att utveckla programvara samt ge stöd för hur en organisation skall fokusera för att bli bättre i sina processer för systemutveckling. (www.sei.cmu.edu/cmmi/)

Problembakgrund

Sedan 1999 har RUP varit den officiella processen för programvaruutveckling inom Volvo IT. Förutom RUP som utvecklingsprocess finns en projektstyrningsmodell, Project Control Model (PCM) som måste följas. PCM är utvecklad av Volvo IT och samverkar med RUP.

Avdelningen Application Development Techniques, ADT på Volvo IT är ansvariga för att ge support kring de utvecklingsprocesser och verktyg som fastställts enligt IT Governance, som ansvarar för att fastställa riktlinjer och policys kring processer, verktyg et cetera. Volvo IT kan idag se att införandet av RUP inte lyckats till fullo då processen inte använts i den utsträckning som var förväntat. Detta beror bland annat på att Volvo IT har expanderat exempelvis i USA och Frankrike genom att AB Volvo förvärvat andra bolag samt att RUP delvis sannolikt använts på ett felaktigt sätt. ADT vill därför undersöka uppfattningen och attityderna kring RUP och lättrörliga metoder.

För att effektivisera och optimera den del av Volvo IT som sysslar med utveckling och förvaltning av applikationer har man inom Volvo IT:s styrelse och ledning fattat ett beslut om att starta ett Software Process Improvement (SPI) – program, ett program för att förbättra sättet man bygger och förvaltar applikationer. Volvo IT:s målsättning är att med hjälp av SPI-programmet, i ett första skede nå till nivå tre på CMMI:s mognadsmatris. Detta beräknas ske inom en tre till femårsperiod. Ett gemensamt arbetssätt, en gemensam terminologi, gemensamma

verktyg et cetera är några aspekter som måste till för att nå nivå tre, vilket, enkelt uttryckt innebär att man måste ha väl definierande processer.

Problemformulering

Huvudfrågan i denna uppsats är:

Hur kan Volvo IT optimera sina processer för systemutveckling för framtida bruk?

För att lättare kunna få svar på denna fråga har vi till vår hjälp haft tre underfrågor.

- Vad har upplevts som positivt respektive negativt i användandet av RUP respektive en lättrörlig metod?
- Skulle man kunna kombinera de positiva aspekterna från RUP respektive en lättrörlig metod för att få fram mer optimala processer?
- Hur svarade projekten upp till hanterandet av de olika arbetsområdena mot en skala på CMMI?

Genom att fördjupa oss i ett RUP projekt och Agile Software Development projekt, har vi försökt att besvara ovan nämnda frågor.

Avgränsningar

I denna uppsats har inte avsikten varit att generellt jämföra RUP och Agile Software Development som processer. Jämförelsen grundar sig snarare i hur de båda processerna har använts i projekt på Volvo IT och hur resultatet av dessa i framtiden kan tillämpas på likvärdiga projekt inom Volvo IT av samma storleksgrad. Vi skall heller inte redogöra för varför RUP inte fungerar fullt ut idag på Volvo IT. Vi kommer på grund av vår begränsade tid, 20 veckor, samt utifrån uppdragsgivarens önskan inte att studera processernas samtliga områden. De områden som skall studeras är projektplanering, kravhantering, test, och arkitektur. Även i vårt mätinstrument CMMI har vi endast valt ut delar som stämmer överrens med de områden som vi valt att undersöka. Man skall även ta i beaktande att vi inte gått på djupet i detta studium utan gjort en så enkel bedömning som möjligt, vilket innebär att vi ställt utvärderingen mot de inledande styckena ”Purpose” och ”Introductory Notes” för de processområden vi valt. (Appendix D)

Målgrupp

Uppsatsen skrevs på uppdrag av ADT och vänder sig i första hand till Volvo IT och de personer som arbetar med utveckling av mjukvara. Därför förutsätts att fundamental kunskap innehas av läsaren avseende systemutveckling och dess processer. I uppsatsen utesluts därför fördjupad beskrivning av vissa centrala begrepp. Vid intresse eller behov vänligen se referenslista för ytterligare information.

Metod

I detta kapitel har vi för avsikt att redogöra för och motivera uppsatsens angreppssätt och insamlingsmetoder. Vidare beskrivs utformning, genomförandet av intervjuer, urvalsdiskussion, hur det insamlade materialet analyserats, samt slutligen en utvärdering av undersökningsmaterialet. Syftet med att redogöra för valet av metod och angreppssätt är att ge läsaren möjlighet att upprepa studien för en kontroll samt för trovärdigheten i uppsatsen (Backman, 1998).

Angreppssätt

Metoder är ett verktyg för att hjälpa forskaren att kartlägga och tolka det problemområde som skall undersökas, att försöka förstå miljön som denne verkar i. Genom det valda tillvägagångssättet skall man komma fram till en lösning - syftet med uppsatsen. (Holme & Solvang, 1997)

Vi har studerat två likvärdiga systemutvecklingsprojekt, likvärdiga i den meningen att de var lika stora vad gäller antalet medlemmar och grad av framgång. I det ena projektet användes RUP som systemutvecklingsprocess. I det andra projektet arbetade man efter en egendesignad process med lättrörliga principer som underlag. De processområden som studerades var projektplanering, kravhantering, arkitektur och testhantering. Genom att intervjua representanter från de olika projekten kunde vi analysera fram de styrkor och svagheter som upplevdes vid användandet av processerna. Vi valde att använda CMMI som ett mätinstrument för att granska projektens hantering av de olika processområdena. Vi valde de processområden på CMMI:s skala som passade bäst för de i projekten undersökta processområden. (Appendix D) Utifrån de valda processområdena kunde vi mäta projektens respektive mognad på respektive processområdes nivå på CMMI:s skala över processmognad. Därefter kunde vi föra fram förslag om hur Volvo IT skulle kunna optimera sina processer för systemutveckling.

Kvalitativt och kvantitativt är två olika angreppssätt för att systematiskt inhämta kunskap för att försöka förstå den omvärld man ämnar undersöka. Det kvantitativa perspektivet, även kallat deduktion, utgår från en teori om omvärlden och försöker sedan bevisa att dess antaganden är sanna eller falska. Det kvalitativa perspektivet, även kallat induktion drar generella slutsatser från erfarenheterna. Omgivningen betraktas subjektivt och intresset ligger ofta i hur individer tolkar, upplever och strukturerar dess omgivande verklighet i relation till sina tidigare kunskaper och erfarenheter. (Backman, 1998; Holme & Solvang, 1997) Många kvalitativa studier använder sig av fallstudier. En fallstudie liksom den kvalitativa strategin undersöker en speciell företeelse, till exempel en organisation eller ett system. En induktiv slutsats är aldrig hundra procent säker, men gäller till dess motsatsen är bevisad. Vår studie gick ut på att tolka våra respondenters upplevelser av att arbeta utefter en viss process och inom valda disciplinområden och granska det insamlade materialet och på så sätt med hjälp av informationen komma fram till våra egna slutsatser och teorier. Vi har därför valt att utföra vår studie utifrån det kvalitativa perspektivet med hjälp av en fallstudie.

Studier som huvudsakligen drar sina slutsatser med hjälp av siffror och statistik brukar kallas kvantitativa. Undersökningar som däremot har för avsikt att beskriva och skapa förståelse för ett problem genom att undersöka och analysera data som inte kan uttryckas numeriskt brukar kallas kvalitativ. (Backman, 1998; Holme och Solvang, 1997) I vår uppgift ingick att tolka våra respondenters upplevelser av valda områden, data som inte gick att presentera i sådan karaktär att den enligt vår mening inte gick att omsätta numeriskt. Därför bedömde vi att det kvalitativa angreppssättet passade bäst till vårt arbete. Det gav oss också en möjlighet att gå på djupet för att få en uppfattning och skapa oss en förståelse över det valda problemområdet. (Backman, 1998; Holme & Solvang, 1997)

Optimalt hade dock varit att göra en etnografisk studie, där vi skulle ha kunnat delta i de båda projekten och på så sätt betraktat fenomenet och utifrån de upplevelserna dragit våra egna slutsatser. Det som omöjliggjorde en etnografisk studie var den förmodade tidsåtgången och att det vid denna tidpunkt inte fanns projekt med angivna förutsättningar att studera.

Mätinstrument

Vi har, som tidigare nämnts, valt att i vår studie använda The Capability Maturity Model Integration (CMMI) som ett instrument för utvärdering över projektens hantering av de olika processområdena. Det finns två dimensioner av CMMI, Staged och Continuous, varav det är Staged som vi har för avsikt att fokusera på. Valet av det perspektivet beror på att den dimensionen är särskilt lämpad för att starta en förbättring av processens mognad. Staged baseras på Software CMM som är en väldefinierad och beprövad vägledning, dessutom en framgångsrik sådan med bevisat resultat. (Ahern, Clouse & Turner, 2001)

Datainsamling

Sekundärdata

Sekundärdata samlades in innan och parallellt med studien i form av vetenskapliga artiklar genom databaser via Internet och i form av böcker inom ämnet. Databaserna som användes för att söka var AMC, IEEE och Science Direct. När vi via databaserna sökt efter sekundärdata i form av vetenskapliga artiklar var ämnesorden: Agile, Agile Software Development, RUP, Rational Unified Process, CMMI, Software Development, systemutveckling samt Software Engineering. Samma begrepp har genomgående använts för att söka efter böcker på bibliotek samt artiklar i populärvetenskapliga tidskrifter. För att undersöka om vårt problemområde hade ett visst allmänt intresse har vi använt oss av de populärvetenskapliga tidskrifterna ComputerSweden, Computer, ComputerWorld, Software Development Magazin och Cutter IT Journal. Vissa vetenskapliga artiklar har i sin tur givit oss vägledning till intressanta referenser i form av ytterligare sekundärdata som vi har fördjupat oss i. Vi har även sökt i Google och Scholar Google.

Primärdata

Primärdatan har, som den kvalitativa metoden inbjuder till, samlats in med hjälp av intervjuer. Vi började vår studie med två stycken informantintervjuer för att få en överblick över vad de båda processerna vi avsåg att studera stod för. För att få information om arbetet i projekten som vi ämnade undersöka, har vi intervjuat deltagare i projekten. Vi har för detta använt oss av två olika typer av intervjuer, besöksintervju och telefonintervju. Nio stycken semistrukturerade intervjuer genomfördes. Intervjutillfällena var en till två timmar långa, sju stycken av dessa var

besöksintervjuer och en var telefonintervju. Mejlkorrespondens har också förekommit regelbundet. Intervjun med en av systemutvecklarna från Polen sköttes helt via mejl då vi inte hade någon möjlighet att träffa honom på grund av det långa avståndet.

Informantintervju
I en informantintervju talar man med en person som har förstahandskunskap om den företeelse som studeras. Denne står utanför händelsen men har mycket att säga om den. Informant kan även kallas för en slags ”ersättningsobservatör”. (Holme & Solvang, 1997; Andersen, 1998)

Besöksintervju
Fördelen med besöksintervju är att man får en personlig kontakt med respondenten, öga mot öga (Holme & Solvang, 1997). Detta medför att intervjuaren får möjligheten att ha en god kontroll över vem som svarar på frågorna, likaså minimeras risken för svarsbortfall. De nackdelar som finns med denna typ av intervju är att kontrollen av miljön är begränsad, respondenten har ingen chans att vara anonym för de som intervjuar och metoden kräver mycket tid för bearbetning. (Lekvall & Wahlbin, 2001) Vid de åtta besöksintervjuerna som utfördes närvarade båda författarna vid samtliga tidpunkter. Detta för att vi skulle få en så bra bild som möjligt över situationen. En av oss agerade huvudintervjuare. Dennes uppgift var då att till största del hålla i konversationen undertiden som den andre personen kunde göra iakttagelser och anteckna eventuella uppföljningsfrågor.

Telefonintervju
En telefonintervju gjordes för denna studie. Fördelen med telefonintervju är att den kräver mindre resurser och mindre tid än en besöksintervju. Den går även förhållandevis snabbt att utföra utan att så mycket av den personliga intervjuns goda egenskaper förloras. De nackdelar som finns med denna typ av intervju är att man inte får ögonkontakt med respondenten, och då förloras möjligheten till att kunna läsa av respondentens kroppsspråk et cetera. Likaså förloras en viss del av det sociala förtroendet som vanligtvis uppstår vid en omedelbar konfrontation med respondenten. (Lekvall & Wahlbin, 2001) Till intervjun använde vi en högtalartelefon för att försöka få fram den diskussion vi ville ha snarare än en strikt intervju där moderatorn ställer en fråga som sedan respondenten ger ett kort svar på. Vi tror att detta medförde att vi kom respondenten närmare och fick den sorts svar vi var ute efter. Även vid denna intervju agerade en person huvudintervjuare och ledde intervjun varpå den andre personen antecknade stolpar och nya frågor som kom upp.

Utformning av intervjuer
En intervjuguide utformades till varje disciplinrepresentant som skulle intervjuas. Vi var noga med att i förhand inte ge respondenterna några frågor från denna guide då vi ville ha en så öppen intervju som möjligt. Enligt Kvale (1997) är det viktigt när man använder intervjun i forskningssammanhang att försöka bygga upp en atmosfär så att respondenten känner sig tillräckligt trygg för att kunna tala fritt om sina erfarenheter och tankar. Detta har vi tagit hänsyn till genom att hålla intervjuerna på respondenternas respektive arbetsplats alltså en miljö som de känner väl. Vidare bokades vissa intervjuer av vår uppdragsgivare och vissa av oss själva. Detta gjordes via mejl där vi kortfattat beskrev vad vår studie gick ut på och varför vi önskade intervjua just denne. Samtliga intervjuer spelades in med hjälp av diktafon och sammanställdes därefter.

Varje intervju startades med att vi, än en gång, kortfattat förklarade vårt syfte med intervjun och förhörde oss om att det gick bra att spela in intervjun och referera till respondenten med namn i uppsatsen.

Urval

Som nämnts tidigare var vår studie ett uppdrag av Volvo IT. Tillföljd av detta fick vi tilldelat oss de två projekt som skulle studeras. De respondenter som var av intresse för oss var därför medlemmarna i de båda projekten. När man gör en sådan bedömning av vilka personer som kan vara intressanta för undersökningen kallas det för bedömningsurval. Denna bedömning kan grunda sig på erfarenhet från det aktuella forskningsområdet. (Lekvall & Wahlbin, 2001) Vår handledare på Volvo IT kontaktade de båda projektledarna vilka var positiva till att medverka som respondenter i studien. Av projektledarna fick vi även förslag till övriga medlemmar i respektive projekt som kunde vara av intresse att intervjua. På så sätt visste vi att respondenterna var av rätt karaktär då valet av undersökningspersoner är väldigt viktigt. Består urvalet av fel personer kan hela undersökningen bli oduglig i relation till den utgångspunkt undersökningen hade från början (Holme & Solvang, 1997).

Metod för analys

Enligt Holme och Solvang (1997) är det viktigt att redan innan studien påbörjas, veta vilka problemområden som kommer att utredas. Vi kartlade därför vad vår uppdragsgivare ville få utrett, definierade problemområdet och syftet för uppsatsen och sökte reda på ämnets aktualitet och omfattningen på forskningen inom området. Därpå fortsatte vår studie med att söka rätt på ett urval av redan etablerade teorier samt synsätt som fanns i omlopp inom systemutveckling för att på så vis se de olika uppfattningar som fanns. Vi utformade intervjumallar med de fyra processområdena (projektplanering, kravhantering, arkitektur och testhantering) i åtanke. Efter att intervjuerna var genomförda och utskrivna strukturerade vi upp materialet utefter dessa områden. Detta för att få en överblick och kunna analysera materialet lättare då det material som belyser samma processområde, fanns under en gemensam rubrik. På så vis kunde en mer komplett bild av materialet tas fram och frågeställningarna kunde belysas utifrån olika synvinklar. (Holme & Solvang, 1997). Den då skapade empirin ställde vi sedan mot den teoretiska referensram vi skaffat oss och kunde på så sätt analysera resultaten för att därpå generera våra egna hypoteser i uppsatsens diskussionskapitel.

Utvärdering

Reliabilitet och validitet

I den kvalitativa undersökningen har reliabilitet och validitet inte det fokus som de har i den kvantitativa studien. (Holme & Solvang, 1997) Vissa forskare inom området forskningsmetodik menar också att det är svårt att uppnå hög reliabilitet då det är svårt att upprepa resultatet av en undersökning i en kvalitativ studie. Men Holme och Solvang (1997) menar att det inte är det primära problemet, snarare skulle det vara svårt att återskapa mötesförloppet med respondenterna.

Vi har flera gånger ställt oss frågan huruvida den information vi har samlat in är trovärdig och huruvida vår undersökning är tillförlitlig. Vad det gäller respondenterna anser vi att de varit representativa och besuttit stor kunnighet för området som studerats. De har dessutom fått god tid på sig på intervjuerna för att ge så korrekta och fullständiga svar som möjligt. Det är svårt att uppnå konstans i en längre tidsaspekt då vår undersökning bygger på kvalitativa intervjuer där vi

inte kan förutsätta att våra respondenter är statiska i sin attityd till ämnet. Vi individer, är alla mer eller mindre aktiva och receptiva vilket gör att de perspektiv vi utgår från är föränderliga. Dock kan det förefalla som dess processer inte är en process då förändringen sker så pass långsamt att vi då får uppfattningen om att de är statiska. (Trost, 2005) I ett kort tidsperspektiv vågar vi utifrån detta resonemang anta att våra respondenter är konsekventa till sitt synsätt på det fenomen vi studerat. Vissa respondenter har vi intervjuat mer än en gång vilket har givit oss möjlighet till att kontrollera ett svar. De har själva dessutom haft möjligheten att kontrollera tillförlitligheten i den information som sammanställts (Holme & Solvang, 1997).

Vår studie hade i det närmaste kunnat nå en mycket hög nivå av reliabilitet förutsatt att alla projekt på Volvo IT skulle ha samma yttre och inre förutsättningar. Men ett projekt anser vi, är en skiftande företeelse. Med det menar vi att projekten av de slag vi studerade, aldrig kan vara det andra likt på grund av de skilda omständigheter som råder. Dessa omständigheter kan vara, varierande slag av kunder, olikartade team (dels till antalet och dels till grad av erfarenhet), tid, ekonomi et cetera. På samma sätt tillämpas dess processer olika beroende på yttre och inre omständigheter. Därtill är RUP ett processramverk som bygger på att processen skall skräddarsys och tillämpningen blir med andra ord olika sätt i olika projekt. I det Agile Software Development utgår man istället från ett manifest och ett antal principer. På så sätt kan man enkelt uttrycka att även den lättrörliga metoden kan skräddarsys beroende på vilket projekt som den skall appliceras på, även om där finns olika former av metoder såsom XP, Scrum et cetera. I det fallet vi har titta på har processen formats och influerats av olika lättrörliga synsätt.

Uppgiften från våra uppdragsgivare var att studera två stycken projekt som var likvärdiga. Dessa båda projekten mätte denna grad av likvärdighet som önskades. Detta genom att; de grundade sig på samma teknik (Java/Cobol), de hade samma kund (Volvo Parts), de var av samma storlek och längd (cirka 7 stycken, sex månader) och de hade samma grad av framgång. Utifrån dessa argument anser vi att reliabiliteten på studien dock blir relativt hög.

Hand i hand med reliabilitet går validitet, bara för att reliabilitet är hög innebär inte det att validiteten är hög. Man måste vara säker på att det man mäter är det man avser att mäta, mätningen måste vara pålitlig (Patel & Davidson, 1998). Vi har intervjuat deltagare med olika roller från de båda projekten och ställt, tycker vi, relevanta frågor för att kunna få svar på vårt problem. Vi har båda två deltagit vid samtliga intervjuer och har tack vare detta kunnat diskutera och analysera intervjuerna tillsammans i efterhand.

Källkritik

Enligt Patel och Davidsson (1998), är det viktigt att förhålla sig kritisk till dokumenten för att kunna göra en bedömning om fakta eller upplevelser är sannolika. Detta tog vi hänsyn till genom att försöka kombinera litteratur, artiklar, intervjuer och egna erfarenheter och i möjligaste mån försöka sålla partiska uppfattningar från fakta. Den informationen som var mest frekvent vägde tyngst.

Vi har försökt att välja ut de teorier som passar vårt studieområde bäst. Det har varit svårt att hitta artiklar som endast handlar om RUP. Samtidigt har det varit förhållandevis enkelt att hitta artiklar om lättrörliga metoder, varför vi efter mycket diskussion, resonerande och sållning försökt att få med så för studien relevanta åsikter som möjligt.

Vad anbelangar uppsatsförfattarnas subjektivitet vill vi ge reservationer. Risk kan finnas att personlig information, exempelvis efter besök på Volvo IT, kan ha färgats av individuella åsikter. Vi har emellertid försökt beskriva synpunkterna och erfarenheterna på ett så objektivt sätt som möjligt, genom att vi kontinuerligt fört en diskussion kring subjektivitetsproblemet.

Teoretisk referensram

I detta kapitel har vi för avsikt att redogöra för de teorier vi utgår från i vår undersökning. Vi börjar med att presentera olika resonemang kring de processkategorier vi studerat. Därefter redogör vi för några grundläggande modeller inom systemutvecklingen som kombineras i Rational Unified Process eller Agile Software Development. Vi fortsätter kapitlet med att ge en beskrivning av RUP och en övergripande beskrivning Agile Software Development med utgångspunkt i det manifest och de principer som fastställdes 2001 av the Agile Alliance[1]. Avslutningsvis ger vi en kort beskrivning av Capability Maturity Model Integration, det mätinstrument vi använt oss av.

Teoretiska aspekter

Boehm och Turner (2004) anser att RUP traditionellt setts som en plandriven och en så kallad tung process. Däremot innehåller RUP-filosofin också utmärkande lättrörliga egenskaper. Men, fortsätter författarna, dessa särdrag från lättrörliga metoder har överskuggats av den överlag omfattande detaljerade processen. Plandrivna metoder är ofta omfattat utvecklade och inkluderar flera olika sorters tillämpningar för utveckling vilket resulterar i att dessa måste skalas ner för användning. Vidare menar författarna att det är svårt för gemene man att skala ner en omfattande process och en konsekvens av det är att processen felaktigt tillämpas fullt ut. Boehm och Turner (2004) menar att lättrörliga metoder erbjuder en bättre förutsättning och utgångspunkt för att skalas upp.

Fowler (2005) menar att den kritik som oftast förekommer mot plandrivna metoder beror på att de är för formalistiska och på grund av de många moment som skall följas minskas utvecklingens effektivitet. Som en reaktion på dessa metoder utvecklades lättrörliga metoder som, enligt författaren, är ett försök till att finna en balans mellan *”ingen process och för mycket processer”* för att erbjuda en användning av processer som ger ett rimligt stöd. Om en process används alltför strikt finns risken att innovation stävjas eller leder till en mekanisk ”*checklistsmentalitet*” vilket resulterar i att processen får en större fokusering och produkten riskerar att komma i andra hand. (Boehm & Turner, 2004) Tunga processer, menar författarna, ökar risken för att folk fastnar i processen med att generera dokumentation istället för att agera systemutvecklare.

Larman (2003) menar att lättrörliga metoder verkar generellt för principer som avspeglar en mer lättviktig anda än dess äldre föregångare såsom RUP med flera, men med rätt insikt kan även dessa användas på ett mer lättrörligt inspirerat sätt. IID[2] metoder helt utan rörlig, *agility*, föreställning i någon mån, kan dock vara sällsynta att hitta i praktiken. (Larman, 2003) Larman skriver vidare om att kategorisera metoder dels utefter en definierad process och dels utefter en empirisk process. Lättrörliga metoder, menar författaren, faller under empiriska metoder och lämpar sig bra till osäkra och ofta föränderliga projekt. De definierade processerna passar bättre

[1] Agile Alliance är ett nätverk som kom att formas av en grupp ”gurus”. Gemensamt utvecklar och utbyter de erfarenheter och arbetssätt som grundar sig i de lättrörliga metoderna. *The Agile Manifesto* ses som ett riktmärke för lättrörliga metoder. (www.agilesweden.org)

[2] Iterative och Incremental Development (IID)

till projekt med motsatta förutsättningar det vill säga förutsägbara och stabila projekt. (Larman, 2003)

Boehm och Turner (2004) skriver att både lättrörliga och plandrivna processer har sådana egenskaper att var och en lämpar sig bäst till olika typer av projekt. Till de projekt som har karaktär av både lättrörlig och plandriven approach är det en nödvändighet och fullt möjligt att kombinera båda metoderna till en hybrid. (Boehm, 2002) För att fastställa den bästa balansen av de två metoderna, skriver författarna vidare, att en riskanalys över projektets karaktäristiska kontra en given metods specifika kännetäcken kan vara behjälplig. I framtiden kommer det även stå högt i kurs om man som företag kommer att kunna tillämpa metoder som klarar av att kombinera lättrörlighet och disciplin utefter ett projekts karaktäristiska. (Boehm & Turner, 2004)

I sin artikel ”The new Methodology” beskriver Fowler (2005) de förutsättningar som han tycker borde finnas i ett företag för att börja arbeta mer lättrörligt. Fowler menar att eftersom det lättrörliga synsättet ser ett tätt samarbete mellan projektmedlemmarna som fundamentalt, är det nödvändigt att alla i gruppen är motiverade att jobba enligt lättrörliga principer. Författaren belyser vikten av kundkontakten när man jobbar lättrörligt, därför är det viktigt att även kunderna är samarbetsvilliga och deltar i projekten. Finns det inga samarbetsvilliga kunder är det svårt att se fördelarna med en lättrörlig metod till fullo. Vidare hävdar författaren att en lättrörlig metod med få steg att följa är lättare att ta till sig än en formalistisk tungviktsmetod. Det som Fowler emellertid anser vara det mest väsentliga är dock att en mentor finns tillgänglig för projekten som skall börja tillämpa ett lättrörligt synsätt. Vid sådana tillfällen kan mentorn hjälpa projekten att undgå nybörjarmisstag.

Jacobson (2002) beskriver behovet av att ha en mentor vid sidan av ett projekt, och en mentor som har god erfarenhet av att anpassa en process utefter det specifika projektets karaktär. Problemet är dock, vilket Jacobson belyser, att även kunniga och skickliga mentorer är svåra att hitta. Det finns dessutom alltid risker med att ha mentorer då det vid tidsbrist lätt blir att denne får andra ansvar än just mentor eller risken att den som tar på sig rollen som mentor applicerar sin egen processteori vilket, hävdar Jacobson, slår ner effektiviteten i projekten. (Jacobson, 2002)

Brooks (1995) beskriver i sin essä ”No Silver Bullet” från 1986 att programvaruprojekt har en förmåga att växa till ett monster på grund av förseningar, överskridande av budgetar och svaga produkter. Många önskar sig därför en mirakulös lösning, så kallad *”Silver Bullet”*, till att genomföra ett projekt och undvika dessa motgångar. Slutsatsen är enligt Brooks att det inte finns eller i framtiden kommer att finnas några magiska fenomen som löser de svårigheter man ställs inför i ett programvaruprojekt.

Brooks presenterar i essän de svårigheter som han anser ligger till hinder för programvaruteknikens utveckling. Han delar in svårigheterna i två kategorier, de essentiella, *essence*, och de tillfälliga, *accidents*, svårigheterna. De essentiella svårigheterna är de inneboende svårigheter som ligger i programvarans natur, vilka enligt Brooks är komplexitet, anpassning, föränderlighet och osynlighet.

Författaren avslutar med att belysa de tillvägagångssätt som kan angripa de essentiella problemen i programvaruutveckling. Han föreslår att man skall undvika att konstruera något man likaväl kan köpa. Undersök därför noga vad marknaden har att erbjuda. För att sedan etablera systemets kravbild förordas, som en del i iterationerna, att bygga snabba prototyper. Istället för att bygga

system, menar Brooks att programvaran metodiskt skall växa fram genom att tillföra funktion för funktion allteftersom de körs, används och testas. Sist men inte minst pekar författaren på vikten av att inneha skickliga designers inom företaget som en nödvändighet för att med relativt liten insats konstruera smarta, enkla och rena strukturer. Författaren menar härvidlag att det är betydelsefullt att identifiera och utbilda skickliga designers i den kommande generationen. (Brooks, 1995)

Vattenfallsmodellen
Vattenfallsmetoden är en linjär utvecklingsprocess vilken involverar en sekvens av aktiviteter där påbörjad aktivitet skall vara avslutad innan nästa tar vid. I praktiken överlappas dessa aktiviteter vilket bidrar till iterationsliknande beteende där delar av utvecklingsförfarandet avslutas, varvid man fortsätter med nästa aktivitet. Problemen skjuts därav på framtiden vilket kan resultera i kostsamma projekt där, i slutändan kunden, också i många fall inte får ett system som motsvarar dennes förväntningar och behov. (Kruchten, 2002; Sommerville, 2004) I små projekt, som pågår i några veckor eller månader och som är förutsägbara och stabila och utan större överraskningar är det fullt möjligt att utveckla utefter ett sekventiellt tillvägagångssätt. (Kruchten, 2002)

Iterativ och inkrementell utveckling
Iterativ och inkrementell utveckling innebär en utvecklingscykel som bygger på en mängd iterationer i ordningsföljd. Denna process möjliggör till att viktiga problem och risker uppdagas tidigt i utvecklingen och att man därmed kan hantera och förebygga dessa i god tid så att man slipper missa felaktiga antagningar sent, likt vattenfallsmetoden. Varje iteration är uppbyggd som en mindre vattenfallsliknande process, där vanligen de flesta aktiviteter i utvecklingens livscykel gås igenom. I slutet av varje iteration bör en intern release göras av systemet där man genomför test som ger feedback på det som producerats. En poäng med en iterativ process är att man går tillbaka för att göra nya värderingar av kraven, vilket leder till att systemet uppfyller kundens behov bättre än vattenfallsmodellen. (Kruchten, 2002; Larman, 2004) Kruchten (2002) menar att användaren oftast inte vet vad de vill ha förrän efter det att de sett produkten, vilket benämns som IKIWISI effekten. *”I Know It When I See It”*. Systemet utökas inkrementellt i nästföljande iterationer. Iterative och Incremental Development (IID) är ett annat namn på samma företeelse. (Larman, 2004)

Komponentbaserad utveckling
Komponentbaserad utveckling möjliggör för återanvändning och anpassning av komponenter från otaliga källor. (Kruchten, 2002) Angreppssättet bygger på en bas av återanvändbara programvarukomponenter och ett visst integrerad ramverk för komponenterna. (Sommerville, 2004) Fördelen med denna utvecklingsmetod är att leverans möjliggörs snabbare och den bidrar även till avsevärd minskning av utvecklingen av programvara, vilket leder till minskade kostnader och risk. Kompromisser av kraven är oundvikliga, då man i analysfasen letar efter komponenter som bäst motsvarar de krav man har och dessa modifieras sedan för att återspegla komponenterna. Detta medför att systemet inte kan leva upp till alla de viktiga krav som beställaren är i behov av. Ytterligare nackdelar kan vara att man förlorat kontrollen över nya versioner av komponenterna vilket leder till att viss kontroll av systemförändringen går förlorad. (Sommerville, 2004)

Rational Unified Process

Rational Unified Process, RUP, är en generell process för systemutveckling och är således avsedd för de flesta utvecklingsprojekt för programvara. RUP har en approach som bygger på vissa nyckelegenskaper och principer som bland annat iterativ utveckling, arkitekturcentrisk process, risk och användarfalldrivet. Processen baseras på en mängd viktiga erfarenheter dragna ur andra brukliga systemutvecklingsmetoder, främst objektorienterade sådana, vilka har koncentrerats till praxis. Med det menas ” … väl beprövade metoder, principer och processer som har visat sig vara framgångsrika i tidigare programvaruutvecklingsprojekt.”. (Lunell, 2003) RUP ger därav riktlinjer för hur man på ett tillförlitligt sätt skall tilldela och hantera de olika arbetsmoment som kan ingå i ett systemutvecklingsprojekt. (Kruchten, 2002; Lunell, 2003)

Det är viktigt att förstå att RUP är ett processramverk där varje projekt eller organisation kan anpassa och utöka RUP genom att plocka de bitar man anser är relevanta och som tillgodoser de behov projektet eller organisationen har. RUP är komplext och omfattar instruktioner för många olika slags systemutvecklingsprojekt och skall därför inte tillämpas fullt ut. (Kruchten, 2002; Lunell, 2003)

Processen är också en produkt som ständigt är under utveckling och uppdateringar görs kontinuerligt. Produkten har utvecklats av Rational Software men levereras och underhålls i skrivande stund av IBM. Till själva processen finns även sammankopplade verktygsstöd som även de tillhandahålls från IBM. (Kruchten, 2002; Lunell, 2003)

Praxis

Utveckla programvara iterativt

Detta angreppssätt är en av tyngdpunkterna i RUP och syftet är att kontinuerligt upptäcka, uppfinna och realisera. Varje iteration i RUP är cirka två till sex veckor och skall resultera i någon form av release som skall kunna utvärderas och granskas så att projektet vet dess status och att man följer rätt väg. Tillvägagångssättet går också ut på att utöka systemet efter varje iteration, det vill säga inkrementell utveckling. På detta sätt kan man tidigt verifiera att tänkta lösningar, såsom exempelvis arkitekturen fungerar. Förfarandet möjliggör också för processen att förbättras då projektteamet har chans att lära av eventuella misstag. (Kruchten, 2002; Lunell, 2003)

Hantera krav

Det iterativa tillvägagångssättet gör det möjligt att beakta kravförändringar. Ett krav är ett uttryck för de behov ett system skall uppfylla utifrån de olika intressenternas perspektiv. Kraven är inte beständiga utan det finns alltid olika omständigheter som gör att dessa är föränderliga med tiden. RUP tillhandahåller en mängd råd för hur man skall hantera kravförändringar, och hur prioritering av krav kan hanteras bland annat så att inte spårbarheten går förlorad. Där finns också vägledning för att fånga upp och identifiera kraven. Genom att också dokumentera krav genom exempelvis Use Cases eller Scenarios underlättas identifiering och kommunikation av kraven. (Kruchten, 2002; Lunell, 2003)

Komponentbaserad arkitekturcentrisk utveckling

Även denna praxis är starkt sammankopplad till det iterativa och inkrementella angreppssättet på så sätt att arkitekturen utgör en mycket viktig stomme och styrredskap för utvecklingsapproachen, då systemet stegvis skall växa fram. Ett programvaruintensivt system, skriver Kruchten (2002),

betraktas ur flera olika perspektiv och ett viktigt redskap för att visualisera och hantera dessa är dess arkitektur. En exekverbar arkitektur produceras i varje iteration och bygger dessutom på en komponentbaserad arkitektur där man skall kunna kombinera inköpta, existerande och nyutvecklade komponenter som skall integreras med varandra. Komponentbaserad arkitektur möjliggör också för återanvändning, definierar de olika utvecklingsgrupper, isolerar beroendet och förenklar underhåll av systemet. (Kruchten, 2002; Lunell, 2003)

RUP definierar arkitekturen i både tekniska och icke tekniska perspektiv. Att skapa en ritning till systemet, precisera samverkan mellan olika delar av systemet, fastställa principer för utbyggnad av systemet samt använda arkitektoniska mönster, är definitionen av de tekniska aspekterna, dessa kan kombineras för att utgöra en viss stil. Arkitekturens stil kan dokumenteras och användas som mall, en så kallad referensarkitektur. Definitionen av de icke tekniska perspektiven grundar sig delvis på 4 + 1 vymodellen i vilken produktens omgivning representeras i form av en vy på användningsfall. (Kruchten, 2002; Lunell, 2003)

Modellera programvara visuellt

En modell är en abstraktion av verkligheten, som i modellen kan beskrivas utifrån olika perspektiv. Visuell modellering hjälper till att hantera komplexiteten i programvaran och genom ett standardiserat modelleringsspråk visualiseras modellen på ett tydligt sätt. En tydlig design kan i sin tur bidra till att motstridigheter upptäcks snabbt. (Kruchten, 2002; Lunell, 2003)

Kontinuerlig kvalitetskontroll

Kontinuerlig verifiering av systemets kvalitet innan driftsättning minskar kostnaderna dramatiskt, för att åtgärda problem eller fel i programvaran. Kontinuerlig utvärdering och verifiering uppnås genom att vid varje iteration genomföra test av programvaran. I varje iteration ges dessutom tillfälle att utvärdera processens kvalitet. (Kruchten, 2002; Lunell, 2003) Kvalitet sätter sin prägel genom hela processen och ligger som en grundbult i varje disciplin. (Lunell, 2003)

Konfigurations- och ändringshantering

Konfigurationshantering innebär dels att hålla reda på och kontrollera de komponenter ett program är uppbyggt av och hur de förhåller sig till varandra, dels att upprätthålla spårbarheten över de olika enheterna och vilken effekt en ändring har på programvaran. Denna praxis är nödvändig för att koordinera de aktiviteter som förekommer och de artefakter som produceras i den komplexa miljö som ett systemutvecklingsprojekt är. Att man efter varje iterationsavslut tar fram och testar en version av programvaran krävs för att samordna iterationer och utgåvor. Kruchten, 2002; Lunell, 2003)

Struktur

Strukturen i produkten RUP är uppdelad i två dimensioner. Den ena är vertikal och utgör en statisk struktur. Där ryms de discipliner, så kallade arbets- eller processområden, som innehåller de kombinerade huvudaktiviteterna med tillhörande roller i ett systemutvecklingsprojekt. Vi utgår här från den produkt som har nio primära discipliner, av vilka sex utgör aktiviteter för utveckling och de tre återstående stödjande discipliner för projektledning, konfigurations- och ändringshantering och utvecklingsmiljö. (Kruchten, 2002)

Rollen är ett centralt begrepp i RUP och definierar ett beteende hos en individ som bestäms utifrån de aktiviteter som rollen skall utföra. En individ kan besitta flera roller. En disciplin innehåller en sekvens av aktiviteter som knyts samman med de olika roller som producerar någon

form av resultat, artefakter. En arbetsenhet som förväntas utföras av en viss roll är en definition av en aktivitet i RUP. En aktivitet har oftast för avsikt att skapa och uppdatera artefakter. (Kruchten, 2002)

Den andra dimensionen har en dynamisk struktur och löper horisontellt, vilken kan beskrivas som en utvecklingscykel. För att kontrollera utvecklingscykeln är RUP organiserad i faser. De fyra faserna består av förberedelse, *inception,* etablering, *elaboration,* konstruktion, c*onstruction* och överlämning, *transition.* Varje fas innehåller en till flera iterationer. I varje iteration förekommer det aktiviteter från nästan alla discipliner, beroende på i vilken fas processen för tillfället är i. Varje fas avslutas med en utvärdering där man kan besluta om projektet skall fortsätta eller avbrytas. Varje sådan punkt kallas milstolpe och specifika delmål skall uppfyllas för att gå vidare. I vissa fall är det befogat att även inkludera mindre milstolpar mellan iterationerna. (Kruchten, 2002, Lunell, 2003)

Agile Software Development

The Agile Alliance författade 2001, ett manifest som bygger på tolv principer. (Appendix B) Principerna i Agile Software Development inspirerar till rörlighet i processen. Manifestet värderar och framhäver vissa företeelser inom systemutvecklingen som prioriteras framför andra. Manifestet värdesätter individerna och ett bra samarbete mellan dessa inom projektet. Man tror således inte att enkom en bra process gör ett lyckat projekt om projektet inte har motiverade, engagerade och samarbetsvilliga medlemmar. En dålig process kan däremot ställa ett projekt på ända trots ett bra team. Fortsättningsvis förespråkar manifestet en väl avgränsad mängd dokumentation som skall vara avgörande och betydelsefull för att inte hindra utvecklingen av en körbar produkt. Att involvera kunden för feedback regelbundet och ofta, värdesätter manifestet som en del av att lyckas med ett projekt, och inte enbart basera samarbetet via ett kontrakt. Slutligen menar alliansen att en långt i förväg beslutad plan ofta hindrar ett öppet synsätt för förändringar. Därför värdesätter manifestet att ett projekts planering skall vara så anpassningsbar att man kan svara upp till förändringar under hela projektets gång. (Martin, 2003)

En av utgångspunkterna i lättrörliga metoder är att de tillämpar en inom tidsramen, *timebox*, iterativ och inkrementell utveckling. Iterationerna är oftast korta och man förutsätter en intern release i slutet av varje iteration där man genomför tester med kund för att återfå respons. (Larman, 2004)

Principer

Vi har valt att sammanfatta en förklarande redogörelse för de tolv principer som ingår i manifestet.

En viktig attityd i lättrörliga metoder är att fokusera på kunden. Kundes delaktighet i projektet och samarbetet med denne är centralt. Här uppmuntras kunden till att komma med fortlöpande förändringar av kravbilden, även sent i projektet. (Jobson; 2006, Larman, 2004; Fowler & Highsmith, 2001) Den alltmer växande oförutsägbarheten och en turbulent omgivning bidrar till förändringar, menar Fowler och Highsmith (2001). Anpassning till förändring är därför också ett viktigt kärnbudskap. Kravförändringar ses som en möjlighet som utnyttjas till kundens konkurrensfördelar. (Fowler & Highsmith, 2001) Värdet för kunden bör omvärderas ofta då ett programvaruprojekt av idag är nära förbundet med instabilitet. Att leverera något av värde till kunden är då att prioritera vilket inte behöver innebära att en version släpps, utan att en körbar del

av en programvara endast levereras för utvärdering. En körbar programvara är också, enligt alliansen, det grundläggande tillvägagångssättet för att fånga kravens subtilitet. (Fowler & Highsmith, 2001)

Ett nära samarbete mellan projektteamet och kunden som fortgår genom hela projektet är av betydelse för teamet för att förstå och lära sig om kundens verksamhet. (Fowler & Highsmith, 2001) Det synsättet bidrar också till ökad förståelse för omgivningens krav. Vidare skall man försöka knyta projektet kring motiverade individer och bygga upp ett förtroende så att varje medlem tar ett personligt ansvar. (Jobson, 2006; Larman, 2004) Ledningen måste kunna ge individerna i gruppen förtroendet och samtidigt förstå att de som har kunskap om ett område också är de som kan fatta ett bra beslut. I ett självstyrande team ges en bättre förutsättning för att få fram den bästa arkitekturen, kraven och designen, då samspelet blir högt och reglerna få. Den lättrörliga metoden förespråkar en mindre mängd dokumentation för en given uppgift (Fowler, 2001). För mycket dokumentation tar udden av ett effektivt utvecklingsarbete. (Martin, 2003) Vid sidan av dokumentation belyser den lättrörliga metoden vikten av att mer använda direkt kommunikation och att förena dessa två medel. En effekt av att använda en direkt, ansikte mot ansikte kommunikation är att utvinna den tysta kunskapen och sprida kunskapen vidare. För att utvecklingen skall vara hållbar i längden, skall projektteamet hitta ett lagom arbetstempo som håller genom hela projektet. Teamet bör också regelbundet granska och utvärdera processen och lära av sina misstag för att anpassa den därefter och på så sätt effektivisera arbetssättet. (Fowler & Highsmith, 2001)

Enkelhet är en väsentlig del då det underlättar vid förändringar. (Fowler & Highsmith, 2001; Larman, 2004) Ett system bör byggas generellt och enkelt och vara förberett för utökning av funktioner. (Fowler & Highsmith, 2001; Jobson, 2006) För att öka rörligheten i processen krävs en kontinuerlig uppmärksamhet på teknik och design. Designen är en aktivitet som är framträdande och kan förbättras stegvis genom hela projektet. (Fowler & Highsmith, 2001)

Scrum

Scrum är en lättrörlig process för att hantera och kontrollera utvecklingsarbeten och beskrivs som en IID metod med betoning på projektledningens värderingar och principer. Scrum utgår från att systemutvecklingsprocessen är komplicerad och oförutsägbar.

Projektet delas upp i så kallade Sprintar, där varje Sprint är cirka 30 dagar. Varje Sprint börjar med ett planeringsmöte under en heldag där alla kraven gås igenom av produktägaren inför hela teamet. En tidsupskattning görs på de presenterade och prioriterade kraven. Tillslut bestäms vilka krav som kommer och hanteras under Sprinten. Varje dag i en Sprint startas med ett möte med teamet där man går igenom, med varje medlem, vad denne har gjort sedan mötet dagen innan, vad som kommer att åstadkommas under dagen och om det finns några hinder.

Varje Sprint avslutas med en granskning där en prestation av en ny version av produkten visas upp för produktägaren, kunderna och övriga intressenter. Här visas och bevisas de kravändringar som gjorts.

Det som Scrumanhängarna menar uppnås med Scrum är bland annat att teamet blir självständigt, motiverat och på så vis effektivt. Fokus bevaras genom att alla vet att efter fyra veckor skall alla kravändringar visas upp. (Larman, 2003)

Extreme Programming (XP)

XP består av fyra värderingar: kommunikation, *communication*, enkelhet, *simplicity* återkoppling, *feedback* och mod, *curage.*

En iteration pågår en till tre veckor. Grundtanken är att utvecklarna arbetar bättre och mer effektivt när de är utvilade, varför man försöker ha 40-timmars arbetsveckor och övertid tillåts inte två veckor i rad. Enhetstester skrivs först, därefter testas själva koden. Ingen dokumentation görs utan allt är så kallad *tacit knowledge,* vilket betyder att all kunskap finns ”i huvudet” hos utvecklarna och koden skall vara så enkel som möjlig. Kommer någon ny med i teamet paras denne ihop med en mer erfaren utvecklare som lär ut kunskapen.

XP är väldigt kommunikations- och team-orienterat. Kunder, utvecklare och ledare bildar tillsammans en enad grupp som sitter i samma rum. Detta för att, som tidigare nämnts, kommunikationen är viktig. Kunderna sitter med bland annat för att kunna ge detaljerade förklaringar till utvecklarna och skriva acceptanstester tillsammans med dem.

Utvecklarna sitter tillsammans två och två vid en dator, så kallad parprogrammering, där en kodar och den andre granskar koden. De följer en strikt kodningsstandard som sätts upp i början av projektet. Vem som helst av utvecklarna kan ändra i koden, det spelar ingen roll vem som skrivit den. När en koduppgift är färdig, testas den och byggs sedan på den redan färdiga koden, detta sker flera gånger per dag.

Det är utvecklarna som estimerar hur lång tid de olika kraven kommer att ta och kunderna som sedan prioriterar vilka krav som är viktigast. På detta vis blir förhoppningsvis de viktigaste kraven gjorda. (Cockburn och Highsmith, 2002)

Capability Maturity Model Integration

Capability Maturity Model Integration (CMMI) är en processförbättringsmetod som kan användas till att utvärdera vilken mognadsgrad en process har nått eller till att vägleda en processförbättring i ett projekt, i en division eller i en hel organisation. CMMI har utvecklats av Software Engineering Institute (SEI) och kombinerar ett valt set av Best Practices som är baserat på erfarenheter utifrån utvecklingsteamets skilda bakgrunder vilka representerar bland annat systemanalys och design, programvaruutveckling och ledning. (Ahern, Clouse & Turner, 2001)

CMMI består av två olika arkitektoniska representationer, *continuous* och *staged*. Perspektiven skiljs åt genom att continuous fokuserar på processområdets kapacitet, *capability* och staged fokuserar på organisationens mognad, *maturity*. (Ahern, Clouse & Turner, 2001)

Continuous möjliggör för organisationen att välja de förbättringar som bäst stämmer överens med deras företagsmål och som minimerar organisationens riskområden. Continuous gör det även möjligt med jämförelser mellan och i organisationer i samma processområde med hjälp av processområdesbasis eller genom att jämföra resultat genom att använda likvärdiga plattformar. (http://www.sei.cmu.edu/cmmi)

Staged tillhandahåller organisationen med en beprövad sekvens av förbättringar. Modellen består av en serie av nivåer, så kallade mognadsnivåer där varje nivå ger anvisning om vad organisationen skall lägga fokus på för att förbättra dess arbetsprocesser. (Ahern, Clouse & Turner, 2001)

Struktur

Mognadsnivåerna består av fem klassificeringsnivåer som går från Initial - Nivå 1 till Optimerad - Nivå 5. Varje nivå består av så kallade processområden och varje processområde består i sin tur av generella och specifika mål. De generella målen tillämpas i alla processområdena medan de specifika målen är unika för ett specifikt processområde och tillämpas endast där. Samtliga mål måste vara uppnådda inom ett processområde innan området ses som avklarat. Som hjälpmedel för att nå varje mål finns det övningar, *practices.* Det finns inget krav på att dessa övningar skall genomföras. Alternativa övningar kan leda lika bra till att nå målen. Det är också nödvändigt att varje processområde inom en nivå är avklarad innan man går vidare till nästa nivå. Anledningen till att en nivå måste vara avklarad innan en ny påbörjas är att varje nivå står till grund för en effektiv implementation av processer i nästa nivå.

Mognadsnivåer

Nedan följer en kort beskrivning av varje nivå i *Maturity Level*. (Ahern, Clouse & Turner, 2001; http://www.qlabs.se; http://www.computerworld.com; http://www.sei.cmu.edu/cmmi)

Nivå 1 - Initial

Detta är starten för en ny process. Här levererar man för det mesta produkter som fungerar men processen är, kaotisk, oförutsägbar, dåligt kontrollerad och reaktiv. Lyckade projekt beror oftast på individuella insatser. Man har ofta problem med: att hålla planering och budget, att kraven är instabila och att ha en bra koordinering.

Nivå 2 - Hanterad

Denna nivå fokuserar på det som rör programvaruprojekten för att etablera en enkel kontroll av projektledning. Även på denna nivå är processen oftast reaktiv men den är definierad och dokumenterad. Projekten levererar för det mesta rätt. Lyckade projekt kan nu repeteras på andra projekt med liknande applikationer. Elementära projektledningsprocesser, *Basic Project Management Processes* finns för att kunna söka kostnader, scheman och funktionalitet. Man har ofta problem med: att man är osäker på hur man skall förbättra sig inom organisationen, att använda sina erfarenheter från tidigare lyckofulla projekt och att man avviker från den definierade processen vid tidsbrist.

Nivå 3 - Definierad

På den tredje nivån involveras både projektet och organisationsfrågor. Här är processen definierad och den är numera proaktiv. Uppmärksamhet ges till dokumentation, standardisering och integrering. Projektet följer en definierad process även vid tidsbrist. Tidigare lyckofulla projekt återanvänds på ett ordnat sätt. Man har ofta problem med: att projektledaren inte är tillräckligt bevandrad med processens förmåga, att bedöma om processen fungerar effektivt och att avgöra om kraven från kunden kommer att infrias.

Nivå 4 - Kvantitativt hanterad
Här vill man etablera en kvantitativ förståelse för både programvaruprocessen och de programvaruprodukter som byggs. Processen är med andra ord mätbar, kontrollerad och förutsägbar. Processen kan justeras och anpassas till specifika projekt utan att tappa kvalitet eller frångå specifikationerna. Man har ofta problem med: att felfrekvensen är stabil men inte minskade, att fokus ligger på upptäckandet av fel istället för förebyggandet av dem och att det är svårt att förutspå om kundkraven kommer att uppnås.

Nivå 5 - Optimerad
Detta är den sista nivån. Här fokuseras det på ständig processförbättring. Detta sker genom kommentarer och gemensamma idéer. Det introduceras innovativa processer för att bättre tjäna organisationens olika behov. Pilotprojekt förekommer ofta.

Processområden för undersökningen

Nedan text är en sammanfattande översättning av de utvalda, passande processområdena i CMMI vilka användes för att utvärdera de områden som undersökts i studien. Se Appendix D eller läs CMMI SE/SW version 1.1 Staged för fullständig information.
(www.sei.cmu.edu/pub/documents/02.reports/pdf/02tr002.pdf)

Projektplanering – Nivå 2
Syftet med detta område är att etablera och underhålla en plan som definierar de aktiviteter som ryms inom projektet. Processområdet involverar följande aktiviteter såsom utvecklandet av en projektplan, ändamålsenligt samspel med intressenter, åtagande mot planen samt uppdatering och underhåll av planen.

Planeringen startar med kraven som definierar produkten och projektet. Planeringen inkluderar att en bedömning av produkten och uppgifter görs, att behovet av resurser uppskattas, att åtaganden förhandlas samt att risker identifieras och analyseras. Allteftersom projektet fortskrider kommer säkerligen projektplanen behöva granskas och uppdateras.

Kravhantering – Nivå 2
Syftet med processområdet Kravhantering är att hantera en produktkomponents krav samt att identifiera motsägelser mellan kraven och projektets plan och arbetsprodukt. Processområdet omfattar alla krav mottagna eller genererade av projektet och inkluderar både tekniska och icke tekniska krav såväl som krav ställda av organisationen. Stegvis och i lämplig takt försäkrar sig projektet om att överenskommelserna av kraven klarar av att stödja planeringen och verkställandet av behoven i projektet. Inkommande krav granskas för att lösa problem och förebygga missförstånd innan kraven införlivas med projektplanen. Allteftersom kravförändringar dyker upp hanterar och identifierar projektet motsägelser som kan förekomma kring planen, arbetsprodukten och kraven. I hanteringen av krav ingår att dokumentera kravförändringar och rationalisera och upprätthålla dubbelriktad spårbarhet mellan källkraven och alla produkter och produktkomponenter.

Tekniska lösningar – Nivå 3
Ändamålet för processområdet tekniska lösningar är att designa, utveckla och implementera lösningar baserade på kraven. Lösningar, design och implementeringar omfattar produkten, produktkomponenter och produktrelaterade livscykelprocesser var för sig eller kombinerade på ett lämpligt sätt. Processområdet fokuserar på att utvärdera och välja en lösning som kan uppfylla

de krav som ställts, utveckla en detaljerad ritning på den valda lösningen samt tillämpa designen som en produkt eller produktkomponent.

Prototyper kan användas som ett medel för att erhålla viktig kunskap för att utveckla teknisk datapaket eller ett komplett set av krav.

Verifikation – Nivå 3

Meningen med att ha verifikation som ett processområde är för att försäkra att en vald arbetsprodukt avspeglar dess specifika krav det vill säga att man byggt produkten rätt. Processområdet involverar förberedelse och utförande av verifiering och identifiering av korrigerande handling. Verifikation är en inkrementell process och lever igenom hela projektet med start i att verifiera kraven, fortsätter genom utvecklandet av produkten och når till slut en färdig produkt. En verifiering av produkten ökar väsentligt sannolikheten att produkten svarar upp till kravställningen. En så kallad *peer review*, sakkunnig bedömning, är en viktig del av verifieringen och en bevisad mekanism för att effektivt avlägsna brister. En *peer review* innefattar en metodisk utvärdering av produkten genom att tillverkarens jämlike urskiljer fel och andra förändringar som behövs.

Validering – Nivå 3

Validering påvisar att en produkt uppfyller dess avsedda användning när den tillämpas i den miljö den är avsedd för. Man validerar alltså att man byggt rätt sak. Användaren är ofta med att validera produkten. Verifikation och Validering används oftast tillsammans, man validerar en produkt på liknande sätt som man verifierar en produkt, till exempel med tester, analyser och demonstrationer.

Empiri

I detta kapitel ges en sammanställning av de intervjuer som utfördes med respektive projekts medlemmar. Innan varje sammanställning presenteras projekten. Avsikten med presentationen av projekten är att ge en viss inblick i vad projekten gick ut på, hur stora de var, vilken teknik som användes och varför de genomfördes. Även en presentation av respondenterna erbjuds.

Projekt Parts Order Web

- Process: Rational Unified Process

> *”Här är de inte jättevana med att använda RUP överhuvudtaget. Jag tog det för givet när jag kom att det skulle vara RUP. Så de[organisationen] sa mer ja okej va bra, klart du skall köra RUP. För det är ju sagt att det skall vara så. Så de tyckte det var jättebra att vi körde RUP för det är det ju sagt från ledningen att det skall vara.”*

Bakgrund

Parts Order Web (POW) var ett projekt som startade hösten 2005 och som pågick under tiden för studien. Leverantören av applikationen var Volvo IT och systemägare var Volvo Parts. Användarna var Volvo Parts olika affärsområden och dess externa leverantörer, *dealers*. Exempel på affärsområden är Volvos Construction Equipment (Volvo CE). Projektet var cirka 7 medlemmar stort, bestående av en projektledare som även var arkitekt och systemanalytiker, en testresurs och domänexpert, en person från förvaltningsorganisationen som var med för att lära sig systemet, en intressent från det tidigare projektet och tre stycken utvecklare från Volvo IT, Polen. Projektet bemannades utefter de resurser som var tillgängliga. Från Volvo Parts sida fanns det ett krav att ta in en resurs med arkitektonisk spetskompetens. Tekniken som användes var J2EE (Java) och IMS (Cobol). Syftet med projektet var att skriva om tre gamla applikationer från systemet Parts OnLine till en applikation. Applikationerna i Parts OnLine utvecklades i .Net men hade en ogenomtänkt arkitektur och upplevdes som ”gungig”. Eftersom projektet var en omskrivning, utgicks det från en kravbild baserat på Parts OnLine:s funktioner. Förhållandet kallas 1:1. Detta medförde att förutsättningarna var något annorlunda gentemot hur de hade varit om applikationen byggts utefter en helt ny kravbild. Man utvecklade endast applikationen vilken skulle integreras med de tre legacysystemen. Systemet var en enkel applikation för orderhantering för att beställa reservdelar, enkelt i den bemärkelsen att det innefattade ett begränsat område av systemfunktioner.

Utvecklingsprocessen var en avskalad RUP, anpassad till Volvo IT vilket också var den officiella utvecklingsprocess som tillämpades i organisationen. Kunskapen om RUP var ojämn bland medlemmarna. Ingen utbildning i RUP genomfördes innan projektet sattes igång. En mentor fanns dock till hjälp, vilken hjälpte till med vilka delar av RUP respektive PCM som projektet skulle koncentrera sig på samt vilka artefakter som skulle produceras.

Respondenterna

Mats Ekhammar
Mats var inhyrd konsult. Han hade flera olika roller i POW-projektet, de största var projektledare och arkitekt. Han har varit med i projekt tidigare på Volvo IT och visste därför hur verksamheten fungerade.

Inger Stening
Inger hade tidigare varit med och utvecklat Parts OnLine och kände därför till logiken bakom systemet och agerade därför som domänexpert. Hon testade och specificerade alla interface mot legacy. Inger har varit anställd på Volvo sedan 1989 och har tidigare arbetat som projektledare, testare och utvecklare.

Habte Woldu
Habte var anställd på Volvo IT, Polen. Han jobbade som chefsdesigner och utvecklare i POW. Han designade och byggde de externa kommunikationslagren. Habte har varit på Volvo IT sedan september 2005.

Åsa Forsberg
Åsa arbetade på Volvo Parts och var systemägare till POW. Hon var Business-projektledare på kundens sida.

Projektplanering

Med hjälp av checklistor i projektstyrningsmodellen PCM kunde projektledaren hålla reda på vilka artefakter som var tvungna att ingå i projektet. Projektledaren upplevde det bra med en sådan anpassning. På grund av att projektet POW var såpass litet bedömdes det att anpassningen kunde reduceras ytterligare.

”Bra att ha checklistor så att man inte glömmer av något. Men risken är att man drunknar, om man inte anpassar RUP, i det för att det är så jäkla mycket. Bra att ha en Volvo IT-anpassning som säger strunta i detta och gör så, sen måste man anpassa det ytterligare för att passa projektet. Det hade varit jättejobbigt att börja om från scratch. Framför allt är det bra om någon skall reviewa det eller ta över, då vet man vart sakerna finns. Det är det bästa med RUP.”

I POW-projektet användes Project Charter som finns i PCM. Där beskrevs alla iterationer, vad som skulle göras i varje iteration och vilka dokument som skulle produceras et cetera. De huvudartefakter som ingick i projektet var en projektplan, Use Case, Use Case Realizations och Software Architecture Document. Use Casen var viktiga för slutdokumentationen för att visa vad systemet hade för funktioner. Däremot förhöll sig respondenten lite tveksam till vissa dokument, såsom Use Case Realization. Dessa menade respondenten var svåra att hålla uppdaterade. Han upplevde också att dokumenten innehöll mycket formalia. De innehöll mycket överskrifter och indexeringar och han ansåg vidare att det som egentligen var väsentligt att dokumentera var förhållandevis lite.

”Oftast går man ut rätt hårt med RUP och börjar med alla dokument och är duktig. Men sen kommer ju verkligheten ikapp en, man gör alla iterationer och planering. Det blir svårt att hålla ihop allt, man skall ju ändra i dokumenten när det ändras, det tror jag inte att man gör alltid så mycket som man borde göra. Då bör man sitta och ägna sig åt projektledning enbart och ha den

fokusen att det här det påverkar ju faktiskt kravdokumenten. Uppdateringen av dokument blir svår. Vissa dokument är ju till för att stötta i processen. Vissa saker skall man ju slänga också. Vissa dokument produceras ju för att exempelvis visa för att hjälpa utvecklaren att göra rätt. Som i slutdokumentation skall det vara saker som stämmer. Skall ju inte sparas för att det står RUP. Sen när alla ändringar kommer så ändar de ju naturligtvis, men i grundbulten [systemet] är det ju rätt, det kommer ju aldrig att uppdateras några ändringar [i dokumentationen]. Men det tror jag inte är meningen heller. Man måste nog lära sig att kasta mer, vissa dokument som har gjort sin nytta och lyft fram projektet till en viss nivå. Och sen går man vidare. Sedan får det vara en slutdokumentation som stämmer. Man fastnar lätt i att tänka att alla dokument måste vara [slut] dokumentation. En risk i sig."

Fördelen var dock att dokumentationen var normaliserad och när man sökte efter en viss dokumentation visste man var den fanns att hitta. För att förstå ett dokument kunde det dock krävas referenser vilket kunde göra dokumentationen svår att tyda. På grund av det här projektets omfattning, var det hanterbart. Dokumentation var något som ibland upplevdes som besvärligt, bland vissa av projektmedlemmarna, vilket resulterade i att dokumenten höll olika standard. Även detta var relativt hanterbart på grund av att projektets ringa storlek. Hade det däremot varit ett större projekt där bara en viss del av dokumentationen görs av en person, hade det varit viktigare att det levererades dokument av samma kvalitet, så att alla förstod innehållet.

När vi ställde frågan till en av respondenterna om hon upplevde att dokumentationen som gjordes var tillräcklig, svarar hon att den kunde förbättras något.

"I och med att den skall kopplas till ett större system skall vi se över det, hur man skall göra det, men det finns Guide Lines som vi brukar säga om hur man skall göra det. Tiden har väl inte alltid räckt till, till att göra bättre dokumentation. Skall det lämnas över till någon annan så behöver man säkert dokumentera något mera men det är väl ganska bra ändå. Det kan alltid bli bättre."

Vidare ingick det i projektplaneringen att planera iterationerna i varje fas. Samtliga iterationer planerades i förväg utefter de Use Case[1] som fanns från förlagan POL. Tidsberäkningen av varje iteration baserades på valda Use Case och estimerades utefter projektledarens erfarenhet till största del. Allteftersom anpassades iterationslängden. Grundtanken var att iterationerna skulle vara tidsbestämda, *timeboxed*. Det gjordes försök att hålla tidsramen, men applikationens externa beroenden samt svårigheter med att samköra utvecklingsflödet till tre olika system, gjorde det svårt.

"Vi försökte ha det så [timeboxed], men det går ju aldrig. Eftersom det hela tiden uppstår krockar i och med att det är tre system."

Enligt respondenten är det generellt starten av ett projekt som är arbetsam. Han tycker det är värt att lägga tid i början, men upplever i regel att få vill se och ta den kostnaden då inget produceras, utan alla sitter tillsammans och diskuterar. Respondenten menade att en ordentlig genomgång brukar löna sig i längden då gruppen får upp ett stabilt arbetsflöde.

[1] Ett användarfall innehåller en eller flera scenarios som beskriver hur systemet skall interagera med en slutanvändare eller annat system. (www.wikipedia.org)

I den första fasen var utgångspunkten ett lättare Use Case som med säkerhet fungerade, för att på så sätt ta fram en mall som beskrev tillvägagångssättet. En gemensam genomgång hölls där specifikationer skrevs tillsammans och var och en löste en specifik uppgift, därefter gjordes en genomgång på resultatet. Eventuella ändringar diskuterades och påbyggningar av funktioner presenterades. Tillvägagångssättet medförde att medlemmarna fick en slags utbildning och förståelse för arbetssättet.

Alla faser förutom konstruktionsfasen som innehöll tre iterationer var utan iterationer. Releaser utfördes endast sent i projektet. Däremot visades en prototyp för kunden i starten som påvisade att tekniken fungerade.

På vår fråga hur kommunikationen i projektet gick till fick vi reda på att projektmedlemmarna satt nära varandra och att de hade veckovisa projektmöten. Där hölls en genomgång över vilka Use Case som skulle prioriteras, nya problem som kommit upp, nya releaser et cetera. Systemägaren från Volvo Parts och POW:s projektledare träffades också minst en gång i veckan och då var det mycket planering av olika slag som diskuterades, exempelvis vad och vem som behövdes vid de olika testerna och hur långt gången processen var. Systemägaren upplevde att hon la ner mycket tid i POW-projektet då det hade första prioritet. Systemägaren tyckte att den tid hon la ner på projektet var ansträngande men nödvändig i och med den bristfälliga kravställningen. Hon upplevde dock att POW-projektet hanterade kravförändringarna mycket bra.

Projektledaren ansåg att det var viktigt att kommunikationen både i gruppen och mellan gruppen och kunden fungerade, då det krävdes mycket verksamhetskunskap för att veta hur rutinerna fungerade i organisationen. Även utvecklarna från Polen kunde ibland uppleva problem med att inte känna till verksamheten.

"Misunderstandings were occurring due to the lack of common understanding of the business in the development team. We new how to code, which frameworks to use and how to solve common problems but didn't have precise knowledge how does the business we are supposed to improve work. Fortunately Mats and Inger were the people who had such understanding and were able to explain it."

Affärsområdenas styrgrupper hade sina egna möten, så kallade gatemöten, utefter PCM, där man gick igenom projektets fortsättning. Vid speciella tillfällen bjöds de olika styrgrupperna in till möten med systemägaren och projektledaren. Ett exempel på detta var när användargränssnittet visades upp för feedback. I andra situationer då bara ett affärsområde varit involverat, hölls det möten enbart med dem.

I det Volvoanpassade RUP fanns det även så kallade riskmallar utefter PCM som kunde följas. De risker som analyserades fram dokumenterades och kodades. Därefter gjordes uppföljningar tillsammans med kunden för att kontrollera om risken hade förstärkts, förminskats eller åtgärdats. Projektet arbetade utefter att försöka minimera riskerna genom att låta respektive ansvarsområde bli medvetet om problemet och ta sitt ansvar.

En faktor som kunde ha blivit en stor risk i projektet, var Competitive Sourcing. I och med att inte möjligheten fanns att träffa resurserna från Polen innan projektet sattes igång, valdes de ut med endast deras CV till grund. En annan stor risk var kravbilden från kunden, eftersom *"gör som det gamla"*, uppfattades som en väldigt svår kravbild. Vid tiden för studien upplevde respondenten

att det fortfarande inte fanns någon bra kravbild, trots det sena skedet i processen. Åtgärden för att reducera risken var att skriva om Use Casen på nytt och omarbeta kravbilden efterhand. Den tredje risken var kommunikationen mellan applikationen och de tre stordatorsystemen. En risk som visade sig vara befogad då tester mot dessa fallerade. De riskerna hanterades genom att designa applikationer så att riskerna blev så isolerade som möjligt. Externa beroenden av andra system var ytterligare en risk. Vidare bedömdes att användargränssnittet var en risk, en konsekvent design var vad som skulle göras enligt kundens styrgrupp. Till hjälp fanns då på Volvo IT en Design and Usability grupp som följde vissa Guide Lines för att utforma användargränssnittet.

Att vara tvungen att ersätta en projektmedlem vid till exempel sjukdom kunde ha lett till en risk. Flera av medlemmarna hade dock ett brett kunskapsfält som medförde att de var tämligen utbytbara. Om en sådan situation hade uppstått, tror respondenten att det i hans fall hade medfört att fokus på projektledarrollen minskat. Det var en risk i sig enligt respondenten, att en individ hade rollen som både arkitekt och projektledare. I motsats till hur det var sagt att fördelningen av rollerna skulle ha sett ut, upplevde han att projektledningen tagit större delen av hans tid och därför fick hans roll som arkitekt mindre fokus. En konsekvens av att kombinera dessa två roller, blev att han förlorade kontrollen över att de fastställda arkitektoniska riktlinjerna följdes vid utvecklingen av applikationen.

Kravhantering

Innan projektet startades gjordes en review av det gamla systemet, där det kom fram att det fanns en dokumentation som var ofullständig och en teknik som inte fungerade. Efter denna review togs det även beslut om en omskrivning av systemet till en annan teknik. Detta innebar att kravbilden blev densamma som tidigare, ett så kallat 1:1-förhållande.

”Kravbilden var ’det skall vara som det gamla’, samma funktionalitet, men det skall vara bättre uppbyggt, så man kan utveckla den, så rättningarna funkar bättre.”

Med en sådan kravbild måste en förståelse skapas initialt för hur det gamla systemet fungerar och vad det är för applikation som skall byggas, förklarar en av respondenterna. Kravbilden granskades utifrån den dokumentation som tillhörde förlagan, POL, som sedan visade sig vara bristfällig eller saknad. Att skriva om Use Casen ingick i projektet och när omskrivningarna av Use Casen var klara, visades dessa upp för kunden. Kunden fick verifiera att kraven stämde överens med deras uppfattning över systemets funktionalitet. Allteftersom uppdagades det att den nya kravbilden inte helt stämde överens med förlagan, trots att kunden bekräftat kravbilden. Det problemet bidrog till att felaktiga funktioner byggdes in i det nya systemet.

”… samma som förut märker man efteråt är sämsta kravställning som finns.”

”Men behöver man lyfta på locket då, som i vårt fall, när det var 1:1 då och upptäcker att egentligen har ingen förstått varför man pratar på det här viset med det här systemet, till exempel, det funkar inte. Följer man det de säger så funkar det inte utan det är flera års anpassningar som har gjorts utan att dokumenteras någonstans. Så faller ju liksom lite den här fina vägen med iterationen framåt. Ja ha, då får du fixa det och så väntar vi i två kalenderveckor för de kommer de och fixar det. Så börjar man med nästa Use Case då eller liknande och då fragmenteras det rätt lätt. Så det viktigaste är att man, skulle ha kollat lite bättre, stämmer det

verkligen med verkligheten? Problemet är ju att det finns system som funkar och funkar systemet så måste det ju vara bra, eller grundbultarna bör ju vara på plats, annars skulle det ju inte funka. Men det behöver ju inte stämma hundra då, för det kan ju hända att man anpassat det efter årens gång, man fixar och trixar."

En direkt konsekvens av den dåliga kravbilden, trodde respondenten var att det kunde komma att bli svårt att mäta slutresultatet. Det vill säga i vilken utsträckning kunden fick det systemet de förväntade sig. En annan effekt av bristerna i kravbilden var att projektledaren upplevde det svårt att planera arbetsgången.

Systemägaren kände att det förmodligen krävdes en hel del erfarenhet som arkitekt för att hantera denna vaga kravställning som de hade. Denne hyste dock förtroende för den inhyrde arkitekten/projektledaren och hans kompetens att hantera kravbilden. Respondenten som inte var med när projektet startades och när beslut togs att göra en omskrivning av POL, hade förståelse för att IT-projektledaren säkerligen hade mycket mer att önska från Volvo Parts som kravställare.

I början av året 2006, en bit in i utvecklingsfasen gjordes en avskalad prototyp, utan färg och form, av applikationen, som de inblandade skulle reflektera över och ge feedback på. Prototypen togs fram i samarbete mellan Volvo Parts och Volvo IT:s Design and Usability grupp och med hjälp av den kunskap arkitekten hade. Syftet med prototypen var att enas om och förbättra användargränssnittet.

När vi frågade respondenten, hur projektet hanterade kravförändringar, fick vi reda på att arbeta utifrån 1:1 också medförde diskussioner om vilka krav som ingick i omskrivningsavtalet, då funktioner som lagts till i det gamla systemet inte uppdaterats i dokumentationen.

"Eftersom detta är ett, 1:1-projekt, så blir det ju lite speciellt då, men allt är ju inte bra i det gamla, därför skriver man ju om det också och då blir det ju ändringar. Och har vi kommit till diskussionen, ingår det eller ingår det inte i omskrivningen eftersom ofta vill man ju ha mer pengar när det är något som inte stämmer som de har sagt."

Vidare berättade respondenterna att vid större förändringar som påverkade projektets kostnad och tid, skulle Change Control Board användas. Kundens styrgrupp måste då godkänna och dokumentera ändringens kostnad. Användarna skulle vid stora eller kritiska SCR:er[2] gå via Volvo Parts och en förhandling skulle göras dem emellan. POW-projektet skulle också tidsbedöma och ge ett kostnadsförslag innan ett beslut skulle tas. Tillvägagångssättet var det formella men har inte använts aktivt i POW-projektet vid tidpunkten för studien, då det ännu inte inkommit några större kritiska förändringar.

"Problemet är ju att de [kravförändringar] kommer att komma nu. Eftersom vi hade så dålig kravbild, 1:1-förhållande. Så nu kommer det: 'Varför gör ni så…' ."

En effekt av 1:1-förhållandet, anser en av respondenterna, var att många större kravförändringar sköts på framtiden och skulle komma att hanteras av förvaltningsavdelningen.

[2] System Change Request, kravförändringar

”Man försöker ju mota allt det nu, inga ändringar in, så mycket som möjligt då å då blir ju det, det är ju 1:1 så det är väldigt enkelt att säga det! Så det är ett lite speciellt projekt.”

”De har en stor lista på kravförändringar som de vill ha gjort men vi vill först ha det här på plats innan vi börjar med det. Så man inte lägger in för mycket på en gång men vi vet ju vad det är som skall in.”

En anledning till det är också att det fanns ett uppdämt behov att få genom förändringar och ny funktionalitet. En bidragande orsak till detta tror systemägaren var att koden varit ”fryst” så länge som två år, då man direkt när föregångaren till POW var klar märkte att systemet inte var bra.

Vid mindre förändringar, som inte påverkade processen eller arbetsflödet, förhandlades dessa mellan IT-projektledaren och systemägaren. Eftersom det inte fanns någon klar kravbild från början tvingade man lösa kraven allteftersom dessa uppdagades. Systemägaren från Parts tyckte att det dagligen hanterades små kravförändringar och upplevde att hon fått bra respons från POW-projektet. Systemägaren upplevde inte förfarandet som ett bra arbetssätt men det fungerade utefter de givna förutsättningar man hade.

Kundsituationen gjorde också kravhantering speciell. Det var många kunder inblandade till samma system och det var svårt att tillgodose alla önskemål. Det var meningen att de olika affärsområdena alltid skulle gå via systemägaren med önskemål om kravförändringar, men det förekom att Volvo Parts kunder gick direkt till utvecklarna i projektet. Ibland fanns möjligheten att hantera det tillvägagångssättet också.

”Det är oftast så tyvärr då, kunden försöker alltid gå till utvecklaren direkt, ’Du, kan inte du ta in detta är du snäll!?’ Det går ju ibland.”

På vår fråga om det inte hade varit enklare att ha proceduren att användarna gick direkt till POW-projektet, då användarna visste vad de ville ha och om Volvo Parts verkligen var tillräckligt insatta i kravställningarna, svarade en av respondenterna att dels så ville respektive affärsområde så väldigt mycket och dels så kände de inte till vad förändringen skulle komma att kosta. Respondenten menade också att Volvo Parts var insatta eller tvingades bli insatta då många ändringsförslag kommer att rapporteras in vid tidpunkten för releasen.

Arkitektur

Vi bad respondenten redogöra för rollen som arkitekt. Han förklarade att arkitekten bland annat skall hitta en teknisk lösning som fungerar för applikationen. I det här projektet var arbetsgången dock något annorlunda. För det första var uppgiften i POW-projektet att skriva om ett befintligt system med en annan redan förbestämd teknik. För det andra har det på Volvo IT tagits fram en referensarkitektur som påvisade hur tillvägagångssättet skulle se ut för en Javalösning, en slags strukturerad mall som projektet följde till stor del. Respondenten upplevde att de fick bra hjälp av den referensarkitekturen så att det blev rätt från början. Dessutom fanns det verktygsstöd utformat av Volvo IT, för att snabbt kunna få fram en implementation av referensarkitekturen.

”… samtidigt har man ju väldigt stor hjälp av den här referensarkitekturen som jag pratade om, om vad som finns redan, så vi har ju den stöttningen direkt, det blir inga problem för oss liksom, vi tar den, så är det klart till mångt och mycket med lager och skiktningar och olika designer.

RUP pekar inte ut hur man skall dela upp en applikation i lager utan mer runt omkring Best Practices brukar man säga. Det står nog att man skall ha en lagerarkitektur, men inte hur man gör det och vilka lager och varför. Men det har de [Volvo IT] specificerat då. Så här visar ni gränssnittet med mera, hur bygger vi applikationer på företaget, man känner igen paketnamn och benämningar."

Utefter en analys av kravbilden samt de externa beroenden som fanns, tog arkitekten fram en design över applikationen baserat på J2EE. Därefter skapades ett Proof Of Concept[3] för att verifiera arkitekturen samt för att använda den som konstruktionsriktlinjer för utvecklarna. Kunden upplevde detta förfarande mycket positivt.

I arkitektrollen i POW-projektet ingick dessutom att dokumentera de artefakter som var väsentliga. Den viktigaste dokumentationen ur ett arkitektoniskt perspektiv, men även ur slutdokumentationssynpunkt, var Software Architecture Document[4] (SAD). Dokumentet skrevs inte initialt utan fylldes på allt eftersom. Respondenten menade att SAD var det största dokumentet som täckte i princip allt och det hade även referenser till andra dokument. Andra dokument som arkitekten ansvarade för var Supplementary Specification[5] och Visionsdokument[6].

Respondenten bekräftade vår teori angående att kraven var starkt sammankopplade med arkitektens arbete och att det har varit problematiskt på grund av 1:1-förhållandet i kravbilden. Arkitektmodellen påverkades dock inte av kravförändringar då dessa mestadels handlade om användargränssnittet.

"Jag har ju frågat efter dem [kraven] men det har ju kommit så där halvbra svar då. För det ingår i mitt jobb annars är det svårt att täcka en lösning när man inte vet vad det är som krävs."

Testhantering

Vi bad de olika respondenterna oberoende av varandra redovisa för vilka slags tester som gjordes och hur man utförde dessa i projektet. De tester som gjordes i POW-projektet enligt RUP:s två första faser, förberedelse och etablering baserades på en Proof of Concept. Där verifierades att den valda tekniska lösningen fungerade med verkligheten. För att inte göra den för stor valdes ett huvudflöde, ett visst testscenario ut, för att säkerställa funktionaliteten. Proof of Concept användes sedan som en slags mall för hur tillvägagångssättet skulle se ut rent arkitektoniskt. Även till Proof of Concept var det tvunget att skrivas tester som *"… testar att testet funkar…"* .

Utvecklarna utförde enhetstester väldigt sällan eftersom större delen av logiken fanns sedan tidigare. Gränssnitten och kommunikationen mellan de olika systemkomponenterna skulle inte skrivas om, det var bara att se till att tekniken fungerade.

---

[3] Proof of Concept, i det här sammanhanget, verifierar tekniken för ett huvudflöde, för att se att tekniken fungerar hela vägen genom systemet. (Ekhammar, 2006)

[4] Visar hur systemet är uppbyggt utifrån olika vyer, 4+1-vymodellen över arkitektur. (Kruchten, 2002)

[5] Visar krav som inte passar in i användningsfallen. (Kruchten, 2002)

[6] Visar intressenternas och användarnas behov samt en högnivåbeskrivning av systemets egenskaper. (Kruchten, 2002)

Tillsammans med representanter från Volvo Parts genomfördes tester med hjälp av prototyper. Dessa tester startades i början av den tredje fasen, konstruktionsfasen. I slutet av konstruktionsfasen utförde slutanvändaren så kallade User Validation Test. User Validation Testerna ingick i ett så kallat acceptanstestpaket och var mindre omfattande då slutanvändaren endast skulle testa ett fåtal scenarios. Affärsområdena testade mer rigoröst med andra tester i de fall som slutanvändaren inte kommer i kontakt med.

”De på Volvo CE kommer att installera först och de är duktiga på att testa. Vissa dealers har blivit utvalda att testa men de som testar väldigt noga är de som beställt systemet och som jobbar på Volvo CE Eskilstuna. Där finns tre stycken som sitter och testar. Dealarna är inga riktiga testare de går mer in och tittar att de förstår. Sen har de på Parts skrivit olika testfall, lägg denna artikel här gör si och så förväntat resultat med mera som slutkunden gör.”

I konstruktionsfasen började POW-projektet med systemtester där man testade de stora huvudflödena som ofta användes. Systemtesterna utfördes manuellt och höll på kontinuerligt. Ett försök till att automatisera systemtesterna gjordes men de fungerade inte fullt ut. Volvo Parts skrev tesfallen för acceptanstesterna men ansvaret att skriva testfallen för systemtesterna låg hos utvecklarna. Projektledaren granskade utvecklarnas testfall, då kvaliteten var skiftande. Respondenten upplevde att systemtesterna påbörjades i tid men att problemet med bland annat kravställningarna gjorde det svårt att veta om testfallen var korrekta.

”De här integrationstesterna kom in sent. Ja, allt kom lite sent egentligen. Men systemtesterna gör man sist, de som Inger håller på med nu då. Så där kom hon ju in rätt så sätt. Men jag vet inte ifall alla tänkbara scenarios var genomtänkta från början riktigt eftersom vi inte riktigt visste hur systemet var uppbyggt med alla de konstiga fallen då, eller features som vi inte visste fanns som måste in.”

Så kallade Performancetester där prestanda kontrollerades genomfördes parallellt med systemtesterna och utfördes av en speciell avdelning på Volvo IT.

Integrationstesterna kom in sent i projektet på grund av att de inte hade tillgång till de resurser som krävdes.

”Det är ett verktyg [Crusie Control] som då gör att de här testerna åt en automatiskt. Så det krävs en del uppsättningar också rent tekniskt och att även förbereda ramverken och så för testerna då, det är ju ett jobb och den personen hade vi inte förrän i februari.”

Vid integrationstesterna, *Continuous Integration*[7], gjordes tester med hjälp av testfall i de ”högre lagren” som anropade funktioner mot det riktiga systemet och teststubbar. Teststubbar användes som ett hårdkodat mellanskikt där modulerna istället för att kommunicera med varandra, pratar med stubben. Detta gjorde att tester kunde göras i ett tidigt stadium innan olika moduler var klara. Det verktyg som användes för att göra testerna automatiskt kallades Cruise Control. Varje gång ny kod hade skrivits, checkades den in i Cruise Control och testfallen kördes. Om ett test fallerade skickades ett mejl manuellt ut till den person som checkade in den felaktiga koden.

[7] Innebär kontinuerlig integration, utveckling och test. (Ekhammar, 2006)

Som verktygsstöd för att registrera fel som hittades användes IBM:s ClearQuest. De som testade, externt som internt, rapporterade buggar, via webbgränssnittet. IT-Projektledaren gick dagligen in i ClearQuest och granskade de fel som rapporterats och delade sedan ut dem, via systemet, till rätt utvecklare. Detta krävde att även utvecklarna gick in dagligen och kontrollerade om någonting skulle åtgärdas. När felet var reparerat, granskades och testades det av den person som rapporterade in buggen.

”Då får jag dem [buggarna] och så kan jag dela ut den till rätt person. Så det blir ju väldigt snabb kontaktväg. Då går man ju inte vägen som kund, via Åsa [systemägaren av POW, Volvo Parts] utan de som testar går direkt via ClearQuest till mig.”

Projekt Core Management System

- *Process: Agile Software Development*

> *”Men det fungerade utmärkt, det gjorde det. Men jag tror inte att det hade fungerat så bra om inte Charles hade som projektledare varit väldigt drivande, det krävs ju verkligen att någon har fått mandat och kan sin sak och orkar driva det. Om det hade vart någon som hade tyckt att det här hade vart en bra idé och sagt det i början på ett möte i projektet och alla hade sagt att ja, det hade kanske vart bra, då hade det ju inte hänt. Utan att det krävs ju att det är någon som är en stark person som har mandatet och kan driva det här. Och ordna upp det liksom, det krävs.*”

Bakgrund

Core Management System (CMS) var ett projekt som startade våren 2003 och pågick fram till mars 2004. Leverantören av systemet var Volvo IT och systemägaren var Volvo Parts. Användarna var Volvo Parts affärsområden, bland annat Volvo Construction Equipment (Volvo CE), och dess underleverantörer, *dealers*. Projektet var cirka 7 medlemmar stort. I det lättrörliga synsättet finns inga definitiva rollindelningar men i huvudsak fanns det, en projektledare tillika arkitekt, testansvarig och databasadministratör, en arkitekt som även var chefsdesigner, två Cobolutvecklare, två Javautvecklare, en kundrepresentant från Volvo Parts och en användarrepresentant från Volvo CE. Tekniken som användes var J2EE (Java) och IMS (Cobol).

Då det var en strikt tidsplan önskade IT-projektledaren att få mandat att driva projektet med en utvecklingsprocess inspirerat av Agile, främst XP och Scrum. Projektledaren hade tidigare brukat detta tillvägagångssätt i andra projekt utanför Volvo IT.

Volvo Parts behövde en gemensam applikation som skulle stödja samtliga affärsområden för att hantera affären kring återanvändning av gamla komponenter. Tidigare fanns inget centralt system som hanterade denna affär utan det fanns enbart systemstöd till vissa delar. Volvo CE, som var först ut att implementera CMS, skötte till exempel hela sin begagnataffär via Excel.

Medlemmarna valdes ut bland dem som fanns tillgängliga på avdelningen med fokus på deras tekniska kunskaper. Endast en person handplockades direkt av projektledaren på grund av hans tekniska kompetens. Det var även viktigt att få tillgång till en representant från kunden, Volvo Parts, då det lättrörliga arbetssättet kräver mycket tid och ett nära samarbete med kunden. Därför blev projektet tilldelat systemägaren från ett av de gamla systemen. Även en användarrepresentant från Volvo CE deltog aktivt i projektet.

Det var endast projektledaren som hade erfarenhet av det lättrörliga synsättet varför det ordnades en genomgång för medlemmarna där grunden för det lättrörliga synsättet lärdes ut. Någon mentor från organisationen fanns inte till hands, projektledaren axlade även den rollen.

Respondenterna

Charles Jobson
Charles var inhyrd konsult. Charles arbetade som, Business-projektledare, IT-projektledare, arkitekt, DBA och testansvarig. Han har utformat en egen lättrörlig utvecklingsprocess med inspiration från bland annat XP och Scrum.

Nils Rosén
Nils jobbade på Volvo IT under projektets gång. Han arbetade med arkitekturen, testningen, skrev tjänstespecifikationer och en del systemutveckling på J2EE sidan.

Anders Henriksson
Anders har arbetat på Volvo IT som Cobolutvecklare i fem år. Han var en av två Cobolutvecklare i projektet.

Alexander Pajari
Alexander jobbade på Volvo Construction Equipment. Han representerade användarna av systemet.

Projektplanering

Projektet följde två projektstyrningsmodeller. Enligt AB Volvo skulle man följa IS-GDP och från Volvo IT fanns interna riktlinjer att följa PCM. En av huvudartefakten för projektledaren var projektplanen. I denna beskrevs vad som skulle göras under projektets gång. Här bedömdes även hastigheten, *estimated time of arrival*, till de olika milstolparna, så kallade *gater*, i de två projektstyrningsmodellerna, IS-GDP och PCM.

Dokumentationen i CMS-projektet ansågs vara viktig. Däremot såg respondenten till att dokumentationen var normaliserad, det vill säga att bara de dokument som var nödvändiga för vidareutvecklingen av applikationen hölls levande. Inom projektet hanterades minimalt, men viktig dokumentation.

”Återigen ur ett arkitekturellt perspektiv så bestämde vi ganska tidigt då, vilka dokument vi vill jobba med som vi tycker driver detta framåt istället för att utgå från, vad säger PCM att vi skall jobba med, vad har vi nytta av? Vi har inte råd att hålla på med massa uppdatering av grejer som inte ger oss någonting utan vi vill uppdatera precis det som ger oss nytta framåt och som vi har nytta av att underhålla och hålla korrekt, dokumentation som vi inte håller uppdaterad och som

inte är korrekt vill vi inte släpa med oss under resan. Och det var en viktig del utav den här arkitekturella setupen att bestämma vilka de här artefakterna var och att detaljspeca de."

Av den dokumentation som skrevs var det bara några få som var av värde för utvecklingen av systemet. De var också viktiga i den bemärkelsen att de efter projektets slut skulle leva vidare till stöd för förvaltningsorganisationen. Den övriga dokumentationen på listan riktades i huvudsak mot de olika styrgrupperna.

"Så RUP plus Volvos interna arbetssätt med PCM och annat har drivit fram en väldans massa dokument som skall underhållas."

"Däremot finns det mycket artefakter i PCM som är bra, jättebra grejer, en verktygslåda bakom, men som gatemodell betraktat är det inte bra att vi har olika gatemodeller i PCM och IS-GDP, det ställer bara till röra och problem, det har ingenting med RUP eller Agile att göra."

Ett viktigt nyckeldokument att hålla uppdaterat var en så kallad Korsreferens. Då systemet var baserat på två plattformar med ett integrationslager däremellan fanns denna dokumentation som påvisade och höll ordning på hur funktionerna mellan de båda plattformarna och integrationslagren kommunicerade med varandra.

"Så var vi tvungna att hålla väldigt god ordning på fält, vad fälten heter och hur de var specade mot de olika världarna så att varje gång vi gjorde en ändring så hade vi järnkoll på hur dominoeffekten slog. Om man gör en SCR eller en ändring eller ett tillägg måste man veta både vad de fältnamnen heter i databasen, i DB2, i Cobolprogrammen, i meddelandet ut, och i javavärlden och i gränssnittet."

Projektet delades upp i faser om en månad och varje fas innehöll veckovisa iterationer som var strikt tidsbestämda. Iterationsplanen byggdes upp utifrån en mängd olika indatakällor såsom projektplanen, buggar från föregående veckas tester och SCR:er. På så sätt konstruerades en meny över vad som skulle genomföras i kommande iteration.

"Fördelen med en veckas iterationer är att man hela tiden kan få feedback och man kan se vad som händer på ett annat sätt. Man har avstämningspunkter som är viktiga avstämningspunkter."

Vid varje ny fas gjordes en övergripande plan på vad som skulle göras. Varje vecka gjordes sedan en uppföljning för att se hur grovmallen följdes.

"Buggarna från testen dagen innan, vilka ska rättas, vilka vi inte skall rätta. Vilket i det av månadsplanen vi hade planerat att göra, vad skall vi göra nästa vecka och vad skall vi ta bort och inte göra alls. Bygga ut konceptet, göra nya prioriteringar och ta in nya krav löpande. Så jobbade vi då"

"Det här veckomötet som vi hade på en halvdag där vi gjorde planering, det är ingen big deal tycker jag och det är ett bra verktyg för projektledaren att jobba med."

"Jag tycker ju att det var en väldigt tilltalande metod. Att korta de här ställtiderna, och få så snabb feedback och göra så lite av de här förarbetena som möjligt, att inte behöva planera så där väldigt, väldigt långt i förväg, utan försöka göra småbitar och försöka göra de klara för att få

feedback på de så fort som möjligt. Jag tycker det är ett jättebra angreppssätt, mycket bra på många sätt."

För att bedöma hur mycket medlemmarna i projektet skulle klara av att utföra i varje iteration, användes en kalkyl[8]. Varje projektmedlem var delaktig i att skatta minimum oh maximum för hur lång tid en viss uppgift skulle ta.

"Då kan man även i tidiga lägen tidsbedöma saker men med en definierad osäkerhet. Och det jobbade vi med i tidig fas rätt igenom hela projektet och när det började bli tal om att bygga, i de här veckoiterationerna, då var det ju en del av veckoplaneringen att bedöma så att vi fick tillräcklig munsbit att göra för en vecka så att vi inte fick för mycket som vi inte skulle kunna klara av under en vecka. Och också kontinuerligt bedöma speeden framåt. Det är något som Agile projekt har i fokus. Att hela tiden stämma av och mäta med vilken hastighet man rör sig: Vad är vår utvecklingsspeed?"

"Då tog man med sig det till nästa sak man bedömde så lär man sig, hela projektet lär ju sig i princip hur mycket man levererar och hur fort det går och så vidare. Det är ju bra både för personerna och för projektledaren och kunderna som då kan få en bättre uppskattning hela tiden"

I projektet ställdes det krav på att kunden aktivt skulle vara med i utvecklingen genom att delta i tester och prioritera vilka funktionaliteter som skulle ingå i systemet. Respondenten menade att trots att kunden ibland tyckte det var påfrestande, så var kundens delaktighet viktig, då kunden alltid "har rätt". I planeringen ingick även en testsession med kunden en halv dag varje vecka. Även utvecklarna kunde uppleva situationen påfrestande med att träffa kunderna såpass tätt då de alltid skulle kunna visa upp körbar kod för denne.

"Det är ju också väldigt stressande. För inför något som du skall publicera och som de skall testa, så måste det ju fungera liksom, och då får du ju mer såna här hårda punkter som måste bli klara. Man kan inte dra på det utan det här är ju en deadline. Och så har man sådana här deadlines väldigt, väldigt ofta. Som i till exempel ett RUP projekt där man har deadline var sjätte vecka i princip, så har vi här nu en varje vecka! Det är ju väldig skillnad, men å andra sidan så blir det ju otroligt mer produktivt. Så när du gör något så blir du tvingad att göra det klart. Och så får du ju begränsa ditt arbete så att du liksom får timeboxa ditt scope hela tiden så att du får göra något som blir klart, som du kan visa och testa, börja om från början, göra något som är hyfsat klart och så testa igen."

En av utvecklarna belyste det han tyckte var en fördel med aktivt deltagande från kunden – enligt nedan.

"För då vet de var de får. Och kommer inte i slutet och lägger till svåra funktionaliteter då. Om man säger och ändra då. Utan man får de här grejerna ganska tidigt, vad de vill ha, vad de kräver egentligen. Och de förstår själva vad det é de har beställt, och då är det tacksamt att få det tidigt, innan man byggt på för mycket funktionalitet så att det blir svårt att ändra."

[8] En kalkyl som tar min och max och räknar ut en ny hoptryckt min och max intervall, som påvisar osäkerheten i bedömningen (Jobson, 2006).

Två gånger per vecka var uppbokade i förhand med kunden för test och planering, däremellan fanns utrymme för daglig konversation. Angreppssättet var att från första början få kunden att känna sig som ägare till applikationen, varför respondenten tyckte att det inte det var några större problem att få kunden att medverka i den mån som behövdes. Han upplevde att kunden hade ett stort ansvarskännande och var entusiastisk inför utvecklingen av systemet. Det var emellertid för kunden naturligt att känna viss tvekan mot tillvägagångssättet i början.

”Ja, det var ju lite knorr i början det var det ju. ‘Skall vi behöva vara med så ofta… ‘ och så. Men det släppte jättefort när de väl hade kommit in i det. Då tyckte de att det var kalasbra och sitta med, och se vad som händer De får ju en helt annan upplevelse av projektet då, när det ser vad som händer, vilka funktionaliteter som finns klara i olika skeden och så vidare. De kände sig mycket mer delaktiga och ansvariga för det som kom ut också. Så, det är väl en tillvänjning för kunden också. ”

Användarrepresentanten upplevde sin delaktighet som fördelaktig då utfallet blev att systemet i högre utsträckning motsvarade Volvo CE:s förväntningar i jämförelse mot ett traditionellt förfarande. Däremot ställde sig respondenten tveksam till att i fortsättningen delta i ett liknande projekt utifrån hans perspektiv som individ.

”Det är alltid påfrestande att vara med i projekt under en sån här lång tid. (---)Med tanke på att du har ett linjejobb att utföra. Det krävs ju mycket utav dig.”

Hela projektteamet från Volvo IT satt tillsammans. Kundrepresentanten från Volvo CE var lokaliserad i Eskilstuna och deltog därför oftast vid testerna via NetMeeting. Ibland åkte delar av projektet också till Eskilstuna för att genomföra tester tillsammans med kund. Systemägaren var lokaliserad i Göteborg och kunde på plats medverka vid testerna.

Projektet arbetade med riskhantering från första dagen, berättade respondenten. Tillsammans med uppdragsgivare Volvo Parts och Volvo CE gjordes en riskmodell. Modellen byggde på att alla parter tillsammans identifierade de risker som kunde uppstå. Därefter prioriterades och organiserades riskerna, och konstruktiva lösningsförslag diskuterades gemensamt fram. Det ansågs nödvändigt att alla parter var överrens om alla risker från första början.

”För är man inte överens med problemen när man börjar lösa grejer så spelar det ingen roll vad man hittar på för käcka saker det blir aldrig bra ändå.”

Riskhanteringen fortsatte sedan löpande under hela projektets gång. En riskdriven ansatts är inget unikt för att arbeta lättrörligt, förklarade respondenten och menade att det är ett generellt förfaringssätt för projektledning. Projektet följde en mall enligt PCM, över hur risker värderades och hur stor sannolikheten var att risken skulle komma att inträffa.

En huvudrisk i projektet var beroendet av externa system som respondenten kände att han inte hade kontroll över. En annan risk var att arkitekturen inte skulle fungera, vilket hanterades genom att testa den mycket i början och alltid ha alternativa lösningar. Arkitekturen tillämpades för första gången i CMS-projektet men hade testats i en mindre skala i det projekt där den togs fram.

Vi avslutade vår första intervju med en av respondenterna med att fråga om han upplevde att processen gav stöd för hur han utförde sitt jobb som IT-projektledare. I huvudsak menade

respondenten att projektledarens inställning till arbetssättet och hur kundorienterad denne är spelar stor roll. Har man inte denna kundorienterande inställning så kan det lättrörliga synsättet ses som mödosamt.

”Det beror på vilket mindset du har som projektledare. Om du som projektledare har mindsetet på att dina kunder skall bli lyckliga och att det skall bli en implementation som ger dem mervärde i businessen i slutändan, fokuserar på Business Caset, att få ut effekterna i affären, om det är det som är ditt fokus då bidrar Agile jättemycket. Men om du istället har lite mera inskränkt traditionellt projektledarfokus, att få ut, Most of the Buck, i businessen, det är inte mitt ansvar det skall Change Managementpersonen i businessen ta hand, om du har det mindsetet, då kan detta sättet uppfattas som jobbigt för då måste du hela tiden ändra ditt scope, du skall ta in nya krav som kommer in och som ger ett bättre Business Case och arbeta om design och att arbeta om vad du skall leverera, ta bort saker som ni har kommit överens om innan och lagt möda på osv, så har du det mindsetet så uppfattas ju det här som jobbigt. Det handlar om hur du funkar som projektledare och vad du har för mindset i din roll som projektledare hur du uppfattar Agile kontra traditionellt, skulle jag vilja säga. ”

Kravhantering

Innan CMS-projektet sattes igång utfördes en gedigen kravanalys genom att först se om det fanns någon paketlösning som kunde tillämpas. Så fort det hade bestämts att det blev ett in-house projekt gjordes en webbaserad prototyp som levde under hela projektets gång. Där visualiserades funktionaliteten som hårdkodades med ett enkelt användargränssnitt. Prototypen visades upp på en workshop där kunderna fick komma med åsikter.

”Så jag satt och hackade frontpage live med massa, ett stort gäng människor och frågade menar ni så här, menar ni så här, målade upp grejer, exemplifierade Use Case skrev Use Case texten, tillbaka till HTML-hacket, fram och tillbaka, fram och tillbaka.”

En av respondenterna antydde vikten av att vara tydlig i kommunikationen vad gäller användning av prototypdriven utveckling.

”Och det var med den [prototypen] man kommunicerade väldigt mycket med kunden, innan man började utveckla. Lite farligt är det ju när man visar en sådan, det ser ju ut som om den är klar. Man lägger ju länkar och trycker man på orderlänken et cetera, fast allt är hårdkodat. Så kunden tror att ”jamen det här är bra”. Vi använder det som är liksom. Så är det alltså. Man får vara väldigt noga med i början där att kommunicera att det här är bara hårt, det finns ingen funktionalitet liksom, ingen logik i detta här. För annars blir det missförstånd, helt klart. Det är lätt för oss att säga som är systemutvecklare men den som inte är systemutvecklare förstår ju inte detta.”

Samma respondent fortsatte:

”Fast det är ändå en fördel att jobba så. När vi jobbade med HTML då är det ju typ samma beskrivningsspråk som vi jobbar med sen när vi implementerar alltihop. Det är ju också en fördel. Så man kan ju i princip ta de hårdkodade byggena och utgå från sen när man skall göra originalet.”

Projektet var kontrakterat till ett fast pris och vid större kravförändringar, SCR:er, gjordes därför nya värderingar av kraven för att prioritera bort de krav som var minst viktiga och inte kunde utföras inom tidsplan och budget. Respondenten menade att utgångspunkten för projektet var att kunden alltid hade rätt och att varje kravförändring hade ett värde för kundens affär. För att hjälpa kunden att prioritera bland kraven, visualiserades de olika alternativen med hjälp av prototypen. Om kunden fann det omöjligt att prioritera bort något av kraven fick styrgruppen fatta ett beslut, då ett tillägg av krav innebar ett tillägg av kostnad och tid.

”Och just att få fram de valsituationerna, driva de besluten, var vi väldigt aktiva med. Det uppskattade ju kunden, att man inte lämnade de här stenarna ovända utan tidigt med en gång bottnade i och förstå vad de ville åstadkomma.”

Arkitektur

Innan starten av projektet gjordes en enkel beskrivning av arkitekturen inför utvärderingen av en eventuell paketlösning. I nästa fas fastställdes principerna på hur arkitekturen skulle se ut, vilka huvudkomponenterna var och de mest väsentliga funktionerna och deras ansvar. Därefter vidareutvecklades, allteftersom i iterationerna, en mer detaljerad ritning över arkitekturen på systemet, en blueprint.

Projektets huvudarkitekt ansåg att det var väldigt viktigt att ta in utvecklarna i ett så tidigt skede som möjligt när man arbetade med arkitekturen för att få deras feedback och bekräftelse på att de förstod tankegången. En annan respondent menade att det är en framgångsfaktor som arkitekt att vara med och koda för att få en större förståelse för strukturen.

”Annars kan man tro att man pratar samma språk och så bygger de på ett annat fall i slutändan ändå som gör att det blir tokigt då.”

Respondenten ansåg att kraven var viktiga att arbeta med när man tog fram arkitekturen men att han sett många exempel på arkitektur som inte var särskilt genomtänkta, där Use Case efter Use Case hade implementerats och där en så kallad stuprörseffekt genererats.

”Alltså bruttomängden av vad alla de här komponenterna skall göra är ju helt drivet av kraven det är inte snack om det, det är bara det att de är omsorterade för att supporta en smart design, det är inte så att man tagit analysen och varit lat och tagit de kraven rätt av och sagt att nu kodar vi rätt upp och ner från analysspecen, pang! Och så blir det också design här per default då blir det fel hävdar jag, då blir det riktigt fel.”

I CMS-projektet arbetades det istället utefter en normaliserad approach där de olika arkitektoniska skikten hade separerade ansvar. En del av den arkitektoniska inställningen var också att driva en regeldriven ansats där det skrevs generella tjänster som kunde utökas. Målet med arkitekturen var att få den såpass flexibel att även andra verksamheter skulle kunna dra nytta av systemet och kunna bygga på funktionalitet efterhand. Båda arkitekterna hade i ett tidigare projekt varit med och utformat en ny arkitektur. CMS-projektet blev en pilot för att tillsammans med Volvo Parts och Volvo IT verifiera den framtagna IT-arkitekturen. En pilot där Volvo IT för första gången byggde på ett servicebaserat tankesätt i Cobol.

Huvudarkitekten och arkitekten fick frågan om de upplevde något i det lättrörliga synsättet som underlättade deras roll som arkitekt. En av dem nämnde att den testdrivna ansatsen som finns i det lättrörliga angreppssättet, och som tillämpades i CMS-projektet, genom att testkoden skrevs innan designspecifikationen var färdig fullt ut, hjälpte arkitekten att upptäcka detaljer i designen i ett tidigt skede.

”Poängen med det, att man skriver testspecifikationen först innan man är klar med designspecen fullt ut, det hjälper arkitekten så till vida att man får mycket tidigare koll på detaljer i implementationen. Man får mycket tidigare en handskakning på att kraven, så som de var beskrivna i Use Case, och en analys, är konsistenta och håller för en implementation. Då kan man ta upp en diskussion med businessen i ett mycket tidigare läge runt inkonsistens, mot motstridande krav, prioriteringar, är det här viktigare än det här och så vidare. Den typen av viktiga beslut och driva, vad är viktigast att bygga. Hela det arbetssättet underlättar arkitektens jobb jättemycket. För annars hamnar man lite grand i en de facto situation, att du kodar baserat på en analysspecifikation, när du sitter och kodar upptäcker du massa inkonsistens och frågar analytikern, hur sjutton har du tänkt här det här hänger inte ihop, stackars analytikern har inte satt sig in i det här problemet på den nivån så att han kan svara, han går tillbaka till kunden och kunden fattar ingenting, här har du tänkt helt fel, vad sjutton, dessutom var det ett antal månader sedan han satt med den här analysen så han kommer inte ihåg.(...) Så att ur en arkitekturell sypunkt får du input för att omvärdera din arkitektur och göra refactoring löpande varje dag, varje vecka istället för att du tar en större smäll senare i implementationen och sluttestet där du kanske får jättestora, eller borde göra jättestora arkitekturella förändringar men inte orkar göra det, det tar för mycket stryk så att du kan nå din önskade arkitektur, och landa i en arkitektur som du är nöjd med i och med att du har den här kontinuiteten.”

Respondenten menade vidare att på detta sätt skapas en arkitektur som stöttar det användaren faktiskt har nytta av och inte en arkitektur som stöttar analysen. Han menade vidare att arbetssättet som beskrevs, dock kan upplevas som arbetsamt då beredskap måste finnas för att omvärdera och bygga om arkitekturen. Dessutom krävs det att koden som skrivs är smart designad det vill säga generell och återanvändbar.

I projektet försökte en korsvis programmeringsmetod tillämpas, där en person specificerade och en annan kodade och vice versa. En av utvecklarna menade att det angreppssättet gav honom mer utmaning som programmerare genom att hjälpa till att specificera tjänster. Dessutom fick han en bättre inblick i vad som var möjligt att göra.

”Och det funkade väl jättebra tycker jag. Det är ju ett bra sätt att jobba tycker jag, för då får man ju det dokumenterat på något sätt. Om man skall skriva specen och skriva koden själv, så är ju risken att man inte dokumenterar, att inte det som behövs blir dokumenterat så att det inte är koden som driver jobbet. Men om det är någon annan som skriver specar så blir det ju specningsarbetet som driver jobbet och det är ju en fördel för då blir ju specen lite mer komplett.”

Samme respondent upplevde de korta veckoiterationerna som positiva då det medförde korta ledtider på de tjänstespecifikationer han skrev åt utvecklarna och att utvecklaren då kom igång att skriva kod tidigare. Detta innebar också en snabb respons på specifikationerna och att fel upptäcktes i ett tidigt skede.

”Och så att inte jag sitter i två månader och skriver specen och trycker in alla möjliga varianterna som jag kan tänka mig uppkomma innan jag egentligen får någon feedback, så är det ett mycket mer tidseffektivt sätt att jobba så som jag ser det.”

Testhantering

Projektet arbetade utefter en testdriven ansats, vilket här innebar att koden för testklassen skrevs innan koden för funktionerna. Projektledaren berättade att hans sätt att se på lättrörliga metoder innebär ”breda roller” varför ingen utpekad testare fanns. Alla utvecklare var tvungna att testa, det ingick att göra Unit Test i den delen som hade kodats. I praktiken satt mestadels arkitekten och en till och skrev Unit Tester. I CMS-projektet genomfördes integrationstesterna manuellt och varje dag innan en utvecklare gick hem var dennes kod tvungen att vara kompilerbar då utvecklarna var beroende av varandras koder. Enda verktygsstödet som fanns var utvecklingsmiljön, WebSphere Application Developer, (WSAD), där enhetstesterna byggdes upp och uppdaterades. I projektet använde man så kallade Mockobjekt[9].

”Men när man arbetar med Unit Tester är det också bra att jobba med Mockobjekt. Man kanske sätter upp sitt Unit Test och jobbar mot interface som är Mockade till att börja med sen när man skriver funktionaliteten så skall ju testerna fortfarande fungera. Men vi automatiserade aldrig Unit Testerna, vi hade ingen Automated Build, det hade vi inte, utan vi skrev en del Unit Tester som vi körde manuellt.”

Varje vecka ägnades en halv dag för att testa tillsammans med användarrepresentanter samt att skriva testprotokoll. Testerna genomfördes utefter testspecifikationer vilka var Use Case drivna. Specifikationerna skrevs av arkitekten tillsammans med representanten från Volvo Parts.

”Då satt vi utspridda i små grupper. Så då satt jag ihop med systemägaren kanske och testade sen sprang X vidare till någon annan och testade med någon annan. Så vi satt i små klungor kring tre, fyra maskiner kanske. Så hade vi satt Alexander upp i Eskilstuna att köra sin test. Och så hade vi konversation på telefon och mejl och grejer undertiden som han testade. Vi testade olika saker parallellt under en halv dag.”

”Och sen att sitta tillsammans med användaren att testa det är ju något som jag tycker att utvecklaren också skall göra. Om man jobbar Agile mässigt. Det är då man får en mer detaljerade förståelse för kraven, man får feedback på det man har byggt - Och direkt. Och få en bra diskussion med prioriteringar när det gäller ändringar i det man gjort direkt med kunden.”

”Det jag kan säga också tycker jag är väldigt bra med en metod där man har hyfsat snabb feedback där man inte behöver vänta i 5-6 veckor innan testarna skall dra igång sitt arbete för att få feedback på det man gjort som systemutvecklare, det är en väldigt bra grej då minskar man ju på ledtiderna och snabbar på utvecklingen väldigt mycket alltså.”

Som verktygsstöd till testerna och felrapporteringen utformades ett testprotokoll i Excel där man kunde följa testfallen och deras resultat. Ett synsätt i lättrörliga metoder, menade projektledaren,

[9] Mockobjekt; jobbar med förväntade resultat. Man talar om för objektet hur det förväntas samverka i form av metodanrop, resultat och kastade undantag med det objekt man verkligen vill testa och i slutet av testningen verifierar man att ens förväntningar har uppfyllts. (översättning från http://martinfowler.com)

antyder att kunden skall äga dokumentet och jobba med det själv och skriva in testfall. Projektledaren ansåg att detta var något som inte lyckades fullt ut då han kände det nödvändigt att hålla kontroll över testprotokollet, och stödja kundrepresentanten. Med tiden blev kunden mer van vid arbetssättet och kunde arbeta mer självständigt med testprotokollet. Varje vecka gjordes så kallade regressionstester där det verifierades att föregående veckas förbättringar fungerade.

På vår fråga hur det gick att köra testfall per telefon tyckte projektledaren att det fungerade bra i det här projektet.

”Jag skall säga, att ha kört hela projektet på det viset hade aldrig funkat men jag var i Eskilstuna [Volvo Construction Equipment] ofta, jag och Nils åkte ju dit ganska ofta, säg var tredje vecka, varannan vecka, i början mera och allt efterhand mindre och sen på slutet när vi skulle rulla ut grejerna jobbade vi jättemycket tillsammans igen, åkte till England tillsammans och rullade ut det hos den största dealern där som första pilotdealer. Och Eslöv som var Core Hub och så vidare, så vi hade ett intimt samarbete med Eskilstuna hela vägen ut, i och med att vi kände varandra och lärde känna varandra tidigt hade vi en bra relation så det funkade med telefon men vi kunde ju inte ha kört det här racet bara med telefon, det hade aldrig gått.”

Diskussion

I detta kapitel har vi för avsikt att diskutera och analysera vår inhämtade data gentemot våra teorier och vårt syfte. Liksom i empirin har vi delat upp kapitlet utefter våra fyra processområden. Efter varje område följer en utvärdering utefter CMMI. Avslutningsvis ges ett övergripande resonemang som ett underlag till vår slutsats.

Projektplanering

Båda projekten var beroende av två olika styrmodeller från Volvo IT och AB Volvo. Båda projektledarna för respektive projekt upplevde fördelar med PCM på så sätt att de innehöll riktlinjer för viktiga artefakter. Däremot riktas det delvis kritik från CMS-projektet, mot att följa två gatestrukturer och att styrmodellerna även drev fram artefakter som inte tillförde processen något av värde och inte heller drev projektet vidare i utvecklingen. Från POW-projektets sida nämndes inte det problemet, däremot ansåg projektledaren att föreningen mellan RUP och PCM gav ett bra stöd. Men en av respondenterna från POW-projektet pekade dock på konsekvensen av att dokumentationen skulle följa vissa riktlinjer då resultatet av detta blev för mycket formalia och endast en liten del av dokumentationen var essentiell. En förklaring till varför de båda projektledarna hade olika upplevelser av styrmodellernas inverkan på projekten kan ligga i projektens struktur. Vi anser att PCM med dess delmål, *gater*, följde en sekventiell utvecklingscykel och därför inte synkroniserade med en starkt iterativ utvecklingsprocess som CMS-projektet tillämpade. Då POW-projektet hade få iterationer, är det troligt att PCM:s gatestrukturer passade bättre och därför inte upplevdes som direkt störande utan istället såg till dess fördelar. Men ser man till en process som har fördelar av att vara starkt iterativ upplever vi att PCM:s struktur kanske bidrar mer till att bromsa effektiviteten i utvecklingsprocessen än till att stödja den.

I produkten RUP kan man se att där finns riktlinjer för att kunna producera många artefakter men tanken med processen är dock att denna skall skräddarsys efter projektets karaktär och därför behövs endast en utvald mängd artefakter produceras. (Lunell, 2003; Kruchten, 2002). Risken är att denna typ av process skulle kunna komma att användas för strikt, vilket gör att det läggs större fokusering på att dokumentera än på att systemutveckla. (Boehm & Turner, 2004) Vår uppfattning är att man i lättrörliga metoder ser dokumentationen som en viktig del av en process men det får inte ta över processen att producera körbar kod. Resultatet från vår studie av POW-projektet visar på vikten av att upprätthålla en god dokumentation av systemet, när vi ser vilken utmaning 1:1-förhållandet medförde. Problemet med den vaga kravbilden bottnade sig i att det inte funnits någon fullständig dokumentation från utvecklingen av POL att upprätthålla.

En av respondenterna från POW-projektet belyser risken för att fastna i en process som RUP, med mycket dokumentation att hålla levande, och som inte levererar något av värde för kunden. Vår uppfattning är att projektet inte körde fast med att producera dokumentation efter konstens alla regler. Projektet lyckades istället balansera mängden dokumentation bra, dels tack vare projektledarens kännedom om risken, dels genom den avskalade Volvoanpassningen av RUP och dels med hjälpen av mentorn för att ytterligare anpassa RUP till projektet.

I CMS-projektet bemästrade projektledaren det upplevda problemet med att upprätthålla en mängd dokumentation genom att separera artefakterna åt. Endast en viktig mängd, normaliserad dokumentation hölls levande inom projektet.

Projektledaren i respektive projekt arbetade i förväg ut en plan och delade in projektet i iterationer och faser. Skillnaderna är dock tydliga. CMS-projektet arbetade utefter en metodik som starkt förespråkade korta tidsbestämda iterationer (Larman, 2003). RUP förespråkar däremot längden på iterationer till mellan två till sex veckor. Så kallade normala RUP projekt kan innehålla mellan tre till nio iterationer och typfallet ligger på sex iterationer. (Kruchten, 2002) POW-projektet med sina tre iterationer hade en låg iterationsmängd. Ser vi på CMS-projektet hade de månadsvisa faser likt det Scrum förespråkar med en grovplan över de mål däremellan, där de ytterligare delade upp projektet i veckovisa iterationer influerat av XP (Larman, 2003). Utifrån den information vi erhöll upplever vi både nackdelar och fördelar med båda förfaringssätten. Att bygga upp med veckovisa iterationer upplevdes ibland som stressande och vi får känslan av att det kanske inte fanns tid att ”hämta andan”. Däremot är inställningen att korta iterationer är produktivt och lärande, lärande i den mening att gruppen får en förståelse för hur lång tid en uppgift tar. Det ger i sin tur en bra uppfattning och ett bra mätetal på hur projektet ligger till utefter tidsplanen. Däri ligger också en skillnad mellan projekten, hur uppgifterna skattades. I CMS-projektet byggde hela planeringen på att varje medlem var med och uppskattade tiden. Därefter använde projektledaren en kalkyl för att göra en ytterligare estimering. I POW-projektet skattades uppgifterna och iterationslängden till största delen utifrån projektledarens egna erfarenheter.

Larman (2003) delar in processer utefter ett projekts förutsättningar och menar att det generellt lämpar sig att tillämpa en definierad process på förutsägbara och stabila projekt. Den kravbild som projektet ställdes inför, upplever vi indikerar på en relativt förutsägbar och stabil problemställning. Utifrån ovan nämnda resonemang upplever vi det naturligt för projektet att driva projektet med en definierad process som RUP och dessutom inte planera för många iterationer. Men det projektet senare ställdes mot var en osann kravbild därför anser vi att fler iterationer i planeringen kunde ha varit befogat.

Kunden var i bägge projekten representerad av Volvo Parts systemägare. Vår uppfattning är att kunden har en mer uttalad central roll i lättrörliga metoder än i RUP, på så sätt att denne förväntas aktivt delta under hela utvecklingscykeln (Kruchten, 2002; Martin, 2003). Skillnaden ser vi också i hur kundens delaktighet utnyttjades i båda projekten. Här tror vi att CMS-projektet hade en framgång vad gäller deras inställning till kundens delaktighet. Tidiga tester med kunden och aktiva prioriteringar mellan krav är vad som krävdes av kunden. Kunden hade likaledes en stor delaktighet i POW-projektet, i regel lika täta sammankomster som i CMS-projektet, skillnaden ligger snarare i att tidigt nyttja en kundrepresentant och en användarrepresentant för feedback på systemets avancerande. Här ställer vi oss frågan hur en användarrepresentants delaktighet i tidiga tester skulle ha kunnat påverka och hjälpa POW-projektet att hantera den bristfälliga kravbilden. Om vi ser till IKIWISI-effekten (Kruchten, 2002) hade kanske angreppssättet hjälpt användaren att identifiera applikationens verkliga krav tidigare och på så sätt underlättat för utvecklingen av systemet. En aktiv kunddelaktighet som krävdes i CMS-projektet ställer däremot stora krav på kunden. Kunden kan komma att uppleva deltagandet som ett för stort engagemang vid sidan av sitt vanliga linjearbete och vi tror att det kan vara svårt att övertyga kunden om nyttan i det denne kan göra för utvecklingen av programvaran.

Mallen för riskhantering enligt PCM följdes av de båda undersökta projekten. I CMS-projektet startade de riskhanteringen med att använda en modell för att på så sätt förebygga konflikter mellan samtliga parter om vad som kunde uppfattas som risker. Projektledaren i POW-projektet belyste problematiken kring att kombinera rollen som projektledare och rollen som arkitekt och detta dementeras inte heller av CMS:s projektledare. Projektledaren från POW-projektet pekade också på konsekvensen av att mixa dessa två roller, vilket vi anser att Volvo IT bör ta i beaktande.

Utvärdering utefter CMMI: Projektplanering, Nivå två

Enligt CMMI skall en planering finnas, följas och uppdateras. Detta tycker vi att båda projekten har uppfyllt. I POW-projektet följdes de fyra faserna i RUP och i CMS-projektet gjordes varje månad en grovplan som uppdaterades varje vecka. Båda projekten hade kontinuerlig kontakt med kunden och var flexibla vad det gäller hantering av kravförändringarna. I POW-projektet följdes PCM:s mall för riskhantering, och även CMS-projektet hanterade riskhanteringen väl. Vi tycker utifrån CMMI:s förutsättningar att hantera projektplanering, att både POW-projektet och CMS-projektet uppfyller kraven för Nivå två.

Kravhantering

Kravhantering är en central aktivitet i systemutvecklingen av en programvara. (Fowler & Highsmith, 2001; Kruchten, 2002; Lunell, 2003) Så som Kruchten (2002) och Lunell (2003) beskriver kraven, är dessa ett utryck av de behov en eller flera intressenter har och som skall uppfyllas av ett system. En bristfällig kravhantering kan resultera att kravkvaliteten försämras och kunden blir missnöjd (Kruchten, 2002). Förändringarna kan antingen ses som ett hot eller en möjlighet i utvecklingen av en programvara. Det lättrörliga synsättet ser då snarare kravförändringar som en möjlighet som utnyttjas till kundens konkurrensfördelar och uppmuntrar därför till löpande kravförändringar. (Fowler & Highsmith, 2001)

Vid startpunkten för CMS-projektet fanns en sammanställd kravanalys som låg till grund för utredningen om alternativet att köpa in en paketlösning. När POW-projektet påbörjades fanns det inte någon ny analys av kravbilden, ”*gör som det gamla*” var utgångspunkten. Ett missnöje har uttryckts från både IT-projektledaren och systemägaren kring den olyckliga kravställningen. I och med att applikationen i sig var ett ”lätt” system att utveckla, uppfattar vi problemet relativt hanterbart men hade ändå en viss negativ inverkan. Vi har uppfattningen att IT-projektledaren och systemägaren också hade god kontinuerlig kontakt och därför kunde hantera mindre kravförändringar på ett smidigt och flexibelt sätt allteftersom.

När vi i vår undersökning ställde frågan hur ett lyckat projekt definieras, svarade kunderna i stort att leverantören skall ha levererat i tid med kvalitet och uppfyllt de krav som fanns. (Appendix C) Kruchten (2002) skriver också att, hur mycket ett system uppfyller kundens behov, är ett sätt att mäta kvalitet. Vi är förvånade över hur det både från Volvo Parts och Volvo IT:s sida inte fanns rutiner för att hantera en omskrivning och dess krav på ett bättre sätt än vad som påvisats i resultatet från vår studie. Vi menar att det rimligtvis från organisationens sida borde finnas ett intresse att tillgodose kunderna och bygga programvara som uppfyller kundernas behov. Att initiera ett projekt med den kravbild POW-projektet hade finner vi en aning godtroget med tanke på att det gamla systemet hade uppenbara tekniska brister. ”*Gör som det gamla*” kan ge en intuitiv känsla, tror vi, av att problemet är förutsägbart och enkelt. Kan effekterna av den dåliga kravbilden ha mildrats, om man likt ett lättrörligt synsätt tagit med kunderna i ett tidigare skede.

Kunden kunde i det här fallet också ha bidragit genom att på ett bättre sätt säkerställt att den kravbild som visades upp faktiskt pekade på de behov deras system skulle uppfylla.

En framgångsfaktor hos CMS-projektet, som vi upplever det, är att man tidigt med utgångspunkt från grovanalysen och tillsammans med kundrepresentanterna visualiserade och analyserade fram en initial kravbild med hjälp av en prototyp. Användandet av en prototyp i HTML kan hjälpa kunden att på ett verklighetsnära sätt illustrera och förstå deras egna verkliga behov.

Vi upplevde att CMS-projektet arbetade utefter en flexibel inställning till hur kravförändringarna hanterades genom att värdesätta varje kravförändring som ett gagn för kundens affär. Att användarrepresentanten från Volvo CE endast till viss del kände det gensvaret som önskades inför kravförändringarna, tror vi beror på den arkitektoniska lösningen, att implementationen av de specifika krav ett affärsområde har, kan komma att dröja något. Genom att designa ett system utifrån en defaultlösning, för att efterhand kunna realisera specifika funktionella krav tror vi ökar förutsättningarna till konkurrensfördelar för Volvo Parts på så sätt att Volvo Parts kunder, affärsområdena, får möjligheten att skräddarsy systemet utefter sin verksamhet.

Utvärdering utefter CMMI:s Kravhantering, Nivå två

När vi ställde Kravhanteringen på CMMI:s Nivå två mot kravhanteringen i POW-projektet såg vi att projektet hade en definierad kravhantering. Allt eftersom kraven kom in granskades de och övervägdes om de ingick i omskrivningen eller inte. I CMS-projektet fanns en gedigen kravhantering där kunden var med och prioriterade och diskuterade vilka krav som skulle införlivas. Vi fick uppfattningen av att båda projekten försökte upprätthålla en dokumentation utav kraven under projektens gång. Men eftersom vi inte har haft någon kontakt med förvaltningsgruppen går det inte att verifiera om uppdateringen fortskridit. Eftersom kravbilden inte var den bästa kan man inte rikta kritik bara mot POW-projektet utan även mot kravställarna. Vi tycker att trots den olyckliga kravbilden hanterade POW-projektet kraven på ett sådant sätt att de kom upp på Nivå två. Vi anser även att CMS-projektet lyckades uppnå Nivå två.

Arkitektur

RUP och lättrörliga metoder har liknande förhållningssätt vad gäller arkitekturens centrala plats i utvecklingen av ett system, och rekommenderar en så tydlig och förändringstålig design som möjligt för att systemet sedan skall kunna byggas och bytas ut. (Kruchten, 2002; Lunell, 2003; Fowler & Highsmith 2001) Båda projekten hade ett arkitektoniskt tankesätt till grund när utformningen av arkitekturen för applikationerna ägde rum. Båda systemen skulle utvecklas med en stabil och pluggbar arkitektur.

Vi ser arkitekturen som en initial risk hos de båda projekten. I CMS-projektets fall då respondenterna menade att den arkitektur de skulle tillämpa var ny för Volvo IT och Volvo Parts. I POW-projektets fall gällde det att inte begå samma misstag med att implementera en ostadig arkitektur likt föregångaren. Lunell (2003) menar att i vissa fall när det råder en osäkerhet kring de idéer man har kan ett ”arkitektoniskt provskott” utföras. I POW verifierades arkitekturen genom en Proof of Concept. Det var något som kunden uppskattade och hade saknat i tidigare projekt. CMS-projektet hanterade den arkitektoniska risken med tidig testning och dessutom alternativa lösningar. POW använde sig av en referensarkitektur vilket också RUP rekommenderar (Kruchten, 2002; Lunell 2003). Denna referensarkitektur upplevdes positiv då RUP beskriver vad som skall göras men inte hur, vilket däremot referensarkitekturen beskriver.

CMS-projektet utgick också från en mall som tagits fram från ett tidigare projekt. Det korsvisa programmeringssättet som användes i CMS-projektet upplevdes som en positiv metod av utvecklarna, de tyckte att de fick en mer komplett dokumentation. Det här tror vi underlättar för förvaltningsgruppen för att förstå hur systemet är uppbyggt. Vi vet dock inte huruvida förvaltningsgruppen upprätthåller denna dokumentation. Om arkitekturen skall fortsätta hållas generell gäller det att förvaltningsorganisationen verkligen följer och tillämpar grundstrukturen.

Utvärdering utefter CMMI:s Teknisk lösning, Nivå tre

Tekniska lösningar som har med arkitekturen att göra kommer inte förrän på Nivå tre på CMMI:s skala. När vi ställer projektens arkitektoniska synsätt mot CMMI:s Tekniska lösningar, finner vi att båda projekten använder sig av en prototyp vilket förespråkas. Även mallar är någonting som båda projekten använder sig av. Båda projekten baserar också sin arkitektur på de kravanalyser som på olika sätt gjordes i början av projekten. Projekten utvecklade båda en mall för den valda lösningen och man har medvetet valt en lösning som skall passa kraven. Vi anser därför att båda projekten uppnår Nivå tre.

Testhantering

Likheterna i de båda teoriernas angreppssätt är att kontinuerligt testa kvaliteten genom hela utvecklingscykeln. (Kruchten, 2002; Larman 2003). Båda projekten genomförde många olika sorters tester, såsom acceptanstester, systemtester och integrationstester. CMS-projektet var även, av en delvis testdriven ansats. I projektet var kundrepresentanterna delaktiga från första dagen. Projektledaren sammanförde kundrepresentanternas kontinuerliga och aktiva deltagande med de veckovisa testsessionerna. Detta gav direkt feedback på resultatet och upplevdes positivt från både utvecklare och kundrepresentanter. Men deltagandet kunde också upplevas som krävande för kunden. I POW-projektet upplevdes det att alla tester utan systemtesterna startade sent vilket medförde att den dåliga kravbilden uppdagades senare än vad som hade behövts. Tidigare tester ihop med kunden i kombination med kortare iterationer kanske hade lett till att man i POW-projektet tidigare hade hittade felen i kravställningen. Man hade dessutom sluppit dels en omskrivning av Use Casen med uppenbara fel och dels programmera in felaktiga funktioner som redan var åtgärdade i POL men som inte var dokumenterade. Ett resultat av att i ett sent skede upptäcka många av bristerna tror vi också bidrar till att det uppfattas som svårt att hinna med att uppdatera dokument såsom de viktiga Use Casen.

Hur tester dokumenteras och vilket verktygsstöd som används kan ske på olika sätt och vi tycker att både POW-projektet och CMS-projektet skött detta på ett passande sätt till respektive projekt. POW-projektet använde sig av ClearQuest och CMS-projektet av excelformulär för felrapportering. Medlemmarna i respektive projekt uppfattade vardera metoden som positiv. I POW-projektet satt som sagt kunden och projektmedlemmarna på olika håll och testade, då tycker vi att det passar bra med användandet av ClearQuest där de som testade efterhand kunde skicka in vad de testat, det hade gissningsvis varit svårt att ha ett exceldokument till detta. I CMS-projektet däremot, där den stora testningen gjordes ihop med kunden, passade det bra, enligt vår mening med ett enklare verktyg.

Utvärdering utefter CMMI:s Validering och Verifiering, Nivå tre

Validering och verifiering, är även de på Nivå tre på CMMI:s skala och går in i varandra. Då båda processområdena är representerade under området Testhantering, har vi valt att ta med de båda och ställa dem gentemot testhanteringen i projekten. CMS-projektet har lyckats bra även här

tycker vi då man har haft tester en gång i veckan med kundrepresentanter och användarrepresentanter för att se att de byggde produkten rätt och att det blev rätt produkt som byggdes. Tester utfördes under hela projektets gång. Även i POW utfördes tester men de flesta testerna utfördes sent och användarna var inte med i den utsträckning som de var i CMS-projektet. Därför anser vi att CMS-projektet når upp till Nivå tre men att POW-projektet har en liten bit kvar.

Slutdiskussion

Vi upplever att denna studie har varit för liten för att kunna ge Volvo IT några konkreta svar på hur de ska optimera sina processer i framtiden. Vi kan emellertid urskilja några företeelser som pekar åt en viss riktning och som vi tycker att Volvo IT ska ta i beaktande.

Utifrån det resultat vi presenterat ser vi att respondenterna uttryckt både positiva och negativa aspekter i respektive projekt. Det vi dock upplever är att det var enklare för medlemmarna från CMS-projektet att spontant reflektera och uttrycka både bra och dåliga aspekter med den lättrörliga process som tillämpades. I POW-projektet var det svårare att tydligt urskilja att alla de positiva eller negativa faktorerna var ett direkt resultat utifrån användandet av processen RUP. Dessutom fanns det organisatoriska faktorer, de projektstyrningsmodeller som skulle tillämpas och tas hänsyn till.

Det synsätt som tillämpades i CMS-projektet på Volvo IT förefaller vara disciplinerat och flexibelt. Flexibelt genom att man bland annat kontinuerligt under projektets gång har varit beredd på fortlöpande förändringar. När vi menar disciplinerat tänker vi på det sätt som CMS-projektet håller fast vid rutiner och har kontroll över utvecklingscykeln genom sin struktur med de veckovisa iterationerna, som dessutom är strängt tidsbestämda.

I POW-projektet kan vi se ett mönster av lättrörlighet på olika sätt, såsom ett öppet flexibelt sätt att hantera nästan dagliga kravförändringar och med en testdriven ansats. Men vi upplever att då förutsättningarna gav tecken på en förutfattad uppgift, var man kanske inte initialt inställd på den ökade påfrestning som uppenbarade sig i form av ständiga omvärderingar och förändringar i planeringen. Därav anser vi också att fler och kortare iterationer skulle kunna bidra till att erhålla bättre kontroll över ett projekt. De korta iterationerna skall dock inte missbrukas till att vara ett verktyg för att kontrollera hur väl individen har presterat. Även om en veckas iterationer fungerade med överväldigade positiv respons från projektmedlemmar i CMS-projektet, tror vi det kan upplevas extremt för gemene man och ser det inte som någon större nackdel att öka iterationslängden med en vecka. I POW-projektet ser vi emellertid att en skicklig projektledare/arkitekt ändå bemästrade situationen med den ofullständiga kravbilden och ingav förtroende hos kunden. Där ser vi också, det som påpekas av flertalet experter inom området systemutveckling (Brooks, 1995; Martin, 2003; Fowler & Hihgsmith, 2001), att individen spelar stor roll för hur utfallet av ett projekt ter sig.

Boehm & Turner (2004) menar att RUP inger en möjlighet till att ge en god balans mellan lättrörliga och plandrivna metoder. Vi tror dock att det är svårt att se RUP som en tillfredsställande balans mellan de två koncepten. Den information vi samlat på oss ger dock indikationer på att det är svårt att skala ner RUP och att det krävs kunskap och mentorer för att på ett bra sätt anpassa RUP till ett projekts förutsättningar. Boehm & Turner (2004) menar att det är lättare att skala upp en process utan experthjälp än att skala ned den. Vi tror att en konsekvens av

att försöka tillämpa processer som är såpass omfattande som RUP, gör att den inte tillämpas fullt ut eller tillämpas på fel sätt trots en anpassning. Vår uppfattning är att det också krävs hjälp för användandet av RUP och inte bara till att anpassa RUP till projektets karaktär. Därför ser vi fördelen med, som Fowler (2001) också belyser, en lättrörlig metod som lätt kan läras ut och där man lär av varandra under projektets gång. Vi tror att det är mycket viktigt att redan innan projektet startar i planeringsfasen, att sätta sig i inte allt för stora grupper, för att alla ska få göra sin röst hörd, och diskutera igenom projektet ordentligt. Redan i detta skede kan experthjälp tas in för att anpassa en mer lättrörlig metod till det enskilda projektets karaktär. Vi tycker också man skall förvalta de erfarenheter och kunskaper som finns med avseende av tidigare projekts såväl positiva som negativa resultat. Proceduren kan eventuellt komma att kräva mer av mentoravdelningen än vad den gör idag. Man bör även beakta de problem som kan uppdagas med att ha mentorer och problematiken med att hitta kunniga sådana (Jacobson, 2002). Det ger ingen stor produktivitet i början av projektet att ha en så pass detaljerad planeringsfas som föreslås ovan men vi tror ändå att Volvo IT tjänar in den tiden under projektets gång då man härigenom undgår att hamna i trångmål.

Vi anser vidare att RUP:s fyra faser, i viss mån, hindrar en lättrörlig metod då respektive fas indikerar ett vattenfallsdrivet angreppssätt. Vi har fått uppfattningen att jobb med korta iterationer ger en högre produktivitet från utvecklaren och även bättre feedback från kunden. Man ser att det går framåt i utvecklingen av systemet, vilket vi tror ger en positiv känsla både för utvecklaren och för kunden och alla känner ett större engagemang. Sitter utvecklaren och tragglar månadsvis utan att se något resultat eller utan att få feedback på det som produceras, kan denne lätt tröttna och inte producera i samma utsträckning som under kortare iterationer. Kunden upptäcker tidigare kraven som denne vill skall ändras ifall denne ser hur systemet utvecklas. Precis som IKIWISI-effekten, nämligen att kunden inte direkt vet vilka krav som skall vara med i systemet, istället är det alltid lättare att undan för undan se vad som behövs.

Volvo IT bör beakta sin tillämpning av PCM som verkar vara ett orosmoment hos projektledarna med mer stjälpande än hjälpande verkan under projektets gång. Viss positiv feedback har erhållits avseende denna modell, såsom bra checklistor men till största delen har kritik lämnats då det är så pass många dokument med mera som skall skrivas och som egentligen inte har med utvecklingen i projektet att göra. Ett förslag vi har är att Volvo IT skall se över hur modellen istället skulle kunna anpassas för att i framtiden bättre kunna passa ihop med mer flexibla metoder och processer. Inspirerade av Boehm & Turner (2002) och utifrån vårt tidigare resonemang om att det är svårt att se RUP som en hybrid mellan plandrivna och lättrörliga koncept, föreslår vi att Volvo IT istället kombinerar en nyutvecklad styrmodell med någon form av lättrörlig metod. Det är också viktigt att organisationen förstår vikten vid att förvalta en implementation av systemutvecklingsprocesser. Att uppmuntra till användandet av arbetssättet tror vi leder till engagerade medarbetare.

Liksom Brooks (1995) tror vi inte att det finns och inte heller kommer att finnas några mirakulösa processer som enkelt kan appliceras på vilket projekt som helst och som löser alla problem. Däremot tror vi att det krävs en gedigen arbetsinsats och ett stegvist förfarande för att förbättra systemutvecklingen. Individen och dennes beteende har en betydande roll i hur ett projekt lyckas. Det hänger inte enbart på vilken utvecklingsprocess som tillämpas. Processen skall vara ett stöd för systemutvecklingen, inte ett hinder. Planera noga innan projektet dras igång. Satsa också på motiverade och intresserade individer. Vi tror att om den enskilde individen uppmärksammas genom och uppmuntras till både kompetensutveckling och personlig utveckling leder det till

engagerade medarbetare vilket rimligtvis borde ge positiv effekt på utvecklingsprocessen. Låt individerna inspirera och lära av varandra. Sprid kunskapen. Om dessa ledord i framtiden genomsyrar företagets projekt tror vi att Volvo IT eventuellt kan klättra på CMMI-skalan och få mer nöjda utvecklare, kunder, och övriga intressenter och således säkrare projekt. Som så många experter emellertid redan har sagt är det trots allt individerna som är den största bidragande faktorn till hur väl ett projekt faller ut. Det är att sätta ihop det optimala ”arbetsteamet” som är den svåra uppgiften!

Slutsats

I detta uppsatsens sista kapitel redogör vi för slutsatserna vi dragit utifrån den analys vi gjort. Vi avslutar med förslag till vidare forskning.

Hur kan Volvo IT optimera sina processer för systemutveckling för framtida bruk?

Vi tror på kombinationen väl genomtänkta checklistor och disciplinerade lättrörliga metoder. Att kunden är med mycket kan leda till att ytterligare effektivisera en process på så sätt att kunden får ett system som mer motsvarar dess förväntningar. Volvo IT bör beakta sin tillämpning av PCM och se över hur den istället skulle kunna anpassas för att i framtiden bättre kunna passa ihop med mer flexibla metoder och processer. Vad Volvo IT än väljer för processer i framtiden måste dessa förankras i organisationen. Satsa på medarbetarna, det är de som bygger systemet, inte processen!

Förslag till vidare forskning

Vår studie antyder på en omognad från kundperspektivet för att tillämpa metoder där denne förväntas delta i stor utsträckning. Förslag till fortsatta forskning är därför att studera det lättrörliga synsättet från kundens perspektiv.

Mot den bakgrund som presenterades i inledningen, att RUP inte anses tillämpas i den utsträckning den borde, finner vi det intressant att studera vilka inverkande faktorer som hindrar till användningen av en process.

Studera fler projekt på Volvo IT för att undersöka i vilken utsträckning de har tillämpat RUP.

Referenser

Artiklar

Anthes, H. G. (2004, mars, 8). Quality Model Mania. Computerworld,

Boehm, B. Get ready for agile methods, with care. Computer (januari. 2002), 64–69.

Boehm, B. och Turner, R. Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-Driven Methods. Preceedings of the 26th International Conference on Software Engineering (ICSE'04), Edinburgh, Skottland

Danielsson, L. (2004, mars, 12). Kvalitet kräver dessa modeller. ComputerSweden, nr 29, 16.

Fowler, M. & Highsmith, J. The Agile Manifesto. Software Development Magazin (augusti. 2001).

Haldén, B.(2006, april, 7) Lättrörliga metoder leder rätt. ComputerSweden nr 40, 16-17.

Hoffman, T. (2005, mars, 28). Sidebar: Comparing Methodologies. ComputerWorld,

Jacobson, I. (2002, januari) A resounding Yes to agile processes – but also to more. Cutter IT Journal, Nr 1

Lindvall, M.(2002, september, 10). Programutveckling som religion. ComputerSweden, 28.

Ljadas, M.(2006, februari, 20). Nya och kaxiga Systemutvecklare. ComputerSweden, nr 20, 18.

Nerur, S., Mahapatra, R. & Mangalaraj, G. Challenges of migrating to Agile methodologies. Communications of the ACM, May 2005/Vol. 48, No 5

Tallung, P.(2006,januari,20). Se upp med användningsfall. ComputerSweden Teknik, nr 7, 8.

Wallström, M. (2005, oktober, 31). Renässans för stämpeln. ComputerSweden, nr 107, 9.

Böcker

Ahern, D. M, Clouse, A & Turner, R. (2001) CMMI *Distilled: A Practical introduction to Integrated Process improvement.* Massachusetts: Addison-Wesley.

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

Björklund, M. & Paulsson, P. (2003) *Seminariehandboken – att skriva, presentera och opponera*. Lund: Studentlitteratur

Boehm, B. & Turner, R. (2004) *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston: Addison-Wesley.

Brooks, Jr, F. P. (1995) *The Mythical Man-Month: Essays on Software Engineering*. Massachusetts: Addison-Wesley.

Cockburn, A. (2002) *Agile Software Development*. Massachusetts: Addison-Wesley.

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

Kruchten, P.(2002). *Rational Unified Process: en introduktion*. Boston: Addison – Wesley

Larman, C. (2003). *Agile & Iterative Development: A managers guide.* Massachusetts: Addison-Wesley

Lekvall, P. & Wahlbin, C. (2001) *Information för marknadsföringsbeslut*. Göteborg: IHM Publishing

Lunell, H. (2003). *Fyra rundor med RUP*. Lund: Studentlitteartur.

Martin, C, R. (2003) *Agile Software Development: Pricniples, Patterns, and Practices.* New Jersey: Pearson Education Inc.

Patel, R. & Davidson, B. (1998): *Forskningsmetodikens grunde*. Lund: Studentlitteratur

Trost, J. (2005): *Kvalitativa forskningsmetoder*. Lund: Studentlitteratur

Länkar

http://www.agilealliance.org

http://www.agilesweden.org

http://www.computerworld.com/developmenttopics/development/story/0,10801,99159,00.html

http://www.martinfowler.com

http://www.projektplatsen.se/press_arkiv/20051128.html

http://www.q-labs.se/1hem/index.htm

http://www.sei.cmu.edu/cmmi/

http://www.wikipedia.org

Intervjuer

Besök

Mats Ekhammar 23/3-06, 29/3-06
Åsa Forsberg 20/4-06
Anders Henriksson 19/4-06

Charles Jobson 27/3-06, 5/4-06
Inger Stening 23/3-06

Telefon
Alexander Pajari 19/5-06

Informant
Charles Jobson 2/3-06
Anders Jonsson 16/3-06

Appendix A

Project Control Model

PCM är en projektstyrningsmodell som används på Volvo IT, gjord för att få styrning på IT-projekten. Den består av fem olika gater:

G0 Change initiation gate
G1 Concept study gate
G2 Final development contract gate
G3 Release gate
G4 End gate

Projektet skall till varje gate ha klarat av vissa delmål. De olika gaterna har försökt sammankopplas med RUP:s fyra faser. Ett projekt börjar i G0, den förberedande fasen. För att ett projekt skall få gå vidare till G1 måste en planeringsplan göras. Samtliga parter det vill säga projektledaren, kunden och styrgruppen skriver på en överenskommelse innan projektet kan starta. Det är sedan IT-styrgruppen som bestämmer ifall projektet uppnått målen för att gå vidare till nästkommande gate.

Appendix B

Manifesto for The Agile Software development

We are uncovering better ways of developing software
by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right,
we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

We follow these principles **:**

Principles behind the Agile Manifesto

Our highest priority is to satisfy the customer
through early and continuous delivery
of valuable software.

Welcome changing requirements, even late in
development. Agile processes harness change for
the customer's competitive advantage.

Deliver working software frequently, from a
couple of weeks to a couple of months, with a
preference to the shorter timescale.

Business people and developers must work
together daily throughout the project.

Build projects around motivated individuals.
Give them the environment and support they need,
and trust them to get the job done.

The most efficient and effective method of
conveying information to and within a development
team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development.
The sponsors, developers, and users should be able
to maintain a constant pace indefinitely.

Continuous attention to technical excellence
and good design enhances agility.

Simplicity--the art of maximizing the amount
of work not done--is essential.

The best architectures, requirements, and designs
emerge from self-organizing teams.

At regular intervals, the team reflects on how
to become more effective, then tunes and adjusts
its behavior accordingly.

(www.agilealliance.org)

Appendix C

Lyckat projekt?

När vi i vår undersökning ställde frågan till samtliga inblandade parter om vad de definierar ett lyckat projekt, svarar leverantörerna följande:

”I tid, avtalad kostnad och med rätt funktionalitet.”
och
”Håller kostnaden. Kostnad och tid, är nog viktigast. Sedan, naturligtvis, funktionaliteten. Men det förutsätter man ju bara. Man förutsätter att man kan fixa till funktionaliteten med hjälp av cost och tid.”
och
”Uppfylla fastpriskontraktet på kost och tid. Leverera en lösning som är av hög kvalité och som är utbyggbar. Projektmedlemmar som trivs tillsammans och har ett bra samarbete med Kund.”

Kunderna definierade ett lyckat projekt:

”För mig handlar det om att man uppfyller de krav som man ställer på produkten inom utsatta tidsramar”
och
”Lyckat projekt för XX är i tid med kvalitet.”

I det senare fallet anser respondenten att tid och kvalitet prioriterades lika. På vår fråga vad de ansåg om kostnaden, svarade respondenterna att kostnad var viktig. I det första fallet menar respondenten att projektet

”… medför en viss profit med att få ut ett sånt här system därför vill vi ha ut det i tid och att det uppfyller de krav man har för då tjänar du pengar också och kan säkerligen tjäna in de pengarna du lägger ut. Men det är självklart, kostnaden är också en viktig detalj.”

I det andra fallet antydde respondenten att projektet var fastpris och därför inte nämnt kostnaden. Den tredje respondenten svarar oss:

”Leverera en lösning till Business som ger maximal nytta och används effektivt så att Business Caset uppfylls. Leverera en lösning som är hållbar och utbyggbar ur Business Process perspektiv. Projektmedlemmar, Business linje personal och IT supplier personal som samarbetar väl. Och konstruktivt tar fram lösningar tillsammans.”

Appendix D

CMMI SE/SW version 1.1 Staged

Requirements Management

Maturity Level 2

Purpose

The purpose of Requirements Management is to manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products. [PA146]

Introductory Notes

Requirements management processes manage all requirements received or generated by the project, including both technical and nontechnical requirements as well as those requirements levied on the project by the organization. In particular, if the Requirements Development process area is implemented, its processes will generate product and product-component requirements that will also be managed by the requirements management processes. When the Requirements Management, Requirements Development, and Technical Solution process areas are all implemented, their associated processes may be closely tied and be performed concurrently. [PA146.N101]

The project takes appropriate steps to ensure that the agreed-upon set of requirements is managed to support the planning and execution needs of the project. When a project receives requirements from an approved requirements provider, the requirements are reviewed with the requirements provider to resolve issues and prevent misunderstanding before the requirements are incorporated into the project’s plans. Once the requirements provider and the requirements receiver reach an agreement, commitment to the requirements is obtained from the project participants. The project manages changes to the requirements as they evolve and identifies any inconsistencies that occur among the plans, work products, and requirements. [PA146.N102]

Part of the management of requirements is to document requirements changes and rationale and maintain bidirectional traceability between source requirements and all product and product-component requirements. [PA146.N103]

Specific and Generic Goals

SG 1 Manage Requirements [PA146.IG101]

Requirements are managed and inconsistencies with project plans and work products are identified.

The project maintains a current and approved set of requirements over the life of the project by doing the following: [PA146.IG101.N101]

- Managing all changes to the requirements
- Maintaining the relationships between the requirements, the project plans, and the work products
- Identifying inconsistencies between the requirements, the project plans, and the work products
- Taking corrective action

Refer to the Technical Solution process area for more information about determining the feasibility of the requirements.

[PA146.IG101.N101.R101]

Refer to the Requirements Development process area for more information about ensuring that the requirements reflect the needs and expectations of the customer. *[PA146.IG101.N101.R102]*

Refer to the Project Monitoring and Control process area for more information about taking corrective action. *[PA146.IG101.N101.R103]*

For Software Engineering

The requirements may be a subset of the overall product requirements, or they may constitute the entire product requirements. *[PA146.IG101.AMP101]*

For Systems Engineering

Each level of product-component design (e.g., segment, subsystem) receives the requirements from the higher level. *[PA146.IG101.AMP102]*

PROJECT PLANNING

Maturity Level 2

Purpose

The purpose of Project Planning is to establish and maintain plans that define project activities. [PA163]

Introductory Notes

The Project Planning process area involves the following: [PA163.N101]

- Developing the project plan
- Interacting with stakeholders appropriately
- Getting commitment to the plan
- Maintaining the plan

Planning begins with requirements that define the product and project. [PA163.N102]

Planning includes estimating the attributes of the work products and tasks, determining the resources needed, negotiating commitments, producing a schedule, and identifying and analyzing project risks. Iterating through these activities may be necessary to establish the project plan. The project plan provides the basis for performing and controlling the project's activities that address the commitments with the project's customer. [PA163.N103]

The project plan will usually need to be revised as the project progresses to address changes in requirements and commitments, inaccurate estimates, corrective actions, and process changes. Specific practices describing both planning and re-planning are contained in this process area. [PA163.N104]

The term "project plan" is used throughout the generic and specific practices in this process area to refer to the overall plan for controlling the project. [PA163.N105]

Specific and Generic Goals

SG 1 Establish Estimates [PA163.IG101]

Estimates of project planning parameters are established and maintained.

Project planning parameters include all information needed by the project to perform the necessary planning, organizing, staffing, directing, coordinating, reporting, and budgeting. [PA163.IG101.N101]

Estimates of planning parameters should have a sound basis to provide confidence that any plans based on these estimates are capable of supporting project objectives. [PA163.IG101.N102]

Factors that are typically considered when estimating these parameters include the following: [PA163.IG101.N103]

- Project requirements, including the product requirements, the requirements imposed by the organization, the requirements imposed by the customer, and other requirements that impact the project
- Scope of the project
- Identified tasks and work products
- Technical approach
- Selected project life-cycle model (e.g., waterfall, incremental, spiral, etc.)
- Attributes of the work products and tasks (e.g., size or complexity)
- Schedule
- Models or historical data for converting the attributes of the work products and tasks into labor hours and cost
- Methodology (models, data, algorithms) used to determine needed material, skills, labor hours, and cost

Documenting the estimating rationale and supporting data is needed for stakeholders' review and commitment to the plan and for maintenance of the plan as the project progresses. [PA163.IG101.N104]

SG 2 Develop a Project Plan [PA163.IG102]

A project plan is established and maintained as the basis for managing the project.

A project plan is a formal, approved document used to manage and control the execution of the project. It is based on the project requirements and the established estimates. [PA163.IG102.N101]

The project plan should consider all phases of the project life cycle. Project planning should ensure that all plans affecting the project are consistent with the overall project plan. [PA163.IG102.N102]

SG 3 Obtain Commitment to the Plan [PA163.IG103]

Commitments to the project plan are established and maintained.

To be effective, plans require commitment by those responsible for implementing and supporting the plan. [PA163.IG103.N101]

TECHNICAL SOLUTION

Maturity Level 3

Purpose

The purpose of Technical Solution is to design, develop, and implement solutions to requirements. Solutions, designs, and implementations encompass products, product components, and product-related life-cycle processes either singly or in combinations as appropriate. [PA160]

Introductory Notes

The Technical Solution process area is applicable at any level of the product architecture and to every product, product component, product-related life-cycle process, and service. The process area focuses on the following: [PA160.N101]

- Evaluating and selecting solutions (sometimes referred to as "design approaches," "design concepts," or "preliminary designs") that potentially satisfy an appropriate set of allocated requirements
- Developing detailed designs for the selected solutions (detailed in the context of containing all the information needed to manufacture, code, or otherwise implement the design as a product or product component)
- Implementing the designs as a product or product component

Typically, these activities interactively support each other. Some level of design, at times fairly detailed, may be needed to select solutions. Product-component prototypes may be used as a means of gaining sufficient knowledge to develop a technical data package or a complete set of requirements. [PA160.N102]

Technical Solution specific practices apply not only to the product and product components but also to services and product-related life-cycle processes. The product-related life-cycle processes are developed in concert with the product or product component. Such development may include selecting and adapting existing processes (including standard processes) for use as well as developing new processes. [PA160.N103]

Processes associated with the Technical Solution process area receive the product and product-component requirements from the requirements management processes. The requirements management processes place the requirements, which originate in requirements development processes, under appropriate configuration management and maintain their traceability to previous requirements. [PA160.N104]

For a maintenance or sustainment organization, the requirements in need of maintenance actions or redesign may be driven by user needs or latent defects in the product components. New requirements may arise from changes in the operating environment. Such requirements can be uncovered during verification of the product(s) where actual performance can be compared against the specified performance and unacceptable degradation can be identified. Processes associated with the Technical Solution process area should be used to perform the maintenance or sustainment design efforts. [PA160.N105]

Specific and Generic Goals

SG 1 Select Product-Component Solutions [PA160.IG101]

Product or product-component solutions are selected from alternative solutions.

Alternative solutions and their relative merits are considered in advance of selecting a solution. Key requirements, design issues, and constraints are established for use in alternative solution analysis. Architectural features that provide a foundation for product improvement and evolution are considered. Use of commercial off-the-shelf (COTS) product components are considered relative to cost, schedule, performance, and risk. COTS alternatives may be used with or without modification. Sometimes such items may require modifications to aspects such as interfaces or a customization of some of the features to better achieve product requirements. [PA160.IG101.N101]

One indicator of a good design process is that the design was chosen after comparing and evaluating it against alternative solutions. Decisions on architecture, custom development versus off the shelf, and product-component modularization are typical of the design choices that are addressed. [PA160.IG101.N102]

Sometimes the search for solutions examines alternative instances of the same requirements with no allocations needed to lower level product components. Such is the case at the bottom of the product architecture. There are also cases where one or more of the solutions are fixed (e.g., a specific solution is directed or available product components, such as COTS, are investigated for use). [PA160.IG101.N103]

In the general case, solutions are defined as a set. That is, when defining the next layer of product components, the solution for each of the product components in the set is established. The alternative solutions are not only different ways of addressing the same requirements, but they also reflect a different allocation of requirements among the product components comprising the solution set. The objective is to optimize the set as a whole and not the individual pieces. There will be significant interaction with processes associated with the Requirements Development process area to support the provisional allocations to product components until a solution set is selected and final allocations established. [PA160.IG101.N104]

Product-related life-cycle processes are among the product-component solutions that are selected from alternative solutions. Examples of these product-related life-cycle processes are the manufacturing and the support processes. [PA160.IG101.N105]

SG 2 Develop the Design [PA160.IG102]

Product or product-component designs are developed.

Product or product-component designs must provide the appropriate content not only for implementation, but also for other phases of the product life cycle such as modification, reprocurement, maintenance, sustainment, and installation. The design documentation provides a reference to support mutual understanding of the design by relevant stakeholders and supports future changes to the design both during development and in subsequent phases of the product life cycle. A complete design description is documented in a technical data package that includes a full range of features and parameters including form, fit, function, interface, manufacturing process characteristics, and other parameters. Established organizational or project design standards (e.g., checklists, templates, object frameworks) form the basis for achieving a high degree of definition and completeness in design documentation. [PA160.IG102.N101]

SG 3 Implement the Product Design [PA160.IG103]

Product components, and associated support documentation, are implemented from their designs.

Product components are implemented from the designs established by the specific practices in the Develop the Design specific goal. The implementation usually includes unit testing of the product components before sending them to product integration and development of end-user documentation. [PA160.IG103.N101]

VERIFICATION

Maturity Level 3

Purpose

The purpose of Verification is to ensure that selected work products meet their specified requirements. [PA150]

Introductory Notes

The Verification process area involves the following: verification preparation, verification performance, and identification of corrective action. [PA150.N101]

Verification includes verification of the product and intermediate work products against all selected requirements, including customer, product, and product-component requirements. [PA150.N102]

Verification is inherently an incremental process because it occurs throughout the development of the product and work products, beginning with verification of the requirements, progressing through the verification of the evolving work products, and culminating in the verification of the completed product. [PA150.N103]

The specific practices of this process area build upon each other in the following way: the Select Work Products for Verification specific practice enables the identification of the work products to be verified, the methods to be used to perform the verification, and the requirements to be satisfied by each selected work product. The Establish the Verification Environment specific practice enables the determination of the environment that will be used to carry out the verification. The Establish Verification Procedures and Criteria specific practice then enables the development of verification procedures and criteria that are aligned with the selected work products, requirements, methods, and characteristics of the verification environment. The Perform Verification specific practice conducts the verification according to the available methods, procedures, and criteria. [PA150.N110]

Verification of work products substantially increases the likelihood that the product will meet the customer, product, and product-component requirements. [PA150.N104]

The Verification and Validation process areas are similar, but they address different issues. Validation demonstrates that the product, as provided (or as it will be provided), will fulfill its intended use, whereas verification addresses whether the work product properly reflects the specified requirements. In other words, verification ensures that "you built it right;" whereas, validation ensures that "you built the right thing." [PA150.N105]

Peer reviews are an important part of verification and are a proven mechanism for effective defect removal. An important corollary is to develop a better understanding of the work products and the processes that produced them so defects can be prevented and process-improvement opportunities can be identified. [PA150.N106]

Peer reviews involve a methodical examination of work products by the producers' peers to identify defects and other changes that are needed. [PA150.N107]

Examples of peer review methods include the following: [PA150.N109]

Appendix A Inspections

Appendix A Structured walkthroughs

Specific and Generic Goals

SG 1 **Prepare for Verification** [PA150.IG101]

Preparation for verification is conducted.

Up-front preparation is necessary to ensure that verification provisions are embedded in product and product-component requirements, designs, developmental plans, and schedules. Verification includes selection, inspection, testing, analyses, and demonstration of work products. [PA150.IG101.N101]

Methods of verification include, but are not limited to, inspections, peer reviews, audits, walkthroughs, analyses, simulations, testing, and demonstrations. [PA150.IG101.N102]

Preparation also entails the definition of support tools, test equipment and software, simulations, prototypes, and facilities. [PA150.IG101.N103]

SG 2 **Perform Peer Reviews** [PA150.IG102]

Peer reviews are performed on selected work products.

Peer reviews involve a methodical examination of work products by the producers' peers to identify defects for removal and to recommend other changes that are needed. [PA150.IG102.N101]

The peer review is an important and effective engineering method implemented via inspections, structured walkthroughs, or a number of other collegial review methods. [PA150.IG102.N102]

Peer reviews are primarily applied to work products developed by the projects, but they can also be applied to other work products such as documentation and training work products that are typically developed by support groups. [PA150.IG102.N103]

SG 3 Verify Selected Work Products [PA150.IG103]

Selected work products are verified against their specified requirements.

VALIDATION

Maturity Level 3

Purpose

The purpose of Validation is to demonstrate that a product or product component fulfills its intended use when placed in its intended environment. [PA149]

Introductory Notes

Validation activities can be applied to all aspects of the product in any of its intended environments, such as operation, training, manufacturing, maintenance, and support services. The methods employed to accomplish validation can be applied to work products as well as to the product and product components. The work products (e.g., requirements, designs, prototypes) should be selected on the basis of which are the best predictors of how well the product and product component will satisfy user needs. [PA149.N105]

The validation environment should represent the intended environment for the product and product components as well as represent the intended environment suitable for validation activities with work products. [PA149.N106]

Validation demonstrates that the product, as provided, will fulfill its intended use; whereas, verification addresses whether the work product properly reflects the specified requirements. In other words, verification ensures that "you built it right;" whereas, validation ensures that "you built the right thing." Validation activities use approaches similar to verification (e.g., test, analysis, inspection, demonstration, or simulation). Often, the end users are involved in the validation activities. Both validation and verification activities often run concurrently and may use portions of the same environment. [PA149.N102]

Refer to the Verification process area for more information about verification activities. *[PA149.N102.R101]*

Where possible, validation should be accomplished using the product or product component operating in its intended environment. The entire environment may be used or only part of it. However, validation issues can be discovered early in the life of the project using work products. [PA149.N103]

When validation issues are identified, they are referred to the processes associated with the Requirements Development, Technical Solution, or Project Monitoring and Control process areas for resolution. [PA149.N104]

The specific practices of this process area build on each other in the following way. The Select Products for Validation specific practice enables the identification of the product or product component to be validated and the methods to be used to perform the validation. The Establish the Validation Environment specific practice enables the determination of the environment that will be used to carry out the validation. The Establish Validation Procedures and Criteria specific practice enables the development of validation procedures and criteria that are aligned with the characteristics of selected products, customer constraints on validation, methods, and the validation environment. The Perform Validation specific practice enables the performance of validation according to the methods, procedures, and criteria. [PA149.N107]

Specific and Generic Goals

SG 1 Prepare for Validation [PA149.IG101]

Preparation for validation is conducted.

Preparation activities include selecting products and product components for validation and establishing and maintaining the validation environment, procedures, and criteria. The items selected for validation may include only the product or it may include appropriate levels of the product components that are used to build the product. Any product or product component may be subject to validation, including replacement, maintenance, and training products, to name a few. [PA149.IG101.N101]

The environment required to validate the product or product component is prepared. The environment may be purchased or may be specified, designed, and built. The environments used for product integration and verification may be considered in collaboration with the validation environment to reduce cost and improve efficiency or productivity. [PA149.IG101.N102]

SG 2 Validate Product or Product Components [PA149.IG102]

The product or product components are validated to ensure that they are suitable for use in their intended operating environment.

The validation methods, procedures, and criteria are used to validate the selected products and product components and any associated maintenance, training, and support services using the appropriate validation environment. [PA149.IG102.N102]