Handelshögskolan vid Göteborgs universitet
Institutionen för informatik

Versionshantering
med inriktning på Harvest på IFS AB

Författare:
Lovisa André
Kristina Falkenström

Examensarbete I
Vårterminen 1999

Handledare: Wera Tegner Johansson

Sammanfattning

På IFS AB utvecklas ett affärssystem som är uppbyggt av en stor mängd filer. Versionshanteringsverktyget CCC/Harvest används för att hålla reda på vilka filversioner som finns och även för att göra det möjligt att återskapa en produktversion hos kund om så skulle behövas. Utifrån det material som vi har samlat in under ett antal intervjuer med anställda på IFS har vi studerat arbetssätten i två större projekt. Arbetssätten har sedan jämförts med de nyligen framtagna riktlinjerna för användningen av Harvest på IFS-kontoren i Sverige. I vår studie har det framkommit att det är av mycket stor vikt att en gemensam arbetsmetodik används vid versionshantering och att den följs av samtliga systemutvecklare för att undvika de problem som annars kan uppstå. De nya riktlinjerna behöver dock kompletteras med en utförlig arbetsbeskrivning för att nå sitt syfte och i vår uppsats ger vi förslag på vad som bör beaktas vid denna komplettering.

Innehållsförteckning

1 Inledning 5
1.1 Bakgrund 5
1.2 Syfte 5
1.3 Målgrupp 5
1.4 Problem 6
1.5 Avgränsning 6
1.6 Metod 6
1.7 Disposition 7
1.8 Språkbruk 7

IFS AB 8
1.9 IFS Applications 8
1.10 Arbetssätt 9

2 Versionshantering 10

3 Harvest på IFS 12
3.1 QDS 12
3.2 Repository 12
3.3 Environment (miljö) 12
3.4 Livscykel och states 12
3.5 Transitions 13
3.6 Processer 13
3.7 Paket och paketgrupper 13
3.8 Användargrupper 14
3.9 Harvest Workbench 14

4 Nya riktlinjer för användningen av Harvest 16
4.1 Språkbruk 16
4.2 Arbetsmetodik i Harvest 16
4.3 Begreppet States 17
4.3.1 ToDo 17
4.3.2 Implementation 17
4.3.3 Build & Test 17
4.3.4 Release 17
4.3.5 Release Archive 17
4.4 Flödesbeskrivning för Nya riktlinjer 18
4.5 Namnstandard 18
4.5.1 Environment 18
4.5.2 Standardpaket 19
4.5.3 Paket för projektdokument 19
4.5.4 Paket för anpassningar 19
4.5.5 Buggar och patchar 19
4.5.5.1 Standardbuggar, rapporterade, rättade av PDO 20
4.5.5.2 Standardbuggar, registrerade, rättade av PDO 20
4.5.5.3 Buggar i STD, rättade av R&D 20
4.5.5.4 Component patches 20
4.5.5.5 Borttag av buggar och patchar 20
4.5.5.6 Buggar i kundanpassningar 20

4.5.6 QDS-klassade dokument 21
4.6 Bugg- och patchhantering 21
4.7 Katalogstrukturer 21

5 Resultatbeskrivning 22
5.1 Projekt Kronans Droghandel (KD) 22
5.1.1 Presentation av företag och projekt 22
5.1.2 Arbetsmetodik i Harvest 23
5.1.3 States 24
5.1.3.1 ToDo 24
5.1.3.2 Implementation 24
5.1.3.3 Build & Test 24
5.1.3.4 Release 24
5.1.3.5 Release Archive 24
5.1.4 Flödesbeskrivning för KD-projektet 25
5.1.5 Namnstandard 25
5.1.5.1 Environment 25
5.1.5.2 Standardpaket 26
5.1.5.3 Paket för projektdokument 26
5.1.5.4 Paket för anpassningar 26
5.1.5.5 Buggar och patchar 26
5.1.5.6 QDS-klassade dokument 27
5.1.6 Bugg- och patchhantering 27
5.1.7 Katalogstruktur 28
5.1.8 Problem/kommentarer 28
5.2 Projekt Stora Hylte (Hylte) 29
5.2.1 Presentation av företag och projekt 29
5.2.2 Arbetsmetodik i Harvest 29
5.2.3 States 30
5.2.3.1 ToDo 30
5.2.3.2 Implementation 30
5.2.3.3 Build & Test 30
5.2.3.4 Release 30
5.2.3.5 Release Archive 30
5.2.4 Flödesbeskrivning för Hylte-projektet 31
5.2.5 Namnstandard 31
5.2.5.1 Environments 31
5.2.5.2 Standardpaket 32
5.2.5.3 Paket för projektdokument 32
5.2.5.4 Paket för anpassningar 32
5.2.5.5 Buggar och patchar 32
5.2.5.6 QDS-klassade dokument 33
5.2.6 Bugg- och patchhantering 33
5.2.7 Katalogstrukturer 33
5.2.8 Problem/kommentarer 34
5.3 Sammanfattning 34

6 Diskussion 35
6.1 Överblick 35
6.2 Problem 35
6.3 Slutsats 36

6.4 Synpunkter från användarna....38

7 Allmän diskussion....39
7.1 Metodutvärdering....39
7.2 Efterkommande forskning....39
7.3 Slutord....40

8 Referenslista....41
8.1 Litteratur....41
8.2 Material från Internet....41
8.3 Opublicerat material: C-uppsats....41
8.4 Kontaktpersoner på IFS i Göteborg....41

9 Bilagor....42

Figurer

Figur 3:1 Backward delta....11
Figur 3:2 Forward delta....12
Figur 3:3 Branch och merge....12
Figur 4:1 CCC/Harvest Workbench....16
Figur 5:1 Flödesbeskrivning för de nya riktlinjerna....19
Figur 6:1 Flödesbeskrivning för KD-projektet....26
Figur 6:2 Flödesbeskrivning för Hylte-projektet....32

1 Inledning

1.1 Bakgrund

IFS AB[1] är ett internationellt företag som utvecklar och säljer affärssystem. IFS:s produkt heter IFS Applications och är ett komplett affärssystem uppdelat på olika moduler. Dessa moduler kan kombineras på olika sätt för att passa kundens verksamhet och önskemål. Exempel på moduler är Lön, Tidrapportering, Inköp, Lager, Redovisning och Anläggning.

Modulerna i IFS Applications består av en mängd olika filer och, eftersom man erbjuder sina kunder specialanpassningar av IFS Applications, behöver systemutvecklarna på IFS ett system för versionshantering. Det är också en del av IFS:s affärsidé att kunna erbjuda kunderna en garanti för att man verkligen håller reda på kundens version av applikationen och att man snabbt kan återskapa denna om så skulle behövas.

På IFS används sedan i december 1997 verktyget CCC/Harvest[2] för att hantera olika filversioner. För att underlätta och effektivisera arbetet i Harvest är det nödvändigt att ha en gemensam arbetsmetodik att följa. På IFS har det hittills inte funnits en sådan, utan Harvest har använts på olika sätt i olika projektgrupper.

Under våren 1999 utvecklas nya riktlinjer för hur arbetet i Harvest ska ske på leveransavdelningen på IFS. Med anledning av det, och med tanke på det faktum att vi inte har kommit i kontakt med begreppet versionshantering under utbildningen på ADB-programmet, var det motiverat att välja versionshantering på IFS som ämne för vår uppsats. Vid stora projekt, där många systemutvecklare, kunder och filer är inblandade, är det av stor vikt att man använder sig av versionshantering, bland annat för att arbetet ska kunna bedrivas så effektivt som möjligt.

1.2 Syfte

Syftet med examensarbetet är att jämföra de olika sätt som Harvest har använts på i två större projekt på leveransavdelningen på IFS AB, samt att se hur dessa metoder skiljer sig från den arbetsmetodik som har utvecklats för användningen av Harvest på IFS i Sverige. Genom att studera de nya riktlinjerna och de båda projekten i detalj kan vi se huruvida arbetet i respektive projektgrupp är strukturerat eller inte. Vi vill också studera några av de problem som har uppstått när man inte har använt sig av samma arbetsmetod, vad som har orsakat problemen och hur de kan undvikas i den fortsatta användningen av Harvest.

1.3 Målgrupp

Uppsatsen vänder sig först och främst till anställda på IFS som använder sig av Harvest. Även för personer som är allmänt intresserade av problematiken kring

[1] Industrial & Financial Systems
[2] Verktyget benämns fortsättningsvis Harvest.

versionshantering i samband med systemutveckling, är uppsatsen beaktningsvärd, trots att den är en studie i hur ett utvalt verktyg tillämpas i samband med versionshantering på ett specifikt företag. Att ha ett praktikfall som exempel gör att begreppen konkretiseras, blir mer lättförståeliga och också överförbara till andra verktyg och programtillverkande företag.

1.4 Problem

På leveransavdelningen på IFS har en gemensam arbetsmetodik för användningen av Harvest hittills inte funnits, vilket har fått som följd att olika rutiner har använts i olika projekt, och även inom projekt. På grund av att man inte har arbetat på samma sätt har det uppstått problem vid användandet av Harvest. När Harvest används är det viktigt att arbetet sker utifrån gemensamma rutiner. Anledningen är att arbetet då kan göras mer effektivt och samtidigt kan användningen göras mer lättförståelig för nyanställda.

De frågor vi vill besvara i vår uppsats är följande:

- Hur skiljer sig arbetssättet i de två olika projekten från det förslag på riktlinjer som nyligen har utarbetats?
- Vilka problem har uppstått på grund av olika arbetsmetoder inom en projektgrupp?
- Hur kan dessa problem undvikas i framtiden?

1.5 Avgränsning

Vi kommer inte att göra någon utvärdering av Harvest som verktyg för versionhantering. Inte heller kommer vi att utarbeta ett nytt förslag på hur en gemensam arbetsmetodik bör se ut.

1.6 Metod

Vi har valt att redovisa de nya riktlinjer som under våren 1999 har utarbetats samt att studera arbetsmetodiken i två pågående projekt på leveransavdelningen för kunderna Kronans Droghandel (KD-projektet) och Stora Hylte AB (Hylte-projektet).

De två utvalda projektens arbetssätt i Harvest sammanställs och jämförs därefter med de nya riktlinjerna. Vi analyserar hur arbetet hittills har fungerat och vill med utgångspunkt från det klargöra vilka problem som uppstått och peka på orsakerna till dem. Resultatet blir ett förslag på vilka aspekter som bör beaktas i arbetet med att eliminera andelen problem vid användningen av Harvest.

Då vår uppsats främst grundar sig på en empirisk studie i användningen av Harvest på IFS, har resultatet från utbildningstillfällen och intervjuer varit den stora källan till material i uppsatsen. I viss mån studerade vi även litteratur som behandlar ämnet versionshantering.

För att inhämta kunskap om Harvest tog vi del av det utbildningsmaterial som finns tillgängligt på IFS. Vi deltog också vid ett par utbildningstillfällen, där en av de

Harvest-ansvariga på IFS demonstrerade versionshantering och användning av Harvest. Dessutom fick vi ytterligare en genomgång av Harvest, där vi hade möjlighet att ställa frågor som berörde innehållet i vår uppsats. Via en kontakt på IFS:s kontor i Malmö fick vi tillgång till de nya riktlinjerna.

Insamlandet av material för studier av projektgruppernas arbetssätt skedde genom ett fåtal intervjuer enligt kvalitativ metod. Intervjuerna gjordes med lämpliga representanter från IFS: två systemutvecklare från vart och ett av de projekt som ingår i studien, sammanlagt fyra personer. Mot slutet av arbetet med uppsatsen bidrog ytterligare personer med information. Urvalet av intervjupersoner skedde genom rekommendationer från vår handledare på IFS. Vi valde att göra längre och mer ingående intervjuer för att få en så omfattande beskrivning av arbetssätten som möjligt. Ett antal intervjufrågor (se bilaga) förbereddes med de nya riktlinjerna som utgångspunkt och sedan fördes en diskussion kring nämnda frågor tillsammans med de personer som intervjuades. Vi blev senare tvungna att ställa kompletterande frågor för att få fram vissa detaljer som inte framkom vid första intervjutillfället. Detta på grund av att vi, när vi studerade de nya riktlinjerna, fann vissa otydliga formuleringar, vilket gjorde det nödvändigt att vid ett flertal tillfällen kontakta upphovsmännen för att få förtydliganden.

1.7 Disposition

Uppsatsen inleds med en presentation av företaget IFS, dess verksamhet och dess produkt, för att placera resten av uppsatsen i det sammanhang som valts för fallstudien. Därefter följer ett allmänt avsnitt om versionshantering, och en beskrivning av Harvest och hur Harvest har anpassats till IFS:s verksamhet. Med detta som bakgrund görs en detaljerad beskrivning av hur de nya riktlinjerna för användningen av Harvest på IFS ser ut. Därpå följande avsnitt behandlar de uppgifter som vi har fått in under våra intervjuer. Resultatet är en grundlig beskrivning av arbetssätten i de båda projekt som vi har studerat. Presentationen av dessa arbetssätt sker enligt dispositionen för de nya riktlinjerna, för att tydliggöra skillnader och likheter vid en jämförelse mellan de olika projekten och IFS:s riktlinjer. Avslutningsvis för vi en diskussion om vad vi har kommit fram till. Vi diskuterar varför det är så viktigt att ha en gemensam arbetsmetodik och ger också förslag på vad man bör tänka på för att undvika de problem som kan uppstå. Till sist gör vi en utvärdering av uppsatsen – metodval och upplägg kommenteras och förslag till fortsatt forskning presenteras.

1.8 Språkbruk

Då den här uppsatsen behandlar ett ämne där engelska uttryck är vanligen förekommande, som t ex *patch*, *bugg* och *state*, och då sådana ord är allmänt accepterade på IFS AB, kommer vi att använda oss av dem i uppsatsen.

För att skilja på olika miljöer, kallar vi en kunds miljö i Harvest för ‘environment’ och använder ordet ‘miljö’ då en databasmiljö avses, t ex kundens ‘produktionsmiljö’ eller ‘testmiljö’. Samtliga nämnda begrepp är vedertagna på IFS.

IFS AB

IFS AB är ett företag som utvecklar och säljer affärssystem till mellanstora och stora företag över hela världen. Från IFS:s internationella websida har följande fakta hämtats. Företaget grundades 1983 i Linköping, där huvudkontoret än idag ligger. Företaget finns representerat i 32 länder och har 43 egna kontor runt om i världen - tio av dessa kontor finns i Sverige. IFS i Sverige sysselsätter ca 600 personer och av dessa arbetar 220 på Göteborgskontoret. 1998 hade hela IFS en nettoomsättning på 1 238 Mkr.

Göteborgskontorets organisation är uppdelat i följande avdelningar: Marknad, Sälj, Utveckling (R&D[3]) och Leverans (PDO[4]). Leveransavdelningen sköter kontakten med kunden från det att kunden beställt IFS Applications till det att alla anpassningar är gjorda och projektet är avslutat. På leveransavdelningen arbetar projektledare, gruppledare, systemutvecklare och applikationskonsulter. En gruppledare är administrativ ledare för en grupp, medan en projektledare är ansvarig för ett kundprojekt. Det förekommer att de här två rollerna innehas av en och samma person. En systemutvecklare sysslar med programmering och gör anpassningar i applikationen, medan en applikationskonsult är mer koncentrerad på användningen av själva applikationen och ger slutanvändarna utbildning och support. En systemutvecklare i projektgruppen utses till tekniker och får då ansvaret för databashanteringen.

1.9 IFS Applications

Den produkt som IFS erbjuder sina kunder heter IFS Applications och är ett komponentbaserat program som används av över 1600 företag runt om i världen. Applikationen består av ett 40-tal fristående standardmoduler som kan kombineras och installeras, i ett antal olika fördefinierade standardlösningar eller helt fritt var för sig, för att skapa individuella lösningar för företag i olika branscher. Modulerna är indelade i följande kategorier: Konstruktion, Produktion, Distribution, Underhåll, Personal och Ekonomi (se bilaga).

IFS Applications kan modifieras efter kundens önskemål och byggas på efterhand för att växa med kundens verksamhet. IFS Applications är ett av marknadens första affärssystem som är helt utvecklat med objektorienterad teknik och utvecklas ständigt för att kunna erbjuda kunderna ett väl fungerande och aktuellt affärssystem. Systemet är anpassat för att hantera övergången till år 2000 samt även valutan Euro.

IFS:s affärsidé är att genom sitt standardiserade affärssystem erbjuda sina kunder handlingsfrihet och konkurrenskraftighet, och därför gör man också många anpassningar i modulerna enligt kundens önskemål. Varje modul består av en stor mängd filer av olika slag. Eftersom systemutvecklarna, som gör både anpassningar och uppgraderingar, arbetar i projektgrupper, där flera personer samtidigt kan komma att arbeta med samma filer, är det av mycket stor vikt att ett system för versionshantering används. Detta för att undvika att en systemutvecklare sparar

[3] Research & Development
[4] Project Delivery Organization

över en annans arbete eller utgår från fel filversion, när han eller hon ska göra anpassningar. Genom att IFS använder sig av ett versionshanteringsprogram, kan man också garantera sina kunder att man håller reda på kundens version av IFS Applications och tillhörande anpassningar, så att dessa kan uppdateras på ett korrekt sätt eller återskapas snabbt om så skulle bli nödvändigt.

1.10 Arbetssätt

För varje ny beställning av IFS Applications bildas en projektgrupp bestående av projektledare, applikationskonsulter och systemutvecklare. Projektledaren tar tillsammans med kunden fram en specifikation över det system som kunden vill ha. Sedan gör systemutvecklarna anpassningar i modulerna utifrån denna specifikation. I början av ett projekt gör man också upp en tidsplan för hur produkten ska levereras till kunden. I tidsplanen anges vilka anpassningar som ska levereras och vid vilken tidpunkt det ska ske. Vanligtvis sker leveransen stegvis. För varje planerad leverans skapas ett leveranspaket.

I projekten är ofta många systemutvecklare inblandade och det är viktigt att hålla reda på vem som har programmerat vad i de anpassade filerna samt när det gjordes. Därför använder sig IFS av versionshanteringsprogrammet Harvest. Det har dock uppstått problem på grund av att systemutvecklare har arbetat på olika sätt med Harvest inom samma projektgrupp. För att komma till rätta med dessa problem, effektivisera arbetet och säkra garantin till kunderna, har man nyligen tagit fram riktlinjer för hur arbetet med Harvest ska ske på IFS:s kontor i Sverige.

2 Versionshantering

Följande beskrivning av versionshantering bygger på uppgifter hämtade från Sommerville (1997). Vid systemutveckling utvecklas en produkt som genom tillägg och ändringar kontinuerligt förses med ny funktionalitet. Flera *versioner* av produkten skapas. När uppställda mål för produkten har uppnåtts skapas en *release*. En release är en version av produkten som släpps till kund.

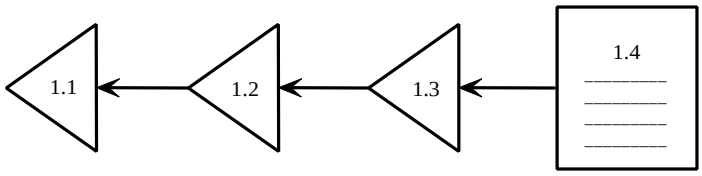
För att underlätta hanteringen av de olika versionerna och releaserna används ofta automatiserade verktyg för versionshantering. Exempel på funktioner i ett verktyg av den typen är:

- *Identifikation av releaser och versioner*. Varje version av en systemfil tilldelas en unik beteckning för identifikation, vilket gör det möjligt att identifiera filen under hela utvecklingsprocessen.

- *Kontroll av ändringar*. För att kunna göra en ändring i en fil måste den checkas ut, vilket innebär att den hämtas från en databas för systemfiler med hjälp av verktyget för versionshantering. Filen reserveras, systemutvecklarens namn och datum registreras och filen är därmed inte tillgänglig för någon annan. Det förhindrar att en fil ändras av misstag och att två systemutvecklare samtidigt ändrar i samma fil utan vetskap om varandra. När filen har ändrats och därefter lämnas tillbaka (checkas in) skapas en ny version. Även den gamla versionen finns bevarad.

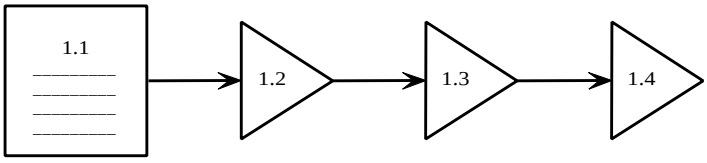
- *Effektiv lagring*. En fil lagras endast en gång i sin helhet. Övriga versioner lagras genom att skillnaderna mellan de olika versionerna sparas. Det gör att det totala lagringsutrymmet som krävs för att spara alla versioner av en fil minskar. Ändringarna kan lagras på två olika sätt:

    - Backward delta – Den senaste filversionen finns lagrad i sin helhet. Tidigare versioner nås genom att de ändringar som har gjorts mellan den sökta versionen och den senaste versionen tas bort.



Figur 3:1 Backward delta (Andersson & Bergand, 1998)

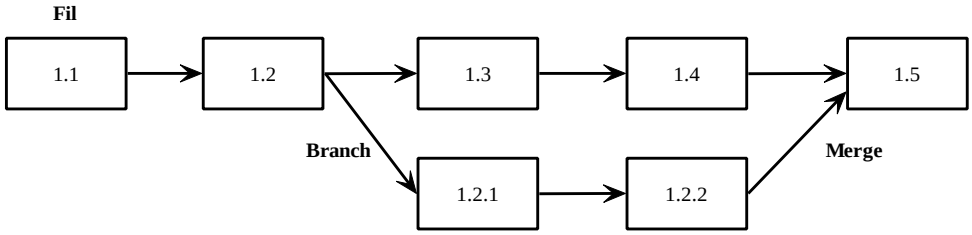
- Forward delta – Den ursprungliga filversionen finns lagrad i sin helhet. Efterkommande versioner lagras genom att ändringarna som har gjorts mellan de olika versionerna sparas. När en senare version söks, läggs ändringarna till den ursprungliga filversionen.



Figur 3:2 Foward delta (Andersson & Bergand, 1998)

- *Ändringshistorik*. Verktyget för versionshantering tillhandahåller funktioner för att lista alla ändringar som har gjorts i en fil. Olika versioner av en fil kan hämtas när så önskas. Det är också möjligt att ta ut en release som har gjorts tidigare för att på nytt installera hos kund.

Många verktyg för versionshantering har funktioner som möjliggör *parallell utveckling*. Parallell utveckling innebär att en fil kan utvecklas i två olika riktningar oberoende av varandra. Det är möjligt att vid ett senare tillfälle slå ihop de båda versionerna till en fil. I samband med parallell utveckling används begreppen *branch* och *merge.* En branch skapas då en fil vidareutvecklas vid sidan om filens egentliga utveckling. Sammanslagningen av filversioner kallas merge.



Figur3:3 Branch och merge (Andersson & Bergand, 1998)

Share innebär ytterligare ett sätt att utveckla parallellt i en fil. Vid share görs ingen kopia av filen som utvecklas parallellt. Istället finns koden lagrad på ett ställe och är tillgänglig för flera utvecklare samtidigt. Ändringarna i filen kan ses av alla systemutvecklare som arbetar mot samma version.

3 Harvest på IFS

Informationen om Harvest på IFS har vi införskaffat vid utbildningstillfällen på IFS. Harvest-ansvarige Stefan Pervik ledde kurserna samt ställde upp vid en enskild genomgång av Harvest med anledning av vår uppsats.

3.1 QDS

Harvest ingår tillsammans med IFS Online[5] i IFS:s kvalitetssystem QDS[6]. Harvest är ett windowsbaserat verktyg för versionshantering och används på IFS sedan i december 1997, då det ersatte det teckenbaserade verktyget CMS. Vissa anpassningar gjordes i Harvest för att man skulle kunna fortsätta att arbeta på ett sätt som liknar det arbetssätt som tidigare användes i CMS. Nedan följer en beskrivning av de komponenter som ingår i Harvest.

3.2 Repository

Alla kunders systemfiler i olika versioner finns lagrade i ett bibliotek, *repository*, i en databas.

3.3 Environment (miljö)

Ett environment i Harvest är en utvecklingsmiljö för ett avgränsat system och dess olika steg i en livscykel. På leveransavdelningen på IFS motsvaras ett environment av ett kundprojekt.

3.4 Livscykel och states

En systemprodukt utvecklas och anpassas kontinuerligt. Utvecklingsarbetet sker i olika steg som följer på varandra och skapar en livscykel. I Harvest motsvaras varje steg i livscykeln av ett *state.* En fil som är under utveckling genomgår de olika stegen och har för varje steg en viss status. I IFS:s Harvestmiljö finns sammanlagt fem states för olika stadier i utvecklingen:

- ToDo – Planering
- Implementation - Utveckling
- Build & Test – Testning
- Release – Klar för installation hos kund
- Release Archive – Tidigare releaser

[5] IFS Online – system för kommunikation mellan kunder och IFS
[6] QDS (Quality & Distribution System) – kvalitets- och säkerhetssystem för att hantera information om versioner, releaser, kunder och anpassningar/buggrättningar.

3.5 Transitions

Transitions, eller förflyttningar, mellan olika states i livscykeln kan ske i två riktningar:

- Promote - förflyttning framåt
- Demote - förflyttning bakåt

Det är endast möjligt att flytta ett steg i taget. När state Release har uppnåtts är det inte längre möjligt att göra en förflyttning bakåt.

3.6 Processer

I varje state finns ett antal processer definierade. Exempel på processer i state Implementation är:

- *Check out for browse*: En fil hämtas för granskning från databasen för systemfiler. Då en fil checkas ut på detta sätt är det inte möjligt att göra ändringar i den.

- *Check out for update*: En fil hämtas från databasen för systemfiler för uppdatering. Vid utcheckning av en fil för uppdatering hämtas alltid den senaste versionen från fildatabasen. Filer som checkas ut för uppdatering måste knytas till ett paket (se nedan). Filen blir samtidigt reserverad av den systemutvecklare som checkat ut den. Den är då låst för alla andra utvecklare.

- *Check in existing*: När ändringar har gjorts i en fil, checkas den in i Harvest igen. En ny version av filen skapas. Samtidigt blir den åter tillgänglig för övriga systemutvecklare.

- *Check in new*: En helt ny fil checkas in (registreras) och lagras i Harvest.

3.7 Paket och paketgrupper

Förändringar i filer kopplas till *paket* i Harvest. Ett paket skapas t ex för en ny funktion i produkten. Alla filer som berörs av förändringen kopplas till detta paket. Enstaka filer kan inte flyttas från ett state till ett annat. Endast förflyttning av hela paket är möjlig. Syftet med paket är att underlätta hanteringen av anpassningar som förändrar mer än en fil. Paket kan kopplas till *paketgrupper*. Paketgrupper fungerar som samlingspaket för flera paket, och kan t ex underlätta hanteringen av de leveranspaket som skapas under projekteringen.

3.8 Användargrupper

IFS har definierat fyra användargrupper som har olika roller vid användningen av Harvest. Användargrupperna har olika rättigheter till processerna i livscykeln. De fyra användargrupperna är:

CM Administrator	Tekniskt ansvarig med fullständiga rättigheter i Harvest. Skapar environments och releaser.
Project Administrator	Administrativ projektledare med begränsade rättigheter. Skapar paket i state ToDo.
Developer	Systemutvecklare. Rättigheter främst i state Implementation.
Approver	Godkänner paket för uppflyttning till nästa state.

3.9 Harvest Workbench

Applikationen för filhantering i Harvest kallas för Harvest Workbench. Vid uppstart av applikationen loggar användaren in med användarnamn och lösenord. Fönstret som då öppnas liknar en vanlig filhanterare, indelad i fem olika flikar. Överst i fönstret anger användaren vilken miljö och vilket state han/hon vill söka filer i. De fem flikarna visar olika *views*, vyer, av det som finns i filbiblioteket:

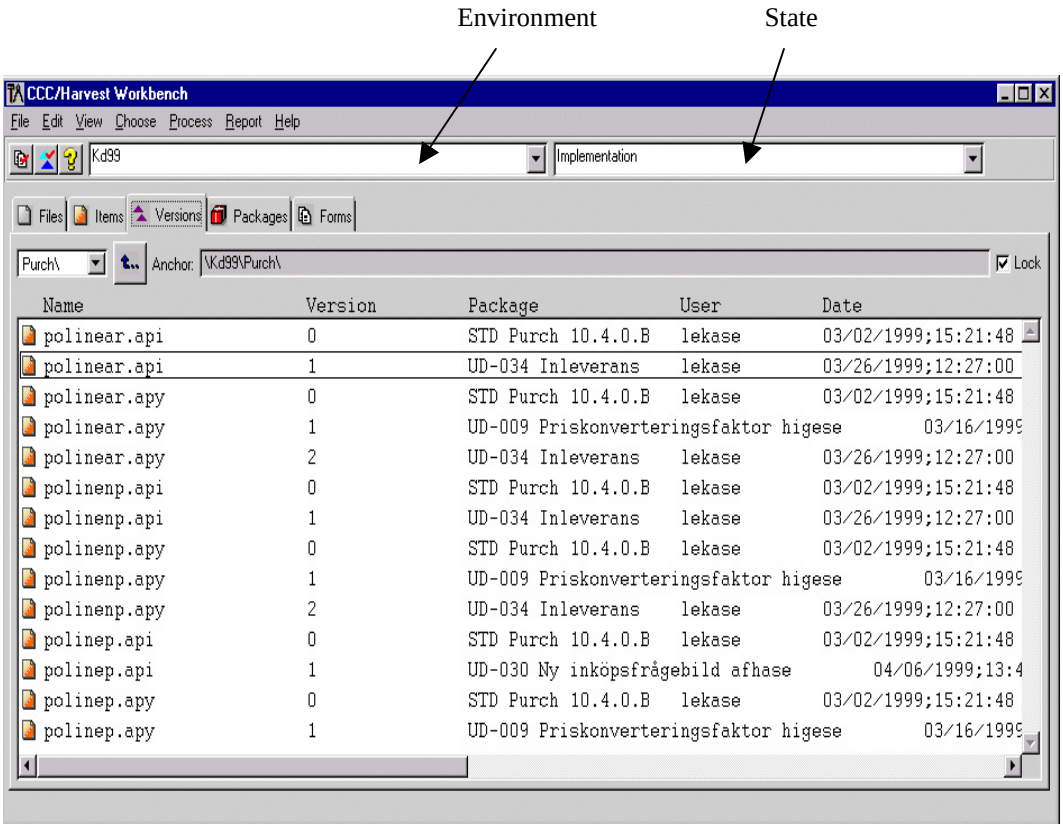
- *Files* är en filhanterare som speglar filstrukturen på disk och på utvecklingsservrar. I denna flik anges till vilken plats på disk som filer önskas checkas ut. Det sker genom att ett lås, en markering i en kryssruta, sätts när önskad sökväg har angivits.

- *Items* är en filhanterare som speglar filhierarkin på filer som finns i databasen för systemfiler i Harvest. De filer som ska hämtas ut markeras. I denna flik hämtas alltid senaste filversionen. Ett lås sätts i den position där filen hämtas. Det som finns under låset i trädstrukturen av filer följer med vid utcheckningen och placeras i den position som har angivits i fliken Files.

- I *Versions* listas alla existerande versioner av en fil. Förutom versionsnummer finns uppgifter om associerande paket och senast ändrad-datum.

- I fliken *Packages* visas de paket som finns inom valt state och environment

- *Forms* – används ej på IFS



Figur 4:1 CCC/Harvest Workbench
I fliken Versions visas samtliga filversioner av en fil. I detta exempel är miljön Kd99 i state Implementation

Efter att användaren har angett projekt och state i fönstrets överkant listas kundens filer, filversioner, releaser och paket i de olika flikarna. Användaren markerar önskad fil och väljer därefter en process i menyn, t ex *Check out for update* samt till vilket paket filen ska kopplas. Filen hamnar i den mapp på disk som har markerats med lås. När systemutvecklaren är klar med sin ändring checkas filen in i Harvest igen. En mer detaljerad beskrivning av användningen av Harvest på IFS finns i kapitlet Nya riktlinjer.

4 Nya riktlinjer för användningen av Harvest

För att underlätta arbetet, minska andelen problem och öka kvaliteten i projekten har man nyligen tagit fram riktlinjer för hur användningen av Harvest ska ske på IFS-kontoren i Sverige. Nedan följer en beskrivning av dessa riktlinjer (Löfgren 1999):

4.1 Språkbruk

Beroende på kundens önskemål används svenska eller engelska, förslagsvis samma språk som används i avtal och PQD[7]. Vedertagna engelska facktermer som t ex bugg och patch är tillåtna även då svenska språket används.

4.2 Arbetsmetodik i Harvest

Arbetet i Harvest börjar med att teknikern skapar ett environment för aktuell kund. Sedan skapar den som är projektansvarig paket och ev. paketgrupper för kommande anpassningar och leveranser. Det görs lämpligen med utgångspunkt från den plan som gjorts för inplanerade leveranspaket till kunden. Paketen skapas i state ToDo, där de sedan ligger kvar fram till att det är dags att börja specificera och programmera de planerade anpassningarna. Paketet flyttas då till state Implementation.

Filer som ska anpassas hämtas från state Release, vare sig det gäller standardversionen av filen eller redan anpassade filer. Filerna checkas in mot det för anpassningen avsedda paketet i state Implementation, där all systemutveckling sedan sker. När systemutvecklaren anser sig vara färdig med en anpassning, testar han/hon först själv att den fungerar enligt specifikationen. Paketansvarig[8] godkänner anpassningen och flyttar paketet till state Build & Test.

I Build & Test görs en installation i en testmiljö och därefter följer en större systemtest av de nya funktioner som har utvecklats i Implementation. Om paketet inte uppfyller alla krav som finns i specifikationen, flyttas paketet tillbaka till Implementation för rättning. När paketet har godkänts för installation i kundens produktionsmiljö flyttas det till state Release.

Alla paket som är godkända för leverans ligger förvarade i state Release. När en release skapas, kommer alla filversioner som är kopplade till ett paket att ingå. Vid en release gör Harvest en ‘ögonblicksbild’ av filerna i de paket som ligger i state Release när releasen skapas.

State Release Archive innehåller alla de releaser som tidigare har skapats. Det är möjligt att när som helst plocka ut dessa för att åter leverera till kund.

[7] Projekt Quality Documents - standardiserade dokument för projektbeskrivning
[8] Kan vara projektledaren, systemutvecklaren eller teknikern

4.3 Begreppet States

IFS Life Cycle Model består av fem states: ToDo, Implementation, Build & Test, Release och Release Archive.

4.3.1 ToDo

Här skapar projektledaren paket för anpassningar som ska göras i projektet. När anpassningen ska påbörjas flyttas paketet till nästa state, Implementation.

4.3.2 Implementation

I detta state görs specifikationen och programmeringen. När det är klart skall anpassningen testas och godkännas för uppflyttning till Build & Test.

4.3.3 Build & Test

I Build & Test plockas filer ut för en större systemtest. Om paketet inte uppfyller funktionsspecifikationen skickas det tillbaka till Implementation. Godkänns paketet för installation skickas det upp till Release.

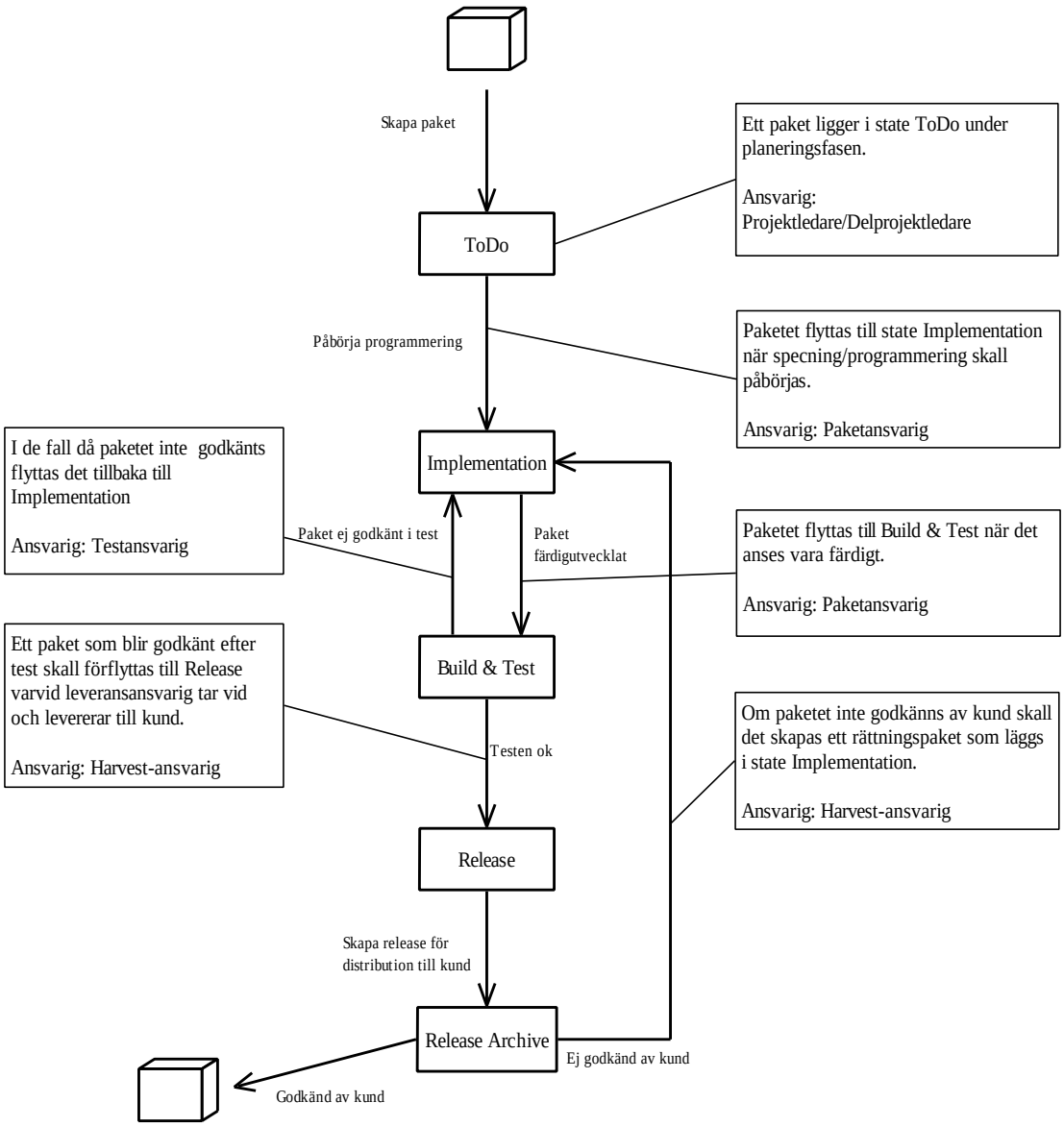
4.3.4 Release

I detta state finns de paket som har godkänts för installation hos kund. Vid en release ingår alla aktuella filversioner som är kopplade till paketet.

4.3.5 Release Archive

Release Archive innehåller alla de releaser som har skapats och levererats.

4.4 Flödesbeskrivning för Nya riktlinjer



Figur 5:1 Flödesbeskrivning för de nya riktlinjerna

4.5 Namnstandard

I riktlinjerna ingår även en rekommendation för hur namngivning av Harvest-relaterade element bör ske:

4.5.1 Environment

En produktionsmiljö ska motsvaras av ett environment i Harvest. En produktionsmiljö består av de filer som en kund använder sig av i sitt datasystem i sin verksamhet. Miljön i Harvest namnges med kundens namn och eventuellt projektnamn. Skulle projektnamn saknas kan årtal för

projektstart läggas till. Namnet på miljön inleds med en versal och efterföljande bokstäver ska vara gemener. Blanktecken och specialtecken är inte tillåtna.

Exempel: Kd, Kd99, Hylte

4.5.2 Standardpaket

När en anpassning ska göras för första gången i en fil, hämtas standardversionen av filen ut från standardmodulen i Harvest och checkas in i projektets Harvestmiljö mot ett standardpaket. Paketet namnges med namnet på den modul som filen tillhör samt modulens version: STD<Modul><Version>. Därefter checkas filen ut mot ett anpassningspaket varvid den reserveras och anpassningen kan påbörjas. Standardpaket kopplas inte till någon paketgrupp, eftersom de inte kommer att ingå i någon release. Denna typ av paket skapas enbart för att fungera som en referens till den version av en fil/modul som är utgångspunkt för en anpassning.

Exempel: STD Purch 10.4.0

4.5.3 Paket för projektdokument

PQD:er och installationsrapporter ska placeras i ett paket som namnges efter miljön eller projektet det tillhör. Paketet placeras i katalogen Projdoc.

Exempel: Projektdokument för Hylte

Specifikationer för anpassningar (funktionsspecifikation) placeras under aktuell modulkatalog och paketspecifikationer under aktuellt paket.

4.5.4 Paket för anpassningar

För varje anpassning skapas ett paket i state ToDo eller Implementation.
Paketet namnges med en beskrivning av anpassningen och namnet på leveranspaketet: <Lev.paketnamn:> Anpassning(funktionsspecifikation) och paketgruppen: <Paketleverans i PQD>. I paketets noteringsfält beskrivs sedan anpassningen utförligt och uppgifter om specifikation, utvecklare och ansvarig projektledare anges.

Exempel:

Paket:	Lev.paket 1: Tvingande förkontering inköpsorder
Paketgrupp:	Installation 980530

4.5.5 Buggar och patchar

För buggar[9] i standardmodulerna (standardbuggar) ska följande paket och paketgrupper användas. De skapas i Implementation. Buggen ska beskrivas i noteringsfältet för paket.

[9] Felaktig/bristande funktionalitet

4.5.5.1 Standardbuggar, rapporterade[10], rättade av PDO[11]

Exempel: Paket = STD<modul>CALLID<callid[12]>
Paketgrupp = STD-BUGGAR<fr o m datum - t o m datum>

4.5.5.2 Standardbuggar, registrerade[13], rättade av PDO

Exempel: Paket = STD<modul>BUGGID<buggid[14]>
Paketgrupp = STD-BUGGAR<fr o m datum - t o m datum>

4.5.5.3 Buggar i STD, rättade av R&D

Exempel: Paket = STD<modul>BUGGID<buggid>
Paketgrupp = STD-BUGGAR<fr o m datum - t o m datum>

4.5.5.4 Component patches

Eftersom R&D tar ansvar för innehållet i patchar[15], hålls dessa separerade från övriga buggrättningar. Textfil med R&D info om patchar ska också checkas in i Harvest.

Exempel: Paket = STD<modul>PATCH<patchid[16]>
Paketgrupp = STD-PATCHAR<fr o m datum - t o m datum>

4.5.5.5 Borttag av buggar[17] och patchar

När buggar och patchar ska tas bort från filer[18] skapas paket som namnges enligt följande standard:

Paket: Borttag STD modul PATCH <patchid>
Borttag STD modul BUGGID <buggid>
Borttag STD modul CALLID <callid>

För paketen skapas en egen paketgrupp med namnet

Paketgrupp: Borttag STD-rättningar (BUGGID, PATCH, CALLID)

4.5.5.6 Buggar i kundanpassningar

Buggrättningspaket namnges lämpligen med buggid och en beskrivning av rättningen för att man lättare ska kunna se vad som har åtgärdats.

---

[10] Bugg som har rapporterats av kund, men som inte har registrerats av IFS
[11] Project Delivery Organization - Leveransavdelning
[12] Unikt löpnummer för rapporterade buggar
[13] Bugg som har registrerats av IFS
[14] Unikt löpnummer för registrerade buggar
[15] Paket med buggrättningar i standard från R&D som baseras på huvud- eller servicerelease
[16] Unikt löpnummer för patchar
[17] I riktlinjerna avses här buggrättningar
[18] Detta sker bl a vid uppgradering till en servicerelease, se kapitel 5.6 Bugg- och patchhantering

Exempel:
Paket: "Lev.paket nr": <Kort beskrivning>
Paketgrupp: KUNDBUGGAR <fr o m datum - t o m datum>

4.5.6 QDS-klassade dokument

QDS[19]-klassade dokument ska finnas i respektive kunds Harvestmiljö. All dokumentation som är nödvändig för att projektet ska kunna slutföras ska enligt rekommendation lagras i Harvest, t ex funktionsspecifikationer och specifikationer från kunden. Dokumentfilerna ska följa vedertagen dokumentstandard och lagras med extensionen '.doc' i katalogen Projdoc.

Exempel:

PQD	<kund>_PQD.doc
Installationsrapport	<kund>_InstRep.doc
Paketspecifikation	<kund>_<paketnamn>_spec.doc
Projekteringsrapport	<kund>_ProjRep.doc
Testprotokoll Databas	<kund>_TestServer.doc
Testprotokoll Windows	<kund>_TestClient.doc

4.6 Bugg- och patchhantering

Buggar och patchar från R&D läggs in i kundens environment i Harvest enligt paketstandarden som beskrivs i kapitel 5.5.5. Filerna med rättningar från R&D läggs in under respektive modul och hanteras därefter som vanliga kundanpassningar. Filerna kan ligga till grund för nya kundanpassningar. Om filerna har kundanpassats, ska anpassningarna tas bort, dvs markeras som icke aktuella (set item obsolete), då en installation av produkten uppgraderas till en servicerelease[20]. Vid uppgradering till en ny standardversion, flyttas eventuella kundanpassningar i rättningsfilerna med till den nyinstallerade versionen.

4.7 Katalogstrukturer

Katalogstrukturen i en kunds environment ska se ut enligt följande.

Exempel: <Kund/projekt>
<modulnamn>
<modulnamn
Applfiles (innehåller kund-app:ar[21], språkfiler o dyl)
Projdoc (ska innehålla samtliga QDS-klassade dokument)

[19] Quality & Distribution System
[20] En servicerelease är en release från R&D som kan innehålla buggrättningar och patchar, men som ändå inte är en ny version av produkten.
[21] Används som utgångsfil vid skapandet av en exe-fil

5 Resultatbeskrivning

I detta kapitel redogörs för resultatet från genomförda intervjuer med personer från de två utvalda projekt som ingår i studien. Projekten presenteras vart och ett för sig, enligt uppställningen i föregående kapitel om de nya riktlinjerna.

5.1 Projekt Kronans Droghandel (KD)

För att få fram den information om KD-projektets arbetssätt som här presenteras, intervjuades systemutvecklarna Johan Planmo (även kvalitetsansvarig) och Lovisa Tholerus. I slutskedet av arbetet med uppsatsen har även Fredric Travaglia, systemutvecklare och tekniker, bistått med uppgifter om projektet. Nedan följer en sammanställning av deras uppfattning om arbetet med Harvest i projektet.

5.1.1 Presentation av företag och projekt

Kronans Droghandel är en tredjeparts-logistiker som arbetar med att distribuera läkemedel och sjukvårdsmaterial till sjukhus och apotek. IFS arbetar sedan två år tillbaka med att införa systemen IFS/Ekonomi och IFS/Distribution i KD:s verksamhet. Det har gjorts många anpassningar i standardsystemen, bl a har en ny modul för EDI[22] och generella gränssnitt skapats och en för att föra över statistikdata till KD:s eget statistiksystem.

KD-projektet är IFS:s största projekt och som mest har över tjugo personer från IFS varit inblandade i arbetet. När projektet startade användes CMS som verktyg för versionshantering, men sedan början av 1998 används istället Harvest. Rutinerna från användningen av CMS överfördes i möjligaste mån till Harvest. I början var det svårt att anpassa dem till det nya verktyget, men med tiden växte ett lämpligt arbetssätt fram. Idag finns riktlinjer för allt arbete i projektet. I riktlinjerna ingår tydliga instruktioner för hur arbetet i Harvest ska ske.

Sedan en tid tillbaka håller projektgruppen på att uppgradera KD:s produktionsmiljö[23] och i samband med det har ett nytt environment i Harvest skapats, nämligen Kd99. Ambitionen är att environment Kd99 ska fungera helt enligt projektgruppens gemensamma riktlinjer. Det har medfört att detta environment är både enhetligt och strukturerat. Med anledning av det, samt det faktum att Kd99 inom kort kommer att bli det environment som används för kundens produktionsmiljö, koncentreras beskrivningen nedan på just detta environment.

[22] Electronic Data Interchange
[23] Arbetet med uppgraderingen kallas ”Uppgraderingsprojektet”

5.1.2 Arbetsmetodik i Harvest

För tillfället arbetar man i två environment i KD-projektet: Kd och Kd99. Kd används idag för filer i kundens nuvarande produktionsmiljö, medan Kd99 används i uppgraderingsprojektet. När uppgraderingen tas i drift i maj 1999 övergår man till att använda bara Kd99.

Det finns tre paketansvariga i gruppen som håller reda på vilka paket som finns. Från början var det endast de paketansvariga som fick skapa paket i Harvest. Numera, när de inblandade har tydliga riktlinjer för hur arbetet ska ske, skapar även systemutvecklare paket.

State ToDo används inte i environment Kd99. Systemutvecklingen börjar i state Implementation med att ett paket för varje anpassning skapas. Paketet ligger sedan kvar i Implementation under hela utvecklings- och testfasen. State Build & Test används alltså inte. Testningen utförs av programmeraren och godkänns av den som är paketansvarig innan paketet flyttas till state Release. En release installeras först i kundens testmiljö. Installation i produktionsmiljön sker inte förrän den nya funktionaliteten har testats och godkänts av kunden.

I KD-projektet används paketgrupper: för varje planerad leverans till kund skapas en paketgrupp. Paket kopplas sedan till aktuell paketgrupp. Det medför att det blir lättare för teknikern att få en överblick över vilka paket och filer som ingår i en viss leverans.

För att hålla reda på vilka anpassningar som har gjorts, används i uppgraderingsprojektet ett dokument som kallas för Uppgraderings-anpassningar.doc. I dokumentet finns uppgifter om vem som har gjort en anpassning, vilka filer som har ändrats och när det har skett. Anpassningen beskrivs kortfattat och blir tilldelad ett unikt nummer. Numret används sedan i kommentarer i programkoden och vid namngivning av paket.

Parallell utveckling undviks för det mesta genom att ytterligare anpassningar i en fil, som redan är reserverad, skjuts upp tills den åter har checkats in i Harvest. Eftersom systemutvecklarna i KD-projektet ofta arbetar med olika anpassningar i ett fåtal centrala filer, är det dock i vissa fall ändå nödvändigt att utveckla parallellt i en fil. För att hantera det, används dokumentet Filkollisioner.doc. När en anpassning påbörjas i en redan reserverad fil, registrerar systemutvecklaren det i Filkollisioner.doc och står därmed på tur att reservera filen. När filen checkas in i Harvest av den som först reserverade den, stryks anmärkningen i dokumentet. Systemutvecklaren som utvecklade parallellt i filen reserverar den aktuella filen och lägger manuellt in sina anpassningar.

5.1.3 States

5.1.3.1 ToDo

State ToDo används inte i KD-projektet.

5.1.3.2 Implementation

All systemutveckling sker i Implementation. För varje ny anpassning skapas ett Harvestpaket av systemutvecklaren. Paketet ligger sedan kvar i Implementation under både utvecklings- och testfasen. Paketansvarig godkänner ändringarna som systemutvecklaren har gjort och paketet flyttas därefter till state Release.

5.1.3.3 Build & Test

State Build & Test används inte i KD-projektet. Paket flyttas från Implementation till Release via Build & Test, men ingenting sker i detta state.

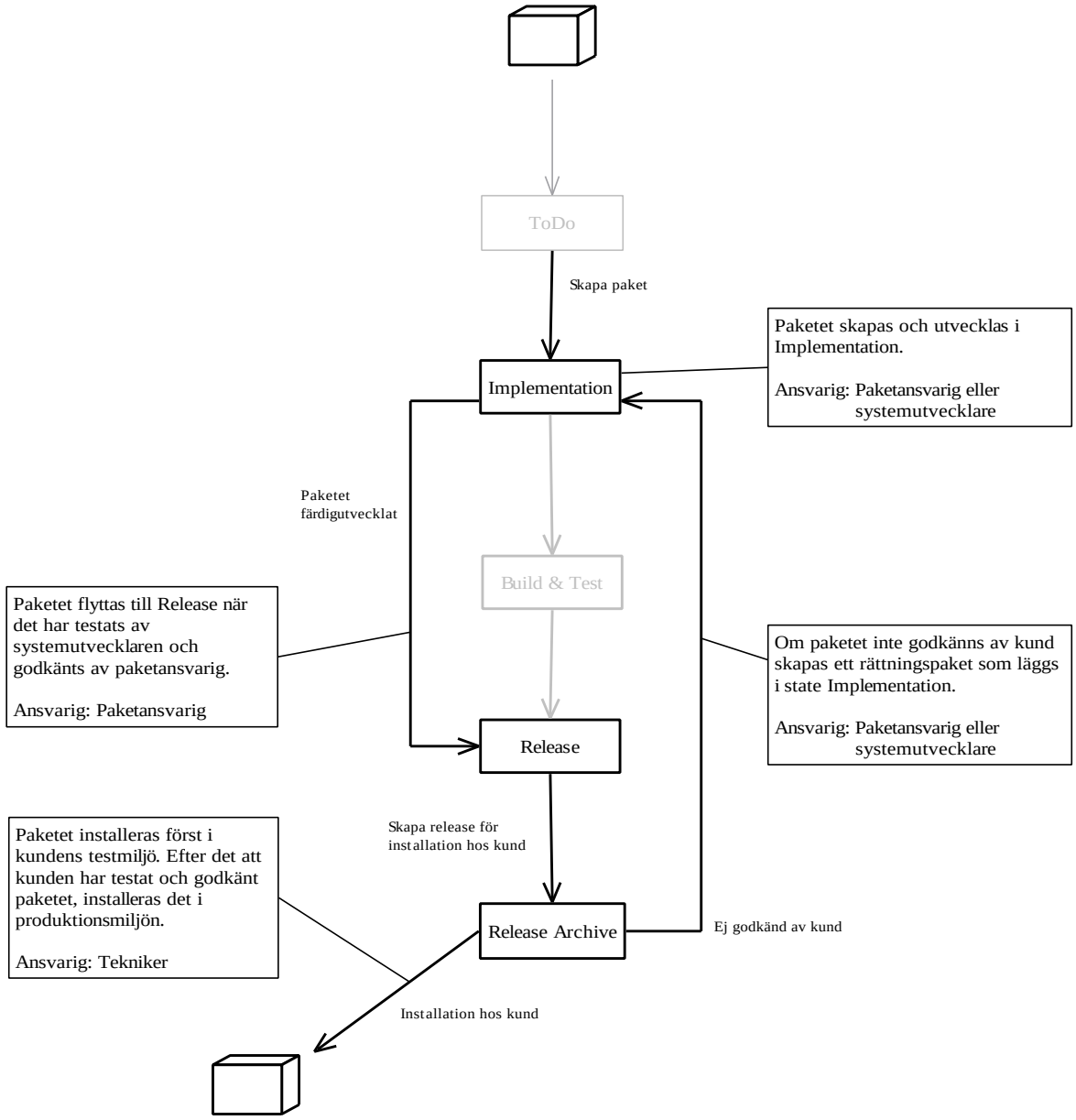
5.1.3.4 Release

I state Release ligger de paket som är godkända för installation hos kund.

5.1.3.5 Release Archive

När det är dags för installation hos kund, skapas en release med utgångspunkt från de paketgrupper och paket som finns i state Release. Först skapas en release för installation i kundens testmiljö. Kunden testar och godkänner den nya funktionaliteten. Därefter installeras releasen i kundens produktionsmiljö.

5.1.4 Flödesbeskrivning för KD-projektet



Figur 6:1 Flödesbeskrivning för KD-projektet

5.1.5 Namnstandard

KD-projektets nya environment har skapats med utgångspunkt från de riktlinjer som används för arbetet i projektet. I dessa ingår tydliga instruktioner för hur namngivning ska ske:

5.1.5.1 Environment

I KD-projektet finns förutom Kd99 ytterligare ett environment för KD i Sverige, vilket har namngivits enbart med projektnamnet. Dessutom finns

två environment för KD:s verksamhet i Finland och Norge. De har namngivits med projektnamn och anvisning om vilket land som avses.

Exempel: Kd, Kd99, KdFi, KdNo

Fortsättningsvis behandlas bara Kd99.

5.1.5.2 Standardpaket

Standardpaket namnges med STD, modulnamn samt versionsnummer.

Exempel:
Paket STD Finrep 8.1.1.A

5.1.5.3 Paket för projektdokument

Projektdokument finns inte lagrade i Harvest, utan ligger sparade tillsammans med annan information på IFS:s kundserver.

5.1.5.4 Paket för anpassningar

För varje ny anpassning skapas ett paket. I paketets namn anges om paketet gäller ekonomi- eller distributionsdelen i projektet: UD för Distribution, UF för Finance (ekonomi). U:et anger att det gäller anpassningar i uppgraderingsprojektet. Därefter följer ett löpnummer och en kort beskrivning av anpassningen.

Exempel:
Paket UD-002 Uppgradering av script
Paketgrupp Leverans 990421

5.1.5.5 Buggar och patchar

- Standardbuggar, rapporterade och registrerade

Kunden rapporterar in buggar direkt till IFS Online, vilket medför att alla buggar som rättas är registrerade.

Exempel:
Paket UD-020 Rättningspaket 1
Paketgrupp Rättningspaket 5 Leverans 990415

I de fall då buggar som ännu inte har rättats av R&D upptäcks, skapas paket som namnges med beskrivning av vilken modul som berörs. Det finns ingen standard för namngivning av paket av denna typ.

Exempel:
Paket Fixass Generellt rättningspaket 1
Paketgrupp Rättningspaket 5 Leverans 990415

- Buggar i STD, rättade av R&D

  Denna typ av buggrättningar har ännu inte förekommit i uppgraderings-projektet.

- Component patches

  När det gäller patchar från R&D skapas paket som anger vilken modul och version som berörs av patchen.

  Exempel:
  Paket STD Finrep 8.1.1.A Patch 1
  Paketgrupp STD-Patch 990301-990419

- Borttag av buggar och patchar

  Det har ännu inte varit aktuellt att ta bort buggar och patchar i uppgraderingsprojektet.

- Buggar i kundanpassningar

  Vid rättningar av buggar i kundanpassningar skapas rättningspaket. Dessa namnges med ‘Rättningspaket’ och ett löpnummer. I namnet finns också en hänvisning till vilken del av projektet anpassningarna hör: UD för Distribution, UF för Finance (ekonomi), följt av ett nummer. I dokumentet Uppgraderingsanpassningar.doc specificeras vilka anpassningsnummer som omfattas av rättningspaketet. För rättning av buggar i anpassningspaketet ‘UD-002 Uppgradering av script’ skapas ett paket som namnges enligt följande exempel:

  Exempel:
  Paket UD-002 Rättningspaket 1
  Paketgrupp Rättningspaket 5 Leverans 990415

5.1.5.6 QDS-klassade dokument

Denna typ av dokument finns inte samlade på ett gemensamt ställe och följer inte heller någon namnstandard.

5.1.6 Bugg- och patchhantering

Buggar och patchar läggs till enligt vad som beskrivs i kapitel 6.1.5.5. Det har inte varit aktuellt att uppgradera till servicereleaser i uppgraderingsprojektet. Hittills har det inte heller varit aktuellt att ta bort buggar och patchar.

5.1.7 Katalogstruktur

Under mappen Kd99 ligger mappar för samtliga moduler i systemet samt för vissa stora, avgränsade anpassningar. Mappen Specifikationer, som också ligger under Kd99, innehåller diverse dokument, dock inte alla qds-dokument.

Kd99
- Accrul
- Finrep
- Specifikationer
- Troll
- Etc.

5.1.8 Problem/kommentarer

Anledningen till att Build & Test inte används är att flera anpassningar ofta görs i ett fåtal, centrala filer. Om en fil med en ny anpassning checkas in i Harvest och sedan flyttas till Build & Test för att testas, kan en annan systemutvecklare checka ut filen för att göra ytterligare anpassningar i den. Risken är att systemutvecklaren inte upptäcker den första, ofullständiga anpassningen. Det kan resultera i att det blir svårt att spåra kompileringsfel eller att en fil som innehåller både en färdig anpassning och en halvfärdig installeras i kundens databasmiljö. I KD-projektet har problemet lösts genom att en fil flyttas från Implementation till Release, utan att state Build & Test används.

De personer som intervjuades i KD-projektet betonar följande:

En förutsättning för att parallell utveckling med hjälp av Filkollisioner.doc ska fungera är att de inblandade utvecklarna kommunicerar med varandra. Alla som arbetar med samma fil har ansvar för att informera varandra om när filen checkas in och ut etc. Dessutom är samtliga inblandade ansvariga för att se till att inga gamla poster finns kvar i Filkollisioner.doc.

För att dokumentet Uppgraderingsanpassningar.doc ska kunna fungera, måste det alltid uppdateras efter det att en anpassning är godkänd och installerad. Detta för att garantera att dokumentet innehåller korrekt information.

Kvalitetsarbete, med ständigt uppdaterade rutindokument[24], utförliga kommentarer i koden och framför allt god kommunikation mellan medarbetarna, är nödvändigt för att ett projekt av KD-projektets storlek ska kunna lyckas. Därmed kan man minimera risken för fel och misstag.

[24] Uppgraderingsanpassningar.doc och Filkollisioner.doc

5.2 Projekt Stora Hylte (Hylte)

Nedan presenteras en sammanställning av den information som framkom vid intervjuerna med systemutvecklarna Claes Håkansson och Karin Jansson i Hylte-projektet.

5.2.1 Presentation av företag och projekt

Pappersbruket Stora Hylte AB i Hyltebruk vände sig till IFS våren 1998 för att ersätta sitt gamla teckenbaserade system med IFS Applications. Företaget beställde standardversionen av modulerna Tid, Lön, Underhåll, Lager och Inköp. Projektet inleddes i juni 1998.

Projektet delades upp i två delar - Tid/Lön och Underhåll, som består av Underhåll, Lager och Inköp. Det var som mest sju till åtta personer som arbetade med Tid/Lön och idag är man två till tre personer, som sköter rättningar. Det var ingen större omsättning på personal under projektets gång. Det var större personalomsättning i Underhållsprojektet. Fem till sju personer arbetade med Underhåll, men man lånade ofta in systemutvecklare från andra projektgrupper, när arbetsbelastningen blev för stor. Idag arbetar en till två personer sporadiskt med Underhåll vid behov.

Under projektet gjordes några större anpassningar, t ex ändringar i schemahanteringen och uträkning av frånvaro, sjukfrånvaro och tjänstledighet i Tid/Lön-modulen på grund av skiftarbetet vid bruket. I modulen Inköp gjordes ändringar i kontofördelningen. För övrigt gjordes en del mindre anpassningar eftersom företaget ville att det nya systemet skulle ha några funktioner som fanns i det gamla systemet. Driftstarten skedde runt årsskiftet 1998/1999 och sedan dess har bara smärre rättningar gjorts, men på grund av dessa justeringar är projektet ännu inte avslutat.

5.2.2 Arbetsmetodik i Harvest

Hylte-projektet var det första projektet på IFS i Göteborg där Harvest användes. I början av projektet gjordes en metodik för hur man ska använda Harvest, men den har inte använts kontinuerligt. I övrigt har ingen uttalad arbetsmetodik använts, utan någon form av metodik har vuxit fram under projektets gång.

I projektet har man använt sig av följande arbetsgång:

> Ett paket skapas i Implementation av en systemutvecklare, som sedan testar anpassningen. När systemutvecklaren är nöjd skickar han/hon upp paketet till nästa state, Build & Test. Paketet testas av paketansvarig och godkänns för release och installation eller förs tillbaka till Implementation för korrigering.

Idag, när anpassningarna är gjorda och endast rättningar av buggar etc återstår, är ofta systemutvecklaren och installationsansvarig en och samma person. Det förekommer att man installerar enstaka filer istället för hela paket.

Underhålls-projektet har följt samma arbetsgång med undantag för att man ibland, på grund av brist på tid och personal som kan testa, inte tagit sig tillräckligt med tid för att testa anpassningarna ordentligt och godkänna dem före installation hos kund.

För varje anpassning skapas ett paket och flera paket grupperas sedan till ett större paket. Paketen har först installerats i kundens testmiljö, för att sedan installeras i kundens produktionsmiljö, efter att kunden har testat och godkänt anpassningarna.

5.2.3 States

5.2.3.1 ToDo

State *ToDo* används inte i någon del av Hylteprojektet.

5.2.3.2 Implementation

Utvecklingen sker i state *Implementation*. Ett paket skapas och får ungefär samma namn som anpassningen. Ofta anges vilken del av projektet paketet tillhör (Tid/Lön eller Underhåll). Specifikationen läggs i samma paket. Systemutvecklaren testar paketet, och när det anses färdigt skickas det upp till *Build & Test*.

5.2.3.3 Build & Test

När systemutvecklaren anser sig nöjd, så skickar han/hon upp anpassningen till Build & Test, där sedan någon annan än systemutvecklaren, i det här fallet projektledaren, kontrollerar att anpassningen fungerar som den ska. Sedan hämtar installationsansvarig paketet och installerar det hos kunden.

Underhållsprojektet har ibland inte använt state Build & Test för att testa anpassningar på grund av brist på tid och personer som kan testa. Man har dock skickat upp paketen till detta state.

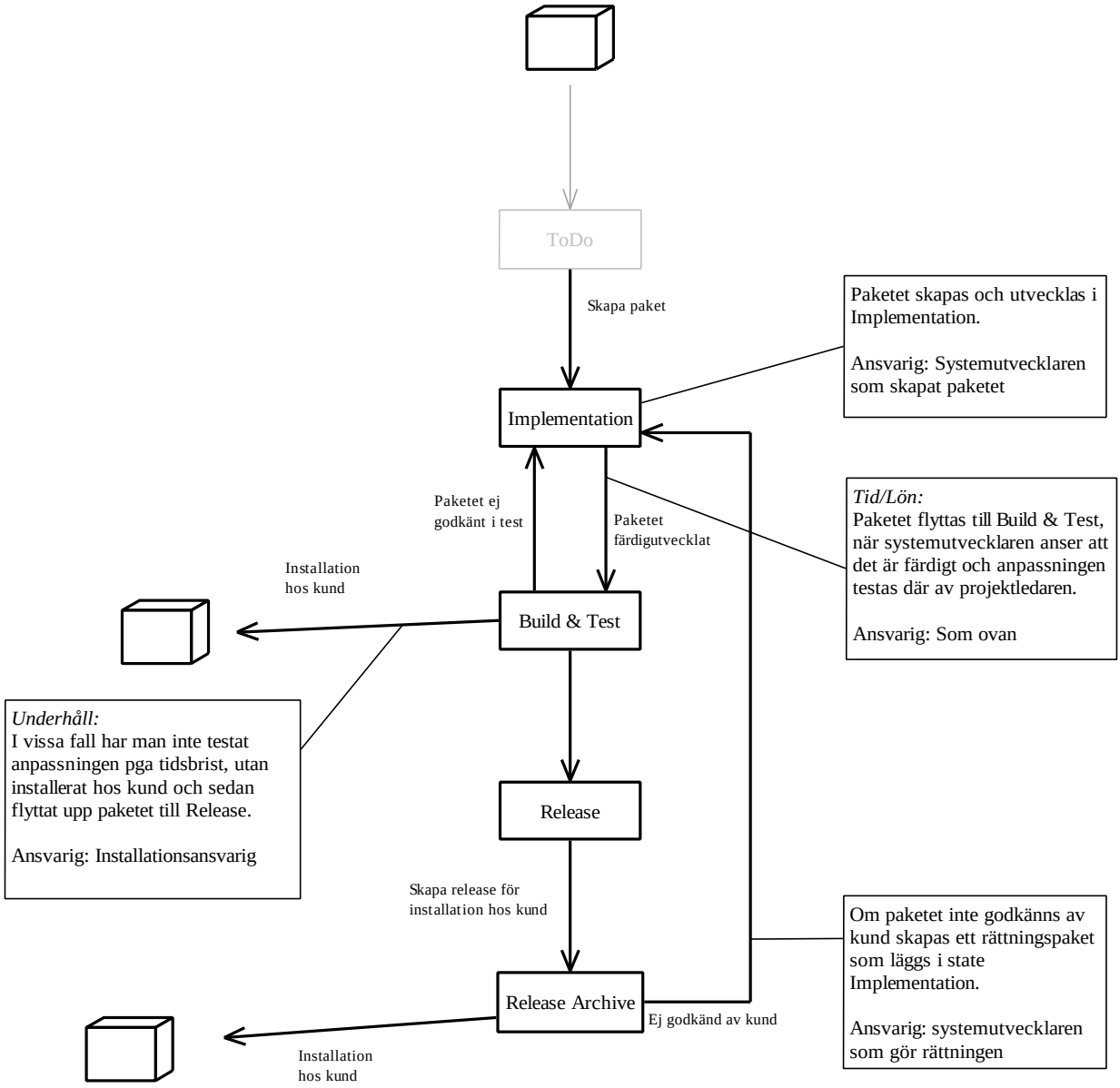
5.2.3.4 Release

En person i projektet, teknikern, är ansvarig för installering och i samband med varje installation för han upp anpassningarna till state *Release*. Ibland har små ändringar installerats hos kund efter driftstart utan att flyttas upp till Release.

5.2.3.5 Release Archive

Detta state användes mest när hela releaser plockades ut, fram till driftstart. Därefter blev det mer att man installerade paket från Build & Test.

5.2.4 Flödesbeskrivning för Hylte-projektet



Figur 6:2 Flödesbeskrivning för Hylte-projektet

5.2.5 Namnstandard

Hylteprojektet har ingen uttalad namngivningsstandard, men Tid/Lön-delen av projeket använde sig av en slags standard där de angav att det var just Tid/Lön samt namnet på specifikationen, t ex PS5 - programspec nr 5. Alla systemutvecklare gjorde inte riktigt likadant, men specifikationsnummer var alltid med.

5.2.5.1 Environments

Projektmiljön kallas för Hylte.

5.2.5.2 Standardpaket

Standardpaketen kallas för STD <modulnamn> <versionsnummer>. När det är aktuellt läggs <servicerelease> till i namnet.

Exempel: STD Purch 10.4.0
STD Purch 10.4.0 A

5.2.5.3 Paket för projektdokument

Ett projektdokument är ett dokument som innehåller information angående projektet, t ex information om aktuella anpassningar. Man har använt sig av två typer av paketnamn i Hylteprojektet: Spec <projektdel> och Dokument.

Exempel: Spec Lön

5.2.5.4 Paket för anpassningar

Paket namnges efter anpassningen och den del av projektet som anpassningen görs i (Tid/Lön eller Underhåll).

5.2.5.5 Buggar och patchar

- Standardbuggar, rapporterade och registrerade

  Kunden rapporterar in buggar direkt till IFS Online, vilket medför att alla buggar som rättas också är registrerade.

  I de fall då buggar som ännu inte har rättats av R&D upptäcks, skapas paket som namnges med beskrivning av vilken modul som berörs. Det finns ingen standard för namngivning av paket av denna typ. Man har dock använt sig av modulnamn och anpassningsnummer.

  Exempel: Inkorder ps223

- Buggar i STD, rättade av R&D

  Ingen namnstandard för detta har använts.

- Component patches

  Patcharna hålls separerade från övriga buggrättningar. Textfilen med R&D-information om patchar ska också checkas in i Harvest. Paketet namnges med modulnamn, version och patchid.

  Exempel: Payrol 8.6.2 Patch 1

- Borttag av buggar och patchar

  Det har ännu inte varit aktuellt med en uppgradering, därför har man inte behövt ta bort några buggrättningar eller patchar i projektet än.

- Buggar i kundanpassningar

  Vid rättningar av buggar i kundanpassningar skapas rättningspaket. De namnges med ‘Buggrättning’ och antingen modulnamn och anpassningsnummer eller, på senare tid, projekt och veckonummer. Det är inte alltid som veckonumret stämmer heller, eftersom en rättning kan ta längre tid att utföra än en vecka. Andra sätt att namnge paketen har också förekommit, t ex Buggr <anpassningsnr>. Något dokument där man anger vilka anpassningsnummer som finns i de olika rättningspaketen har inte använts.

  Exempel: Buggrättning Lön ps 11
  Buggrättning Tid/Lön v. 51

5.2.5.6 QDS-klassade dokument

Program- och funktionsspecifikationer har placerats i projektets environment i Harvest i olika kataloger. När testprotokoll har använts har även dessa lagts in i Harvest. Man har inte haft ett enhetligt sätt att förvara dokumentet och man har heller inte använt en namngivningsstandard. Man har på senare tid checkat in alla dokument mot ett paket som heter 'Spec'.

5.2.6 Bugg- och patchhantering

Man har inte skiljt på standardbuggar och rättningar i anpassningarna, utan man har löst allt på samma gång i ett buggrättningspaket.

5.2.7 Katalogstrukturer

Under projektets huvudkatalog Hylte finns en mapp för varje modul. Dokumenteringen har inte alltid förvarats i Harvest, men man har ett par kataloger som heter Spec <projektdel>. T ex Spec FNE eller Spec Tid. Här förvaras specifikationerna i .doc-format. Det finns även en katalog där som heter Dokument, men den används inte i någon större utsträckning.

Exempel:

Hylte
<modulnamn>
<modulnamn>
Spec <projektdel>

5.2.8 Problem/kommentarer

Ett problem som inte har direkt med Harvest att göra är att man har haft tidsbrist i Underhållsprojeket. Detta har dock medfört att man har slarvat med att Harvest-hantera filer och detta har gett upphov till nya problem.

När det sedan har saknats en gemensam namngivningsstandard, har det ibland varit svårt att hålla isär paket med liknande anpassningar.

Tidspressen har berott mycket på att Stora Hylte AB i början av projektet förhandlade fram att IFS skulle betala ut ett vite om driftstarten av IFS Applications hos pappersbruket blev försenat. Utvecklarna har i vissa fall inte haft tid att testa anpassningarna ordentligt före installation hos kund och kunden har då upptäckt fel som utvecklarna själva borde ha upptäckt.

Om en systemutvecklare hämtar filer från projektets server, istället för att hämta dem ur Harvest, kan det uppstå fel när den nya anpassningen ska testas eftersom det kan finnas halvfärdiga anpassningar gjorda av andra systemutvecklare i filen.

En bidragande faktor till problemen har varit att många medlemmar i projektet har varit nyanställda och därför saknat kunskaper och erfarenhet om hur man bäst arbetar med Harvest. Några problem hade kunnat undvikas med en fastslagen gemensam metodik i skriftlig form på ett tidigt stadium i projektet.

5.3 Sammanfattning

Vid en analys av resultaten från våra intervjuer har vi kunnat fastslå att arbetssätten i KD- och Hylte-projekten skiljer sig från de nya riktlinjerna. Skillnaderna kan kortfattat beskrivas i följande punkter:

- State ToDo används inte alls.
- Build & Test används inte som det är tänkt till att testa anpassningar, utan tester sker i Implementation.
- Release skapas ibland efter installation hos kund, istället för före.
- En uttalad namnstandard har inte använts i Hylte-projektet.
- Bugg- och patch-hantering har inte förekommit.

I övrigt skiljer sig inte arbetssätten från rekommendationerna nämnvärt.

6 Diskussion

Vi kommer i detta kapitel att diskutera de resultat som har framkommit i vår studie.

6.1 Överblick

Gemensamt för de båda projekten är att state ToDo och Build & Test inte har använts som det är tänkt enligt de nya riktlinjerna. Det har förekommit att paket har installerats i kundens testmiljö och produktionsmiljö, innan paketen har förts upp till Release på grund av brist på tid. Det har ännu inte varit aktuellt med bugg- och patchhantering i något av projekten.

I KD-projektet har arbetet blivit mer disciplinerat i samband med det uppgraderingsprojekt som pågår sedan en tid tillbaka. Tydliga beskrivningar för hur arbetet ska ske har utarbetats. KD-projektets arbetsmetodik har idag flera likheter med de nya riktlinjerna, t ex finns det en namngivningsstandard som, trots att den skiljer sig något från rekommendationerna, åtminstone används konsekvent. De har också utvecklat egna finesser, som skulle kunna beaktas av skaparna av de nya riktlinjerna, nämligen de tidigare nämnda dokumenten Uppgraderings-anpassningar.doc och Filkollisioner.doc (se kapitel 6.1.2).

Arbetssättet i de båda delarna av Hylte-projektet skiljer sig åt såtillvida att Tid/Lön-delen av projektet har haft ganska bra rutiner och dessutom mer tid på sig att göra anpassningar. I Underhålls-delen däremot har arbetet varit mer ostrukturerat, även om man vid starten formulerade en arbetsmetodik. Metodiken i Underhåll liknar i viss mån de nya riktlinjerna vad det gäller användningen av states, men den har inte följts systematiskt. Framför allt har testningen av de nya anpassningarna varit undermålig. Även namngivning har skett utan några angivna regler, vilket har skapat en viss oreda. Orsaken till problemen är framför allt tidsbrist. Utvecklarna har varit tvungna att ta genvägar vid Harvest-hanteringen av filer för att bli klara i tid, och man har inte heller hunnit instruera nytillkomna projektmedlemmar i användningen av Harvest i någon större utsträckning.

6.2 Problem

I de båda projekten har det uppstått vissa problem till följd av att man arbetar på olika sätt med Harvest:

- Systemutvecklare hämtar en fil för anpassning från kundservern istället för ur Harvest, för att spara tid eller för att filen redan är reserverad av en annan systemutvecklare. När en fil inte hämtas ur Harvest kan man inte vara säker på att det är den senaste versionen av filen, eller att det är en korrekt och fullständig version. Följande problem kan uppstå:

    - Utvecklaren får kompileringsfel på grund av att det redan kan finnas en halvfärdig anpassning i filen. Kompileringsfelen kan ta lång tid att rätta till. Om paketet installeras hos kund utan att felet har upptäckts, kan följden bli

att kunden uppmärksammar de fel, som utvecklaren borde ha sett på ett mycket tidigare stadium.

  - Filen anpassas och checkas sedan in, utan att den jämförs med den version som redan ligger i Harvest. Risken är att anpassningar som har gjorts i tidigare versioner inte upptäcks och därför inte flyttas över till den nya versionen av filen. Gjorda anpassningar kan därför plötsligt falla bort vid installation och kunden går miste om funktionalitet som funnits tidigare.

- Utvecklaren plockar en fil för anpassning ur state Build & Test. Tidigare gjorda anpassningar har inte testats klart i filen och det kan i ett senare skede leda till svårlösta kompileringsfel eller att det blir fel i applikationen ute hos kund.

- Installation hos kund görs utan att den nya funktionaliteten har genomgått noggranna tester på grund av tidsbrist eller brist på personer som kan testa. Följden blir även i detta fall att fel i applikationen upptäcks av kunden.

- Om det inte finns ett gemensamt namngivningssätt, framför allt när det gäller paket, och projektpersoner döper paket till namn som är snarlika, blir det svårt att hålla isär de olika paketen. Det leder lätt till ineffektivitet, då det kan ta extra tid att identifiera det paket som sökes.

- Om det inte finns klara, tydliga riktlinjer för hur man ska gå tillväga i arbetet med Harvest, blir det svårt för nya personer att komma in i projektet vad det gäller Harvest-användningen. Dessutom ökar risken för att de gör fel, som t ex de fel som nämnts ovan.

6.3 Slutsats

Om man ska dra slutsatser om versionshantering i Harvest utifrån resultatet i vår uppsats, verkar det viktigaste vara att alla gör på samma sätt i en projektgrupp. De fel som uppstår på grund av bristande rutiner upptäcks ofta av kunden, vilket ger ett dåligt intryck av IFS. Eftersom IFS har som policy att garantera kvalitet och säkerhet genom sitt koncept QDS, ställer det stora krav på leveransavdelningen. Därför behövs tydliga riktlinjer, vars syfte är att skapa ett gemensamt, effektivt arbetssätt i Harvest. Det är också viktigt att motivera och bibehålla disciplinen bland gruppmedlemmarna, så att riktlinjerna verkligen följs.

Med tanke på att systemutvecklare byter projektgrupp mer eller mindre tillfälligt på IFS, så skulle det underlätta om man använde samma arbetsmetodik i samtliga projektgrupper. Genom att dessutom ha en person i varje grupp som ansvarar för Harvestanvändningen och som ser till att alla arbetar enligt det gemensamma arbetssättet, kan de flesta problem undvikas.

För nyanställda på leveransavdelningen innebär klara riktlinjer en stor hjälp. Det är dock inte tillräckligt med instruktioner bara på papper. Det är dessutom viktigt med en noggrann genomgång av hur Harvest används, och inte bara information om *att* verktyget ska användas.

De nya riktlinjer som presenteras i kapitel 5 är en viktig del i arbetet med att minska andelen problem och ett mycket bra initiativ. Riktlinjerna är strukturerade och detaljrika, men de innehåller dessvärre vissa oklara formuleringar, vilket medför att de bör kompletteras för att nå den effekt som önskas. I och med att riktlinjerna har formulerats på ett ibland tvetydigt sätt är risken för missförstånd påtaglig och det är svårt för systemutvecklare utan erfarenhet av Harvest att följa instruktionerna. Riktlinjer måste vara lätta att förstå och lätta att följa för att syftet med dem skall kunna uppnås.

Som de nya riktlinjerna ser ut idag, anser vi att de är för otydligt formulerade för att en ny Harvest-användare ska kunna följa dem. Det som behövs är en enkel, tydlig steg-för-steg-beskrivning av hur det dagliga arbetet med Harvest ska bedrivas. En arbetsmetodik som är entydig och lätt att följa befrämjar disciplinen hos både erfarna och nyanställda systemutvecklare.

För att gå närmare in på detaljer om namngivning, kan vi konstatera att det är viktigt att vara kreativ när man namnger paket och ge dem beskrivande namn. Det är lämpligt att ange vilket delprojekt som paketet tillhör i paketnamnet. När det gäller releaser är det bra om de namnges med datum, för att man lätt ska kunna hålla reda på dem. Det kan bli många buggrättningspaket i ett projekt och därför kan det vara lämpligt att numrera dem, vilket beskrivs närmare i kapitel 6.1.5.5 om KD-projektet under rubriken “Buggar i kundanpassningar”.

I de fall när parallell utveckling inte kan undvikas tycker vi att KD-projektets lösning med dokumentet Filkollisioner.doc (se kapitel 6.1.2) verkar lämplig. Det ställer dock stora krav på kommunikationen inom projektgruppen.

Vi har även under arbetets gång sett att det finns yttre faktorer som har inverkan på kvaliteten på användningen av Harvest. Några förutsättningar för att man på leveransavdelningen ska kunna garantera sina kunder en hög kvalitet på produkten är också att det finns tillräckligt med tid avsatt till projektet, rätt kompetens i projektgruppen och en god kommunikation med kunden:

- Tid – Det är för att minska felfrekvensen nödvändigt att det finns tid att genomföra omfattande tester innan installation sker i kundens produktionsmiljö.

- Kompetens - För att ett projekt ska kunna genomföras på ett för både kunden och IFS tillfredsställande sätt, bör det finnas rätt blandning av kompetens inom projektgruppen. Man bör undvika att ha för många nyanställda systemutvecklare i en och samma projektgrupp, utan istället sträva efter en jämn fördelning av nyanställda och erfarna systemutvecklare. Detta för att de nyanställda ska få stöd och hjälp att snabbt komma in i arbetet. Förutom kunskaper i projektledning bör projektledaren ha så stor kunskap i systemutveckling, att han/hon kan göra en rimlig bedömning av vilka anpassningar som ska göras, hur lång tid de tar att utföra och i vilken ordning utvecklingen ska ske. Genom god planering kan parallell utveckling och tidsbrist undvikas.

- Kommunikation – Det bör finnas en öppen kommunikation mellan kunden och representanter från projektgruppen för att kunden ska få den produkt som

önskas. Kunden bör också informeras om betydelsen av att man testar de paket som installeras i kundens testmiljö, för att utvecklarna på IFS ska kunna få en nödvändig respons på sitt arbete inom rimlig tid.

6.4 Synpunkter från användarna

Under intervjuerna och även i samtal med andra systemutvecklare på IFS har följande synpunkter framkommit:

- Förslag på namnstandard för paket: Datum, beskrivning av ändring eller specifikationsnummer ska ingå i namnet.

- Harvest är överlag bra, men man kunde önska bättre sök- och sorteringsfunktioner för filer som finns i Harvest.

- Placering av QDS-klassade dokument: Dokumenten samlas i ett separat Spec-bibliotek, men checkas in mot respektive anpassningspaket i Harvest.

- Det bör vara möjligt att checka ut ett helt paket med samtliga filer, istället för som idag, när det bara är möjligt att checka ut filerna manuellt. Risken för att man missar en fil är stor.

7 Allmän diskussion

7.1 Metodutvärdering

Något som har haft inverkan på resultatet av intervjuerna, och därmed på resultatet i uppsatsen, är att vår handledare på IFS rekommenderade de personer som sedan blev intervjuade, och att dessa har uttryckt sin personliga uppfattning om användningen av Harvest. Projekten rekommenderades av den anledningen att det är stora projekt med många systemutvecklare inblandade, vilket var en lämplig utgångspunkt för vår undersökning. Om vi själva hade valt andra projekt och personer hade resultatet eventuellt blivit ett annat, men då vi saknar kunskap om de projekt som bedrivs på IFS AB under våren 1999 fann vi det för gott att följa de rekommendationer som vi fick. Möjligtvis hade vi kunnat studera fler projekt och/eller intervjua fler personer varvid eventuellt fler problem och alternativa arbetssätt kunnat uppmärksammas. Vi valde dock att begränsa vår undersökning till två projekt för att kunna göra en mer detaljerad studie.

Eftersom vi i vår uppsats studerar användningen av ett verktyg som vi tidigare inte har kommit i kontakt med, var det nödvändigt att först sätta sig in i hur det fungerar. Det har dock varit svårt att hinna inhämta annat än teoretiska kunskaper. Vi har därför varit tvungna att till stor del förlita oss på våra intervjupersoner, eftersom vi själva inte har någon större erfarenhet av att arbeta i Harvest.

Om vi istället för de personliga intervjuerna hade gjort en enkät hade vi eventuellt fått svar som hade varit lättare att bearbeta och jämföra. Vi hade emellertid gått miste om möjligheten att ställa följdfrågor och diskutera intervjupersonens åsikter, om vi hade valt att göra en enkät.

7.2 Efterkommande forskning

Under uppsatsskrivandet har vi kommit att tänka på ytterligare intressanta ämnen för fortsatt forskning, nämligen:

Genom att vidare undersöka arbetet i olika projektgrupper skulle man kunna utforma tydliga steg-för-steg-instruktioner för hur man bäst arbetar i Harvest. Det kunde vara lämpligt att även intervjua nyligen anställda systemutvecklare för att få en klar bild av hur dessa instruktioner borde se ut.

Man skulle kunna göra en utvärdering av Harvest som verktyg för versionshantering på IFS. Uppfyller Harvest de krav som IFS har på ett versionshanteringsverktyg?

Ytterligare ett förslag är att jämföra Harvest med andra versionshanteringsprogram. Vilka alternativ har mjukvaruproducerande företag idag och hur står sig Harvest i konkurrensen?

Det skulle också vara intressant att undersöka vad kunder har för uppfattning om versionshanteringen hos sin programtillverkare. IFS garanterar sina kunder kvalitet och säkerhet genom sitt koncept QDS. Hur uppfattas detta av kunden?

7.3 Slutord

Arbetet med uppsatsen har varit både intressant och givande. Vi har lärt oss en hel del om Harvest, även om vi inser att vi har mycket kvar att lära. Något som har gjort arbetet extra roligt är all den respons vi har fått från anställda på IFS AB. Vi vill tacka alla på IFS som har hjälpt oss genom att dela med sig av sina kunskaper och åsikter. Ett stort tack också till vår handledare på informatik – Wera.

8 Referenslista

8.1 Litteratur

Backman, Jarl, 1998, Rapporter och uppsatser, tionde upplagan. Studentlitteratur, Lund.

Löfgren, Niklas, 1999, Harvest Guidelines, IFS AB, Malmö.

Sommerville, Ian, 1997, Software Engineering, 5th Edition. Addison Wesley Longman Limited, Harlow, Essex.

8.2 Material från Internet

IFS:s internationella websida, maj 1999:
IFS Industrial & Financial Systems – IFS Home
URL: http://www.ifsab.com/

8.3 Opublicerat material: C-uppsats

Andersson, Marie; Bergand, Maria, 1998, Versionshantering – riktlinjer för anskaffning av verktyg, institutionen för informatik, Göteborgs universitet.

8.4 Kontaktpersoner på IFS i Göteborg

Handledare:
Rönnqvist, Liselott, gruppledare samt delprojektledare i Hylte-projektet.

Intervjuer:
Håkansson, Claes, systemutvecklare i Hylte-projektet.
Jansson, Karin, systemutvecklare i Hylte-projektet.
Pervik, Stefan, Harvest-ansvarig systemutvecklare, lärare vid kurstillfällen.
Planmo, Johan, systemutvecklare i KD-projektet.
Tholerus, Lovisa, systemutvecklare i KD-projektet.

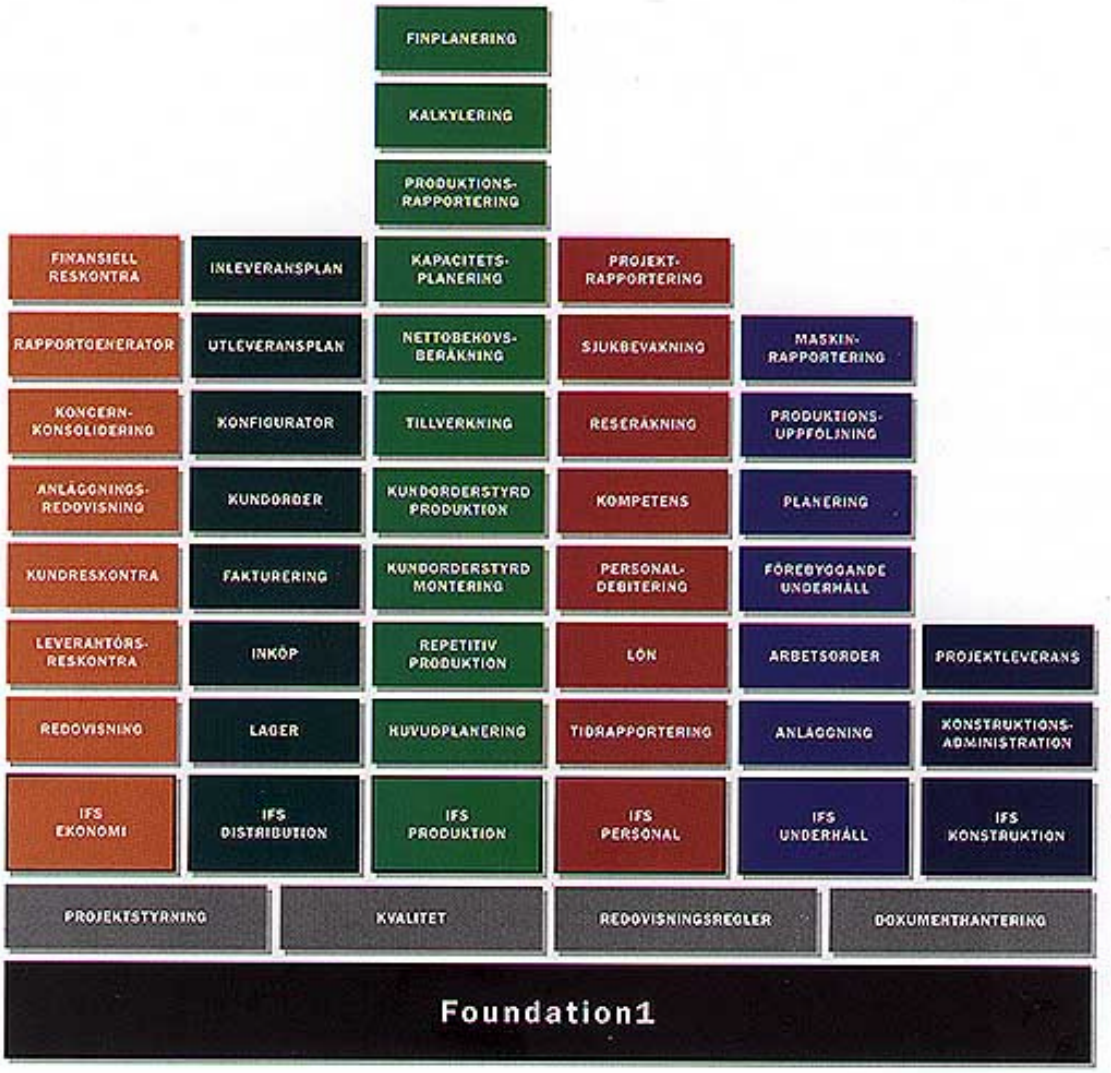
E-post:
Löfgren, Niklas, systemutvecklare på IFS i Malmö.

Övriga:
Ebbeson, Niclas, systemutvecklare/applikationskonsult
Lundberg, Hans-Peter, systemutvecklare/tekniker
Nilsson, Christin, systemutvecklare
Travaglia, Fredric, systemutvecklare/tekniker i KD-projektet

9 Bilagor

Bilaga 1: IFS Applications
Bilaga 2: Intervjufrågor

IFS Applications



Intervjufrågor

Allmänt

Vilket projekt arbetar du med (Hylte eller KD)?
Berätta kort om företaget och de anpassningar de har beställt.
Hur länge har projektet pågått?
Hur många personer arbetar med detta projekt?
Har det varit stor personalomsättning i projektet?
Sedan när används Harvest i projektet?
Uppstod problem vid införandet av Harvest i projektet och i så fall vilka?
Vilken attityd hade systemutvecklarna till införandet av Harvest?
Vilken attityd har systemutvecklarna idag?

Användning i projektet

Finns det en metodik för hur gruppen ska använda Harvest?
Hur ser metodiken ut?
Följs metodiken?
Beskriv hur Harvest används i projektet (KD/Hylte)
T ex Används ToDo-statet?
Vem skapar paket?
Hur skapas paket?
Följs eventuella namngivningsstandarder för paketen?
Vem promotar ett paket till nästa state?
När promotas paket?
När demotas paket?
Vem testar?
När görs en release?

Avslutning

Hur tycker du att Harvestanvändningen fungerar i projektet?
Vilka problem har uppstått?
Vilka förändringar skulle behöva göras för att förbättra användningen av Harvest? (Hur borde Harvest användas?)
Synpunkter på Harvest?