HÖGSKOLAN
TROLLHÄTTAN • UDDEVALLA

Förbättra processen vid småskalig systemutveckling
En fallstudie

Improve process in small-scale softwaredevelopment
A case study

Uppsats i Informatik tio poäng.
Instutionen för informatik och matematik.
Inlämningdatum:15 oktober 2003
Författare: Åke Johansson och Lennart Jonasson
Handledare: Lars Svensson
Examinator: Kerstin Grunden

Abstract

This paper presents an evaluation of the working process at a minor system-development department. The analysis is made from a case study in co-operation with a local company. For the analysis, we studied and used the Capability Maturity Model (CMM). The result of our evaluation was that the system developing process of today did not reach goals of CMM.
We think that the CMM model is far to extensive to be implemented in a minor department. The model must be tailored so it suits the conditions at the department, and the implementation should be carried out in an evolutionary way in small steps. At the tailoring and implementation of the model, the experience and opinions of the employees should be used. The reason for this is to get a quicker improvement.

Sammanfattning

Detta papper presenterar en utvärdering av systemutvecklingsprocessen vid en mindre systemutvecklingsavdelning. Analysen är gjord utifrån en fallstudie i samarbete med en extern uppdragsgivare. Den modell vi studerat och använt vid analysen är Capability Maturity Model (CMM). Vi fann att systemutvecklingen idag inte uppfyller CMM:s intentioner. Modellen som helhet är dock alltför omfattande för att realiseras på små systemutvecklingsföretag, och måste därför anpassas till avdelningens villkor och införas evolutionärt och i små steg. Vid anpassning och implementering av modellen bör personalens erfarenheter och åsikter utnyttjas. Detta för att snabbare få till stånd prosessförbättringar.

FIGURFÖRTECKNING

BILAGA

1. Introduktion

Vårt samhälle genomsyras alltmer av informations- och kommunikationsteknologi. Tekniken påträffas i alla sammanhang, på vår fritid i arbetet och inom skolan. Denna teknologi har stor påverkan på hur vi utformar våra organisationer och vårt samhälle.

Informatik är ett samhällsvetenskapligt ämne med stora inslag av teknikstudier och teknikanvändning. Informatik handlar inte enbart om teknik, utan en väsentlig del av ämnet medför studier av utveckling och förändring av IT-baserade system enligt Goldkuhl (1996). Viktigt inom forskningen är att förändringar och skapandet av IT-stöden sker i ett användarorienterat perspektiv. Införandet av ett nytt informationssystem skall ses som ett kunskapslyft för hela organisationen

Det forskningsområde inom informatiken vi har inriktat oss på är Software Engineering, där
Software Process Improvement (SPI) ingår. Inom detta område ingår forskning som säkrar kvalitet i produkt och process inom systemutvecklingen.

Paulk (1998) anser att en av de första utmaningarna när det gäller att införa förbättrade arbetsprocesser i små organisationer, är att de oftast har små resurser, både vad gäller tid och pengar. De måste hela tiden kämpa för att överleva. Små organisationer tenderar även att tro att

- Vi är alla kompetenta - vår personal är anställd för att utföra jobbet och vi har inte tid och pengar för träning och utbildning.
- Här kommunicerar alla med varandra – flervägskommunikationen fungerar eftersom vi arbetar så ”tätt”
- Vi är alla ”hjältar” – vi gör vad som behöver göras, regler passar inte oss, de blir bara till besvär och kommer att sinka vårt arbete.

Ändå har små organisationer samma problem som stora vad gäller odokumenterade krav, fördelning av resurser, överblick av projekt och produktens dokumentation

1.1. Bakgrund

Vårt uppsatsarbete bedrevs i samverkan med en extern uppdragsgivare. Vi fick uppdraget att objektivt undersöka hur systemutvecklingen utfördes idag på en av företagets mindre utvecklingsavdelningar och hur eventuella förbättringar kunde uppnås. På denna avdelning arbetar fyra anställda, varav tre arbetar med systemutveckling och den fjärde som uppdragsledare. Inom koncernen kallas projekten för uppdrag, därav namnet uppdragsledare. Uppdragsgivaren avsåg att efter undersökningen erhålla en rapport från oss innehållande förslag till förbättringar i systemutvecklingsprocessen.

Den avdelning vi undersökte arbetar med vad vi betecknar vara småskalig systemutveckling. Karakteristiskt för denna systemutveckling är:

- Liten personalresurs (1 – 2 personer).
- Kort livscykel (vattenfallsmodellen).
- Relationsdatabas (systemen innehåller oftast MS SQL databas.)
- Flerskiktslösningar.
- Homogen programmeringsmiljö (programmeringen görs oftast i ASP 3.0 (application service provider) eller Visual Basic 6).

Avdelningen har utvecklat ett flertal webbapplikationer åt stora svenska industriföretag men arbetar även mot mindre lokala företag.

När en kund anlitar vår uppdragsgivare har kunden ett problem som behöver bli löst. Dahlbom& Mathiassen (1995) definierar ett problem med att det kan ha två tillstånd. Initialtillståndet (där kunden är nu) och måltillståndet (där kunden vill vara). För att lösa problemet behövs en väg mellan initialtillståndet och måltillståndet. Problemet kan lösas med hjälp av att man chansar på att finna vägen mellan tillstånden. När målet är nått är det svårt att blicka tillbaka och veta hur vägen dit såg ut. Lösningen av problemet är då inte målet utan hur du ska ta dig dit.
Ett smidigare sätt att lösa problem är att använda sig av en metod. Då blir metoden sättet att ta sig mellan tillstånden och målet blir metoden.
Utvecklingsarbetet ska alltså innehålla metoder för att göra vägen mellan intialtillståndet och måltillståndet så kort som möjligt.

1.2. Modellval

För organisationer som vill införa ett SPI-program, finns två vägar att gå, enligt Jalote (2002) – ett totalt individuellt program, eller ett ramverksbaserat program Vid ett individuellt program analyseras den nuvarande arbetsprocessen, och beroende på vilka brister man upptäcker och vilka mål man har för förbättringsarbetet, vidtas lämpliga åtgärder. Vid ett ramverksbaserat SPI-program används en externt framtagen modell. Denna modell används sedan vid analys av den nuvarande arbetsprocessen och vid framtagandet av lämpliga förbättringsåtgärder. Vi har använt oss av en ramverksbaserad SPI-modell i detta arbete. Anledningen är att dessa modeller erbjuder riktlinjer för analys av nuvarande arbetsprocess och framtagande av lämpliga förbättringsåtgärder.

Det finns flera ramverksbaserade SPI-modeller att tillgå, exempelvis CMM (the Capability Maturity Model), IDEAL, BOOTSTRAP och SPICE. Vid valet av modell är det viktigt att modellen är anpassningsbar så att den passar organisationens arbetssätt. Modellen måste alltså vara möjlig att tolka så att den blir överförbar på organisationen. I annat fall kan modellen bli byråkratisk och rigid, med följd att fokuseringen koncentreras på att följa modellen istället för att uppnå förbättrad arbetsprocess. Jalote (2002) uttrycker detta med orden: ”*Don’t work for the framework, let it work for you”*. Förutsatt att denna flexibilitet finns i modellen, så spelar det mindre roll vilken modell man använder. Jalote hävdar att det är bättre att lägga energi på hur en modell bäst kan användas än att lägga energin på diskussioner om vilken modell som passar bäst. Eftersom CMM-modellen är anpassningsbar och

flexibel (Paulk, 1998), valde vi att använda den som modell i vårt arbete. CMM är dessutom den SPI-modell som är mest använd vid SPI-projekt anser Jalote (2002).

CMM utkom år1991 och består av en femgradiga mognadsskala som ursprungligen tillämpades på programutveckling och härrör från ett projekt initierat och sponsrat av det amerikanska flygvapnet Department of Defefense(DoD). Projektet realiserades vid Software Engineering Institute vid Carnegie-Mellon universitetet. Det avsåg att ge militären en objektiv metod att mäta kvaliteten på programutvecklingsföretag inför militära upphandlingar. Modellen klassificerar programvaruproducerande organisationer i princip efter hur väl de har kontroll över sin programproduktionsprocess. (SEI, 2003a)

Vi använde bara delar ur CMM och inriktade oss på nivå två i skalan. Denna nivå har nyckelområdena; kravhantering, projektplanering, projektuppföljning, hantering av underleverantörer, kvalitetssäkring samt konfigurationsledning. Vid den studerade avdelningen användes inga underleverantörer, varför detta nyckelområde inte behandlas i vår studie. I nivå 2 är hörnstenarna projektplanering och kravhantering. Johnson och Brodman (2000) menar att om metodanvisningarna följs redan från början kan arbetet styras på ett mer målinriktat sätt.

CMM-modellen togs först fram för stora organisationer och företag, och är därför i sin helhet alltför omfattande för företag där de personella resurserna är begränsade konstaterar Paulk(1998). Modellen innebär även att en stor mängd dokumentation upprättas, vilket gör att den kan kännas byråkratisk och betungande för mindre företag. Men strukturerade programutvecklingsprocesser är viktigt även för mindre företag och organisationer. CMM bör därför anpassas till organisationen och ses som en guide snarare än ett diktat. Om organisationen sedan använder guiden med sunt förnuft, så kommer mycket av det som ses som problematiskt att försvinna.

Paulk(1998) påpekar att det är en fördel med att införa processförbättringar med hjälp av anpassad CMM är att modellen då blir lättare att implementera och upprätthålla. Modellen blir lättare att förstå och nyttan av de olika aktiviteterna klargörs tydligare. Det är även viktigt att arbetet med att skräddarsy CMM görs inifrån organisationen, så att alla de vars arbete påverkas av modellen får vara med och utforma den. Processer som definierats av användarna själva har mycket större chans att bli accepterade av användarna

1.3. Problemformulering och syfte

Vi har därför valt att fokusera vår studie på:

Hur använda CMM för att införa SPI vid småskalig system utveckling?

Syftet med denna uppsats är att generera kunskap om en skräddarsydd anpassning av processförbättringsmodellen CMM, samt hur en sådan anpassning kan bidra till förbättringsarbetet i ett systemutvecklande företag.

Syftet ska uppfyllas genom att besvara följande frågor med studier av teori och med hjälp empirin från våra kvalitativa intervjuer.

1. Hur genomförs systemutvecklingsprocessen idag kontra CMM?
2. Hur införa CMM inom småskalig systemutveckling?

1.4. Disposition

Denna uppsats har vi lagt upp så att vi först förklarar vilken metod vi använt och hur intervjuerna gick till. Därefter följer en kort presentation av det företag där studien ägde rum. Efter detta förklarar vi de teorier och centrala begrepp vi använt oss av. Teoriavsnittet är ganska omfattande, så vi har lagt en separat disposition för detta först i teoriavsnittet. Efter att teorier och centrala begrepp är förklarade så finns intervjusvaren redovisade. Vi har valt att redovisa dem efter tre teman. Därefter har vi gjort en analys av intervjuerna. Vi analyserar dem då efter CMM:s nyckelområden kvalitetshantering, kravhantering, projektplanering och konfigurationshantering. I kapitlet som följer därefter tar vi upp några viktiga aspekter att beakta vid införande av modellstyrd systemutvecklingsprocess vid småskalig systemutveckling. Som sista punkter i denna uppsats har vi valt att lägga våra egna synpunkter och en kort sammanfattning av resultatet.

2. Metod

I vår studie har vi avgränsat oss till att studera arbetsprocessen och hur den kan förbättras när det gäller småskalig systemutveckling. När det gäller arbetsprocessen har vi avgränsat oss till att studera den vad gäller planering, arbetsprocessen vid utförandet och kvalitetshantering.

Vid genomförandet av studien använde vi oss av en kvalitativ ansats. En av anledningarna till detta val var att populationen var så liten. En annan anledning var att vår studie har hermeneutisk grund, utifrån vår frågeställning ville beskriva och förstå den nuvarande arbetsprocessen i just detta företag. Vårt val av metod stöds därmed även av professor Jan Trost. I sin bok ”Kvalitativa intervjuer” (1997) beskriver han valet mellan kvalitativ eller kvantitativ metod med orden:

> *”Något förenklat: Om frågeställningen gäller hur ofta, hur många eller hur vanligt så skall man göra en kvantitativ studie. Om frågeställningen däremot gäller att förstå eller att hitta ett mönster så skall man göra en kvalitativ studie.”*
> (sid. 16)

För att kunna förstå och beskriva den nuvarande arbetsprocessen krävs kvalitativa data, som innehåller den intervjuades attityder, åsikter, tankar och erfarenheter. Vi ville även få en helhetsbild av arbetsprocessen, och där ligger styrkan i kvalitativa data. De visar på totalsituationen (Holmberg &Solvang, 1997).

2.1. Intervjuer

Den största källan till data i en kvalitativ studie är intervjuer. Vi intervjuade fyra personer, tre som arbetade med systemutveckling och en uppdragsledare. Intervjuerna genomfördes under perioden 18 till 22 september 2003. Eftersom företagets systemutvecklingsavdelning består av tre personer plus en uppdragsledare så var urvalet enkelt. Före vi tog kontakt med respondenterna skaffade vi sektionschefens tillåtelse att intervjua dem. Sektionschefen kontaktade sedan de tilltänkta respondenterna och förberedde dem på att vi skulle ta kontakt med dem och att de kunde avsätta nödvändig tid för våra intervjuer. Detta ansåg vi vara mycket viktigt, eftersom företaget är ett konsultföretag där all tid måste debiteras. Vid vår första kontakt med respondenterna förberedde vi dem även på vad vi skulle intervjua dem om. Tid och plats för intervjuerna överlät vi till respondenterna att avgöra.

Före intervjuerna informerade vi respondenterna om vad vårt arbete gick ut på och hur intervjumaterialet skulle användas. Vi informerade dem även om att de var anonyma i den bemärkelsen att vi inte nämner några namn i uppsatsen, samt fick deras tillåtelse att citera dem. Intervjuerna, som varade drygt en timme per styck och bandades, genomfördes på företaget där vi fick disponera ett rum där vi kunde prata ostört. Detta för att de intervjuade skulle få befinna sig i sin ”hemmamiljö”. Vi hade inte satt någon tidsmässig gräns för intervjuernas längd, utan de intervjuade kunde ta tid på sig och behövde inte känna sig stressade av klockan. Allt detta för att respondenterna skulle känna sig så bekväma och trygga som möjligt, vilket är av stor vikt för att få ett så bra resultat som möjligt.
Vår intervjumanual delade vi in i olika teman. Dessa teman har de KPA(Key Process Areas) som ingår i CMM-modellens nivå 2 legat till grund för. Dessa KPA:s är Requirements Management, Software Project Planning, Software Project Tracking and Oversight, Software Quality Assurance och Software Cofiguration Management. Nivå två innehåller även Software Subcontract Management, men detta KPA saknar relevans vid den småskaliga systemutveckling vi studerade. Våra teman var:

1. Respondentens bakgrund och nuvarande arbetsuppgifter
2. Planering av uppdrag
3. Arbetsprocessen vid genomförandet
4. Kvalitetshantering

Först ställde vi några inledande frågor om respondenternas bakgrund och nuvarande arbetsuppgifter. Detta för att en av hermeneutikens huvudpunkter är att tolkning måste ske i förhållande till en kontext (Wallen, 1996). Som sista punkt i tema 2, 3 och 4 fick respondenterna ge sin egen syn på vilka problem som de såg inom området och föreslå lämpliga förbättringsåtgärder.
Fullständig intervjumanual medföljer som bilaga.

3. Fallstudien

I vår rapport använder vi oss av begreppen *koncern, företag* och *avdelning.* Med koncern menas alla företag som ingår i industrikonsultkoncernen. Med företag avses det lokala konsultföretaget. Begreppet avdelning används för den del av det lokala konsultföretaget där småskalig systemutveckling pågår.

3.1. Bakgrundsfakta om företaget

Koncernen är en stor industrikonsult som funnits i över hundra år. Företaget har ett brett utbud av konsulttjänster. En del av koncernen hjälper Ericsson i framtagandet av nya mobiltelefoner, en annan bistår Holmen att bygga en ny pappersmaskin. Det är tänkt att deras utbud av discipliner ska matcha alla kundens behov av industrikonsulttjänster. Företagskoncernen är noterad på Stockholmsbörsen. Koncernen hade en nettoomsättning år 2002 på 2miljarder SEK och en personalstyrka på 2600 personer. Koncernen har, som många andra konsultfirmor, fått sämre lönsamhet och dragit ner på antalet anställda till 2400 stycken i år och omstrukturerat verksamheten från 16 bolag till fem. De fem nya områdena är: Energi/Miljö, installation, Papper/Massa, Industri/System och Nya marknader. I Nya Marknader ingår fyra mindre bolag med spridda verksamheter.

Det lokala företaget vi studerade har 35 stycken anställda. Tidigare hade inriktningen hos företagets konsulttjänster varit mot elbranschen, men efter en omorganisation den förste januari 2003 är den mot flera branscher. På företaget finns det tre bolag representerade ur koncernen. Det är Energi/Miljö, i den delen jobbar bara en konsult. Med Installation arbetar fem konsulter. På Industri/System delen, vilken vår studerade avdelning tillhör, jobbar 25 konsulter totalt, av vilka 15 förvaltar ett system åt en stor svensk industrikoncern. Detta system har en iterativ livscykelmodell och systemutvecklingsarbetet är modellstyrt enligt **I**terative **A**pplication **D**evelopmen (IAD), som en annan industrikonsult har utvecklat. Förvaltningsuppdragets arbetsmetoder var färdigutvecklade när företaget tog över.
På kontoret finns ytterligare några anställda som sköter administrationen på företaget.

Regionchefen har sitt kontor på företaget. I regionen ingår också flera andra kontor.

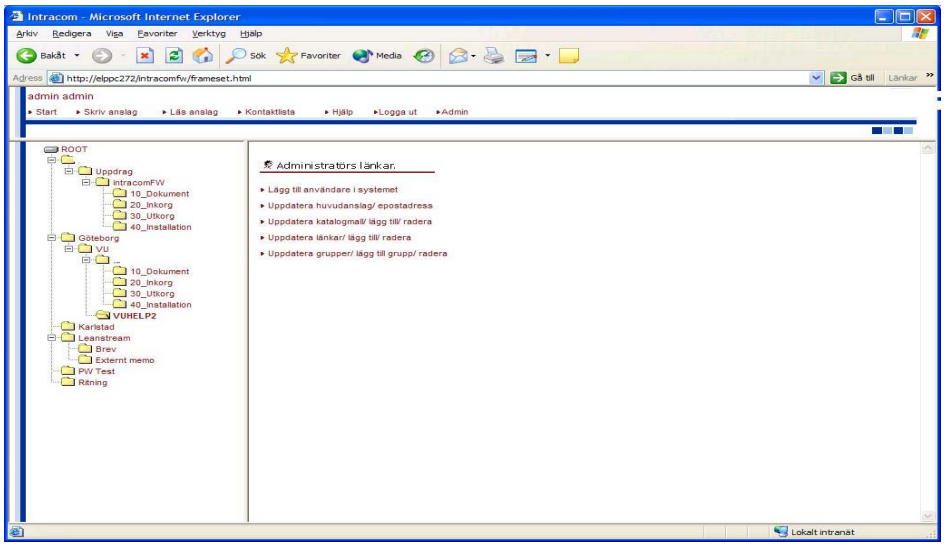
3.2. Avdelningen

Utanför förvaltningsuppdraget jobbar det tre stycken programutvecklare och en uppdragsledare med diverse lokala konsultuppdrag till små och medelstora företag. Inom koncernen används beteckningen uppdrag i stället för projekt, därav beteckningen uppdragsledare.
Den avdelning vi undersökte arbetar med vad vi definierar som småskalig systemutveckling. Betecknande för denna systemutveckling är:

- Liten personalresurs: Uppdragen var ofta inte större än att de innefattade två till tre personer. Arbetsfördelningen skedde utifrån de kunskaper de hade inom programmeringsområdet. Emellanåt utfördes uppdragen i samarbete med

systemutvecklare från andra kontor. Uppdragsledaren hade det formella ansvaret för projekten och skötte vanligtvis kundkontakten.

- Kort livscykel: Avdelningens systemutveckling utfördes sekventiellt. Att den är sekventiell innebär att den består av ett antal faser. Arbetsgången var sådan att man avslutade en fas innan man gick vidare till nästa. Denna typ av systemutveckling följer det traditionella mönstret med analys, design, implementation och testfas. Utöver detta arbete tillkom underhåll och drift av vissa applikationer.
- Relationsdatabas: Bestämdes av systemutvecklarna utifrån det antal klienter som skulle arbeta i systemet. Om det var ett enanvändarsystem så användes MS Access, men mest användes MS SQL, eftersom det oftast rörde sig om större system.
- Flerskiktslösningar: Behovet av arkitekturlösning bestämdes utifrån den modell som gjordes innan programmering inletts.
- Homogen programmeringsmiljö: Programmeringen gjordes oftast i ASP 3.0 (application service provider) eller Visual Basic 6. I något av dessa programspråk hade de flesta av systemutvecklingsarbetena utförts.



Figur 1 Användargränssnitt som utvecklats på avdelningen.

Ett exempel på system som utvecklats på avdelningen är ett ärendehanteringssystem åt en stor svensk industrikoncern *(fig. 1)*. Systemet är uppbyggt så att om en användare av ett annat stort datasystem får problem så loggar användaren in i systemet och skapar ett felmeddelande där de talar om vilken typ av problem de har. Felmeddelandet lagras i databasen och får ett unikt nummer. På ett annat kontor sitter en grupp konsulter och får fram de olika felmeddelandena och åtgärdar dem. Om så behövs kontaktas kunden via telefon, men de flesta problem löses via webben. Detta ärendehanteringssystem användes av klienter runt om i hela landet.

Avdelningen har även utvecklat ett flertal webbapplikationer åt stora svenska industriföretag men arbetar även mot mindre lokala företag

4. Teori

4.1. Teoridisposition

I teorikapitlet börjar vi med att förklara lite kort om vad Software Engineering är. Därefter berättar vi om SPI (Software Process Improvement). I avsnitt 4.4. presenterar vi kortfattat CMM (Capability Maturity Model) och dess nivåer. Efter detta redovisar och förklarar vi Paulks tio viktiga punkter att beakta vid införande av modeller i små företag följt av SSM (Soft Systems Methology), som erbjuder guidelines för kvalitetsbestämning.

4.2. Software Engineering

Software engineering eller programvaruteknik är den fackmässiga disciplinen för att systematiskt och förutsägbart åstadkomma korta ledtider, låga kostnader och hög kvalitet i industriell programvaruutveckling. Software Engineering innehåller generellt tre nyckelelement: metoder, verktyg och procedurer (processer), som gör det möjligt för en chef att styra programutvecklingsprocessen och tillhandahålla dem som praktiserar programutveckling en grund för att bygga högkvalitativ programvara på ett effektivt sätt (Carlsson,2002). Software engineering delas in i ett managementperspektiv och ett teknikperspektiv. Managementperspektivet handlar om modeller och metoder för programvaruutvecklingsprocessen som helhet. Det perspektivet används för att förstå ekvationen "tid-kostnad-kvalitet" och för att kunna mäta och uppskatta dessa attribut. Teknikperspektivet behandlar metoder och modeller för de tekniska aspekterna i programvaruutvecklingens olika faser, t ex specifikation, konstruktion, testning.
(IT-universitetet ,2000).

4.3. Software Process Improvement (SPI)

SPI är en arbetsmetod som passar för samtliga IT-bolag. Konceptet bygger på att IT-företagen med avsikt och systematiskt arbeta med att förbättra systemutvecklingsverksamheten och därmed skapar förutsättningar för att lyckas med sina IT-projekt. Arbetet är en process som ska ske långsiktigt och kontinuerligt. Med hjälp av SPI kan företaget leverera det kunden vill ha och i tid.
Metoden kan inte fungera utan företagsledningens stöd eftersom det krävs resurser att införa och följa upp. Det är mätningar av resultatet som återförs som ett av styrmedlen i processen (Systems Engineering Process Office,2003).

Intresset för Software Process Improvement (SPI) i Europa tog fart i samband med ISO-certifieringen(O Hara,2000).. För att stimulera och införa SPI i Europa beslöt Europa kommissionen att bistå med pengar till organisationer som var med i ESSI European Software System Initiative. Målet för ESSI var att stödja de organisationer som införde bättre systemutvecklingsarbete och det i sin tur skulle skapa ytterligare

förbättringsmetoder och på så sätt på sikt skapa en bättre mjukvaruindustri. Initiativet har medfört att arbetsmetoden är på frammarsch inom IT-bolagen i Europa och har medfört kvalitetsförbättringar. Ändå är SPI inte alls så utbredd och använd här i Europa som i USA.
I USA användes en annan metod för att införa SPI. Där fick utveckling av verktyg som befrämjade SPI stöd. Detta var en av anledningarna att SEI utvecklade CMM och liknande metoder

En ledstjärna i SPI är att kvalitet ska byggas in i processen. För att få SPI måste det finnas en egen ansats hos organisationen av att vilja förändra och förbättra arbetet. Förändringar bör ske där behovet är som störst. Förändringsarbetet bör ske i projektform för det är då enklare att stämma av och mäta arbetet. Det är också viktigt att man använder sig av någon processändringsmodell av typ IDEAL eller CMM. Det finns mätbara vinster för företagen att hämta med hjälp av SPI. Vinster som tid, kostnad och kvalitet är enkla att summera. Men organisationen kan även få vinster som är svåra att räkna i pengar.

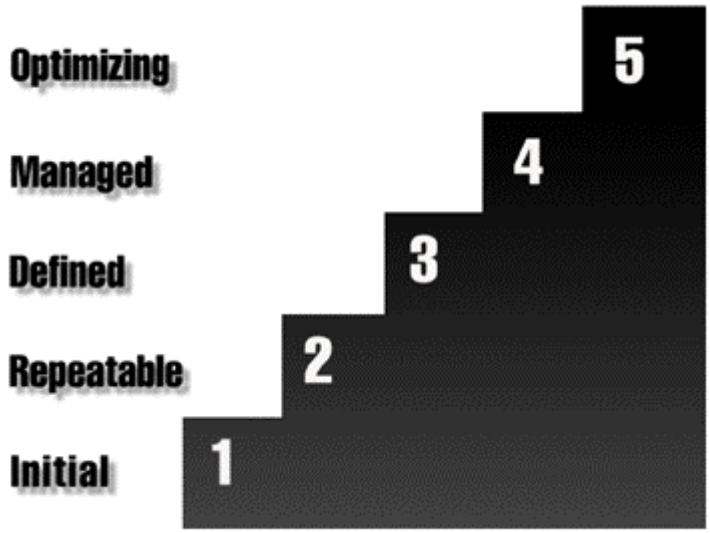
1. Minskat risktagande i projekten.
2. Bättre och stabilare arbetsmiljö för de anställda.
3. Ökat förutseende av resultaten.
4. Bättre anseende hos kunderna efter lyckade projekt.
5. Mätbara projektprestanda.

(IT-universitetet ,2000)

4.4. Capability Maturity Model (CMM)

CMM är en modell som skapades av the Software Engeneering Institute (SEI). SEI är ett federalt forsknings- och utvecklingscenter, som etablerades av USA:s försvarsdepartement 1984. Anledningen till att SEI grundades var att man ville komma tillrätta med de problem som var, och är, förknippade med programutvecklingsprojekt. Exempel på sådana problem är att tids- och ekonomiska ramar överskrids, en funktionalitet som inte svarar upp mot användarens behov och tveksam kvalitet (Paulk, 1998)**.** När det gäller utveckling av programvara så sammanfaller kvalitet i produkt med kvalitet i process. Programvarans kvalitet är alltså beroende av processens kvalitet, och CMM-modellen ger företaget riktlinjer som leder mot bättre kvalitet i processen.

CMM:s grundidé (SEI, 2003b) är att kontinuerliga förbättringar i små steg är bättre än stora genomgripande förändringar. Därför är modellen indelad i fem steg eller mognadsnivåer (se figur 2), som beskriver en evolutionär utveckling från ad hoc till strukturerade programutvecklingsprocesser.



Figur 2 De fem mognadsnivåerna i CMM (SEI, 2003c, s.1)

Nivå 1, initial: På denna nivå finns få eller inga processer för programvaruutveckling definierade. Utvecklingsprocessen är ad hoc och ibland kaotisk. Framgång bygger på individuella insatser och hjältedåd.
Nivå 2, repeatable: För att kunna spåra de individuella projektens kostnader, tidsschema och funktionalitet, har en grundläggande projektledningsprocess införts. Erfarenheter från tidigare projekt tas tillvara. Dessa kan sedan användas när nya liknande projekt skall genomföras. Här införs också mätningar, som är en viktig förutsättning för att kunna åstadkomma en förutsägbar process. Mätetal från tidigare projekt kan sedan ligga till grund för framtida projekts tids- och kostnadsplaner.
Nivå 3, defined: Eftersom olika projekt har olika förutsättningar, skräddarsys processen till varje enskilt projekt enligt givna riktlinjer. Detta innebär att programutvecklingsprocessen, både vad gäller projektledning och teknisk utveckling, är dokumenterad och integrerad med en standardprocess för verksamheten. Utförande av mätningar är integrerat i modellen.
Nivå 4, managed: Detaljerad mätdata samlas in. Mätetalen används för att styra och förbättra process och produkt.
Nivå 5, optimizing: På den här nivån sker kontinuerligt förbättringar baserade på erfarenheter och kvantitativa återkopplingar från tidigare projekt. Nya innovativa idéer och tekniker testas i pilotförsök.

Alla nivåer, utom nivå ett, innehåller ett antal nyckelområden (Key Process Areas), som beskriver element som är viktiga för att uppnå uppsatta mål. Organisatoriskt innebär CMM att ansvar definieras och fördelas på ett stort antal roller, och att projektgrupper med specifika arbetsuppgifter sätts upp.

4.5. Paulks tio punkter

För att på ett framgångsrikt sätt kunna implementera CMM i en liten organisation, har Paulk (1998) tagit fram tio viktiga punkter att beakta. Dessa är:

- Stöd från ledningen (management sponsorship)
- Mätningar (measurement)
- SEPG (Software Engineering Process Group)
- "as is" processer
- Dokumentering av processer (documented processes)
- Anpassning av modellen (tailoring)
- Träning (training)
- Riskhantering (risk management)
- Planering (planning)
- Inbördes utvärdering (peer reviews)

Stöd från ledningen är en ytterst viktig del av arbetet med att förbättra en organisations arbetsprocesser. Som individer kan vi förändra och förbättra arbetsprocessen inom vårt eget arbetsområde, men om en förändring skall ske inom organisationen som helhet, så måste det finnas aktivt stöd från ledningen.

För att kunna se att förändringsarbetet har positiva effekter skall mätningar av dessa effekter utföras. För att mätningarna skall vara meningsfulla, måste man förstå vad mätetalen betyder och hur man skall analysera dem. Därför måste valet av mätetal göras mycket noga. I annat fall kan analysen ge oanvändbara eller felaktiga resultat.

En utvecklingsgrupp (SEPG) eller en ansvarig person bör tillsättas för att koordinera förbättringsarbetet.

Börja med "as is", alltså hur processen ser ut idag, och inte med hur den borde vara. Detta är viktigt för att kunna utföra förbättringsarbetet i små steg och därmed undanröja eventuellt motstånd bland personalen.

Dokumentera processer. Dokumenterade processer stödjer organisationens lärande och undanröjer ett ständigt uppfinnande av hjulet. Dokumentationen bör dock inte vara alltför komplex och omständlig.
Hur dokumentationen skall göras beslutas av organisationen själv.

Processer måste anpassas till organisationens och projektets behov. Standardprocesser skapar ett ramverk, men varje enskilt projekt har unika behov och förutsättningar. Standardprocessen får därför inte vara rigid eller alltför formell.

Meningen med training är att utveckla yrkesskicklighet. Detta kan göras på flera sätt, Paulk förordar "mentorship", alltså att lära sig genom att arbeta och "praktisera" under överinseende av en person med större erfarenhet och kunskap.

CMM handlar på ett sätt om riskhantering. För små projekt kan vattenfallsmodellen användas som livscykelmodell, annars kan spiralmodellen användas. Om kunden är inblandad i systemutvecklingen, kan prototyping eller gemensam applikationsdesign vara bra.

Planering behövs för varje ”större” programvaruprocess, men det är organisationen själv som avgör vad som är ”större” och hur planen skall se ut.

Peer reviews – någon form av inbördes utvärdering under förbättringsarbetet.

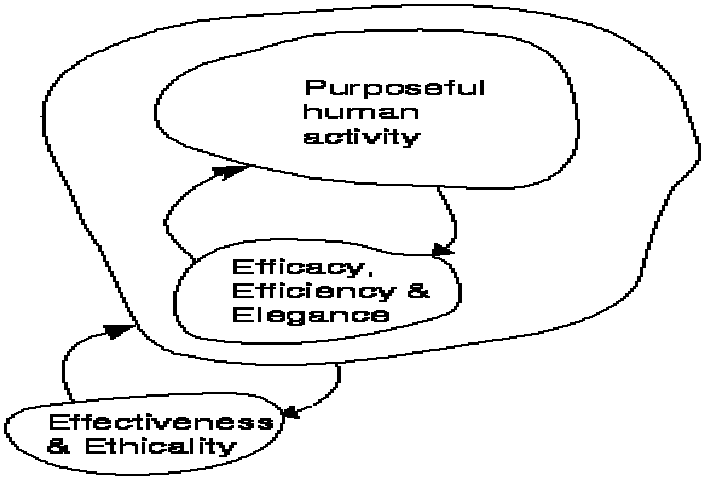
4.6. Soft Systems Methodology (SSM)

Soft Systems Methodology, SSM, är en generell metod för att undersöka och lära sig om en problemsituation. Denna metod undersöker problemsituationen för att sedan hitta en passande lösning till att förbättra situationen. Metoden utvecklades 1960 av Peter Checkland vid Lancasters universitet.

Utifrån denna metod finns fem kriterier för att se om IS-kvalitet uppnåtts. Dessa är:

Efficacy – producerar systemet den output som efterfrågas?
Efficiency – produceras efterfrågad output med minsta möjliga resursåtgång?
Elegance – har gränssnitt och output en bra design? Är de estetiskt tilltalande? Är de för komplicerade, över eller underarbetade?
Effectiveness – möter processen, som systemet utför, organisationens övergripande mål på sikt?
Ethicality – är systemets arbete acceptabelt bedömt ur värdemässiga perspektiv, där de värdemässiga bedömningarna kan vara ”bra” eller ”dåligt”?

Här kanske man kan tycka att även Economy borde vara ett kriterium för kvalitet. Men, menar författarna, ekonomi har med resursåtgång att göra, och därmed finns ekonomi redan med i kriteriet Efficiency.



Figur 3 Yttre och inre kontroll-loop (Vidgen et al, 2000, s.8)

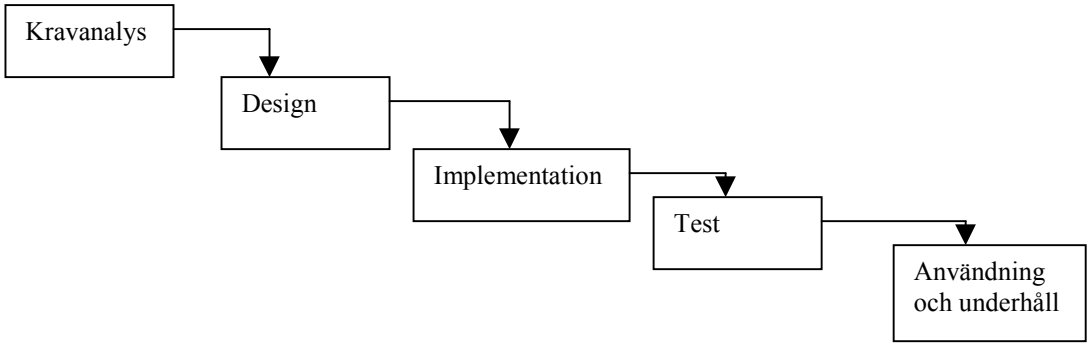
De här fem E-kriterierna behöver inte operera på samma nivå i systemmodellen. Det är bättre om modellen har flera kontrollnivåer. Figur 3 visar detta ur ett övergripande perspektiv. Efficacy, Efficiency och Elegance ligger här i en inre kontroll-loop, medan Effectiveness och Ethicality ligger i den yttre kontroll-loopen. På detta sätt ligger alltså kontrollen på ”detaljnivå” i den inre loopen och kontrollen på ”övergripande nivå” i den yttre. Tillsammans bildar de en helhetssyn på IS-kvalitet (Vidgen et al,) .

5. Centrala begrepp

5.1. Sekventiell och iterativ livscykelmodell

Företaget vi studerade hade en avdelning som hade ett systemförvaltningsuppdrag där det redan fanns en fungerande arbetsmodell, nämligen IAD. Det var då frestande att utgå från denna modell även i vår studie av företagets övriga systemutvecklingsarbete. Uppdragens natur skilde sig dock avsevärt. Avdelningen som arbetade enligt IAD-modellen var relativt stor och arbetade med ett uppdrag som sträckte sig över en längre tidsperiod och uppdragsgivare var ett stort företag. Uppdragen till små och medelstora företag var däremot mera tids- och resursbegränsade. De olika förutsättningarna för uppdragen avspeglade sig även i systemutvecklingens livscykelmodell. Vid systemutveckling för små och medelstora företag användes sekventiell modell, och i förvaltningsuppdraget användes iterativ modell. Få stora projekt, alltså projekt som löper över en lång tid, har en så stabil miljö att vattenfallsmodellen är den livscykelmodell som är att föredra, men att den kan vara utmärkt för mindre projekt menar Paulk (1998).

Vattenfallsmodellen är ett exempel på en sekventiell modell. Att den är sekventiell innebär att den består av ett antal faser. Arbetsgången är sådan att man avslutar en fas innan man går vidare till nästa. Faserna och deras inbördes förhållande ses i figur 4.



Figur 4 Vattenfallsmodellens faser (Näsström et al, 1999, s.8)

Faserna är:

- **Kravanalys:** tillsammans med kunden analyseras vad systemet skall göra och vilka problem som skall lösas. Arbetet i denna fas mynnar ut i en kravspecifikation.
- **Design:** systemet modelleras genom att en övergripande arkitektur tas fram och delkomponenter specificeras.
- **Implementation:** de olika komponenterna utvecklas och testas.
- **Test:** de olika delarna sätts samman och testas tillsammans.
- **Användning och underhåll:** systemet installeras hos kunden och tas i bruk. Fel som upptäcks rättas och eventuella andra anpassningar görs.

I en iterativ modell byggs systemet fram successivt. Man kan då arbeta med mer än en fas åt gången, man måste alltså inte göra färdigt en fas för att kunna påbörja nästa. Oavsett vilken fas man befinner sig på, så kan man alltid gå tillbaka en eller flera faser och upprepa stegen. Man kan därför inte ha samma hårda tidsplanering vid arbete enligt en iterativ modell som enligt en sekventiell. Det blir ju svårare att sätta deadlines för arbetet i de olika faserna, eftersom man hela tiden kan gå tillbaka. Tidsplaneringen blir därför mer som riktlinjer för när olika saker skall bör vara färdiga. Detta innebär att det är svårare att ha kontroll över var man befinner sig i utvecklingsarbetet i en iterativ modell jämfört med en sekventiell modell.

Eftersom ett system som tas fram genom en iterativ modell ständigt förändras och förbättras, behöver det inte bli ”färdigt”.

Vid arbete enligt IAD (Iterative Application Development) – modellen utmynnar varje iteration i en version av systemet som kan börja användas direkt. Varje version

kallas *Pilot,* och feedback från den implementerade piloten används vid specifikation och utveckling av nästkommande piloter (Näsström et al, 1999).

5.2. Kvalitetsbegreppet

Kvalitet är, och har alltid varit, en viktig del för att skapa bra förutsättningar för alla sorters verksamhet. Detsamma gäller för skapande av informationssystem. Men begreppet kvalitet är mycket brett. Därför kan det kanske vara på sin plats att definiera begreppet.

En definition, som fått stor spridning, har gjorts av International Standards Organization (ISO 1986). Den lyder:

> *”The totality of features and characteristics of a product or service that bear on its ability to satisfy specified or implied needs”.*

Den andra är:

> *”Quality is a judgement by customers or users of a product or service; it is the extent to which the customers or users believe the product or service surpasses their needs and expectations”* (Vidgen et al, 2000).

Kvalitet är alltså till vilken grad kundens behov och förväntningar uppfylls.

Detta är ju ganska tydligt och klart, men tyvärr så är inte kvalitet när det gäller informationssystem lika lätt att definiera, och en orsak till detta är att det finns olika synsätt på vad kvalitet är. Pekka Rejonen (2000) hävdar att datavetenskapen och IS-vetenskapen har ganska stora skillnader i sin syn på vad ett IS-system är. Datavetenskapens företrädare ser på ett informationssystem ur teknisk synvinkel. De ser det som ett system bestående av datorer och telekommunikationsteknologi, medan IS-vetenskapen ser det som ett socialt system, som en organisation och dess informationsbehov. För systemutvecklaren är systemet i sig det viktiga medan det bara är ett verktyg för användaren.
Han hävdar att de olika synsätten på informationssystem är någonting positivt som bör uppmuntras. En av anledningarna till detta är att kvalitet ur användarens synvinkel inte är möjlig om inte teknisk kvalitet finns. En annan anledning är att användare av datasystem ofta behandlar det tekniska systemet som en ”black box”, det vill säga som någonting man inte vet så mycket om hur det fungerar. Man är bara intresserad av dess output, och har inte den kunskap som behövs för att avgöra om denna ”black box” har teknisk kvalitet eller inte (Dahlbom & Mathiassen, 1995). Enligt Rejonens resonemang är alltså den tekniska kvaliteten avgörande för systemutvecklare. Användarkvaliteten skapas enligt rapportförfattaren först när användarna lärt sig att använda systemet.

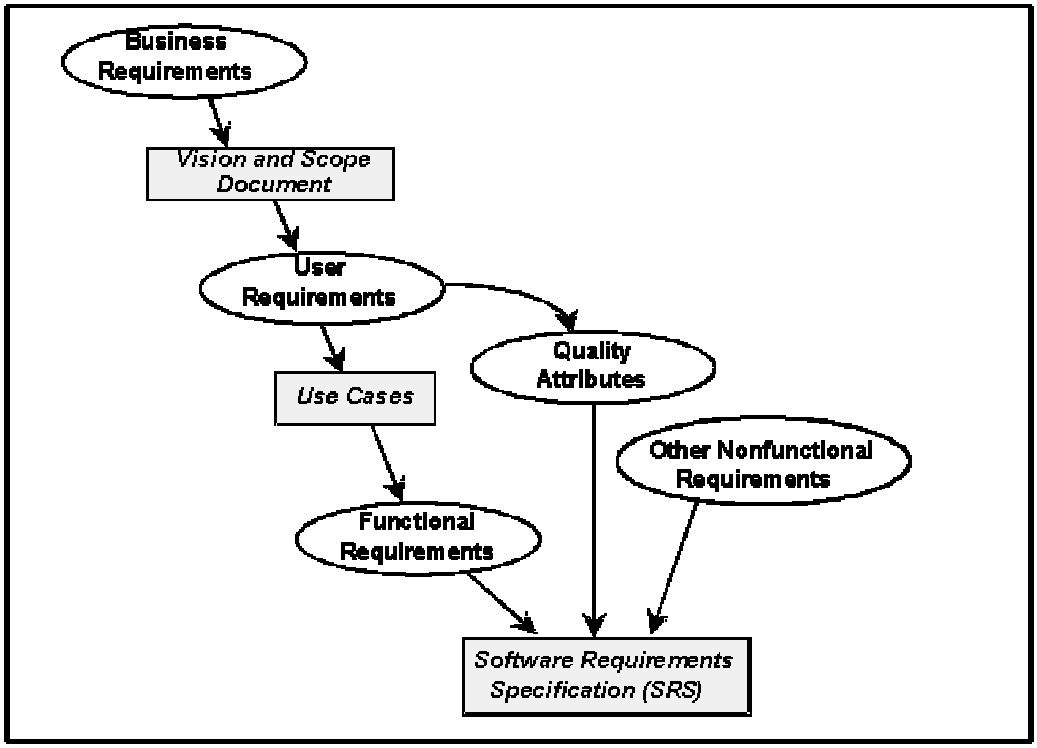
Ett annat exempel på de olika synsättens inverkan på kvalitetsbegreppet ger Vidgen et al (2000) uttryck för. Även här tar författarna upp att den traditionella synen på kvalitet i samband med IS-utveckling har varit inriktad på den tekniska kvaliteten. Detta har gjort att man har sett systemets kvalitet som applikationens kvalitet. De vill

i sin rapport visa på betydelsen av att ha en bredare syn på begreppet kvalitet. Denna syn koncentrerar sig på om systemet kan svara upp mot användarnas behov (se definitionen av kvalitet ovan). Den tekniska kvaliteten är visserligen nödvändig för att säkerställa IS-kvalitet, men enbart teknisk kvalitet räcker inte. Ytterligare ett viktigt problem som tas upp i rapporten är att det finns användare från många olika nivåer och positioner i en organisation, som använder sig av ett system. Man måste då ställa sig frågan: Har de olika användarna samma syn på vad som är kvalitet? Om inte är det viktigt att avgöra vems kvalitet som skall tillgodoses, exempelvis beställarens eller användarnas?

Som vi kan se så finns det flera aspekter att ta hänsyn till när det gäller begreppet kvalitet. När skall vi anse att systemet har fullgod kvalitet, när det tekniskt uppfyller kvalitetskraven (objektiv kvalitet) eller när kunden är nöjd (subjektiv kvalitet)? Svaret är att kvalitet är uppnådd först när både objektiv och subjektiv kvalitet är uppnådd (Dahlbom & Mathiassen, 1995).

5.3. Kravhantering

Kravhantering är en viktig, men långt ifrån enkel, aktivitet vid all systemutveckling. Karl Wiegers (2002a) belyser detta och beskriver ett arbetssätt för att ta fram en bra kravspecifikation. Han hävdar att även själva ordet “krav” kan ställa till besvär. Ordet ”krav” har olika betydelse för olika människor. För en person på chefsnivå kan ”krav” betyda ett koncept eller en affärsvision, medan systemutvecklarens ”krav” mera liknar en detaljerad användargränssnittsdesign och ”krav” från kunden ofta egentligen är idéer till hur lösningen skall se ut. Man bör därför arbeta med kravhanteringen i tre nivåer. Figur 5 visar de tre nivåerna. I toppen finns affärsmässiga krav som representerar mål på en hög nivå. De skall beskriva hur ”världen blir bättre” om detta system implementeras. På mittennivån tas användarkraven fram. Detta kan göras med hjälp av användarfallsdiagram. Utifrån användarfallen tas så de funktionella kraven fram, som i modellen ligger på den lägsta nivån. Kravspecifikationen skall vara en behållare för både funktionella och icke-funktionella krav. Till icke-funktionella krav hör sådant som kvalitetsattribut (exempelvis usability, efficiency, portability, maintainability), affärsregler, förutsättningar för design och implementation mm.



Figur 5 De tre nivåerna vid framtagande av kravspecifikation (Wiegers, 2002a, s.2)

Utöver detta arbetssätt för att få in så många kravaspekter som möjligt, så måste även stor vikt läggas på formulering av kraven. De får inte vara dubbeltydigt, allmänt eller otydligt skrivna, de måste harmonisera med varandra så att inte två krav motsäger varandra, de skall vara korrekta, nödvändiga och genomförbara inom ramen för projektets tillgångar.

6. Sammanställning av intervjuer

Vid vår sammanställning av intervjusvaren har vi valt att gruppera resultaten utefter de teman som ingick i vår intervjumanual.

6.1. Respondenternas bakgrund och nuvarande arbetsuppgifter

De intervjuade har alla någon form av datautbildning som grund och de har jobbat minst två år inom företaget. Vid intervjutillfället hade alla arbetat eller arbetade med systemutveckling. En har arbetat en längre tid i det modellstyrda förvaltningsuppdraget medan tre hade varit där en kortare period. Alla har alltså viss erfarenhet av att arbeta i en modellstyrd arbetsprocess. En av de intervjuade arbetar som uppdragsledare. Dennes uppgift är att skaffa kunder, planera uppdragen och sköta det administrativa arbetet runt uppdragen. I arbetsuppgifterna ingår även att ta fram priser på hårdvara och att ha hand om kontakten med kund under uppdragets

gång. Även systemutvecklarna kan dock ha direktkontakt med kunden om behov finns. Uppdragsledaren är också ekonomiskt ansvarig för uppdraget.

6.2. Arbetsplanering

Det första som sker när ett nytt uppdrag kom in, är att uppdragsledaren tillsammans med kunden tar fram en kravspecifikation. Utifrån denna uppskattar sedan systemutvecklarna den tid som krävs för genomförande av uppdraget. Med tidsuppskattningen och kravspecifikationen som grund tas så ett anbud fram, vilket kunden sedan får godkänna.

Vid upprättande av kravspecifikation finns standardiserade mallar för detta att tillgå. Dessa mallar har framtagits lokalt och användes från början mot elbranschen. Vid mindre uppdrag upprättas dock inte alltid en formell kravspecifikation, men kraven finns alltid dokumenterade. På frågan om kraven alltid finns dokumenterade svarar en respondent:

> *”Ja, men i olika grad detaljerad då förstås. Och en del uppgifter vi får är bara en uppgift: ”Lös detta”, men det finns ingenting som specar önskemål om hur det skall lösas, utan det är helt upp till en själv då och då har man ju fria tyglar. Och då ger ju inte den kravspecen lika mycket, för den skulle ju bara innehålla två rader. Så det är olika, men det finns alltid alltid något dokument i bakrunden som styr, ett mejl eller en beskrivning eller någonting som jag har att gå efter.”*

Om kunden ändrar sina krav så dokumenteras förändringen oftast. Är ändringen så pass stor att den påverkar det avtalade priset så dokumenteras ändringen alltid. Då ändras även avtalet och den nya avtalsversionen gäller. Är ändringen så liten att den ryms inom ramen för avtal och kravspecifikation, så dokumenteras den inte alltid. En respondent uttrycker det med orden:

> *”Generellt inom företaget är att vi är ganska dåliga på det. Det brukar slinka med ändå. Egentligen skulle det vara en tilläggsbeställning.”*

Huvudmålet är dock nöjda kunder. Men de intervjuade påpekar också vikten av att ha en dokumentation att stödja sig på om diskussioner med kunden skulle uppstå.

Uppdragen är klassificerade efter antal timmar de löper på. Denna klassificering är framtagen av koncernen. Större uppdrag innehåller mera dokumentation och all dokumentation rörande uppdraget sparas. Ibland kan mindre uppdrag skötas utan en speciell uppdragsgivare. Systemutvecklaren har då själv kontakt med kunden och fungerar samtidigt som uppdragsledare.

När systemutvecklingen utförs i grupp delas arbetsuppgifterna upp på den medarbetare som är bäst skickad att utföra den. Denna rollfördelning, alltså vem som gör vad, och i vilken ordning det skall göras, sköts av systemutvecklarna själva utan inblandning av uppdragsledaren och utan att det dokumenteras. Det har tagit gruppen ett tag att få sina roller utan någon direkt styrning. Deadlines görs upp med kunden

för leveransdatum, programmerarna diskuterar sedan sinsemellan om när de skall ta över efter varandra. Inom utvecklingsgruppen sköts kommunikationen genom att de pratar direkt med varandra, uppåt rapporterar de till uppdragsledaren, som i sin tur rapporterar vidare. Uppdragsledaren planerar även in möten med utvecklarna för att stämma av arbetet. Eftersom alla inom systemutvecklingen fungerar som konsulter med ansvar för sin egen del av arbetet, så sker styrningen av de mindre uppdragen för det mesta av systemutveclarna själva. Ibland jobbar de även tillsammans med andra kontor inom koncernen, och då blir uppdragsledarens styrande och övervakande roll mera viktig.

Respondenternas egna förslag till förbättringar när det gäller planeringsfasen är:

- Mer direktkontakt med kunden utan uppdragsledaren som filter. En uttrycker denna önskan med att:

  ”Att alla som har med någonting att göra, också går ut och pratar med kunderna om det, för att det är ju vi som är experter på just våra saker, och har man då en uppdragsledare eller någon projektledare som filter däremellan, då tappar man kvalitet och kanske även pengar i slutändan.”

- Dessutom önskades en enklare dokumentation för mindre uppdrag, den idag ansågs alltför omfattande.
- Ett gemensamt regelverk för projekthantering inom systemutveckling från koncernen centralt efterlystes. De olika systemutvecklingsavdelningarna inom regionen har ingen gemensam policy att gå efter.
- Kravspecifikationen bör ses över.

6.3. Arbetsprocessen under systemutvecklingen

Man arbetar idag inte efter någon metod. Arbetsprocessens början skiljer sig åt beroende på vilken typ av uppdrag det är, uppdragets storlek, om uppdraget liknar tidigare de gjort, hur pass specificerad kravspecifikationen är osv. Det första de gör är att göra en modell av arbetet, t ex att ställa upp villkor och specificera alla affärsregler som skall finnas med och var logiken skall ligga osv. Sedan kontrolleras den mot kravspecifikationen, så att modellen och kravspecifikationen stämmer överens. Vid mindre projekt med bara tre eller fyra projekt objekt, eller en typ av uppdrag man gjort många gånger, sätter man igång och programmerar direkt. Detsamma gäller om kravspecifikationen är tillräckligt detaljerad.

Vid problem som gäller själva programmeringen, vänder sig en till Internet först och sedan till kolleger på andra kontor. Däremot kan han få hjälp i huset när det gäller att förstå hur vissa saker funngerar, (t ex Active Directory). De andra vänder sig till kolleger i huset först.
När det gäller sparande av fungerande lösningar, så har en av de anställda en liten kodbank där han sparar lösningar för egen del. Den är inte hans personliga, utan tillhör företaget, men det är han själv som skapat den. Det är dock bara han själv som

använder den. Osäker på om någon annan har någon användning för den. Tidigare projekt sparas dock. Anledningen till detta är enligt en respondent att:

> **”***Vi kanske sitter på flera ställen och gör liknande program och produkter. Detta gör vi utan vetskap om varandra. Det ska samlas på intranätet.”*

Det finns inga regler för vilken dokumentation som skall finnas angående arbetsprocessen. På frågan om någonting dokumenteras svarar en respondent:

> *”Nej, egentligen inte, eller väldigt sällan i alla fall. Jag försöker att kommentera koden, det är väl egentligen den enda återanknytning vi har till själva arbetet och arbetets gång. Att man däri förklarar varför man har tänkt på ett visst sätt och varför man har gjort på ett visst sätt.”*

Utöver detta skapas en checklista där olika aktiviteter bockas av när de är klara och tid för olika delmoment. Denna lista har alla inblandade i uppdraget tillgång till och i den kan uppdragsledaren följa hur arbetet fortskrider. Man dokumenterar även tid som överskrider tidsbudget, men man specificerar den inte på olika delmoment eller aktiviteter.

Respondenternas egna förslag när det gäller förbättring av arbetsprocessen vid genomförande av uppdrag var:

- Fler personer inom fler kunskapsområden för att få ett bredare perspektiv.

Respondentens egna förklaring till detta var:

> *”Varje område skulle täckas med ett antal personer. Alla var inte lika duktiga men de hade lite kunskap. Alla områden skulle ha ett antal personer. Detta är viktigt att vi inte blir beroende av en person.”*

- En tydligare rollfördelning inom arbetsgruppen, rollfördelningen styrd uppifrån.
- En strukturerad och dokumenterad arbetsprocess. En sytemutvecklare säger:

> *”Egentligen skulle jag vilja ha mer på papper utverkat uppifrån. Vi själva har tagit fram den strukturen som finns. Om det skulle komma in ett projekt där det skulle ingå en sju åtta personer i gruppen då skulle det nuvarande inte fungera.”*

- Kontakt utvecklare – användare, inte användarens chef.

6.4. Kvalitetssäkring

De finns övergripande dokument för kvalitet men de gäller för hela koncernen, någon policy för kvalitet när det gäller företagets systemutvecklingsdel finns inte. Samtliga som jobbar där efterlyser även här ett dokument uppifrån centralt som beskriver arbetsgången i systemutvecklingsprojekt. En av de intervjuade säger:

> *”Det är det som är så jäkla dumt i (xxxx), de har ingen tradition när det gäller systemutveckling. Vi har efterlyst det! Vi har inget dokument som beskriver hur våra systemutvecklingsuppdrag ska gå till.”*

Den intervjuade poängterar även vikten av att det dokument som skall beskriva arbetsgången även gäller för hela företagets systemutvecklingsdel:

> *”Det är ytterligare ett problem den här strukturen eller den jag borde sätta upp, den borde sättas upp för hela företaget. Om jag sätter upp en struktur här och de klurar ut en annan på ett annat ställe hur dyrt blir inte det för företaget. Sen blir de olika och sen ska vi samarbeta. Värdelöst!”*

Uppdragsledaren ansvarar formellt för kvaliteten på uppdraget, men även systemutvecklarna själva ansvarar för sin del av arbetet. Systemutvecklarna testar själva programmen för att se om de är bugg fria. De har själva konstruerat sätt att kontrollera kvaliteten, och man gör kvalitetskontrollen så grundligt man bara kan. Det subjektiva kvalitetstänkandet finns med under hela uppdragets gång och man försöker göra en del i taget och testa den. En av respondenterna uttrycker också detta i intervjun:

”Det är ju lite svårt att kvalitetssäkra ett dataprogram, för det är ingen som vet om du har gjort rätt, inom fnuttar, i programmet, men det funkar ändå. Många säger att kvalitet är när kunden är nöjd, eller beställaren är nöjd. Jag vill inte alltid hålla med om det.”
Respondenten säger även att:

”Jag skulle vilja ha ett dokument som specar hur vi skall jobba, för det är ju egentligen upp till oss själva nu idag. Jag menar, man kan ju slarva och ändå få till en ganska bra grej, men den kanske inte håller. Den kanske kraschar inom ett halvår eller det är kanske svårt att bygga på den.”

Om kunden vill pruta på priset, så systemutvecklarna vet att de inte kan testa ett program riktigt, så skrivs detta in i anbudet. Man brukar följa upp sina uppdrag genom att ringa upp kunden efter ett tag och kontrollera hur det fungerar. Det är kunden som avgör kvaliteten, och de har nöjda kunder.
Det finns inte någon samlad bild över kvaliteten utöver nöjda kunder.

Deras egna förslag till förbättrad kvalitetshantering var:

- Återigen dokumenterade riktlinjer för hur ett uppdrag skall genomföras.

- Formella riktlinjer på hur man avslutar ett uppdrag på ett bra sätt. I dag är det så att när kunden godkänt så är det klart, men kunden kan hålla på länge och inte godkänna.

7. Analys

Intervjusammanställningen visar på en rad karakteristiska problem kring företagets systemutveckling idag kontra CMM. Vi skall här analysera de problem som är relaterade till CMM.s nyckelområden på nivå två.

7.1. Software Quality Assurance

Som vi kunnat se i avsnitt 5.2 (Centrala begrepp), så finns det flera aspekter att ta hänsyn till när det gäller begreppet kvalitet. Enligt Dahlbom & Mathiassen (1995), uppnås kvalitet först när både objektiv och subjektiv kvalitet är uppnådd, och det är systemutvecklarens ansvar att se till att så blir fallet. Författarnas beskrivning av detta lyder:

> *”Being responsible for the introduction of a complex and rapidly evolving technology into an organization, the systems developer is responsible for the future consequences of the use of that technology, for problems of maintenance and further development, for compability with future technology, and for the usefulness of the technology as the organization changes.”*
> (sid. 139)

Vid vår studerade avdelning var definitionen av kvalitet att när kunden var nöjd var nödvändig kvalitet uppnådd. Men av resonemanget ovan kan vi se att kvalitet inte till fullo kan mätas i kundens tillfredsställelse.

Respondenterna på avdelningen har samma uppfattning som Dahlbom & Mathiassens resonemang om kvalitetsdefinition. Men för att denna kvalitetsdefinition skall vara möjlig att uppnå, måste ett dokument som innehåller kvalitetspolicy och specificerar kvaliteten utarbetas. Detta för att alla inom systemutvecklingsavdelningen skall arbeta enligt samma riktlinjer. Dokumentet bör även gälla specifikt för systemutvecklingen.

För att kunna uppnå en bra tekniskt kvalitet och samtidigt tillfredsställer användarna, föreslås att Soft Systems Methodology (SSM) används som referensram (Vidgen et al, 2000).

I vår studie framgick även att systemutvecklarna själva till stor del måste avgöra om kvalitet är uppnådd. Att själv objektivt avgöra kvalitet på sitt eget arbete är mycket

svårt. Det är inte säkert att två systemutvecklare ger samma svar på frågan om en viss applikation till fullo möter kvalitetskraven eller ej. Ett enkelt och användbart sätt att utvärdera en applikations kvalitet är att helt enkelt be om någon annan erfaren systemutvecklares åsikter. Att läsa och kritisera andras program är ett effektivt sätt att förbättra kvaliteten, samtidigt som det för systemutvecklare är bra för den egna kompetensen att se hur andra utvecklar program.

Om möjlighet finns, så är blind utvärdering ett mycket bra tillvägagångssätt. Blind utvärdering går till så att den som utvärderar en applikation inte vet vem som har byggt den. På så sätt så elimineras risken att det är applikationens skapare som utvärderas istället för själva applikationen (Dahlbom & Mathiassen, 1995).

7.2. Repeatable Requirements Management

Ett nödvändigt element i kampen för kvalitet är en väl definierad och specificerad kravhantering. Krav som fattas, är inkompletta eller felaktiga leder till fel i applikationen. Den otillräckliga kravspecifikationen förhindrar även vanligtvis att felen upptäcks vid testningen av applikationen, eftersom testen också baseras på kravspecifikationen, ett krav som fattas kan ju inte heller upptäckas vid test. Vid en analys av felen i ett telefonväxelsystem, visade det sig att de fel som fick de mest allvarliga konsekvenserna, hade sin grund i en otillräcklig kravspecifikation och var därmed svåra att lösa (Hecht & Hecht, 2000).

Av detta kan vi se att en kravspecifikation måste vara mycket väl genomtänkt och strukturerad för att ge systemutvecklaren en så bra grund att utgå från som möjligt. Vid vår studerade avdelning kunde kraven vara mer eller mindre väldokumenterade och strukturerade. Personalen var dock mycket medvetna om problemet. Av våra intervjuer framgick att en översyn av dagens kravspecifikationsmall önskades, samt att en tydlig kravspecifikation var önskvärd. Detta inte bara för att ge en god utgångspunkt för utvecklingsarbetet, utan även för att systemutvecklaren alltid skulle kunna luta sig mot det som stod i kravspecifikationen.

En annan sak som framkom vid våra intervjuer var att systemutvecklarna ville ha mera direktkontakt med kund eller användare. På så sätt skulle feltolkningar och ovissheter i kravspecifikationen kunna lösas på ett säkrare och smidigare sätt. Samma tankegångar ger Catherine Connor och Leonard Cajello (2002) uttryck för. De skriver:

> *"Remember, it is critical that the entire project team have a complete understanding of the requirements. In order to achieve this type of quality, the developer must be involved early on to help clarify and drive out any ambiguity that may be inherent in the first version of the requirements"*
> (sid. 2)

Vidare skriver de om vikten av att ha klara regler vid de förändringar av kraven som uppstår. För uppstår gör de. Oavsett hur mycket arbete som läggs på att göra en så bra kravspecifikation som möjligt, så blir den aldrig perfekt (Wiegers, 2002b).

7.3. Software Project Planning

Här tar vi upp både *Software Project Planning* och *Software Project Tracking and Oversight* i samma stycke. Anledningen är att de är starkt korrelerade till varandra. Projektledning innebär styrning, planering och samordning av ett projekt. Vid planering av systemutvecklingsprojekt skall uppskattningar, såsom uppskattad tid för genomförande av projektet och tid för att uppnå eventuella delmål, samt de aktiviteter och åtaganden som ingår i projektet skall vara dokumenterade. Detta för att det skall vara möjligt att ha översikt och följa projektets gång, till exempel se om tidsplanen håller. Om så inte är fallet kan man då vidta lämpliga åtgärder, som att skjuta till ytterligare resurser eller organisera omfördela resurserna på ett tidigt stadium. Dessutom skall eventuella önskemål om förändringar av åtaganden vara godkänd av alla berörda parter innan de godkänns (CMM Online,2002).

Avdelningen vi studerade hade en projektmall man använde vid planeringen. Denna innehöll bland annat avtal, pris, antal timmar och budget. Uppdragen var indelade i olika storlekar, man skilde på uppdrag under 40 timmar och över 40 timmar och uppdrag där man fungerade som underkonsult. Uppdrag över 40 timmar hade mer dokumentation än de under 40 timmar, till exempel riskanalyser. Tidsplanen var även uppdelad på olika aktiviteter. Vid överskridande av tidsbudget lades denna tid som en icke-debiterbar post i uppdraget, men den överskridna tiden var inte uppdelad på de olika aktiviteterna. Man kunde därför inte senare ta fram vilken aktivitet som hade tagit längre tid än beräknat, bara hur mycket uppdraget som helhet hade överskridit tiden med. För att kunna följa uppdragets gång vid större uppdrag, hade man tagit fram en checklista där de olika aktiviteterna var dokumenterade. Där bockade systemutvecklarna av allt eftersom aktiviteterna var färdiga. Denna checklista var sedan tillgänglig för alla berörda inom projektet.

Vid planeringen av uppdrag var även systemutvecklarna inblandade, de stod till exempel för tidsuppskattningen. Att de som skall utveckla systemet även får vara med vid projektplaneringen är viktigt av framförallt två orsaker. För det första så blir bedömningar gjorda med större ackuratess, eftersom de som skall utföra uppgiften har större kunskap om den. För det andra så gör deltagande vid planeringen att systemutvecklaren blir mer motiverad att hålla planen, de flesta människor vill ju gärna leva upp till vad de själva har lovat (Heerkens, 2003).

I vår studie framkom att förändrade krav inte alltid dokumenterades. Detta gällde framförallt när förändringen var liten. Var den stor så dokumenterades den i form av en ny version av kravspecifikationen. Påverkade den dessutom priset så skrevs även avtalet om.

Enligt Connor & Cajello (2002) är det viktigt förändringar som överenskommits dokumenteras för att de skall kunna spåras. För att uppdragsledaren skall kunna ha överblick över uppdragets utveckling, måste han ju veta vad som ändras från den ursprungliga kravspecifikationen som ju planeringen bygger på.

Arbetet under själva produktutvecklingsprocessen hade en mycket liten styrning från uppdragsledaren. Systemutvecklarna styrde till stor del arbetet själva. Rollfördelningen var inte styrd, utan hade tillkommit med tiden genom systemutvecklarna själva.

Den lilla styrningen som framkom i vår studie är dock inte någonting som behöver vara fel.
Varje projektgrupp har en viss mognadsnivå, och den studerade systemutvecklingsgruppen hade en stor mognadsnivå. Vid val av ledarstil måste projektledaren ta hänsyn till denna mognadsnivå. Om projektgruppen har hög kompetens och hög grad av engagemang och intresse, kan ledarstilen vara delegerande. Därmed menas att projektmedlemmarna kan arbeta mer självständigt och ta större ansvar, och det personalorienterade ledarskapet kan tonas ned anser Wisén & Lindholm (2001). Vid förändringar av personalstyrkan, om någon slutar på företaget eller vid nyanställningar, måste dock en hårdare styrning till så att exempelvis rollfördelningen i systemutvecklingsgruppen klargörs redan vid början av ett uppdrag.

7.4. Software Configuration management

Software Configuration Management (SCM) är en viktig del av systemutveckling med målet att dokumentera och kontrollera förändringar hos ett system och dess ingående delar. Förändringar kan komma från både externt och internt håll. Externa förändringar kan komma från exempelvis användare/kund eller från förändrad teknologi. Interna förändringar kan komma från förbättrade metoder eller design, från incrementell utveckling och från rättning av fel. Förändringar under systemutveckling är vanligtvis en signifikant orsak till att ett projekts schema och budget förändras. Okontrollerad hantering av förändringar är troligtvis den största enskilda orsaken till att ett system inte kan levereras i tid eller inom budgetramen skriver Hyatt (2001).

Det är även viktigt att det förändrade eller tillagda kravet blir bedömt utifrån hur det påverkar systemets kvalitet och funktion innan det godkänns. Kravet måste även bedömas utifrån hur det kommer att påverka projektplan och budget. En kravspecifikation bör därför endast förändras om kund, systemutvecklingsteam och testteam har godkänt den. Förändringen skall även dokumenteras för att kunna spåra de förändringar som överenskommits (Connor & Cajello, 2002).

SCM och kravhantering är starkt relaterade till varandra. I vår studie fann vi att kraven alltid dokumenterades på något sätt, de var dock mer eller mindre specificerade. Med tanke på SCM är detta inte fullt tillfredsställande, eftersom en förändring måste kunna utvärderas med kravspecifikationen som utgångspunkt. Om kravspecifikationen är alltför allmänt skriven blir det dessutom svårt att avgöra om ett önskemål från kunden är att bedöma som en förändring eller ej. Studien visade också att förändrade krav inte alltid dokumenterades, vilket även detta gör dem svåra att utvärdera samtidigt som de blir svåra att spåra.

En annan viktig del i SCM är versionshantering. I nuläget har inte avdelningen något versionshanteringssystem, men det har diskuterats att skaffa ett.

8. Förslag till SPI-åtgärder

Vi har nu analyserat hur systemutvecklingsprocessen idag förhåller sig till CMM:s nyckelområden. Resultatet av analysen visar att företagets studerade avdelning idag befinner sig på CMM-skalans nivå ett. De delar vi anser som mest avvikande från CMM:s mål är hantering av krav, hantering av ändrade krav och otillräckligt dokumenterad och standardiserad kvalitetssäkring. Dessa avvikelser är dock inte specifika för det företag där vår fallstudie ägde rum, de finns troligtvis i många systemutvecklingsföretag av den storlek vi studerade.

I resultatet skall vi därför diskutera några punkter som är viktiga när ett företag som sysslar med småskalig systemutveckling ska införa SPI. Punkterna är skräddarsydda för små organisationer och kräver därför inte allt för stora resurser. Vi försöker även belysa de problemen som kan uppstå, eller tror finns, utifrån erfarenheterna från fallstudien och de forskningsrapporter vi tagit del av.

8.1. Kvalitet

För att ett företag ska överleva i längden krävs att det levererar produkter med kvalitet. För ett litet företag är det ofta extra viktigt, eftersom ett misslyckat projekt kan leda till ekonomisk kris. Att få ryktet om sig att inte nå upp till kundens förväntade resultat kan få förödande effekter. Ofta är de mindre företagens resurser mycket begränsade och att rätta till fel i program är kostsamt. Om företaget har fler verksamhetsgrenar är det viktigt att kvalitetspolicyn inte är för generell, eftersom den då kan upplevas svår att översätta till systemutvecklingsuppdragen. Risken med en alltför generell policy är att det tenderar bli subjektiv. Vi hävdar därför att systemutveckling måste ha egna dokumenterade kvalitetsregler speciellt anpassade till systemutveckling. På så sätt vet alla vilka direktiv som gäller vid avgörandet om god kvalitet har uppnåtts eller ej, samtidigt som alla följer samma definition av kvalitet.
Vid utvärdering av kvalitet föreslår vi att SSM:s fem E: *Efficacy, Efficiency, Elegance, Effectiveness* och *Ethicality,* används.

Att systemutvecklare i allmänhet bör vara fokuserade på att skapa hög kvalitet på sina produkter anser vi vara en självklarhet. Vår fallstudie visade att de system som företaget levererade hade genomgått omfattande tester av systemutvecklarna, och dessa tester medförde att kunderna var nöjda. Detta förhållande tror vi inte är ovanligt vid småskalig systemutveckling generellt. Det kan då kanske tyckas vara onödigt med en kvalitetshandbok för systemutveckling, men om det sker en förändring av personalen, som att någon slutar på företaget och en ny medarbetare kommer in, så blir det svårt att definiera begreppet ”nöjd kund” för denne. Med en kvalitetshandbok blir kvalitetsbegreppet lättare att kommunicera till den nye medarbetaren.

För ett systemutvecklingsföretag som idag lyckas upprätthålla hög kvalitet på sina produkter, även om systemutvecklaren själv måste utvärdera kvaliteten på sitt arbete, kan det tyckas vara överkursarbete att införa en oberoende kvalitetsbedömning. Men återigen, vid personalförändringar kan en egen kvalitetsbedömning bli ödesdiger, för den bygger till stor del på den kunskap, självinsikt och erfarenhet medarbetaren har, och dessa färdigheter skiftar från person till person. Utöver vid personalförändringar är en oberoende kvalitetsbedömning nödvändig vid systemutvecklingsuppdrag där systemutvecklarna måste gå utanför sitt huvudsakliga kompetensområde. Så även om inte en oberoende kvalitetsbedömning är direkt avgörande för dagen, så är det någonting som både små och stora företag bör planera införa.

8.2. Senior management

En grundförutsättning för att CMM skall lyckas, är att de anställda inom organisationen verkligen vill ha en förändring till stånd. Paulk (1998) uttrycker det med orden:

> ”*If the employees of an organization are satisfied with the status quo, there is little that the CMM can provide that will lead to true change; change occurs only when there is sufficient dissatisfaction with the status quo that managers and staff are willing to do things differently”*

Organisationens avsikt med att införa CMM måste alltså vara en uppriktig önskan att förbättra processen, och inte bara för att det råkar vara ”inne” just då.

Trots att Paulk tar upp vikten av de anställdas förändringsvilja så har han inte med detta som en viktig punkt att beakta vid införande av CMM. Detta anser vi göra hans tiopunktlista ofullständig. Det går inte att förutsätta att förändringsvilja finns, utan den första och viktigaste punkten borde vara att propagera för förändring bland de anställda. De anställda måste bli medvetna om att det finns problem i nuvarande arbetsprocess och att det krävs förändring för att kunna åtgärda problemet. Om inte denna punkt tas med i beräkningarna vid införande av CMM blir resten av Paulks punkter mycket svåra att genomföra.

I vår fallstudie framgick det att denna vilja till förändring redan fanns på företaget, men högre upp i organisationen blev det stopp. De flesta forskare inom området hävdar att det är personalen som ogillar förändringar, men i det här fallet var det tvärtom

Men för att SPI ska komma igång krävs att det finns engagemang hos företagsledning också, och inte bara hos systemutvecklarna. Uteblivet SPI-engagemang hos företagsledningen har ofta gjort att aktiviteter för systemutvecklare inte kunnat slutföras.(Abrahamsson och Jokela,2000)
Respondenterna i våra intervjuer ville alla ha en förändring av deras systemutvecklingsprocess. De ville ha riktlinjer som passade systemutvecklingen men förändringen måste komma uppifrån företagsledningen. Dessutom är det

meningslöst att skapa en modell bara för den lilla avdelningen, eftersom systemutvecklarna där ibland samarbetar med andra avdelningar.

När en organisationshierarki betraktas så finns mycket mer än den formella maktstrukturen återgiven. Desto högre upp i hierarkin en befattningshavare befinner sig desto mer makt besitter denne. Organisationshierarkin är oftast också en karriärstege. Det medför att förändringsidéer ofta utvärderas utifrån vilka konsekvenser de har för sin egen karriär, status och makt i organisationen. Självintresset är för det mesta minst lika starkt som organisationsintresset vid bedömningar av förändringsinitiativ. Detta medför att personer som ser sin formella makt hotad motarbetar förändringar. Som motsats kan då nämnas att personer som ser sin makt stärkas vid förändring ger förändringen sitt stöd.
Förändringsvilja räcker inte för att förändringar ska genomföras.

”They watch your feet, not your lips”
-Dr. Tom Peters (Systems Engineering Process Office,2003)

Det måste även finnas makt att genomdriva den. Om makt saknas för att genomdriva förändringar blockeras också förändringsambitionerna till slut. Det kan medföra missnöje och frustration bland medarbetarna och en vilja att genomföra en revolution för att få till stånd en förändring. (Bruzelius & Skärvad,2000)

“*It is not enough that top management commits itself for life to quality and productivity. They must know what is that they are committed to- that is, what they must do. These obligations cannot be delegated.*

Support is not enough. Action is required”
-Dr. W. Edwards Deming(*Systems Engineering Process Office*,2003)

Systemutvecklingen uppfattas av många som en relativt ny affärsgren. Företag som tidigare varit inriktade mot en annan mer mogen bransch har insett nya affärsmöjligheter i att införa systemutveckling. Detta för att få inriktningarna att komplettera varandra för att möta kundernas behov. För att förenkla processen med att införa en ny affärsgren, implementeras den nya arbetsorganisationen som en kopia av den som fanns tidigare i företaget, och används i den nya systemutvecklingsorganisationen. Många av de värderingar och grundläggande antaganden lever då kvar från den väl inarbetade och mogna branschen, vilka kan påstås vara eller utgöra företagets kultur. Företag som utfört liknande förfaringssätt bör vara på sin vakt, väl fungerande systemutveckling kräver en genomtänkt strategi för att arbetet ska fungera optimalt. Vi anser att det är bättre att leda förändringsarbetet mot ett förutbestämt mål än att det sker okontrollerat underifrån. Ett starkt ledningsinitiativ måste till för att skapa de incitament som krävs för systemutveckling, i synnerhet om företaget har flera inriktningar mot andra branscher.

Bruzzelius & Skärvad (2000) hävdar att en företagskultur kan vara svår att förändra och utveckla. Kulturen blir mer inrotad desto äldre ett företag är.

.

8.3. As is

För att en modell skall vara tillämpbar och göra nytta för individer i en organisation och organisationen som helhet, måste den anpassas. Modeller kan anpassas efter organisationen eller organisationen efter modellen. Möjligt är också att försöka kompromissa mellan modell och organisation och utföra ömsesidiga anpassningar. Hur än modellinförandet ska gå tillväga för att göra en modell tillgänglig och praktiskt användbar, har man ett omfattande arbete framför sig.

Paulk (1998) skriver att förbättringsarbete skall utföras evolutionärt och i små steg. Därför skall man börja själva förbättringsarbetet med ”as is”, hur systemutvecklingsprocessen ser ut idag, och inte med hur den borde vara. På detta sätt blir förändringen lättare att acceptera av de anställda. Systemutvecklingsprocessen på ett företag kan verka väl fungerande för de inblandade. Om processen granskas med hjälp av någon modell utifrån så uppvisar den i regel brister, men om personal och ledning skaffar sig självinsikt om att arbetet bör förändras och hur det kan förändras. Denna självinsikt bör utnyttjas när det gäller att definiera nuvarande arbetsprocess. Processer som definieras av användarna själva har mycket större chans att accepteras av användarna. De anställdas självinsikt om brister i systemutvecklingsprocessen bör även utnyttjas vid anpassningen av CMM-modellen. Lösningen på många av de problem som finns inom systemutvecklingsprocessen finns redan inom organisationen, det gäller bara att identifiera lösningen och anpassa den för vidare användning (Jalote, 2002). Ett bra sätt att hitta lösningen på några av de problem som finns idag är därför att ta upp systemutvecklarnas egna förslag till förbättringar.

8.4. Software Engineering Process Group (SPEG)

Inom småskalig systemutveckling genomförs projekt oftast i samverkan mellan medarbetare, det är därför viktigt att starta förbättringsarbetet i dessa konstellationer. En SEPG (Software Engineering Process Group) kan då tillsättas för att leda och koordinera förbättringsarbetet (Paulk,1998). Att leda arbetet med att införa en systemutvecklingsmodell är komplicerat och kräver erfarenhet och ett omfattande modellkunnande, så SEPG-gruppen bör därför ledas av en konsult som besitter dessa egenskaper. Det är även viktigt att rätt personer väljs till gruppen. De utvalda måste vara medvetna om att de delar ansvar med hela systemutvecklingsprocessen. Gruppen skulle kunna bestå av olika representanter från organisationen, såväl systemutvecklare som chefer. Modellinförandet bör starta utifrån ett nytt projekt. Att först skriva rutiner och policy och sedan applicera detta på projekt som är i gång fungerar sällan, dessa måste växa fram under projektets gång och var och en i gruppen måste förstå vikten av att ha bra rutiner, som också är

användbara i framtida liknande projekt. Gruppen bör uppstartas utifrån ett pilotprojekt i lagom storlek som drivs i SEPG-gruppens regi. Bäst resultat skulle nås om gruppen bestod av anställda som har erfarenhet från modellstyrd systemutveckling, och utformas som en paraplyorganisation där gruppmedlemmarna kunde ha var sitt ansvarsområde. Exempelvis kunde en vara ansvarig för att förbättra kvalitetshantering, en annan för projektstyrning och så vidare Därefter kunde det utverkas en plan över hur systemutvecklingen skall genomföras i framtiden.

Tillsättandet av en sådan här grupp bör inte bli så ekonomiskt betungande för en liten organisation, eftersom förbättringsarbetet löper parallellt med projektet. I en sådan grupp skulle finnas möjligheter att exempelvis ta fram en gemensam variabelstandard för kodningen samt att skapa en gemensam kodbank där fungerade lösningar skulle finnas sparade på ett strukturerat sätt.
Men återigen, det yttersta ansvaret ligger dock på företagsledaren att styra förbättringsarbetet och ställa nödvändiga resurser till förfogande

9. Diskussion

I diskussionen vill nämna några av de egna reflektionerna vi gjorde under arbetets gång

Ett argument som anförts mot CMM:s filosofi är att all kontroll och struktur dödar den kreativitet som krävs för att utforma och ett programvarusystem. Watts Humphrey bemöter detta genom att påpeka att ett strukturerat arbetssätt gör det möjligt att använda sin kreativitet på rätt sak; att hitta den bästa design som uppfyller de krav som ställs. Genom att skriva ner de uppgifter som vi vet är gemensamma mellan alla utvecklingsprojekt har vi dem ”ur vägen” och kan koncentrera oss på kärnan i den aktuella uppgiften. Filosofin liknar den som Thomas Alva Edison gav prov på när han sa att ”Uppfinningar är 99% perspiration och1%inspiration”.

CMM har heller inga konkreta råd att ge om hur systemutvecklingsarbetet ska utföras, det enda som finns är generella riktlinjer och mål. CMM är en komplex modell att förstå sig på inlärningen tas bit för bit. Modellen är dynamisk så den går att införa i ett nytt företag eller ett som har varit igång några år.

I vår fallstudie kom vi också i kontakt med modellstyrd systemutveckling. På den avdelningen sköttes allt enligt de regler och föreskrifter som modellen krävde. Vi noterade att arbetet kunde liknas vid ett fabriksjobb enligt Taylors principer. De anställda hade var och en sin egen väldokumenterade bit av helheten att sköta. Modellstyrt arbete innehar en styrka genom att det sällan förekommer några obehagliga överraskningar, alla ändringar och vem som gjort dem i systemet finns noga dokumenterade från kravspecifikationen till kod. En risk med att förvalta ett sådant system kan vara att de som äger systemet kan med lite framförhållning flytta systemet till en annan förvaltare. Detta är möjligt genom att det inte behövs så

mycket helhetsutbildning om systemet. Varje person behöver bara kunna en liten del av helheten.
Vi skulle även vilja kommentera Paulks artikel där det inte nämns mycket om underställd personals inställning till förändringsarbete. Paulk kommenterar ledningens roll i processen men nämner bara lite om att förändringen ska ske evolutionärt och i små steg för att möta mindre motstånd hos personalen. Paulk har rätt i att det ska göras stegvis och att förändringarna ska hinna bli rutiner innan nästa förbättring införs. Vi tror dock att personalen måste informeras ordentligt om vilka fördelar företaget får med hjälp av arbetsmodeller innan förändringen sker. Det de anställda måste inse är att metoden ska vara lösningen till att hitta en genväg till kundens behov.
De måste inse att de inte behöver uppfinna hjulet varje gång de får ett nytt projekt. Att införa modellstyrning auktoritärt eller i smyg tror vi inte på det måste ske i samförstånd.

10. Slutsats

I vår fallstudie undersökte vi hur systemutveclingsprocessen utfördes vid småskalig systemutveckling idag kontra CMM. Vi fann då att flera moment i systemutvecklingsprocessen inte nådde upp till CMM:s mål. Kravhantering och kvalitetsstyrning hade de allvarligaste bristerna.

Det är svårt att generalisera utifrån en enda fallstudie, men vi anser att de problem vi fann i fallstudien kan vara giltiga vid all småskalig systemutveckling som saknar modellstyrning. Vi anser därför att CMM eller någon annan modell skall införas för att förbättra systemutvecklingen, men den måste anpassas till de förutsättningar som finns i en liten organisation. Vid anpassandet bör de inblandade systemutvecklarnas erfarenheter och åsikter tas tillvara. Modellen skall införas i små steg och evolutionärt. Vid implementation av modellen bör Paulks tio punkter tas i beaktande. Detta för att snabbare få processförbättringar till stånd. Först och främst måste dock personal och ledning vara överens om att en förändring bör ske, därmed underlättas införandet av det nya sättet att arbeta och modellinförandet blir mindre kontroversiellt för de inblandade.

Referenslista

Abrahamsson, P. & Jokela,T. (2000). *Development of Management Commitment to software process improvement.* [online] http://iris23.htu.se/proceedings/pdf/85final.pdf [030923].

Bruzelius, L.H. & Skärvad, P-H. (2000). *Integrerad organisationslära* (8:e uppl.). Lund: Studentlitteratur.

Carlsson, I. (2002). *Förlorade vi vår religion.* [online] http://sesam.tranet.fmv.se/rendezvous/rv/nr3-02.pdf [030908].

CMM Online. (2002). *Goals by Key Process Area (listed by Maturity Level).* [online] http://www.teraquest.com/cmmOnline/Cross_Gs_Pas.html [030901].

Connor, C & Cajello, L. (2002). *Requirements Management Practices for Developers.* [online] http://www.therationaledge.com/content/jul_02/m_requirementsManagement_cc.jsp [030912].

Dahlbom, B. & Mathiassen, L. (1995). *Computers in Context.* Massachusets: Blackwell Publishers.

Goldkuhl, G. (1996). *Informatik, Ett ämne i, om och för förändring.* Institutionen för matematik och statistik vid HTU, (kompendium), Uddevalla.

Hecht, H. & Hecht, M. (2000). *How Reliable are Requirements for Reliable Software?* [online] http://www.dacs.dtic.mil/awareness/newsletters/stn3-4/howreliable.html [030929].

Heerkens, G. (2003). *Beware of Common Planning Failures*. [online] http://www.software-engineer.org/downloads/Beware_of_Gommon_Planning_Failures_oct01_heerkens.pdf [030929].

Holme, I. & Solvang, B. (1997). *Forskningsmetodik Omkvalitativa och kvantitativa metoder*
(2:a uppl.). Lund: Studentlitteratur.

Hyatt, L. (2001). *Software Configuration Management Guidebook.* 8online] http://satc.gsfc.nasa.gov/GuideBooks/cmpub.html [030926].

IT universitetets Hemsida. (2000). *Kvalitets modeller och standarder.* [online] http://www.dsv.su.se/courses/pvt/F2-PV2.pdf [030919].

Jalote, P. (2002). *Lessons Learned in Framework-Based Software Process Improvement.* [online] http://www.computer.org/proceedings/apsec/1850/18500261abs.htm [030910].

Johnson.D.L. & Brodman .J.G. (2000). *Applying CMM Project Planning Practices to Diverse Environments*. [online] http://www.computer.org/software/so2000/s4040abs.htm [030905].

Näsström, B., Haikara, S. & Larsson, S. (1999). *Systemutvecklingsmetoder vid fen IT-företag i Göteborg.* [online] http://www.handels.gu.se/epc/archive/00002219/01/larss.nasstrom.haik.ia5840.pdf [030917].

O'Hara, F. (2000). *European experiences with software process improvement*. [online] http://portal.acm.org/citation.cfm?doid=337180.337495 [030923].

Paulk, M.C. (1998). *Using the Software CMM in Small Organizations.* [online] http://www.sei.cmu.edu/activities/cmm/papers/cmm-small.pdf [030902].

Rejonen, P. (2000). *Software development and IS use.* [online] http://iris23.htu.se/proceedings/PDF/93final.PDF [030902].

Sei:s Hemsida. (2003a). *A breif history of the SEI.* [online] http://www.sei.cmu.edu/annual-report/about/history.htm [030909].

Sei:Hemsida. (2003b). *Capability Maturity Model® (SW-CMM®) for Software.* [online] http://www.sei.cmu.edu/cmm/cmm.sum.html [030909].

Sei:Hemsida. (2003c). *Capability Maturity Model® for Software (SW-CMM®).* [online] http://www.sei.cmu.edu/cmm/ [030909].

Systems Engineering Process Office. (2003). *Why SPI.* [online] http://sepo.spawar.navy.mil/sepo/index2.html [030918].

Trost, J. (1997). *Kvalitativa intervjuer* (2:a uppl.). Lund: Studentlitteratur.

Vidgen, R., Wood-Harper, T. & Wood, R. (2000). *A Soft Systems Approach to Information Systems Quality*. [online] http://iris.informatik.gu.se/sjis/vol5/vidgen.shtml [030829].

Wallén, G. (1996). *Vetenskapsteori och forskningsmetodik* (2:a uppl.). Lund: Studentlitteratur.

Wiegers, K. (2002a). *Karl Wiegers Describes 10 Requirements Traps to Avoid.* [online] http://www.processimpact.com/articles/reqtraps.html [030929].

Wiegers, K. (2002b). *Writing Quality Requirements.* [online] http://www.processimpact.com/articles/qualreqs.pdf [030929].

Wisén, J & Lindholm, B. (2001). *Effektivt Projektarbete* (upplaga 6:2). Stockholm: Norstedts Juridik.

Bilaga 1

Intervjumall

Problemställning: Hur utförs SPI i småföretag?

Allmänt

1. *Vad är din bakgrund och hur länge har du jobbat på företaget.?*
2. *Vad är dina arbetsuppgifter? Översiktligt, helhet, arbetsbeskrivning.*
3. *I dina arbetsuppgifter ingår programutveckling, hur ofta eller hur stor del av din tid disponerar du på det av din totala arbetstid?*

Arbetsplanering

Det kommer in ett nytt projekt, berätta om hur arbetet planeras.

Dessa frågor vill vi ha svar på i berättelsen. Det som inte kommer fram, eller vi inte uppfattar tydligt eller förstår, får vi flika in frågor om.

4. Programutveckling utförs det i grupp eller görs det enskilt, görs arbetet i delprojekt eller finns det någon standardprojektmall för arbetsgången i utvecklingsarbetet?
5. Får du eller gruppen träffa kunden eller får ni en arbetsrekvisition på vad som ska göras?
6. Skriver ni ner vilka krav kunden har, eller vem förvarar dessa dokument?
7. Finns det någon policy runt dessa kundkravsdokument?
8. Hur försäkrar du dig om att dessa krav överensstämmer med när du påbörjar arbetet?
9. Om kunden ändrar sig, vem rapporteras det till? Ändras kravspecifikationen då?
10. Hur hanteras risker? T ex om det blir så stor förändring så det påverkar ekonomi eller budget? Dvs sådant som kan äventyra projektet?
11. Vad gör du om kraven, kundens önskemål, inte kan uppfyllas?
12. Finns det några huvudmål att styra efter: ekonomi nöjda kunder?
13. Vem styr projekten? Vad innebär projektövervakning och –styrning för dig? Finns det några problem med styrning som är återkommande i projekten? Hur fungerar det med fortlöpande och aktuell överblick om ni jobbar i grupp?
14. Vet du exakt hur mycket som ska göras till en viss tid, (deadlines)? Vilken typ av information lägger du vikt vid för att få grepp om hur projektet följer plan?

Finns det något du skulle vilja ändra på när det gäller nuvarande arbetsplanering? Varför och hur?

Arbetsgången i utvecklingsarbetet!

Berätta hur arbetsgången är.

15.Om du inte kan lösa ett problem, vart vänder du dig då?
16. Sparas fungerande lösningar för återanvändning? I så fall, hur?
17. Vilken information om själva arbetsprocessen sammanställer du och till vem? Ge exempel!
18. Dokumenteras information om projektet, problem, tidsåtgång etc. Hur tas informationen tillvara, kunskapsbank? Används informationen vid planering av nya projekt?
19.Om kunden vill ändra sig när projektet kommit en bit på väg hur gör du då, berätta.
20. Skulle du vilja göra någon förändring angående vem som rapporterar vad? Mellan
de olika nivåerna eller deltagarna i arbetet? Berätta hur arbetet utförs idag?
Finns det någonting du anser problematiskt med den arbetsgång som gäller idag? Varför och hur?

Kvalitetssäkring!

Berätta hur kvalitétshanteringen fungerar idag?
21. Kvalitetmål, finns det någon policy eller standard?
22. Finns kvalitén med från början i projektet?
23. Hur gör du för att få en samlad bild av kvaliteten på ditt arbete, mått?
24. Klagomål från kund hur behandlas de. Hur utvärderas utvecklingsarbetet i efterhand??
25. Vem är ansvarig för kvalitén?
Finns det något inom dagesns kvalitétshantering du skulle vilja ändra på? Varför och hur?