Handelshögskolan vid Göteborgs Universitet
Institutionen för Informatik
Magisteruppsats vt 2002

Författare: Emma Skagerberg
Handledare: Håkan Enquist / Johan Magnusson

Att utföra systemtestning för – och med – kvalitet

INNEHÅLLSFÖRTECKNING

FIGURFÖRTECKNING

SAMMANFATTNING

Denna uppsats syftade till att utröna var i systemutvecklings- och testningsprocessen man borde göra förändringar för att uppnå en så hög kvalitet som möjligt på systemet och utfördes i samarbete med ett systemutvecklingsföretag. Syftet uppnåddes genom att undersöka felrapporter av fel som hittats såväl av testarna vid systemtestningen som av användarna efter att produkten släppts. Dessa fel delades in i olika kategorier för att ge bild av vilken typ av fel som var vanligast. I tillägg till detta utfördes även ett antal intervjuer för att hitta orsakerna till varför felen uppstod.

Undersökningen visade att de flesta felen dels orsakades av att systemet inte från början var byggt för att bli så stort som det blev, och dels av kombinationen låg grad av styrning och en oerfaren utvecklingsstab, vilket har medfört att många av felen har släppts igenom. Dessutom visade det sig att kommunikationen mellan de olika delarna av systemutvecklingsprocessen inte fungerade optimalt.

Uppsatsen resulterade i slutsatsen att det bästa sättet att säkerställa kvalitet är att ställa om blickfånget från systemtestningen till att tänka testning genom hela systemutvecklingsprocessen. Detta kan göras genom att utöka testarnas delaktighet i tidigare faser av processen och genom att införa manuella granskningar av tidig dokumentation och kod.

1 Inledning

1.1 Bakgrund

1.1.1 ”Quality is free”

1950 reste amerikanske ingenjören och statistikern W Edwards Deming till Japan för att där försöka implementera sina visioner om kvalitet. Med hjälp av moderna verktyg i form av statistiska hjälpmedel och processförbättrande styrning fick han japanerna att sakta men säkert kvalitetsmässigt förbättra sina produkter, och detta sätt att tänka fick genomslagskraft på de flesta av den japanska processindustrins områden. Världen förundrades över det japanska industriundret och hur de japanska produkterna, som höll en så pass hög kvalitet, kunde säljas till ett lika lågt pris som övriga produkter. Hela hemligheten ligger i vad Deming lärde japanerna, nämligen att *kvalitet är gratis*. Alla de pengar som läggs ut på ändringar i produkter och manualer och på telefonsamtal, och alla de pengar som förloras på att kunderna inte kommer tillbaka uppgår till en mycket högre summa än den som ett gediget kvalitetsarbete skulle kräva (Tognazzini, 1996).

1.1.2 Kvalitet och systemutveckling – igår och idag

Mjukvarubranschen är, jämfört med annan konstruktionsindustri, en ung bransch och har på senare år upplevt en explosionsartad utveckling. Möjligheterna för den nya tekniken har varit i princip obegränsade och kunderna har hittat fler och fler användningsområden för den. Det största konkurrensmedlet för systemutvecklarna har tidigare varit att få ut så mångsidiga produkter som möjligt till så många som möjligt på så kort tid som möjligt, vilket har bäddat för den snabba utvecklingstakten. Kvaliteten har dock inte hängt med i utvecklingen, men användarna har tidigare accepterat en viss felfrekvens, beroende på att de, trots felen, har haft stor nytta av systemen.

I och med att datorer för oss i västvärlden har blivit en självklar del av våra liv, och inte bara ett bra hjälpmedel i arbetet och i privatlivet, har också kraven på dem förändrats. Man har inte bara sett vilken nytta den nya tekniken kan göra utan även sett dess nackdelar i form av kraschade system och dataintrång. Många företag brottas idag med olika applikationsversioner, uppgraderingar, buggfixar och dålig mjukvarukvalitet. Detta ställer till mycket problem för dem, samtidigt som de inte har något annat val eftersom systemen har blivit fullständigt inkorporerade i företagsmiljön (Hulme, 2002). Den alltmer utsträckta användningen av Internet har också gjort att pålitlighet och säkerhet blivit frågor som står i centrum, och den allmänna tilliten till tekniken och dess säkerhet har minskat betydligt de senaste åren. Frågan är om vi nu inte har kommit upp på en platå i utvecklingen där det, precis som Deming talar om, lönar sig att låta kvaliteten prioriteras och existerande funktioner raffineras och säkras istället för att uppfinna nya funktioner.

I ett PM som skickades ut till Microsoft och dess dotterbolag i januari 2002 tog Bill Gates upp just denna problematik. Han myntar där uttrycket ”Trustworthy Computing”, vilket innebär att datorsystem bör bli lika pålitliga som elektricitet, vatten och telefon är idag. ”Trustworthy Computing” skall ha högsta prioritet inom Microsoft. Målet är att Microsoft skall leda utvecklingen mot att göra datorer och program pålitliga ur alla aspekter, och att deras kunder därigenom i framtiden ska få en annan och bättre bild av Microsoft som företag. Vidare talar han om funktion kontra säkerhet:

In the past, we've made our software and services more compelling for users by adding new features and functionality, and by making our platform richly extensible. We've done a terrific job at that, but all those great features won't matter unless customers trust our software. So now, when we face a choice between adding features and resolving security issues, we need to choose security. (Gates, 2002)

Det är inte bara Microsoft som försöker ändra sin inställning till kvalitet, att satsa på kvalitet ur olika aspekter är just nu en allmän trend inom systemutveckling, och denna trend välkomnas av många fackmän i branschen (Hulme, 2002).

1.2 Företaget

1.2.1 Idé

Grundidén för uppsatsen bestod i att utvärdera inrapporterade fel på programvara som funnits ute på marknaden en tid och utifrån denna utvärdering betrakta systemutvecklings- och testningsprocesser för det undersökta företaget. Denna undersökning utfördes därför i samarbete med ett existerande systemutvecklingsföretag. Vilket det aktuella företaget är har ingen egentlig betydelse för den här rapporten och därför har alla referenser till företaget gjorts anonyma genom att istället för företagets namn endast skriva "Företaget". All data baseras på ett tidigt projekt och ett antal förbättringar har förts in i flera delar av Företagets processer.

1.2.2 Samarbetet mellan Företaget och deras OEM-partners

Företagets affärsidé går ut på att leverera en produkt som är s.k. ”white-labeled”. Detta innebär i praktiken att de har ett antal samarbetspartners, så kallade OEM-partners (Original Equipment Manufacturer), som Företaget levererar sina produkter till. När produkten kommer till OEM-partnern bär den inte längre Företagets namn utan den är ommärkt så att det ser ut som om det är OEM-partnerns egen. Slutanvändarna är alltså kunder till OEM-partnern. För Företagets del innebär detta den fördelen att samma produkt kan säljas till fler företag, för även om de olika OEM-kunderna är konkurrerande företag kan de använda samma produkt i grunden utan att det syns utåt.

Figur 1: Företagets relation till slutkunder

Företagets OEM-partners erbjuder i sin tur företagets tjänster till sina kunder och sprider på så vis Företagets lösningar till över 100 länder och ett stort antal användare över hela världen.

1.3 Problemformulering

1.3.1 Syfte och frågeställning

Undersökningen har som syfte att hitta de brister i systemutvecklingsprocessen hos Företaget som leder till att fel som uppstår i programvaran kommer så långt ut som till användarna. Målet har varit att hitta en alternativ modell som förhindrar att problemen uppstår. Den alternativa modellen har framarbetats genom studier av teori kring testning och systemutveckling, och har sedan anpassats med hjälp av det som kommit fram genom övrig datainsamling till Företagets behov.

I enlighet med uppsatsens syfte har undersökning till uppgift att svara på följande huvudfråga:

> *Hur kan processerna på företaget omarbetas för att förbättra kvaliteten på deras produkter?*

Syftet kan vidare delas upp i tre mindre aktiviteter med tillhörande delfrågor:

- Att få en uppfattning om hur verkligheten ser ut på Företaget.
  Detta framkommer genom studier av företagets interna dokument och genom att studera hur systemutvecklingsprocessen och testningsprocessen fungerar idag.

  Hur ser processerna ut idag?

- Att ta reda på var de flesta problem uppstår

Detta görs genom att dela in de fel som uppkommit vid testning, s.k. issues, och supportfall tillhörande samma produkt, i olika kategorier.

Inom vilken felkategori hamnar de flesta issues resp. supportfall?

- Att ta reda på orsaken till varför problemen uppstår
  Orsakerna till varför problemen uppstår just i den aktuella kategorin kan hittas genom att intervjua berörda personer.

  Vad beror utfallet av undersökningen på?

I nästa kapitel kommer dessa metoder att förklaras närmare.

1.3.2 Definition av begreppet ”fel”

För att kunna föra en diskussion kring kvalitet krävs först en definition av vad ett fel egentligen är. Inom programvara kan man enligt Perry (2002) urskilja två huvudgrupper av fel:

1. Programmet uppför sig inte i enlighet med specifikationen
   Exempel på felorsaker (Sommerville, 2001):
   - Designspecifikationen uppfyller inte systemkraven
   - Designspecifikationen misstolkas och leder till fel i programspecifikationen
   - Programkoden överensstämmer inte med programspecifikationen
   - Datainmatningsfel
   - Avlusningsproblem, dvs. något fel som rättas till orsakar ett annat fel i programmet

2. Programmet uppför sig inte enligt användarens förväntningar
   Exempel på felorsaker (Sommerville, 2001):
   - Systemkraven från användarna misstolkas eller nedtecknas inte tillräckligt väl.
   - Användarna misslyckas med att specificera vad de vill ha, eller ändrar sig och kommer in med krav på produkten för sent.
   - Felaktigheter i dokumentation m.m. om produkten kan leda till att användaren förväntar sig något annat av programmet.

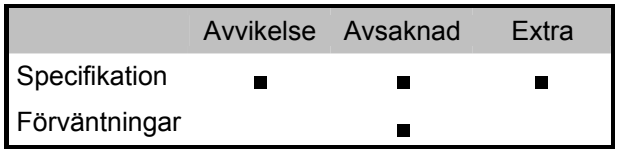
Vidare menar Perry (2002) att dessa huvudgrupper i sin tur kan delas in i tre undergrupper:

a) **Avvikelse** - Programmet innehåller alla de attribut som finns specificerade i specifikationen, men det uppför sig inte så som det är tänkt. Inom denna grupp hamnar de fel som vanligtvis kallas för buggar, både fel av mindre betydelse och fel som kan ställa till stor skada i och utanför systemet.
b) **Avsaknad** - Något attribut som finns i specifikationen saknas i programmet. Här finns alla de fel som beror på missförstånd mellan olika faser i utvecklingen, där

utvecklarna har missuppfattat specifikationen eller användarnas vision av systemet. Av denna anledning hamnar alla fel ur användarperspektivet (huvudgrupp 2) här.

c) **Extra** - Programmet innehåller något attribut som inte finns i specifikationen. Detta behöver inte alltid utgöra ett stort fel, men många fel av denna typ kan ställa till problem t.ex. om programmet blir mer komplicerat än vad som var tänkt.

Figur 2: Fördelning av feltyper beroende på perspektiv



För Företagets del finns det ett par problemområden som försvårar utvecklingsarbetet. För det första är distansen till användarna stor. Det är inte slutanvändaren som fungerar som beställare och som ställer krav, utan det är i detta fall OEM-kunderna som kommer med krav för nya produkter. Dessutom sker i princip all kommunikation med slutanvändarna genom OEM-kunderna. Dessa båda faktum gör att användarnas åsikter antingen inte hörs alls eller filtreras genom OEM-kunderna, vilket försvårar arbetet med att undvika fel ur ett användarperspektiv. För det andra skall Företagets produkter kunna integreras i ett stort antal olika systemkombinationer, vilket ställer höga krav på produkternas flexibilitet.

1.3.3 Avgränsning

I denna uppsats behandlas endast de fel som uppkommit vid testning, s.k. issues, och supportfall som kommit in gällande en specifik produktversion. Versionen som undersökts för denna uppsats är tillräckligt aktuell för att resultatet skall kunna appliceras på nuvarande produkter, men dock inte så pass ny att den fortfarande används, vilket gör att man hunnit få in alla supportfall som man kunnat hos produkten. Detta kan alltså ge en god bild av hela kedjan som utgör systemets livslängd, från start, genom utveckling och test, och till implementering och support.

1.4 Disposition

Kapitel 2 tar upp vilket metod som använts. Det bygger vidare på syftet och arbetsgången och innehåller mer detaljerad information om hur arbetet är upplagt.

Kapitel 3 behandlar bakomliggande teori för att ge läsaren en teoretisk grund att stå på. Här beskrivs först kortfattat vanliga test- och systemutvecklingsmodeller, testtekniker, och kvalitetssäkringsstandarder. Med den allmänna teorin som bakgrund visas sedan den

optimala modell för hur testning skall fungera i samband med systemutveckling som har utarbetats under teoristudierna.

I **kapitel 4** läggs resultatet av undersökningen fram. Det börjar med en presentation av den objektiva information som framkommit genom studier av företagsinterna dokument. Efter det tas resultatet av kategoriseringen av issues och supportfall upp, och slutligen presenteras en sammanfattning av utförda intervjuer.

Kapitel 5 innehåller analys av resultatet av datainsamlingen. Här jämförs den modell som utformats mot bakgrund av teorin med arbetssättet på Företaget.

Kapitel 6 presenterar slutsatserna och svarar på de här ovan ställda frågorna. Dessutom finns här också en diskussion om arbetet kring uppsatsen.

Slutligen visas en förteckning över de källor som använts för uppsatsen.

2 Metod

2.1 Datainsamling

2.1.1 Kvantitativ och kvalitativ metod

Det finns två olika sätt att samla in data, kvantitativ och kvalitativ faktainsamling. Det är sällan en undersökning kan karakteriseras som enbart kvalitativ eller kvantitativ, ofta behövs lite av båda för att man skall kunna få en helhetsbild av problemet. Åt vilken typ av metod undersökningen lutar beror sedan på vad det är man undersöker. (Patel & Davidsson,1994)

En kvantitativ undersökning kännetecknas främst av att man med hjälp av statistik samlar in och sorterar informationen, vilken man sedan på olika sätt kan mäta. Detta ger upphov till handfasta och otvetydiga resultat, vilka sedan relativt lätt kan analyseras. (Holme & Solvang, 1997)

Kvalitativ forskningsmetod använder sig inte av statistik utan sker genom verbala analysmetoder. Kvalitativ data kan sålunda inte mätas eller delas in i fack och är i större grad beroende på vem som utför undersökningen. Syftet med en kvalitativ undersökningsmetod är att skapa en djupare förståelse för det studerade objektet, inte att pröva om resultatet är generellt giltigt. Datainsamlingen kan bestå t.ex. av personliga intervjuer, vilket gör att kvalitativ data snabbt kan bli ganska omfattande. I och med detta begränsas undersökningen till få objekt, men den blir desto mer djupgående. (Wallén 1996)

Denna undersökning lutar åt forskning av kvalitativ karaktär. Detta för att en systemutvecklingsprocess är något som måste ses ur en subjektiv verklighetsbild, eftersom det som utgör själva processen är människorna som arbetar i den, och deras subjektiva uppfattning om arbetet som de utför. Själva datainsamlingen består av tre delar och är av såväl kvantitativ som kvalitativ karaktär. För det första består den av dokumentstudier, för det andra studier och kategorisering av supportfall och issues, och för det tredje ett antal intervjuer.

2.1.2 Dokument

För att kunna uppfylla huvudsyftet, *Att hitta en alternativ modell som förhindrar att problemen uppstår*, krävdes studier av bakomliggande teorier kring systemutveckling och testningsprocesser. De sekundärdata som användes för att få denna grundläggande teoribakgrund består av böcker, rapporter, artiklar och webbplatser som behandlar ämnet. Relevant litteratur hittades genom sökningar i Göteborgs Universitetsbiblioteks databas Gunda, relevanta ämnesdatabaser, och allmänna sökmotorer på Internet, med sökord som ”Software QA”, ”Testing Software”, ”Verification”, ”ISO” m.m.

För att skaffa erforderlig kunskap för att kunna analysera processerna inom den aktuella verksamheten studerades dokumentationen kring den aktuella produkten och företagets processer. Den information som framkom härigenom låg till grund för att förverkliga det första delsyftet, *Att få en uppfattning om hur verkligheten ser ut på Företaget*.

2.1.3 Kategorisering

Nästa delsyfte, *Att ta reda på var de flesta problem uppstår*, uppfylldes genom att samla in kvantitativ data i form av felrapporteringar från supportärenden och från så kallade issues. En förstudie av issues och supportfall lade grunden till uppkomsten av ett antal olika kategorier av fel. Issues och supprtfall tillhörande den aktuella produkten blev därefter indelade i dessa kategorier och grupperade enligt den mall för olika typer av fel som finns presenterade här ovan (sid. 9-10).

2.1.4 Intervjuer

Dokument och siffror stämmer inte alltid överens med hur det fungerar i verkligheten, och därför har i datainsamlingsarbetet för denna uppsats även intervjuer använts. Detta kan, förutom att visa om den upplevda verkligheten överensstämmer med den dokumenterade, ge en fingervisning om varför felen uppstår. En intervju kan föras på olika sätt beroende på vilken typ av data man vill ha fram. Intervjuns grad av standardisering beror på hur frågorna ställs, om man ställer samma frågor till alla respondenter, eller om man anpassar frågorna efter situationen, t.ex. i form av följdfrågor på det som sagts tidigare. Hur stort svarsutrymme intervjupersonen får brukar mätas i grad av strukturering. En fråga som man endast kan besvara ”ja” eller ”nej” är en fråga med en hög grad av strukturering, likaså frågor med fasta svarsalternativ (Gordon, 1971).

I denna rapport hade intervjuerna till uppgift att förverkliga delsyftet *Att ta reda på orsaken till varför problemen uppstår*. För detta användes halvstandardiserade personliga intervjuer av informell karaktär, eftersom möjligheten då finns att kunna ställa följdfrågor och föra diskussioner kring ämnet. Innan intervjuerna fastställdes fyra olika ämnen kring vilka olika frågor ställdes och diskussioner fördes. Dessa fyra ämnen var samma för alla intervjupersoner. Vidare intervjuades personer ur olika delar av processen och organisationen för att verksamhetens helhetsbild och för att synpunkter från alla håll skulle framkomma.

Ämnen som intervjun behandlade:

- Företagets existerande projektmodell
- Kommunikationen mellan olika delar av systemutvecklingsprocessen
- Styrningens vara eller icke vara
- Nyheter, hur de emottas av företagets anställda

Intervjuade personer:

Kvalitetsansvarig
Utvecklingschef
Projektledare
Supportchef
Testare

2.2 Vetenskapsteoretisk bakgrund

Olika forskare har olika sätt att betrakta verkligheten, olika sätt att arbeta på, och olika inställning till insamlad information. Här nedan beskrivs de synsätt och arbetssätt som använts för denna undersökning.

2.2.1 Grundläggande synsätt

Det finns enligt Wallén (1996) tre grundläggande synsätt inom dagens forskningsmetodik och de utgörs av positivism, systemteori och hermeneutik. Systemteorin delar upp verkligheten i olika system och försöker definiera systemets olika delar och delarnas inverkan på varandra och på systemet som helhet. Denna ansats uppstod ur ett behov av att följa, förstå och planera för växt och förändring i komplexa sammanhang där olika faktorer är beroende av varandra. Man utgår från att hela systemet är mer än bara summan av systemets delar. Systemteorin har, liksom positivismen, krav på mätbarhet, men här blir orsak-verkan-förklaringar alltför komplicerade eftersom det finns många samverkande orsaker. Viktigare här är växelverkan mellan systemets delar
Denna rapport ser på verkligheten utifrån ett systemteoretiskt perspektiv eftersom målet är att betrakta verksamhetens olika delar och deras inbördes relationer.

2.2.2 Metodansats

Enligt Söderfeldt (1972) sker det vetenskapliga framåtskridandet:

"genom ständig växling mellan induktion och deduktion. En person uppfattar gemensamma egenskaper hos förut osystematiserade företeelser i verkligheten och inordnar dem under ett begrepp. En frågeställning konstrueras. Genom operationalisering går man åter ut i verkligheten och försöker inbegripa ännu fler data i begreppet. Näste forskare tar vid och sammanställer resultaten från undersökningen av frågeställningen med andra kända resultat och relaterar det nya begreppet till andra begrepp i en hypotes. Genom deduktion från hypotesen får han fram nya

frågeställningar, som ger nya data, som bestyrker eller falsifierar hypotesen. Så småningom kan flera hypoteser uppställas, och deras gemensamma drag kan ligga till grund för induktion av ett teorem, som sammanfattar informationen i flera hypoteser. Återigen kan nya hypoteser och frågeställningar deduceras från teoremet och prövas mot data för att ligga till grund för fler hypoteser och fler teorem. Teoremen kombineras sedan i teorifragment. Hela tiden växlar man mellan induktion och deduktion, man rör sig uppåt eller nedåt i abstraktionsgrad. Det slutliga målet blir att kombinera teorifragmenten till den generella teorin."

Denna uppsats arbetsgång följer den induktiva metodansatsen. Här finns ingen på förhand uppställd datainsamlingslinje som grundar sig i en hypotes, utan datainsamlingen sker förutsättningslöst.

2.2.3 Undersökningsansats

Olika undersökningar har olika ingångar beroende på hur pass väl forskaren känner till bakgrunden till forskningsobjektet. En deskriptiv undersökning går ut på att samla in existerande information och systematisera den för att beskriva en speciell företeelse. Ofta koncentrerar man sig på att beskriva ett par mindre områden av det fenomen man studerar, men beskriva dem grundligt. Normativ undersökning liknar deskriptiv undersökning, men syftar inte bara till att beskriva ett fenomen som det är, utan har som mål att beskriva hur ett fenomen borde vara för att vara bra. (Wallén, 1996) Detta är undersökning av normativ karaktär. Målet är inte att beskriva verksamheten utan att försöka se var man kan göra förändringar och förbättringar för att optimera systemutvecklings- och testningsprocessen.

3 Teori

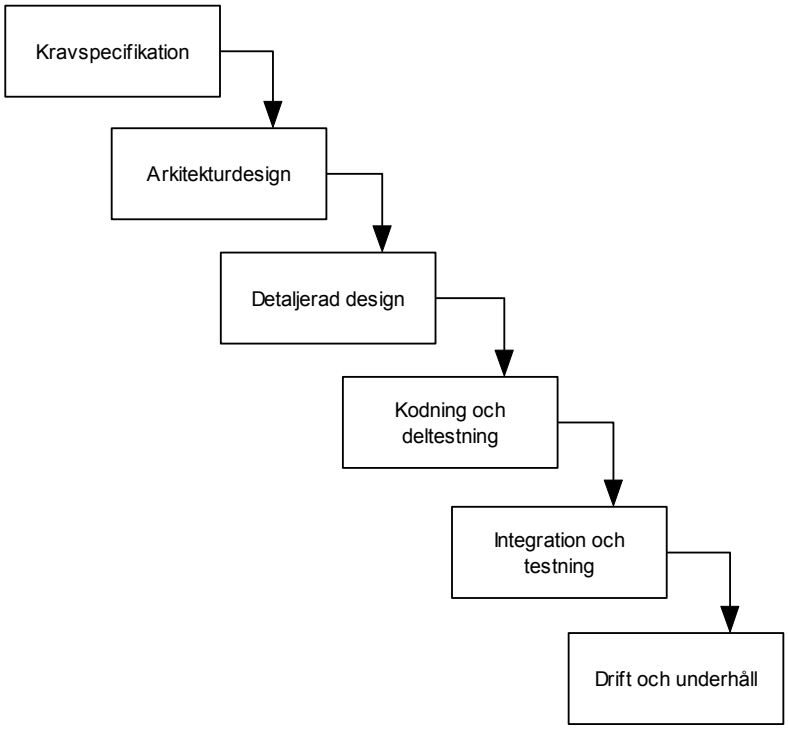
3.1 Systemutvecklingsprocessen

Skapandet av ett system är en process som innehåller många olika aktiviteter. För att få struktur över alla dessa aktiviteter finns det ett antal olika skolor som var och en har utvecklat en egen modell över processen. Dessa modeller fungerar som en abstrakt representation av processen. Här följer en kort beskrivning av ett par av de vanligast förekommande modellerna, baserade på Sommerville (2001), för att fungera som en referens.

3.1.1 Vattenfallsmodellen

Vattenfallsmodellen är en bild över ett systems hela livscykel. Vattenfallsmodellen börjar med kravspecifikationen som kortfattat innebär att man specificerar vad systemet skall göra och inte göra, vilken miljö det skall fungera i och vilka personer, system och annat som systemet kommer att interagera med. Baserat på detta kan man sedan gå vidare och mer specifikt kartlägga miljön runtomkring och inom systemet.

Figur 3: Vattenfallsmodellen



Detta är vad som kallas arkitekturell design. Efter det sker en mer utförlig beskrivning av systemet och dess funktioner, kallad detaljerad design, vilken ligger till grund för kodningen. Testningen sker sedan i (minst) två steg, dels i viss mån samtidigt som kodningen och dels när alla funktioner är implementerade och delsystemen kan integreras och systemet kan fungera som en helhet. Det sista steget visar den aktivitet som pågår så länge systemet är i bruk och som oftast tar mest tid och resurser i anspråk, drift och underhåll.

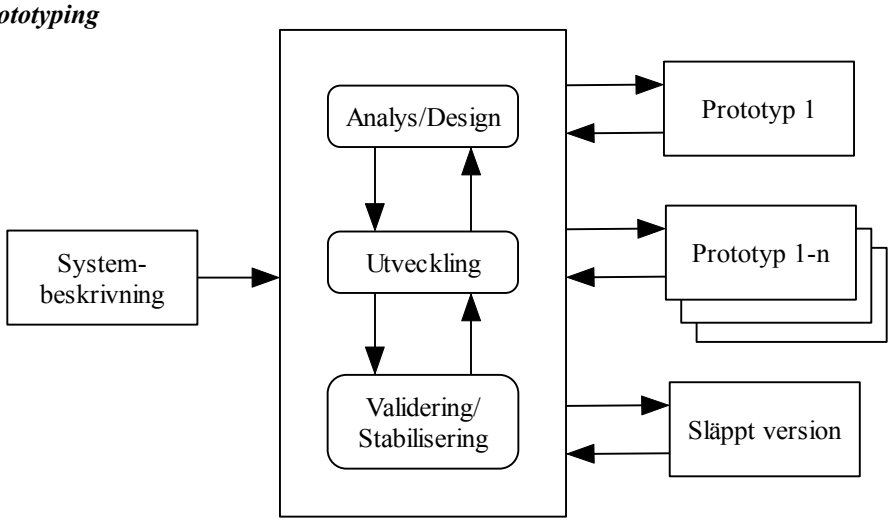
Vattenfallsmodellen har, ju längre utvecklingen har gått och ju mer kunden har blivit en del av systemutvecklingsprocessen, visat sig inte räcka till. Kraven från beställaren kan misstolkas och/eller vara vagt definierade, vilket innebär att de kan komma att ändras när man är mitt i processen. Om man då inte har möjlighet att gå tillbaka och ändra i kravspecifikationen kan det sluta med att beställaren i slutändan får ett system som den inte vill ha.

3.1.2 Iterativ systemutveckling

För att undvika vattenfallsmodellens tillkortakommanden har olika typer av iterativa modeller utvecklats. I stort sett innehåller de samma aktiviteter som i vattenfallsmodellen, men här finns en medvetenhet om att kraven på och designen av ett system av olika anledningar kan komma att ändras. Därför finns möjligheten att gå tillbaka och ändra om något skulle visa sig vara ”fel”. Hela systemet utvecklas inte på en gång utan det sker i etapper.

Prototyping är en typ av iterativ systemutveckling. Utvecklingen börjar med en enkel prototyp baserad på vagt formulerade krav man fått från beställaren.

Figur 4: Prototyping



När man fått klartecken om att den är OK fortsätter man att bygga på prototypen. Samma sak, dvs. ny prototyputveckling och godkännande, sker sedan i så många steg som krävs

för att systemet skall betraktas som klart, och när så är fallet levereras hela systemet till kunden. Inom prototyping är aktiviteterna specifikation, design, kodning och validering i princip samtidiga, vilket ofta har till följd att överblicken över processen blir svår att hantera. Dessutom kräver denna typ av systemutveckling en engagerad beställare som är beredd på att lägga ner tid på att sätta sig in i, testa och godkänna – eller rata - de olika prototyperna. Viktigt är också att prototyperna inte betraktas som färdiga delar av systemet och blir till rena versioner, de får endast ses som ett hjälpmedel för systemutvecklarna att bygga det äkta systemet på rätt sätt.

En liknande process är s.k. inkrementell systemutveckling. Där delas systemet upp i olika delar (inkrement) som utvecklas var för sig. När ett inkrement är klart levereras det som en färdig produkt till beställaren. Processen upprepas och de övriga inkrementen ansluts till det befintliga systemet ända tills systemet innehåller alla de funktioner som beställaren vill ha.

3.2 Testning och planering

3.2.1 Hur planera?

Testningen är den aktivitet som ser till så att inga fel finns. Meningen är ju att utvecklingsprocessen skall vara utformad så att inga fel finns (testkostnader innefattar inte bara kostnaden för att hitta felet i fråga utan också de resurser man använde för att bygga in felet i systemet (Perry, 2002)), men om det finns fel i programvaran skall testningsfasen hitta dessa (Marick, 1997). För att hitta alla fel måste allt testas i alla tänkbara situationer på alla möjliga sätt. Detta är för i princip alla systemutvecklingsprojekt en omöjlighet eftersom det finns ett oändligt antal kombinationer av skeenden. Därför måste en avgränsning göras och problemet med planeringen är hur man gör denna avgränsning.

Om vi först ser på tidsaspekten, dvs hur länge testningen skall hålla på innan den kan anses vara klar, är detta först och främst en resursfråga. Många använder tekniken ”testa så långt pengarna räcker”, vilket kan ställa till problem eftersom det varken är bra att undertesta eller övertesta. Antingen hittas för få fel för att systemet skall kunna hålla den kvalitet som var tänkt, eller så lägger man ut en massa tid och pengar i onödan på att testa för grundligt. Det finns en optimal testpunkt, dvs. hur mycket som är lagom att testa, som kan hittas genom att analysera kringliggande faktorer. Detta innefattar t.ex. faktorer som projektets resurser, systemets kritikalitet, dess syfte och användarnas förväntningar på systemet.

Den andra aspekten gäller vilka delar av systemet som skall testas mest.
Riskstyrd testning är ett sätt att så effektivt som möjligt hitta så många fel som möjligt. En riskuppskattning görs då, vilken ger en fingervisning om var testningen behövs mest. Kortfattat innebär det att de delar av systemet som är mest kritiska och / eller de delar som antas innehålla flest fel testas i större utsträckning än andra delar.

Statistisk testning är en annan ansats där det till skillnad från ovan nämnda ansats inte är risk som styr, utan man testar utifrån hur systemet kommer att användas på marknaden. Både vilka testfall som skall göras och vilka delar av systemet som skall testas genereras slumpmässigt. Man skapar en testgrupp som fungerar som en representation av verkligheten och med hjälp av resultatet av testningen kan slutsatser dras om hur pass pålitligt systemet kommer att vara. Vad som kommer i skymundan med denna ansats är undantagsförhållanden.

Den tredje aspekten gäller vid vilken fas testningen bör ske. I allmänhet gäller att ju tidigare ett fel upptäcks, desto lättare och billigare är det att rätta till felet. Dock uppstår inte vissa fel förrän efter integration av olika delar, vilket gör att testning i slutet av utvecklingsprocessen alltid måste ske.

Slutligen kräver en planering av test också att vilka testtekniker som skall används fastställs.

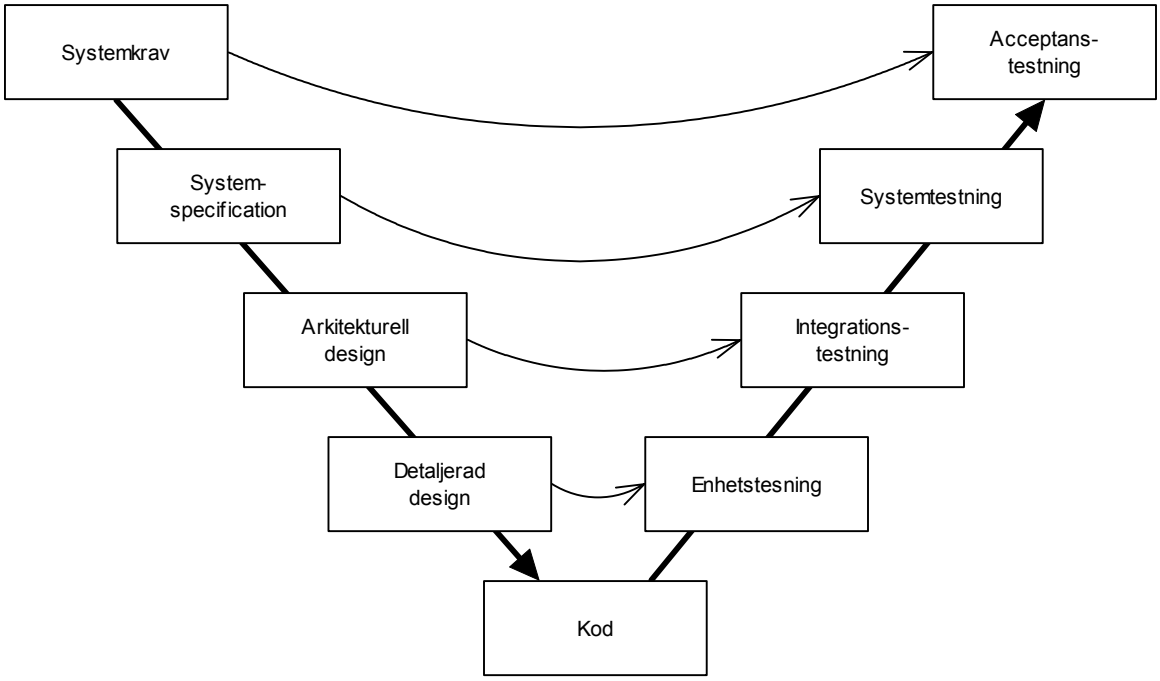
3.2.2 Validering och verifiering, vad är det?

Validering är den aktivitet som man utför för att se så att systemet har alla de funktioner som användaren behöver, med andra ord frågan om man har byggt rätt system. Verifiering däremot, inriktar sig bara på de funktioner som finns i kravspecifikationen, och med utgångspunkt ifrån vad det är specificerat att dessa funktioner skall göra testas de. Frågan handlar alltså istället om man har byggt systemet på rätt sätt (Oskarsson, 1995).

3.2.3 V-modellen

Denna typ av modell, anpassad till respektive organisations egen projektprocess, är den som för närvarande används mest som modell för testning och validering inom systemutvecklingsbranschen. V-modellen bygger på delar av vattenfallsmodellen och delar upp systemutvecklingsfaserna i kravspecifikation, systemspecifikation, arkitekturell design, detaljerad design och kodning. För varje steg i modellen skapas en testplan och testfall som bygger på processfasens dokumentation. När man utformar en plan för t.ex. användartestning använder man kravspecifikationen, medan man vid utformning av integrationstestfallen har större användning för dokumentation kring den arkitekturella designen. Testerna utförs sedan för varje steg i omvänd ordning. Varje enhet testas först separat innan den sätts ihop med övriga enheter för integrationstestningen. Sedan fortsätter man med systemtestningen som är till för att se till så att de funktioner som finns med i systemspecifikationen finns där och fungerar som de skall (verifiering). Processen avslutas med att användarna får testa systemet och avgöra om det överensstämmer med vad de vill ha (validering/acceptanstest).

Figur 5: V-modellen



3.2.4 Cleanroom software engineering

Denna metod har utvecklats och förespråkas av Software Engineering Institute. Huvudsyftet med the Cleanroom process är att utveckla mjukvara som aldrig uppvisar några problem när den väl kommit ut till användarna. Den går ut på att med hjälp av en väl genomarbetad utvecklingsprocess förebygga fel istället för att hitta och ta bort fel. För att åstadkomma detta används följande tekniker (cleansoft):

- Formell specifikation – Utvecklingen är baserad på matematiska principer. En särskild metod används för design och specifikation. Dessutom används funktionell verifiering, ett slags matematiskt bevis (se nedan), för att bekräfta att designen är en riktig implementation av specifikationen.
- Inkrementell systemutveckling (se ovan)– Inkrementen skall specificeras tidigt.
- Strukturerad programmering – Man gör en stegvis förfining av specifikationen för att producera kod.
- Software Inspection (se nedan) – Denna teknik har helt ersatt enhetstestning och modultestning.
- Statistisk testning (se ovan) – Testningen skall endast utföras av användare eller en representativ användargrupp.

3.3 Testning – olika tekniker

Man brukar dela upp de olika testteknikerna i två stora grupper: statiska och dynamiska tekniker. Skillnaden mellan dessa två tekniker är att man vid dynamisk testning kör programmet i fråga, medan man vid statisk testning endast hanterar koden manuellt (Oskarsson & Glass, 1995, Perry, 2000).

3.3.1 Statiska tekniker

Software Inspection
Den första versionen av denna teknik skapades 1976. Efter det har flera olika tekniker som bygger på denna utvecklats, men de har alla samma grundidé: att med hjälp av en formell procedur hitta och identifiera brister och fel i mjukvaran på ett sådant sätt att samma typ av inspektion kan appliceras på alla delar av systemutvecklingsprocessen och på all slags dokumentation. Det finns även ett par andra typer av kod- och dokumentationsgenomgångar, s.k. walkthroughs och reviews, men som först och främst har ett annat syfte, nämligen det att utvärdera produkten. De är alltså inte rena felsökningstekniker och kan variera i djuplodning och är ofta mer informella än Software inspections. Dessa tre begrepp är förklarade i IEEE standarden 1028-1997. Enligt denna är

Software inspections:

”a visual examination of a software product to detect and identify software anomalies, including errors and deviations from standards and specifications”

walkthroughs:

“a static analysis technique in which a designer or programmer leads members of the development team and other interested parties through a software product, and the participants ask questions and make comments about possible error, violation of development standards, and other problems”

och reviews:

“a process or meeting during which a software product is presented to project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval” (Aurum, Petersson & Wohlin 2001).

Strukturell analys (Automated statical analysis)
Detta är en typ av verktyg som fungerar som ett komplement till kodkompilatorn och syftar till att hitta logiska fel. Dessa program letar efter oönskade kodproblem som odeklarerade variabler, deklarerade variabler som ej används, för djupa loopar, metodanrop med fel antal argument, motstridigheter i globala data, brott mot

namnkonventioner etc. Programmen är alltså bara ute efter att leta i kodtexten och kräver inte att koden körs.

Formell verifiering (Korrekthetsbevis)
Formell verifiering innebär att man genom ett matematiskt bevis kan förvissa sig om att systemet uppfyller de krav som ställs. Till skillnad från annan testning kan formell verifiering bevisa att systemet gör rätt i samtliga fall. För övriga testtekniker finns inget som hindrar att systemet gör fel i ett fall som inte blivit testat. Att en formell verifiering lyckats innebär inte att övrig testning blir onödig eftersom alla steg i systemutvecklingen inte kan hanteras formellt. Däremot innebär det att omfattningen av testningen, och framförallt felsökningen, kraftigt kan minskas.

3.3.2 Dynamiska tekniker

”Black box testing”
”Black box testing” innebär att testning sker genom ett externt interface, dvs. testaren vet inte vad som ligger bakom skeenden utan han fungerar som en användare. Testaren har ett testfall där utförandet och det förväntade resultatet finns specificerat, sedan jämförs resultatet med det som förväntades hända. Testningen kan också automatiseras genom att t.ex. spela in en serie knapptryckningar och sedan köra det som ett macro. Om det är värt att automatisera ett test eller inte beror dels på hur bra testet är på att hitta buggar och dels på hur många gånger testet kan användas innan det har blivit inaktuellt. ”Black box testing” är en relativt enkel teknik som är lätt att förstå och skapa testfall till, och där systemets användare kan agera testare eftersom den simulerar verklig användning. Vad som inte är bra är att samma kod kan komma att testas onödigt många gånger och man missar logiska fel i koden.

”White box testning”
”White box testing” är motsatsen till ”black box testing” i den mening att testaren har tillgång till koden och kan bygga testfall på strukturen istället för på programmets funktion. Med denna teknik kan testningen styras att agera på en viss del av programmet genom att dela upp koden i segment och sedan köra test på specifika delar. Oftast täcks fler kodrader in än med ”black box testing” och de logiska felen kan upptäckas.

Felsådd
Först programmeras fel medvetet in i koden och när testningen av den aktuella koden är klar kan man räkna ut hur stor del av alla fel som hittats genom att se på hur många procent av de medvetna felen som hittades.

Mutationstestning
Mutationstestning liknar felsådd, men här är syftet att kontrollera hur pass väl testfallen är gjorda. Man lägger in ett fel i taget och jämför den ”äkta” kodens testfallsresultat med den ”muterade” kodens. Om dessa två inte skiljer sig åt kan man dra slutsatsen att testfallet inte var tillräckligt uttömmande eftersom det inte hittade den muterade kodsnutten.

Prestandaanalys
Prestandaanalys utgörs oftast av verktyg som utvärderar hur effektivt ett program arbetar och har inte något med felsökning att göra.

Täckningsanalysator
Täckningsanalysator är ett verktyg som håller koll på vilka delar av programmet som körts efter att alla testfall har körts. Detta kan vara till hjälp för att få en bild av hur stor del av programmet som har testats.

3.4 Systemutvecklingsprocessen ur ett kvalitetssäkringsperspektiv

3.4.1 Varför standardisera?

Att certifiera sig för en standard kan enligt Oskarsson & Glass (1995) vara nyttigt av två anledningar, dels för kundens skull och dels för organisationen själv. Om en organisation har certifierat sig för en viss standard kan kunden vara förvissad om att organisationen följer en viss procedur för att säkra kvaliteten på produkten. Dessutom kan en standard med uppsatta regler att följa underlätta för ledningen eftersom den då kan ha kontroll över produktionens skeenden. Nackdelen med standarder är först och främst att det kan vara svårt att hitta en standard som passar. Standarder tenderar också att generera en hel del jobb med extra dokumentation och administration, vilket inte alltid känns berättigat. Nedan finns en närmare beskrivning av två standarder, ISO och CMM, men det finns utöver dessa ett antal andra standarder som man kan certifiera sig för t.ex. IEEE och AQAP.

3.4.2 ISO

ISO 9000 är en samling standarder och riktlinjer som ställer krav på leverantören av en produkt. Dessa standarder kan appliceras på i princip alla organisationer. ISO 9001 (”Kvalitetssystem – kvalitetssäkring vid konstruktion, utveckling, produktion, installation och service”) som är en del av ISO 9000-familjen, är utformad för organisationer med inriktning på konstruktion. Detta dokument appliceras följaktligen även på organisationer som berör programkonstruktion. I tillägg till de uppsatta kraven finns även dokumentet ISO 9000-3, som fungerar som en handledning för hur ISO 9001 bör användas vid utveckling av programvara.

Med kvalitetssäkring avses egentligen inte att man säkrar kvaliteten på produkten som sådan, utan kraven gäller snarare leverantörens organisation. Det finns två huvudpunkter i ISO 9001:

- Alla operationer som påverkar kvaliteten skall vara uppstyrda
- Denna styrning skall vara synlig

Detta innebär att organisationen måste vara relativt strikt styrd och att allt måste finnas dokumenterat. ISO 9001 innehåller 20 st. kvalitetselement som upprättar standarder för hur allt ifrån ledningens ansvar till processtyrningen bör skötas och dokumenteras. Inom tillverkningsindustrin är det oftast processen och dess styrning som innebär den största, dyraste och mest tidskrävande delen av arbetet, men när det gäller programkonstruktion ligger huvuddelen av arbetet kring konstruktionen av produkten. Processen är egentligen bara reproducerandet av det konstruerande programmet, dvs. kopiering till CD-skiva eller motsvarande. Det som har relevans för denna rapport är alltså konstruktionsfasen och därför ligger fokus här på att beskriva vad ISO 9001 ställer upp för riktlinjer för just denna del av utvecklingsprocessen.

ISO 9001 kräver att man planerar innan man gör och att man specificerar innan man konstruerar. Två planer skall utarbetas, dels en utvecklingsplan och dels en kvalitetsplan. Utvecklingsplanen skall bestå av:

- Definition av en utvecklingsprocess
- Beskrivning av organisationen och dess ledning
- Beskrivning av olika faser, enheter och områden
- Beskrivning av metoder och hjälpmedel som används

I kvalitetsplanen skall följande finnas definierat:

- Kvalitetsmål
- Kriterier för acceptans för in- och utdata mellan olika faser
- Identifiering och planering av testning, validering och verifiering
- Specifika ansvar för kvalitetsaktiviteter

Allmänt för själva konstruktionsfasen gäller att all utdata från en fas granskas och gås igenom innan den används som indata i en annan fas. Med utdata menas all konstruktionsdokumentation från den första projektplanen till kod. Denna genomgång av dokument skall vara formell och planerad och skall i sin tur även den dokumenteras.

3.4.3 CMM (Capability Maturity Model)

CMM har utformats av SEI (Software Engineering Institute) och är en standard som konkurrerar med ISO 9001 och dess applicering på programutveckling. CMM är uppbyggd kring programvaruutvecklingsorganisationens mognadsgrad och klassificerar den efter fem nivåer. Eftersom denna modell är inriktad på just programvaruutveckling är den mer specifik och detaljerad än ISO 9001. ISO 9001 fokuserar mer på relationen leverantör - kund, medan CMM:s fokus ligger på själva systemutvecklingsprocessen. De fem steg för organisationen som CMM specificerar är (CMM):

Initial: Systemutvecklingsprocessen är mer eller mindre improviserad, och kan även vara smått kaotisk. Få aktiviteter finns definierade och framgång bygger på individuella prestationer.
Upprepad: Grundläggande projektstyrning finns för att registrera kostnader, resurser och funktioner. Det finns också en definierad process som har till uppgift att upprepa tidigare projekts framgångar.
Definierad: Processen för både lednings- och utvecklingsaktiviteter finns dokumenterad, standardiserad och integrerad i organisationen. Alla projekt använder en godkänd och skräddarsydd version av organisationens standardmodell för att utveckla och underhålla mjukvara.
Styrd: Detaljerad mätning av systemutvecklingsprocess och produktkvalitet görs. Både processen och produkten blir kontrollerad.
Optimerande: Ständig processförbättring sker med hjälp av kvantitativ feedback både från processen i sig och från utvärdering av nya idéer och tekniker.

3.5 Teoretiskt ramverk för undersökningen

3.5.1 Sammanfattning av teori

För att kunna ha en bra utvecklings- och testningsmodell i bakgrunden när undersökningen gjordes utvecklades, på basis av teorin, en hypotetiskt god testmodell som kunde tänkas passa Företagets behov. Denna modell bygger på följande punkter:

- **Kvalitetssäkring**

För att uppnå en hög kvalitet på en produkt krävs god ledning och styrning. En väl utarbetad och beprövad process som har visat sig fungera och som ledningen har kontroll över och alla accepterar är en förutsättning för att produkten skall kunna hålla en jämn kvalitet. Det bör alltså inte vara testteamet som är ansvarigt för kvaliteten utan det ligger på ledningens ansvar att se till att kvalitetstänkandet finns med i hela processen (Marick, 1997). Även om ISO 9001 och CMM är konkurrerande standarder är det just denna grundidé de har gemensamt, vilket man bör ta tillvara på. Därmed inte sagt att det krävs en certifiering, om allt redan flyter på bra kan en certifiering ställa till mer problem än vad den löser. Om projektprocessen däremot inte fungerar tillfredsställande kan någon slags styrning hjälpa till, och om den sedan består av ett certifikat eller en modifiering av den egna projektmodellen spelar mindre roll. Dessutom har de olika kvalitetsstandarderna inte haft den genomslagskraft inom systemutvecklingsbranschen som många hade hoppats på, kanske på grund av att det fortfarande är en ung bransch och för att kvalitetsstandarder därför har varit svåra att definiera. (Hulme, 2002)

- **Testplanen**

När det gäller testplanen är det först och främst viktigt att identifiera programmets riskzoner och därigenom avgöra hur de olika testerna bör fördelas. Dock skall man inte låta bli att testa de delar man inte tror innehåller några allvarliga fel, det är för osäkert att vara helt och hållet ovetande om hur bra vissa delar av programmet fungerar. En kombination av riskstyrd testning och statistisk testning gör att det mesta täcks upp utan att testningen blir alltför redundant och ineffektiv. Istället för att lägga ner mycket tid på att testa varje funktion för sig bör ganska stor vikt läggas vid integrationstestningen, eftersom de flesta fel oftast uppstår i interaktionsmomentet (Perry, 2000). Testteknikerna bör också anpassas till systemets natur, generellt kan sägas att en blandning av ett par eller flera olika tekniker oftast ger bäst resultat eftersom olika tekniker täcker upp för varandras nackdelar. Ett exempel kan vara att kombinera black box testing med white box testing (Oskarsson & Glass, 1995). Alla tester kan inte automatiseras, vissa måste köras manuellt. Inte heller kan man förvänta sig att man skall hinna köra alla tester manuellt (Marick, 1998). Slutligen bör en testplan ha möjligheter att förändras om det skulle visa sig att den inte fungerar på ett tillfredsställande sätt.

- **Testningens roll i systemutvecklingsprocessen**

Testningen bör finnas med genom hela systemutvecklingsprocessen och bör ingå som en del i varje fas, inte bara vara en aktivitet som bara sker i slutet av processen. För att kunna gå från en policy att undvika fel genom att hitta dem, till en som undviker fel genom att inte göra några från början, krävs ett nytt sätt att tänka. (Hulme 2002)

All dokumentation som producerats i den aktuella fasen bör gås igenom på ett eller annat sätt t.ex. med hjälp av Software Inspection. 2/3 av alla fel i mjukvara uppstår innan man börjar skriva själva koden och en genomgång av all output från de tidigare faserna kan leda till att många fel hittas tidigt, innan de hinner programmeras in i systemet (Perry, 2000). En annan anledning till att vara noggrann med inspektionen är att dokumentationen för varje fas enligt V-modellen ligger till grund för skapandet av testfall. Även denna testdesign för varje fas kan vara bra att gå igenom innan själva testerna körs.

I ett systemutvecklingsprojekt sker hela tiden förändringar och av denna anledning utvecklades de iterativa systemutvecklingsteknikerna. För att testdesignen skall kunna följa denna utveckling krävs en modell som är anpassad till dessa förändringar. V-modellen kan fungera som grund, men den bör anpassas till ny information som tillkommer projektet i andra faser (Marick, 1999b). Ny information om en tidig fas kan komma i en senare fas, vilket gör att man inte bör sätta gränserna mellan faserna vad gäller testdesignen alltför hårt. En medvetenhet om att testdesign kan komma att ändras och anpassas till nya beslut och situationer längre fram i processen är en nödvändighet för att få testfallen så bra och effektiva som möjligt. Helst bör någon slags procedur som säkerställer att ändringar i systemdesignen även påverkar testdesignen användas.

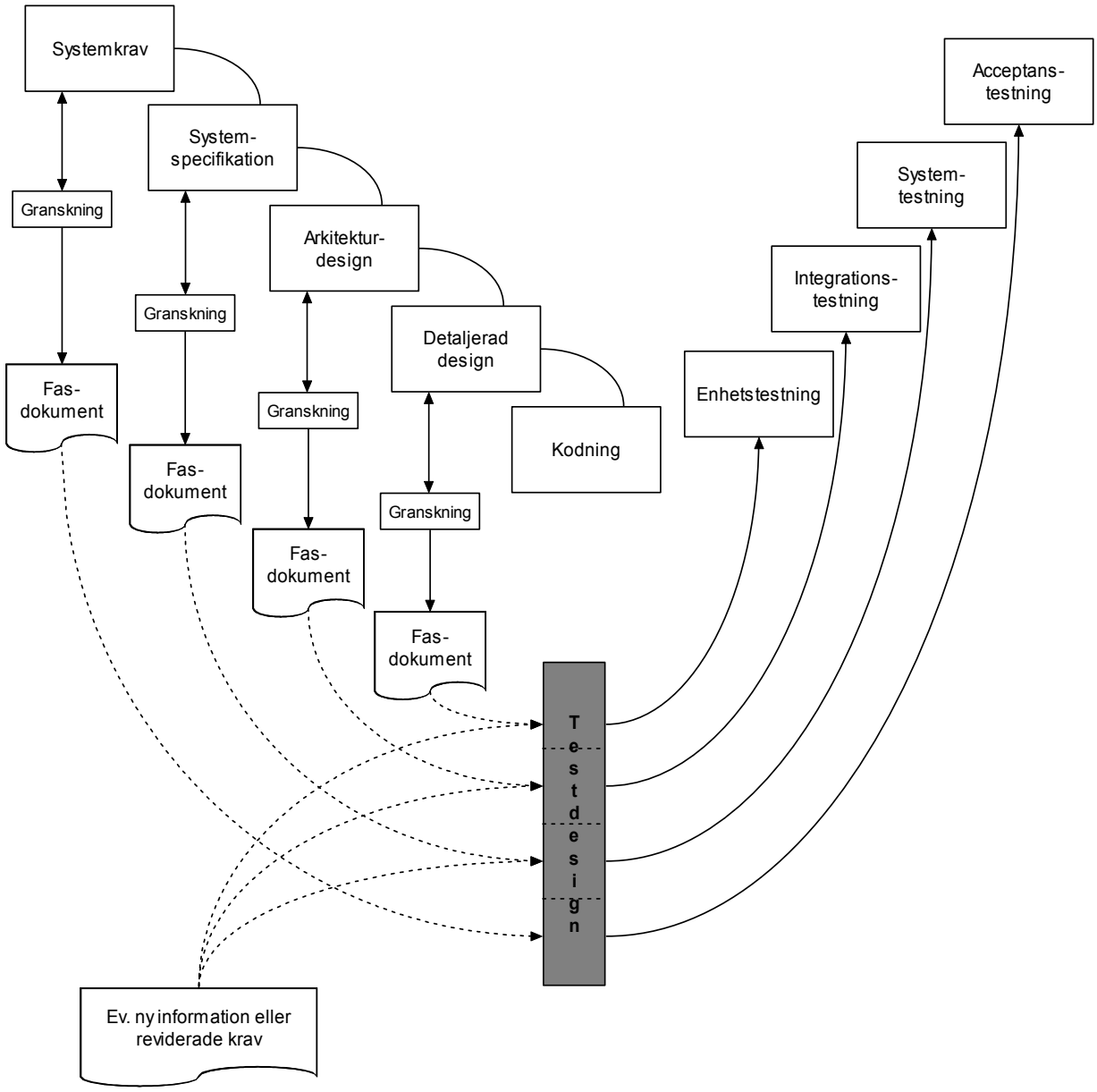
Slutligen är det inte bra att helt och hållet bara vara bunden till den dokumentation som utvecklarna producerar eftersom:

A savvy tester doesn't trust the documentation anyway. After all, the whole point of testing is that people make mistakes. Well, didn't people write those documents? (Marick, 1999b)

3.5.2 Den anpassade V-modellen

Den utarbetade testmodellen är anpassad till de krav som ställts här ovan. Den innehåller samma aktiviteter som V-modellen, men har utökats och gjorts om för att kunna ta hänsyn till tilläggsinformation och gränslös testdesign

Figur 6: Den anpassade V-modellen



Varje fas i utvecklingsprocessen ger upphov till dokumentation som skall användas av bl.a. testgruppen. Denna dokumentation är inte finit så fort den släppts, utan granskas och revideras vid eventuella felaktigheter eller tveksamheter. Dessutom kan den skickas tillbaka om det visat sig att dokumenten inte var tillräckligt utförliga för att kunna bygga testfall på. Testdesignen grundas sedan på - men är inte helt och hållet avhängig – de olika fasdokumenten, som trots granskningarna inte bör ses som helt felfria. De olika fälten i rutan Testdesign symboliserar, nerifrån och upp, design för Acceptanstestning, Systemtestning, Integrationstestning och Enhetstestning. Till skillnad från V-modellen är inte gränserna mellan de olika faserna i testdesignen i denna modell lika strikt dragna. Detta öppna synsätt har två fördelar, dels kan information som är användbar för testdesign av andra faser än just den fas som dokumentationen bygger på komma till användning, och dels kan nya uppgifter och systemändringar även påverka alla faser av testdesignen och inte bara de som inte redan gjorts. Detta gör att testdesignen blir en mer iterativ process där den efterhand som projektet fortgår förbättras och raffineras för att bli så effektiv som möjligt. Testningen kan sedan ske i samma ordning som i V-modellen, dvs. man går från litet till stort, från små enheter och moduler till helhetstestning.

Sammanfattningsvis är detta en modell där:

- All dokumentation från systemutvecklingsfaserna granskas
- Dokumentationen ligger till grund för testdesign av varje fas
- Gränserna har luckrats upp i syfte att göra processen iterativ
- Hänsyn kan tas till ny information och systemändringar

4 Resultat

I detta kapitel redovisas resultatet, med andra ord en objektiv bild av den information som framkommit under arbetets gång. Presentationen av resultatet följer de punkter som beskrevs i syftet (sid. 8-9). Under respektive rubrik finns en kort återknytning till vilken del av syftet som resultatet uppfyller.

4.1 Företagets arbetssätt

Detta kapitel anknyter till punkt 1 genom att beskriva hur företagets processer ser ut ”på pappret”. Här läggs den information fram som behövs för att svara på frågan:

- *Hur ser processerna ut idag?*

4.1.1 Projektmodellen

En stor del av det dagliga arbete som utförs på Företaget sker i projektform. Ett projekt definieras som en aktivitet som:

- Skall leda fram till ett bestämt resultat
- Är en engångsuppgift, dvs. inte direkt repetitiv
- Kräver olika resurstyper
- Är tidsbegränsad

Allt som inte kan definieras enligt dessa principer kommer på Företaget att fungera i linjeorganisation. Exempel på funktioner i Företagets linjeorganisation är Marknadsföring, Produktionsledning och Produktutveckling.

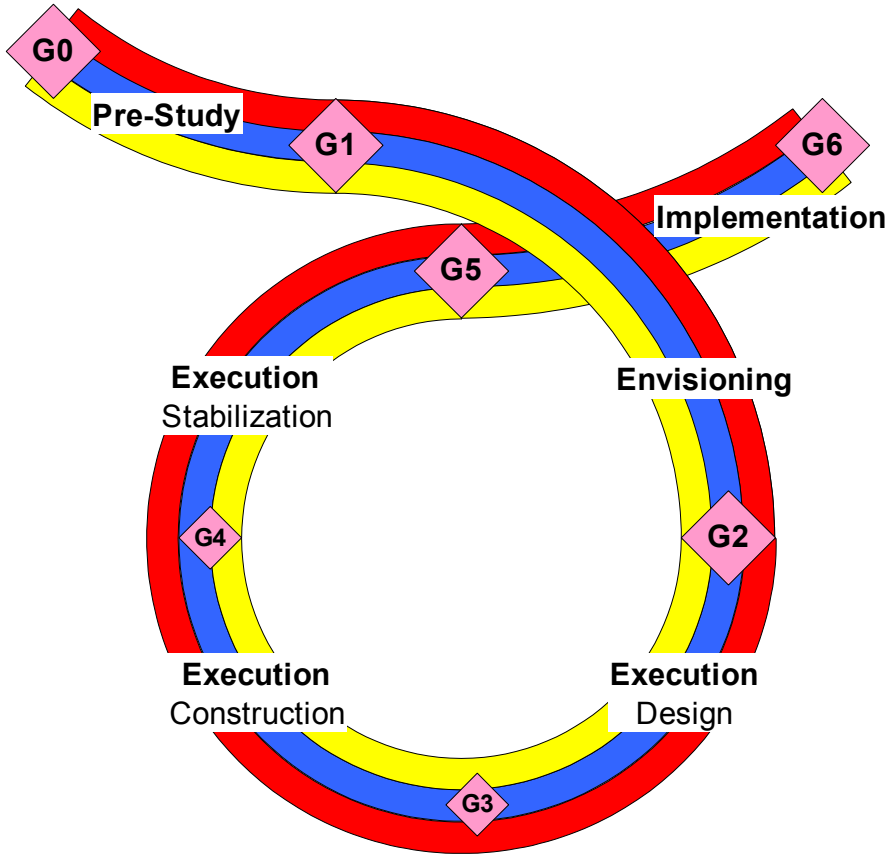
Den projektmodell som Företaget arbetar enligt liknar övriga systemutvecklingsmodeller, och innehåller de faser som finns i dessa modeller. Faserna utgörs av en del där man grovt skissar upp vad man vill ha, en designfas där man bryter ner kraven och gör mer och mer specifik design, en konstuktionsfas där själva systemet skapas, och en stabiliseringsfas där systemet testas. Till dessa faser har lagts en förstudiefas och en implementationsfas för att få med projektets hela livslängd.
I Företagets projektmodell kallas faserna

1. Pre-study phase
2. Envisioning phase
3. Execution phase
    a. Design
    b. Construction
    c. Stabilization

4. Implementation phase

Pre-study phase har till uppgift att klargöra att idén bakom projektet går att genomföra ur både tekniskt och kommersiellt hänseende. I Envisioning phase görs den grova skissen av hur projektet skall se ut och genomföras, detta för att skapa en grund för att kunna lyckas med föresatsen. Execution phase är uppdelad i tre delar och det är här som själva produkten av projektet blir till. Implementationsfasens uppgift är att se till så att den produkt som är resultatet av projektet implemeteras ute hos kunden, vilket i tid kan vara allt från en mycket lång till en mycket kort process beroende på hur projektet ser ut.

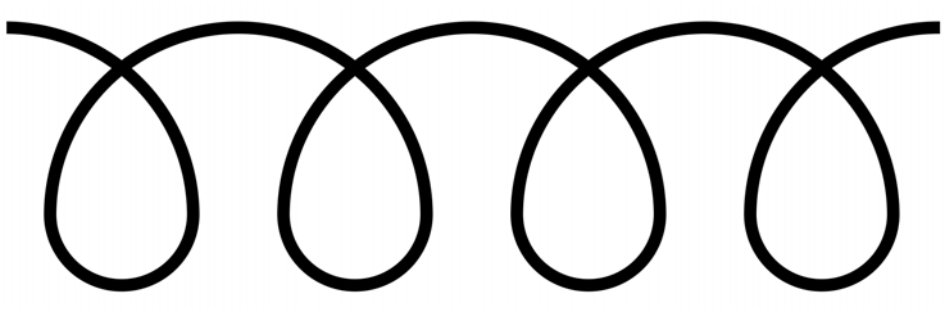
Figur 7: Företagets projektmodell



För att se till så att man är på rätt väg finns ett antal grindar, dvs. utvärderings- och korrigeringspunkter, inlagda i modellen. De ligger i början av Pre-Study och efter det i slutet av varje fas och underfaser till execution (G0 – G6). Den viktigaste grinden är grind 2 som avslutar Envisioning phase, även kallad ”Golden Gate”, där varje del av det framtida projektet skall finnas planerade, och där man utifrån denna plan avgör om projektet kan fortsätta eller inte.

När Implemetation Phase är slut och man nått G6 är projektet också slut, men erfarenheterna av det avslutade projektet kan, och bör, användas till ett nytt projekt. Dessa efterenheter blir då en del av det nya projektets prestudy, och på så vis bildar flera projekt på varandra en spiral, och processen blir iterativ.

Figur 8: Modell över hur projekten bygger på varandra

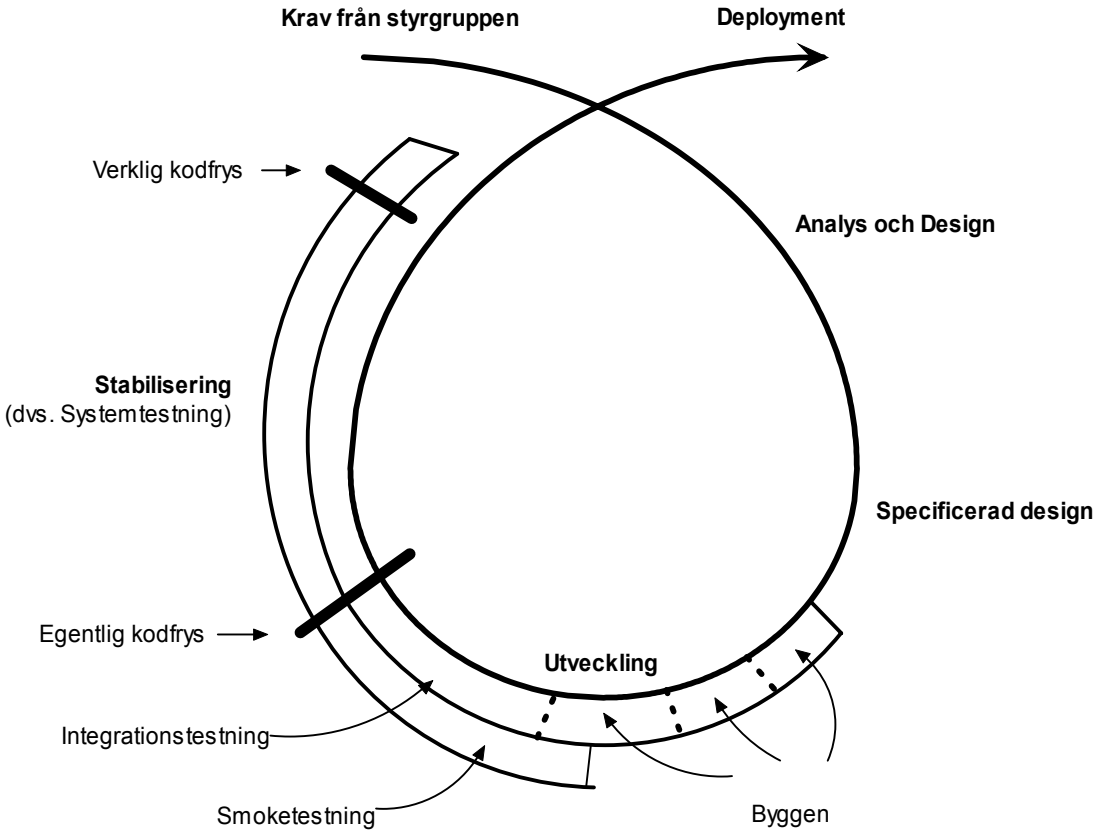


Hela projektprocessen är uppdelad i tre mindre processer som löper parallellt, Projektstyrningsprocess, Projektledningsprocess och Process för projektarbete. Dessa tre processer och dess ägare har olika arbetsuppgifter uppdelade mellan sig.

Process	Processägare	Processens syfte
Projektstyrningsprocess	Projektsponsor	Att starta stoppa eller ändra projekt med syfte att säkra att projekten och dess mål för företaget framåt i vald affärsriktning.
Projektledningsprocess	Projektledare	Att bryta ned projektets mål till leverabler och leda tilldelade resurser mot uppsatta projektmål.
Process för projektarbete	Delprojektledare	Att leverera projektets leverabler med kvalité i rätt tid till rätt kostnad.

4.1.2 Testmodellen

Figur 9: Bild över hur Företagets testning sker ihop med projektmodellen



Under Execution phase i projektmodellen finns testteamet med parallellt med utvecklingen av produkten. När projektplanen och produktens övergripande arkitektur är klar skall det även finnas en testplan med i dokumentationen. När den detaljerade designen och konstruktionsfasen sedan sätter igång görs testfall som bygger på de designdokument som finns tillgängliga. Enhetstestningen och integrationstestningen görs sedan av utvecklarna själva i takt med att produkten och dess byggen växer fram. Innan stabiliseringsfasen sätter igång, med systemtester som största aktivitet, sker så kallade Smoke-tester, vilket är en typ av test som ser till så att allt som krävs för att kunna köra en riktig systemtest finns, i form av gränssnitt m.m. När stabiliseringsfasen har satt igång skall all kod ”frysas” dvs. inget skall utvecklas vidare utan endast bugfixar bör ske här.[1] När systemtesterna är klara lämnas en testrapport med statistisk information och resultat till styrgruppen som med hjälp av denna bestämmer om produkten är tillräckligt färdig för att släppas eller inte. Innan den släpps skrivs även en s.k. Release Note som bla. beskriver den aktuella versionens kända fel och all ny funktionalitet. Därefter går projektet in i Implementation phase.

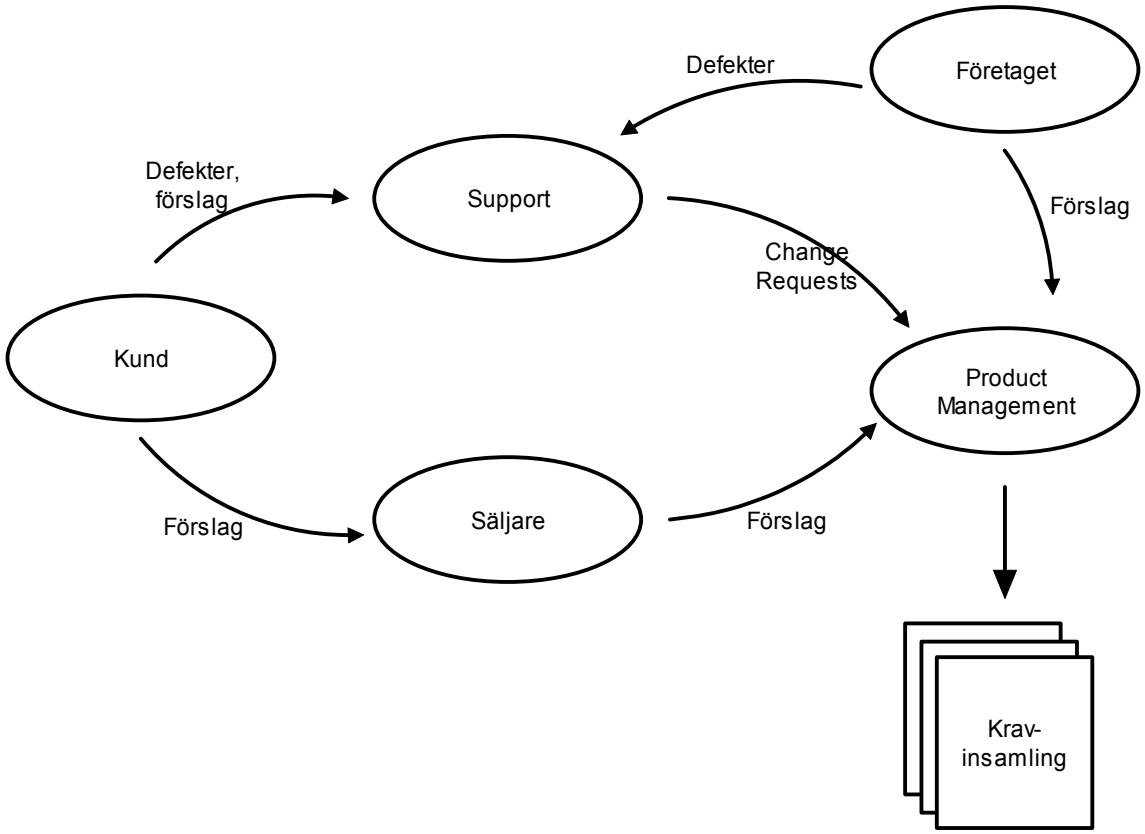
[1] I bilden visas denna gräns som ”Egentlig kodfrys” och är den gräns som eftersträvas. Det som i bilden kallas ”Verklig kodfrys” är den gräns där kodfrysen i realiteten sker, vilket tas upp längre fram (s. 44).

Det faktum att systemet måste stödja ett antal olika typer av hårdvara gör att testningen är svår att automatisera, då varje hårdvarutyp måste testas för sig. I och med detta blir den största delen av testningen ”Black box testing” och tar ganska mycket tid i anspråk. Det är detta som utgör systemtestningen, och den görs av en testpool som inte alls är kopplad till den övriga utvecklingen. Detta är ett medvetet val eftersom det är meningen att testpoolens personal vara ”nybörjare” på systemet, så att man samtidigt får ett usabilitytest.

4.1.3 Kravinsamlingen

Kravinsamlingen ingår inte som en del i projekt processen utan står utanför och klassificeras som en ständig aktivitet, dvs. en linjeaktivitet. Krav kommer in från kunder, medarbetare, säljpersonal och andra som kommit i kontakt med systemet. Dessa krav kan antingen utgöras av förslag till nya funktioner, eller av förbättringar av den redan existerande produkten, vilket inkluderar tillrättning av defekter. Product Management administrerar centralt alla de krav som kommer in på olika sätt.

Figur 10: Företagets kravinsamlingsprocess

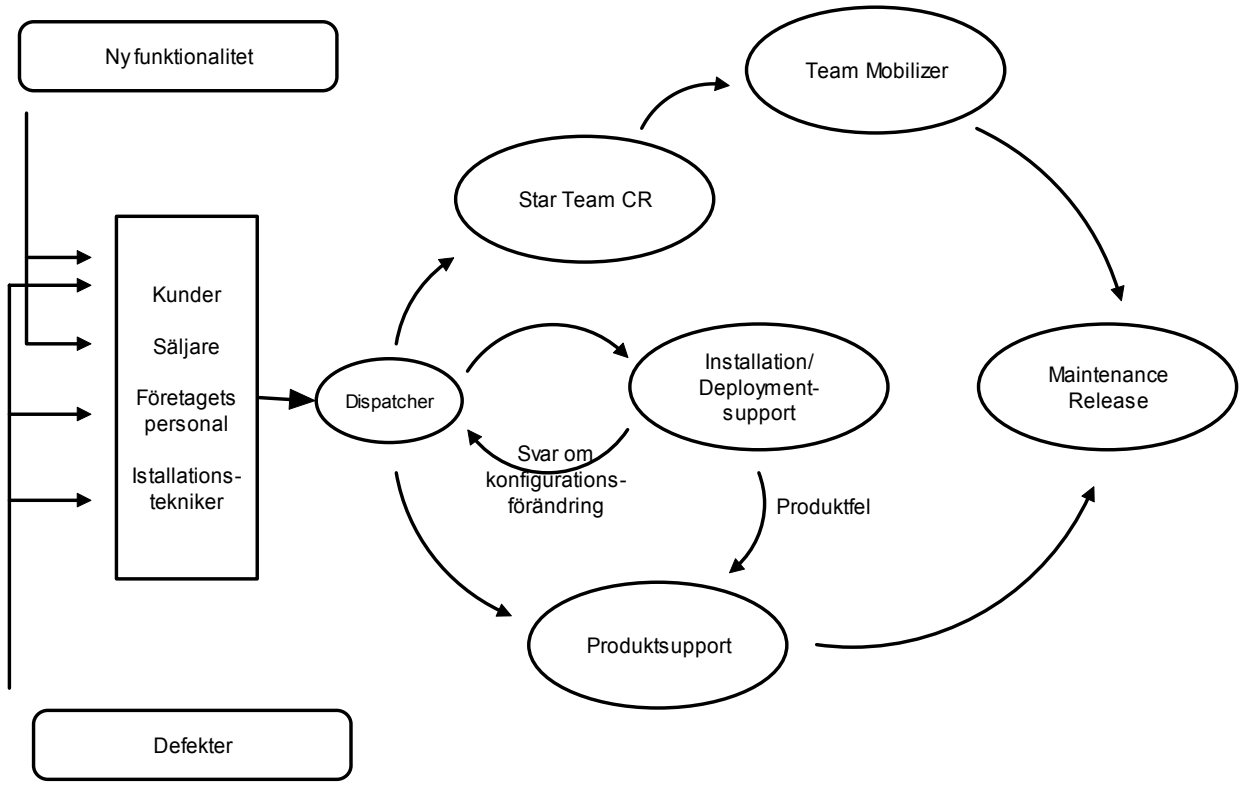


4.1.4 Supportarbetet

Support har som uppgift att serva de kunder som har problem med Företagets produkter. Företagets samarbetsorganisation med deras OEM-partners ställer till vissa problem för support. Dels finns det ingen klar bild av hur stor andel av alla supportfall som hamnar hos OEM-partners och hur stor andel som hamnar hos Företaget, dels är det i och med detta svårt att säga vilken typ av fel som egentligen är vanligast. Dessutom kan det ofta vara så att kunderna själva hittar lösningar på problem som hade kunnat vara till nytta för Företaget, men information om detta når aldrig fram till Företagets support.

I organisationen kring support som den ser ut idag, är maintenance-teamet sammanslaget med support. För att få uppdateringarna att fungera bättre beslutades i våras att man skulle förändra utskicken av maintenance-paket. Tidigare bestod paketet av enskilda filer som skickades tillsammans med en beskrivning för hur man skulle gå tillväga. Nu består paketet istället av en setup-fil som självmant installerar alla rättningar. Detta minskar risken för att personen som installerar rättningarna begår misstag, och med detta hoppas man komma tillrätta med många av de fel som dessa misstag genererar. I paketet skall 40 % av uppdateringarna bestå av maintenance, 40 % av supportförändringar, och 20 % skall reserveras för akuta ärenden.

Figur 11: Hantering av felrapporter från support



Fel eller förslag till ny funktionalitet kommer från kunder, säljare, Företagets personal och installationstekniker. En dispatcher skickar ärendet vidare till rätt person. Frågor kring installation hamnar hos Installation/Deployment support, som för närvarande endast består av en person. Vidare hamnar allt som är rena fel hos produktsupportgruppen, som även tar hand om installationsproblem som visat sig bero på fel i produkten. S.k. Change Requests hamnar i verktyget Star Team CR. En CR kan antingen vara ”defect” eller ”suggestion”. Varje månad sammanställs valda förändringar ur Star Team CR och svar från Produktsupport till en Maintenance Release.

4.2 Felkategorisering

Denna del anknyter till den andra punkten av syftet, att ta reda på var problemen uppstår, och ger svar på frågan:

- *Inom vilken felkategori hamnar de flesta issues resp. supportfall?*

4.2.1 Presentation av kategorierna

Som tidigare nämnts har inrapporterade fel tillhörande en specifik produktversion betraktats ur två olika synvinklar. De utgörs av testarnas synvinkel, dvs. issues, och av användarnas synvinkel, dvs. supportfall. För att kunna hitta bra kategorier gjordes först en studie av slumpvist valda issues och supportfall vilket resulterade i åtta olika rubriker. Nedan följer en närmre beskrivning av kategorierna.

Efter det gjordes en genomgång av alla de supportfall tillhörande den aktuella versionen som rapporterats in till Företagets support. Varje fall lästes igenom för att uppnå förståelse för problemet och vad det berodde på, och för att sedan kunna sätta in problemet i rätt felkategori. På samma sätt gjordes sedan en genomgång och kategorisering av de issues som rapporteras in från systemtestningen, med tillhörande test case.

Inställningar
Här har alla fel som har med användarinställningar att göra hamnat. Det kan röra sig om att användarkontot är inställt på ett visst sätt men inte beter sig enligt det mönstret.

Funktioner
När produktens funktioner inte fungerar enligt design eller enligt användarens tycke har felen klassificerats som funktionsfel.

Navigation
Detta är fel som uppstår när man flyttar runt mellan olika sidor. Ofta är det länkar eller tillbaka-knappar som inte fungerar, eller så kommer man till ett helt annat ställe än vad som var ämnat.

Utskrifter/fax
Det skall finnas möjlighet att via systemet direkt kunna skriva ut eller faxa dokument, och när detta inte har fungerat har felen hamnat i denna kategori.

Grafik
Alla problem som uppstår på grund av att något krånglar med grafiken har hamnat i denna kategori. Det kan röra sig om för stora ikoner, scrollbars när det inte behövs, inga scrollbars när det behövs, m.m.

Text/tecken
Specialtecken som å,ä,ö och € ställer ofta till problem, och alla fel som kan härledas till sådana specialtecken har hamnat i denna kategori.

Ordval
Detta handlar om rubriker, hälsningsfraser och annan text i produkten som inte är helt korrekt.

Format
Fel som uppstår när man försöker öppna ett dokument med ett format som systemet skall stödja har hamnat i denna kategori.

Som ett tillägg till dessa kategorier finns även när det gäller supportfall ett par feltyper som faller utanför dessa ramar. Det gäller fel som inte direkt kan härledas till Företagets produkt, utan fel som uppstår på grund av någon eller något utomstående. Dessa är:

Användarfel
Fel som uppstår på grund av att den som använder produkten inte har följt instruktionerna för hur produkten skall användas, främst handlar det om att användaren inte har varit tillräckligt noggrann vid installationen av produkten.

Miljöfel
Fel som beror på att produkten installerats i en annan miljö än den som rekommenderas, dvs. saker som att fel operativsystem används, att fel versioner av olika program används tillsammans, eller att andra program installerats på samma server som håller Företagets produkter.

Försvinnande fel
Fel som uppstått och som inrapporterats, men som sedan inte kunnat återskapas och som inte uppstått igen, eller som lösts av kunden själv efter att de inrapporterat felet.

Frågor
Består av allmänna frågor om produkten. Betraktas inte som fel och faller utanför denna undersökning.

4.2.2 Tillvägagångssätt

4.2.2.1 Issues

Under systemtestningen har varje issue nedtecknats av testaren. Varje ny issue har fått ett specifikt id och till varje issue har information och kommentarer lagts. Information som behövs för att kunna identifiera och rätta till felet är bla. produktversion, hårdvarutyp, namn på den som testat, testfallsnummer, testfallsmomentnummer och slutligen en kommentar om vad det var som gick snett. För denna undersöknings del var det endast numret på testfallet och kommentaren intressant, så för att förenkla kategoriseringen är överflödig information bortsållad i excelarket med alla issues.

Figur 12: Urklipp ur det bearbetade excelarket med inrapporterade issues

1119	36.1	2	Det går inte att läsa textfältet. Rubriken syns men inte innehållet.
1120	24	6	När man har gått in finns det ingen back knapp. Det är alltså inte möjligt att gå tillbaka igen utan man måste gå till huvudmenyn.
1121	8	1	Sidan vill ej öppnas opening page lägger av direkt däremot fungerar det att öppna resterande sidor

För att problemet skall kunna sättas i ett sammanhang krävs en inblick i det test case där felet uppstod. Ett test case är en beskrivning av hur testaren skall gå tillväga för att testa en viss funktion. Där står, steg för steg, exakt vad testaren skall göra och vad det förväntade resultatet av handlingen är.

4.2.2.2 Supportfall

Företaget har ett verktyg, Star Team, för att rapportera in supportfall där all information som behövs för att kunna få full förståelse av problemet och dess lösning finns. Här lagras allt som har med supportfallet att göra, dvs. förklaring över problemet som uppstod, vem som tag hand om fallet, vem som har sagt vad till vem och när, hur problemet löstes och hur många timmar som lagts ner på att lösa fallet.

För varje issue respektive supportfall har sedan det aktuella id-numret lagts in i passande cell i en tabell som på x-axeln har kolumnerna Avvikelse från specifikationen, Avsaknad enligt specifikationen, Tillägg till specifikationen och Avsaknad enligt användaren, och

på y-axeln kategorierna beskrivna ovan, Inställningar, Funktioner, Navigation, Utskrifter/fax, Grafik, Text/tecken, Ordval och Format.

4.2.3 Resultat i siffror och bilder

Sammanlagt har 175 issues och 41 supportfall betraktats och kategoriserats, alla tillhörande den aktuella versionen. Resultatet av denna indelning kan betraktas nedan.

Figur 13: Fördelning av issues

Issues	*Spec. avvikelse*	*Spec. avsaknad*	*Spec. tillägg*	*Anv. avsaknad*
Inställningar	7	1	0	1
Funktioner	54	28	1	0
Navigation	12	16	0	2
Utskrifter/fax	11	0	0	0
Grafik	7	6	0	4
Text/tecken	9	0	0	0
Ordval	5	1	0	4
Format	7	0	0	0

Figur 14: Fördelning av supportfall

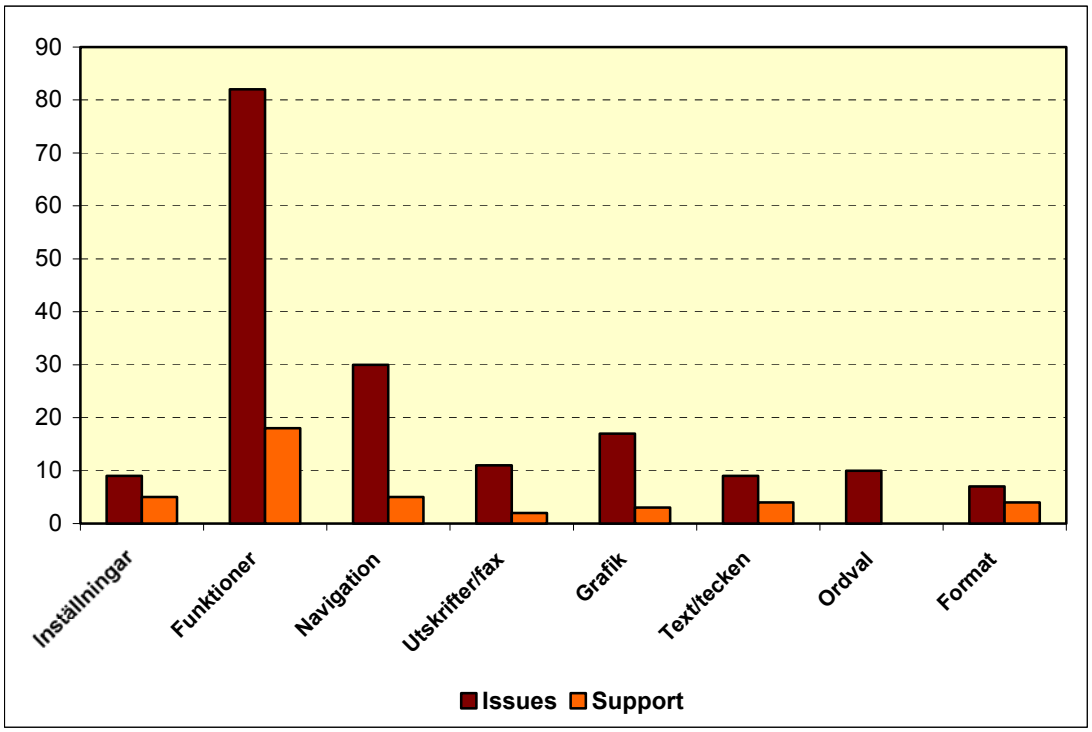
Supportfall	*Spec. avvikelse*	*Spec. avsaknad*	*Spec. tillägg*	*Anv. avsaknad*
Inställningar	2	2	0	1
Funktioner	11	2	0	5
Navigation	4	1	0	0
Utskrifter/fax	1	0	0	1
Grafik	0	1	0	2
Text/tecken	4	0	0	0
Ordval	0	0	0	0
Format	3	0	0	1

Vid en jämförelse liknar bilden för inom vilken kategori man har mest problem vad gäller issues den för supportfall. Det är främst kategorin Funktioner som ställer till problem, och då inom den typ där funktionen i fråga finns, men den inte beter sig enligt specifikationen, och detta gäller såväl issues som supportfall. Procentuellt sett har dock de funktionsfel som kategoriserats som Avsaknad enligt specifikationen minskat rejält när produkten kommit ut till användarna, 33% av funktionsfelen kunde kategoriseras som Avsaknad vid systemtestningen, medan endast 11% av funktionsfelen föll inom samma kategori med produkten ute på marknaden. Däremot har man inte vid systemtestningen

kunnat förutse användarnas behov vad gäller kategorin funktioner, eftersom testarna inte rapporterat in ett enda sådant fel, medan 12% (5st) supportfall handlar om funktioner som fattas ur ett användarperspektiv.

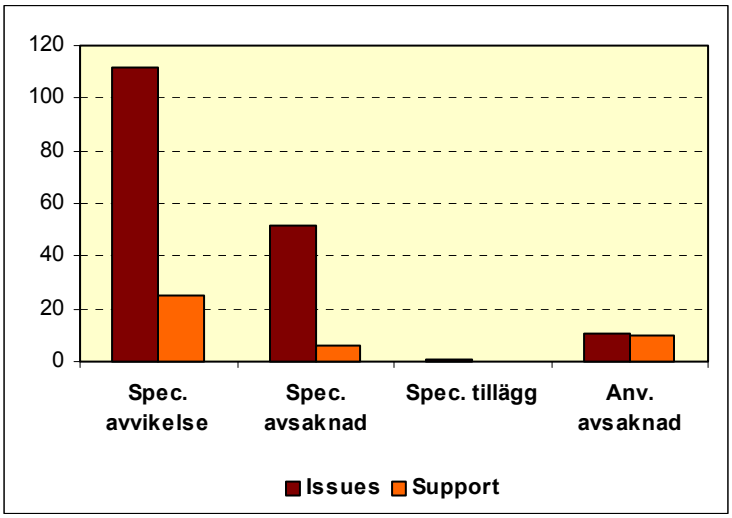
Vidare utgör navigationsfelen med länkar och knappar som inte fungerar en ganska stor del av felen vid systemtestningen, men största delen av dessa har rättats till innan produkten släppts.

Figur 15: Diagram över fördelning av issues och supportfall efter kategorier



Sammantaget kan de flesta av felen klassas som avvikelser från specifikationen. 63% (137/217) av alla fel hamnar inom denna grupp. 27% (58/217) av felen kan klassas som avsaknad av funktioner jämfört med specifikationen. Däremot är de två återstående kategorierna försvinnande små, framför allt de funktioner som klassas som extra, men även det som ur användarperspektivet saknas i produkten.

Figur 16: Diagram över fördelning av issues och supportfall efter feltyp



4.2.4 Supportspecifika fel

Vad gäller de fel som inte kan härledas till Företagets produkt är det Användarfelen som utgör den största gruppen fel. Även Miljöfelen beror till största delen på att användarna av en eller annan anledning inte följer instruktionerna vid installering av produkten och kan också räknas som användarfel, om än inte inom exakt samma kategori.

Figur 17: Fördelning av supportspecifika fel

Feltyp	**Antal**
Användarfel	22
Miljöfel	8
Försvinnande fel	10
Frågor	11

4.3 Intervjuer

Här nedan läggs den information fram som ger svar på frågan:

- *Vad beror utfallet av undersökningen på?*

Intervjuerna anknyter till syftets tredje punkt, dvs. den del som avser arbetet med att ta reda på varför problemen uppstår.

4.3.1 Sammanfattning av intervjuer

Nedan följer en redovisning av vad som sagts under intervjuerna. Informationen är grupperad efter ämne och inte efter respondent för att läsaren skall få en klarare bild av situationen utan att hacka upp den.

4.3.1.1 Processen

Det här avsnittet tar upp projektmodellen som finns beskriven här ovan (sid. 29). Meningen med denna del av intervjun var att försöka utröna om de arbetar enligt den anslagna modellen och om den fungerar. Modellen utformades för ungefär ett år sedan och Företagets personal har sedan dess haft den som mall att arbeta efter.

De av respondenterna som har haft mest med modellen att göra tycker att den i stort sett fungerar bra och att den, i den mån det går, motsvarar verkligheten. Problem uppstår först och främst när man i modellen har fasta gränser, vilket i praktiken är svårt att hålla hårt på. Både projektledaren och utvecklingschefen ansåg att det som projektledarna hade svårt att hålla på i modellen var att undvika att börja med utvecklingen innan man hade kommit fram till den s.k. Golden Gate. Ur kvalitetsansvariges perspektiv är det kodfrys som inte fungerar i praktiken. Den skall egentligen ligga direkt efter utvecklingsfasen och innebära att ingen kodning förutom att fixa till eventuella fel får ske under systemtestningen, men den verkliga kodfrysen ligger i stället alldeles i slutet av stabiliseringsfasen, dvs. strax innan systemtestningen är klar.

I praktiken är det enligt projektledaren inte heller alltid nödvändigt att följa modellen till punkt och pricka. I vilken grad den bör följas beror på projektets storlek och hur många personer det involverar.

Testaren hade aldrig fått information om processmodellen och visste inte att det fanns en modell att följa. Däremot visade det sig att han i praktiken ändå arbetar enligt den modell som finns. Angående hur han anser att processen fungerar tycker han att arbetet ofta hindras av att utvecklingen försenas. Ofta sitter en hel grupp testare och väntar på ett visst bygge utan att kunna göra någonting. Eftersom testpoolen endast är där för att testa blir det på grund av dessa förseningar lång och meningslös väntan. Sammanfattningsvis tycker han att testarna i de flesta avseenden kommer i sista hand i planeringen. Ibland har det inte funnits verktyg för att kunna utföra de specificerade testerna, och ofta får testarna sätta sig ”där det finns plats”.

Kvalitetsansvarige ser gärna att de som skriver testfallen skulle ha en större inblick i analys och design-fasen. Dessutom skulle både han och projektledaren välkomna fler projekt där man använder sig av prototyper och låter användare göra usabilitytester på den direkt efter analys och design för att få en uppfattning om det man utvecklar verkligen kommer att fungera på marknaden.

Kvalitetsansvarige tror också att kodgranskning skulle vara bra att införa, men tror att det blir svårt eftersom man varken har tid som det ser ut nu eller har den företagskulturen. Även projektledaren ser på kodgranskning som något som definitivt skulle inverka positivt på kvaliteten.

4.3.1.2 Kommunikation och styrning

Kommunikationen på Företaget sker mycket via mail och via mun, och är väldigt lite uppstyrd, men fungerar enligt utvecklingschefen trots allt relativt väl på grund av att stora delar av personalen lägger ner mycket jobb och tar ansvar för sin uppgift. De flesta anser dock att det finns mycket att förbättra vad gäller kommunikation mellan de olika delarna av processen. Tex. anser supportchefen och projektledaren att man bättre borde ta tillvara på den information som kommer från support. Supportchefen känner dock att det finns en konflikt mellan de kundönskemål som kommer in från support och de som kommer via säljarna. För att kunna sälja in produkten hos kunderna vill säljarna lägga krutet på nya funktioner medan support vill åtgärda grundläggande saker som länge har genererat problem och därmed öka kvaliteten på de funktioner som redan finns.

Enligt utvecklingschefen finns också kommunikationsbrister i kravinsamlingen. Han kallar den för ”ad hoc”, då den mer eller mindre saknar styrning. Det handlar ofta om att sälja in nya krav på ett bra sätt till styrgruppen, snarare än att man väljer det som verkligen är bäst för produkten.

När det kommer till hur mycket styrning som behövs i framtiden går åsikterna lite isär. De är alla överens om att Företaget är en ung organisation, både företagsmässigt och personalmässigt, som har och har haft väldigt lite styrning, och att det hittills har varit stormigt. Hur det bör se ut är de dock inte helt överens om. Utvecklingschefen och projektledaren anser att det för tillfället inte behövs så mycket mer styrning än vad som redan finns, att det snarare skulle göra processen tungrodd och långsam än bra och smidig. De anser att man har börjat lära sig av sina misstag och att en högre grad av styrning därför inte är så nödvändig. Dessutom kan det till och med löna sig att ha en lösare attityd och ge folk större frihet. supportchefen och kvalitetsansvarige skulle däremot välkomna en högre grad av styrning inom vissa områden av flera skäl, främst för att ha större kontroll över informationsflöden, men även för att ha lite kontroll över vem som gör vad och varför.

Kvalitetsansvarige ser gärna att Analys- och designfasen blir lite mer strukturerad och att all information samlas på ett ställe, så att även de som står utanför just den delen av processen kan ta del av vad som händer. Ett annat exempel på brist på styrning är versionshanteringen. Den har i och för sig en väl inarbetad och fungerande arbetsgång, men den finns inte dokumenterad någonstans utan kvalitetsansvarige har allt i huvudet. Supportchefen anser att bristen på dokumentation över vad folk gör kan ställa till stora problem när det gäller att ersätta personal som försvinner.

4.3.1.3 Felorsaker

Respondenterna verkar ha en ganska klar bild av vad det är som genererar mycket fel och varför. Utvecklingschefen tror att det finns två orsaker. Dels utgör det gamla systemet som dagens produkt bygger på en för dålig grund för att man egentligen skall kunna göra ett felfriare system, och dels har det att göra med att personalen till stora delar är ung och oerfaren. De flesta som arbetar som utvecklare inom Företaget har högst ett par års tidigare arbetslivserfarenhet inom branschen. Detta tillsammans med den låga graden av styrning har kanske inte varit den mest lyckade kombinationen ur kvalitetssäkringssynpunkt, men valet att anställa en ung personal har även haft sina fördelar, anser utvecklingschefen.

Projektledaren tar först upp de installations- och användarproblem som kommit upp, och tycker det är synd att det blir så mycket fel som inte beror på produkten utan på annat. Systemet får inte rätt förutsättningar att kunna fungera på ett bra sätt. Vad gäller de fel som finns i produkten tror också projektledaren att det mesta beror på att systemet från början endast var avsett att bli ett sommarprojekt, men sedan fick utgöra grunden för ett mycket större system. Detta har gjort stora delar av koden till ”spaghettikod”, dvs. kod som är svår att strukturera upp och där en liten förändring någonstans kan ge oanade konsekvenser någon annanstans. Även testaren ser detta senaste fenomen som huvudorsak till fel.

Supportchefen är även han inne på linjen att systemet inte får rätt förutsättningar, och tror att många av de fel som de i supporten stöter på uppstår på grund av att de som installerar produkterna inte är tillräckligt kunniga eller noggranna. Många fel beror helt enkelt på att man som användare eller administratör inte har läst instruktionerna ordentligt. För att komma tillrätta med problemen försöker man nu utforma en enklare installeringsprocedur, som inte kommer att kräva lika mycket av den som installerar den. Detta kommer förhoppningsvis att ta hand om många av de slarvfel som uppstår vid installering av produkten. Dock tror inte supportchefen att man helt kan komma bort från användarfel eftersom man aldrig helt kan automatisera installeringen. Som nämnts tidigare är även supportchefen medveten om de fel som finns i produkten och vad de beror på. Han ser hellre att man i vissa fall ordnar en bra grund för produkten att stå på innan man lägger till fler funktioner.

4.3.1.4 Att införa nyheter

Övergripande tycker inte respondenterna att det är något problem att införa nya idéer eller processer på Företaget. Både respondenterna själva och miljön de arbetar i verkar vara öppna för nya förslag. Utvecklingschefen tycker att så länge man visar att förändringen är till för att förbättra verksamheten och för att öka produktiviteten finns det inga hinder för att komma med nya bud, även om vissa delar av förändringen skulle

innebära ”tråkiga” arbetsuppgifter. På individnivå tror han dock att det kan vara annorlunda. Inte heller projektledaren anser att det är några problem att komma med nya arbetssätt, även om nyheterna skulle vara av mindre positiv karaktär. Han tycker att personalen i stort tar ett större professionellt ansvar nu jämfört med innan, mycket på grund av erfarenheter och utökad kunskap. Supportchefen ser positivt på den nya arbetsgången, och tror att den skall kunna underlätta för honom och hans medarbetare i framtiden, även om inte allt löper smidigt riktigt än.

5 Resultatanalys

5.1 Allmänt

På gott och ont har Företaget som ett relativt litet företag en låg grad av styrning. Om man skulle klassificera Företaget enligt CMM-modellen skulle de troligtvis hamna mellan stadierna Initial och Upprepad. Det finns en definierad process som följs, men ofta beror framgången på individuella prestationer. De nackdelar som denna låga grad av styrning medför är att man inte så lätt kan byta personal, när personalomsättningen ökar i och med att företagets storlek ökar blir det svårt att ta in nya medarbetare och få dem att ha samma funktion som de andra hade tidigare. Det kan också bli en ganska stor snedbelastning av arbetsbördan, där vissa personer gör mer saker än andra. Det finns även fördelar med att inte styra alltför mycket i och med att man lättare kan behålla organisationens gemytlighet och kreativitet, vilket på ett sätt är en nödvändighet för Företaget, då en stor del av deras verksamhets fortlevnad är beroende av kreativitet för att hitta på nya saker som tidigare inte existerat.

5.2 Projektmodellen

Den projektmodell som Företaget arbetar efter är numera väl använd och fungerar som ett bra hjälpmedel när den behövs. Den kontrolleras av ledningen och accepteras och används av övrig personal, vilket utgör en god grund för att produkten som utvecklas skall kunna hålla en jämn kvalitet. Den motsvarar i princip det som i realiteten händer i processen, även om det finns problem med att sätta fasta gränser i verkligheten.

Något som dock skulle kunna förbättras angående projektmodellen är kommunikationen mellan processens olika delar. Det är ett faktum att de flesta av de fel som hittas under systemtestningen hamnar i kategorin funktioner, och att många av dem är ett resultat av dålig kommunikation mellan Designfasen och Konstruktionsfasen. I inledningen där fel som fenomen specificerades listades bl.a. dessa orsaker till att fel där programmet inte uppför sig i enlighet med specifikationen uppkommer:

- Designspecifikationen uppfyller inte systemkraven
- Designspecifikationen misstolkas och det blir fel i programspecifikationen
- Programkoden stämmer inte med programspecifikationen

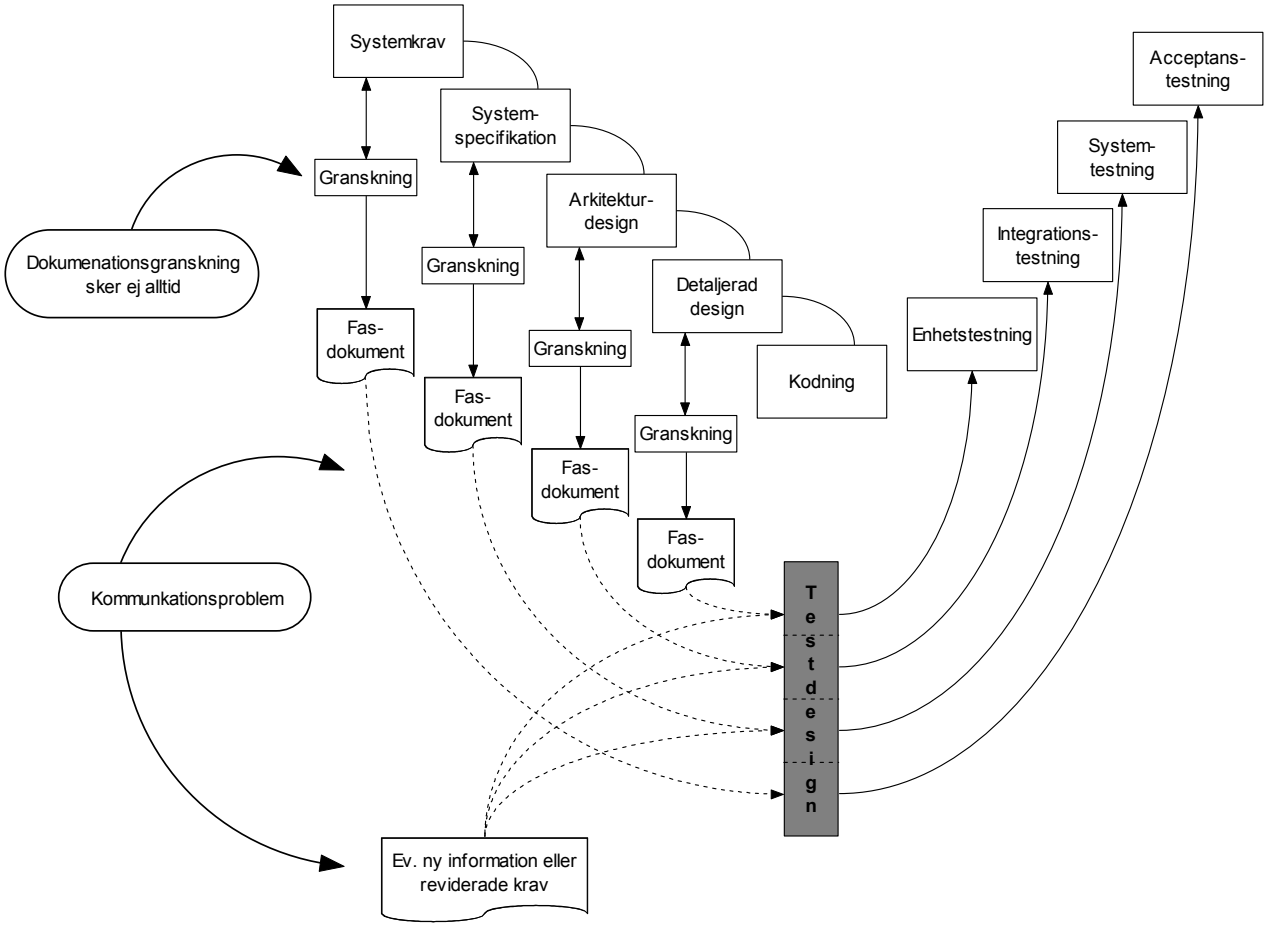
Att ovanstående punkter inträffar beror på missförstånd mellan systemutvecklingsprocessens olika delar och leder till att funktioner som finns specificerade i ett tidigt skede ”försvinner” på vägen och inte implementeras.
Med andra ord, alla fel som hamnat under ”Avsaknad enligt specifikationen” har någon av dessa orsaker, och skulle kunna undvikas genom att förbättra kommunikationen.

En annan del av processen som skulle kunna förbättras är den som skall se till så att samma fel inte begås i flera projekt. Den ständiga processförbättring som skulle kunna ske med hjälp av feedback från tidigare projekt uteblir på grund utav bristen på kommunikation bl.a. mellan support och styrgruppen.

5.3 Testning

Vid en jämförelse mellan den testningsmodell som presenterades i teoriavsnittet och den som finns på Företaget och vad som har sagts om den, ser man först och främst skillnader i informationsflödena. Den återkoppling som behövs för att testteamet skall ha full kontroll finns inte. Det händer att saker sker i utvecklingen som testteamet inte vet om på grund av att allt inte dokumenteras eller att information inte finns tillgänglig, vilket bevisas av att testfallen ibland inte alls går att köra eftersom de har blivit förlegade.

Figur 19: Företagets brister enligt den anpassade V-modellen



Detta ger en känsla av att testningen egentligen inte är något som helt och hållet deltar i processen, utan testteamet får på eget initiativ se till så att de får all den information de behöver. Ny information som tillkommer processen verkar inte heller komma hela vägen fram. Enligt projektmodellen är det projektledaren som är ansvarig för projektets leverabler, alltså borde det också vara projektledaren som ser till så att testteamet får den information de behöver och inte testteamet själva. Även uttalanden som att en bättre kommunikation mellan de som arbetar med design av systemet och de som skriver testfall efterlyses visar även det på att testningen inte ses som en nödvändig del i projektets alla faser.

Granskning av dokument görs i relativt liten utsträckning och granskning av kod görs inte alls. Detta medverkar till att testningsfasen ofta blir lång eftersom upptäckten av alla fel skjuts upp till slutet.

5.4 Otillräcklig kodgrund

Om vi återigen går tillbaka till felorsakerna finns det två punkter inom ramen för fel enligt specifikationen kvar att avhandla och de är:

- Datainmatningsfel
- Avlusningsfel

All den ”felaktiga” kod som programmerats in i systemet i ett tidigare skede och som ännu inte rättats till eller anpassats efter systemets storlek kan räknas som datainmatningsfel. Orsaken till att koden alltså ser ut som den gör och därmed producerar så pass mycket funktionsfel som hamnar inom gruppen ”Fel enligt specifikationen” är att den har en för dålig grund, och om man vill minska på denna typ av fel måste man satsa på att göra om grunden. En orsak till att detta inte gjorts tidigare kan vara att i konflikten mellan användarkrav som kommer från support och användarkrav som kommer från säljarna, har fokus legat på att implementera nya funktioner istället för att förbättra de man redan har. Kravinsamlingen fungerar inte heller tillräckligt väl för att kunna hantera denna konflikt.

5.5 Användare och Företaget

De punkter i teorin som visar på orsaker till fel där programmet inte uppför sig enligt användarens förväntningar är:

- Systemkraven från användarna misstolkas eller nedtecknas inte tillräckligt väl.
- Användarna misslyckas med att specificera vad de vill ha, eller ändrar sig och kommer in med krav på produkten för sent.

Denna typ av fel utgör en ganska liten del av supportfallen, vilket kan ha sin bakgrund i användarna inte själva beställer systemet, utan bara får en ny produkt och har därför inga direkta referensramar. Däremot gör samma faktum, att användarna får en helt ny produkt, ofta att de gör många fel när de har produkten i sin hand, eftersom de inte vet hur de skall hantera den, och det är här problemet ligger med relationen mellan användare och Företaget. Det som tidigare togs upp som orsak till denna typ av fel var att:

- Felaktigheter i dokumentation m.m. om produkten kan leda till att användaren förväntar sig något annat av programmet.

Nu handlar det i realiteten i Företagets fall inte främst om att dokumentationen kring produkten är felaktig, utan det är snarare så att installationen av produkten kräver mer av installationsteknikern och av miljön än vad installeringsteknikerna själva tror.

6 Slutsatser

6.1 Frågesvar och rekommendationer

Här följer en kort sammanfattning av vad av vad undersökningen har resulterat i. Detta görs genom att besvara de frågor som ställdes i inledningskapitlet. Svaret på huvudfrågan fungerar som den rekommendation till Företaget som undersökningen syftade till att uppnå.

- *Hur ser processerna ut idag?*

Det som finns dokumenterat överensstämmer ganska bra med verkligheten. Frågan är snarare om alla de delar av verkligheten som, för kommunikationens och säkerhetens skull, bör finnas nedtecknade verkligen dokumenteras. Det finns delar av arbetsgången på Företaget som fungerar nu, men som inte är tillräckligt dokumenterad eller uppstyrd, vilket i framtiden skulle kunna få konsekvenser för företaget. Om kunskaperna kring en viss arbetsuppgift begränsas till en eller ett högst ett par personer och om den/de av en eller annan anledning skulle försvinna, går värdefull information om intet om detta då inte finns nedtecknat. Sammanhängande med detta är också bristerna i informationsflödet. Det är inte uppstyrt till att informationen alltid skall vara tillgängligt för rätt personer vid rätt tillfälle. Mycket av kommunikationen sker på impuls, vilket ofta leder till att informationen inte alltid hittar fram till alla de personer som berörs av informationen i fråga.

- *Inom vilken felkategori hamnar de flesta issues resp. supportfall?*

Den kategori som uppvisade överlägset flest fel, både som issues och som supportfall, var kategorin Funktioner inom feltypen ”Avvikelse från specifikationen”. Denna typ av fel handlar oftast om ”oförklarliga” och/eller svårfunna småfel som kan ställa till stor skada.

Support hade också en hög grad av fel i kategorin Användarfel, dvs. fel som uppstår på grund av att den som använder produkten inte har följt instruktionerna för hur produkten skall användas.

- *Vad beror utfallet av undersökningen på?*

Att så många fel hamnade i funktionskategorin har sin grund i att systemet från början var tänkt att vara ett sommarprojekt men blev utbyggt och fick utgöra grunden till kommande system. Det var alltså inte anpassat för att bli så stort som det blev, och man har heller inte gjort någon stor satsning på att försöka göra om systemets grund. Detta kan jämföras med risken att utveckla med hjälp av prototyping och använda prototyperna som skarpa versioner på grund av resursbrist istället för att se prototyperna som hjälpmedel.

I tillägg till detta har en kombination av låg grad av kontroll, t.ex. i form av granskningar, och en oerfaren utvecklingsstab gjort att många av felen har släppts igenom i alltför hög grad.

Installationen av produkten och dess krav på rätt miljö har tidigare varit det som har ställt till problem inom support. Det kräver att den som installerar produkten och den som sköter om den dagliga driften innehar viss kunskap om produkten, vilket inte alltid har varit fallet.

- *Hur kan processerna på företaget omarbetas för att förbättra kvaliteten på deras produkter?*

För att kunna undvika fel genom att inte göra några från början, och för att komma tillrätta med alla de fel som hamnar inom kategorin Funktioner, krävs att man tänker om. För det första kan en mer stabil grund anpassad till systemets storlek hjälpa utvecklarna att i högra grad undvika att bygga in buggar. För det andra bör testningen finnas med genom hela systemutvecklingsprocessen och ingå som en del i varje fas, inte bara vara en aktivitet som endast sker i slutet av processen. För det tredje bör man i högre grad använda sig av granskningar. Detta skulle korta ner testningsprocessen och man skulle få mer tid över till annat, t.ex. att göra usabilitytester på prototyper. Granskningen skulle inte behöva ta så mycket tid i anspråk, en lättare version av granskning kan ske vid enhetstestningen. Om den görs av någon annan än den som själv gjort den enhet som testas, och om det finns en vilja att rapportera in misstag som begåtts redan i detta skede, och en procedur för att sköta detta, kan man vinna tid utan att ha satsat alltför mycket resurser. En annan fördel med granskningar är att man alltid har minst två personer som är insatta i dokumentet eller koden, vilket gör att man alltid har en backup-person.

De fel som hamnar inom den kategori där användarna tycker att systemet skulle ha sett ut på ett annat sätt, eller haft andra funktioner än de som redan finns, är inte särskilt många, vilket tyder på att användarna på detta plan är nöjda med vad de får. Av denna anledning finns det inte mycket att vinna på att lägga in usabilitytester. Trots det kan prototyputveckling ändå vara till stor fördel om man tar tillvara på användarnas åsikter eftersom man då ”gratis” får idéer om vad användarna eftersöker och kan på så vis ligga steget före i nästa produkt.

Många av felen inom kategorin Användarfel kan förhoppningsvis till viss del redan vara avhjälpta genom den nya ordningen med Maintenance Releaser, där mindre krav ställs på den som underhåller systemet genom att man nu endast har en fil att hantera.

Vad gäller styrning tror jag att projektmodellen som helhet i nuläget redan är tillräckligt uppstyrd, att börja på något nytt nu när modellen verkligen börjar bli inarbetad leder bara till att det jobb man haft med att implementera den skulle vara ogjort. Däremot kan man förbättra informationsflödet genom att se över kommunikationen mellan de olika processdelarna. Att alla som projektet berör är på det klara med vad som händer och har

full möjlighet att ta tillvara på den information som finns har en positiv inverkan på kvaliteten. Man kan också öka säkerheten genom att låta informationen fördelas ungefär som den sprids på en RAID 5[2]. Om man ser till så att alla till viss del är insatta i varandras respektive uppgifter får man också en ”säkerhetskopia” av informationen, och kan vid bortfall rekonstruera förlorad kunskap.

6.2 Diskussion

Denna uppsats syftade till att utröna var i systemutvecklings- och testningsprocessen man borde göra förändringar för att uppnå en så hög kvalitet som möjligt på systemet och detta anser jag att den har presenterat ett förslag på. Förhoppningsvis kommer denna rapport att leda till att Företaget får insikt i vad det är som ställer till problem och ett alternativ för hur dessa problem kan lösas. Det som framkommit är inget som egentligen överraskar någon, det mesta är Företagets personal redan medvetna om. Trots detta kan det vara en stor fördel att få problemområden beskrivna svart på vitt utifrån ett objektivt perspektiv. Framgent kan det vara en bra idé för företaget att på ett liknande sätt kategorisera fel och studera resultatet för att få exakta siffror på inom vilka områden man behöver lägga mer tid på att försöka undvika fel. Dessutom kan en liknande statistisk undersökning ge en vink om effekterna av eventuella förändringar inom organisationen.

I själva arbetet hade en jämförelse mellan olika versioner av produkten kunnat ge en mer rättvis bild av var felen uppstår. Vad som inte prioriterats i en version kan ha legat högt upp på önskelistan inför nästa, vilket skulle kunna ha gett en annorlunda bild av fördelningen av fel. Dessutom kunde en uppdelning av felkategorin ”Funktioner” kunnat ge en fingervisning om exakt vilken typ av funktioner som ställer till problem. För detta krävdes dock mer tid och större inblick i produkten.

I ett längre perspektiv skulle en fortsatt allmän forskning kring förhållandet mellan gamla stabila och nya instabila funktioner vara ett intressant ämne. Frågan är om man, som nämnts i inledningen, kan jämställa systemutveckling med traditionell processindustri vad gäller kvalitet, och kanske framför allt, om det alltid är lika viktigt. Här kan även en jämförelse mellan äldre och yngre användare och hur deras mentalitet gentemot kvalitet i samband med system är vara intressant och ett sätt att sondera den framtida marknaden.

[2] (Redundant Array of Independent Discs 5). Ett RAID-system består av två eller flera samverkande hårddiskar, vilket bl.a. kan ge säkrare drift och högre feltolerans genom att systemet kan fås att fortsätta att fungera även om en disk gått sönder.

Referenslista

Böcker

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

Gordon, H. (1971). *Intervjumetodik.* Stockolm: Almqvist & Wiksell.

Holme, I.M. & Solvang B. K. (1997). *Forskningsmetodik om kvalitativ och kvantitativa metode*r. Lund: Studentlitteratur.

Oskarsson, Ö. & Glass, R. (1995). *ISO 9000 i programutveckling: att konstruera kvalitetsprodukter.* Lund: Studentlitteratur.

Patel, R & Davidsson, B. (1994). *Forskningsmetdodikens grunder.* Lund: Studentlitteratur.

Perry, W.E. (2000). *Effective Methods for Software Testing.* Vancouver: Wiley

Sommerville, I. (2001). *Software Engineering.* Harlowe: Addison-Wesley.

Söderfeldt, B (1972). *Statsvetenskapliga metoder.* Stockholm: Almqvist & Wiksell.

Tognazzini, B. (1996). *Tog on Software Design.* Reading: Addison-Wesley.

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

Artiklar

Aurum, A. Petersson, H. & Wohlin, C. (2001). State-of-the-art: software inspections after 25 years. *Software Testing, Verification and Reliability, 12,* online.

Miller, J. (1999). Estimating the number of remaining defects after inspection. *Software Testing, Verification and Reliability, 9,* 167-189.

Woodward, M. (2001). Editorial: Putting specifications to the test. *Software Testing, Verification and Reliability, 11,* 141-142.

Roth, A. (2001). How Good Is this Software? *The Software Testing & Quality Engineering Magazine, 3,* online.

Marick, B. (1999a). Maybe Testers Shuoldn't Be Involved Quite So Early. *The Software Testing and Quality Engineering Magazine, vol 3,* no.3.

Wiegers, K.E. (2001) Requirements When the Field Isn't Green. *The Software Testing and Quality Engineering Magazine, vol 1,* no.3.

Rapporter

Marick, B. (1997). *Classic testing mistakes.* Paper presented at *STAR* '97.

Marick, B. (1998). *When Should a Test Be Automated?* Paper presented at *Quality Week* '98.

Marick, B. (1999b). *New Models for Test Development.* Paper presented at *Quality Week* '99.

Marick, B. Bach, J. & Kaner, C. (2000). *A Manager's guide to Evaluating Test Suites.* Paper presented at *Quality Week* 2000.

Internet

Gates, B. (2002). *Trustworthy Computing.*
http://www.informationweek.com/story/IWK20020118S0093

Software QA and Testing Frequently-Asked-Questions, Part 1
http://www.softwareqatest.com/

Software QA and Testing Frequently-Asked-Questions, Part 2
http://www.softwareqatest.com/

Hulme, G. V. (2002). *It's time for developers to think and act differently.*
http://www.informationweek.com/story/IWK20020120S0003

Cleanroom Development Engineering
http://www.cleansoft.com

Cleanroom Development Engineering,
http://www.sei.cmu.edu/activities/str/descriptions/cleanroom_body.html

CMM
http://www.sei.cmu.edu/cmm/cmms/transition.html