Handelshögskolan vid Göteborgs universitet
Institutionen för informatik

Versionshantering på IFS AB

med inriktning på Harvest och egenutvecklat Harvesttillägg.

Uppsatsen gjordes i syfte att öka förståelsen och belysa vikten av versionshantering när det gäller programvaruutveckling. En studie gjordes över hur arbetssättet för versionshantering i versionshanteringsverktyget Harvest hade förändrats under de senaste två åren. Detta gjordes genom att studera flödesbeskrivningar, i projektet med kunden Kronans Droghandel (KD) men även flöde för allmänna riktlinjer och genom att analysera de problem som uppstår vid versionshantering. Dessutom undersöks om det nya, egenutvecklade Harvesttillägget underlättar versionshanteringen för anställda på IFS.
Uppsatsen avgränsas till att studera KD-projektet på IFS AB. Övrigt material har vi i huvudsak fått genom litteraturstudier, intervjuer och ett test som gjordes av Harvesttillägget. Uppsatsen drog slutsatsen att de problem som finns och som man borde arbeta vidare med, inte är flödes-beskrivningarna utan tillägget. Att få ett väl fungerande Harvesttillägg som stöder arbetet speciellt vid leveranser skulle vara tidsbesparande och även ge en ökad kvalitet av programvaran.

Författare:
Emma Albinsson
Cecilia Norén

Examensarbete I 10p
ADB-programmet 80p
Vårterminen 2001
Handledare: Karin Brander

Innehållsförteckning

Bilagsförteckning

Figurförteckning

1 Inledning

1.1 Bakgrund

Versionshantering ingår i ett företags konfigurationsstyrning och är en av grundstenarna för bra kvalitet på programvaran.
Konfigurationsstyrning ingår i ett företags kvalitetssystem och är ett samlingsnamn för ett antal aktiviteter som används vid systemutveckling, men som inte är kopplade till någon speciell utvecklingsfas. Exempel på sådana aktiviteter är versionshantering, dokumentation över ansvarsfördelning, rutiner som underlättar spårbarhet av frisläppta produkter.
Versionshantering används vid systemutveckling när man skapar en ny produkt som genom tillägg och ändringar kontinuerligt förses med ny funktionalitet. Då skapas flera versioner av produkten.

När vi inledde vår praktikperiod på IFS[1] i januari 2001, hade vi näst intill obefintliga kunskaper i ämnet.
Uppgiften som skulle lösas under praktiken var att göra ett tillägg till det redan existerande versionshanteringsverktyget CCC/Harvest[2], som ingår i IFS kvalitetssystem QDS[3]. Vi skulle med tillägget förbättra Harvest både när det gällde säkerhet och snabbhet. För att kunna göra detta fick vi sätta oss in i hur Harvest fungerade. Detta gjordes med hjälp av tekniker Fredric Travaglia, som var vår handledare under praktikperioden. Eftersom vi aldrig arbetat med versionshantering, så var det ganska svårt att sätta sig in i hur tillägget skulle underlätta versionshanteringen för de som arbetar på IFS.

Harvest togs i bruk december 1997. Då kom de första riktlinjerna för hur Harvest skulle användas. I den projektgrupp som vi har valt att följa, KD-projektet[4], har man kontinuerligt arbetat med att förbättra och skapa nya arbetsrutiner i Harvest som fungerar bättre.

Man anser i gruppen att Harvest är ett bra versionshanteringsverktyg, men det finns en del brister i funktionerna. Som examensarbete hade man önskemål från IFS att utreda vilka problem som finns i dagens projektmetodik rörande versionshantering.

Vi kommer att studera utvecklingen av Harvest mellan år 1999 och 2001, fokusera på problem vid versionshantering och undersöka huruvida tillägget kan förbättra hanteringen.

[1] Industrial & Financial Systems
[2] Verktyget kommer fortsättningsvis att benämnas Harvest
[3] Quality Development System, intranet IFS
[4] Kundprojektet Kronans Droghandel

Detta ämne har även tagits upp av studenter i tidigare examensarbete[5], vilket vi kommer att referera till. Då jämfördes de olika sätt som Harvest har använts på i två större projekt på leveransavdelningen på IFS AB, samt hur dessa metoder skiljde sig från den arbetsmetodik som hade utvecklats för användningen av Harvest på IFS i Sverigerutiner från 1999.
En av de projektgrupper man undersökte var KD-projektet. Det finns därför dokumenterat hur man arbetade år 1999 och detta kan ge en intressant bakgrund till utvecklingen av arbetsrutinerna i Harvest.

1.2 Syfte

Syftet med examensarbetet är att ge en ökad förståelse och belysa vikten av versionshantering när det gäller programvaruutveckling. Vi vill jämföra hur arbetssättet för versionshantering i Harvest har förändrats under de senaste två åren och analysera de problem som uppstår vid versionshantering. Dessutom vill vi undersöka om det nya, egenutvecklade Harvesttillägget kan underlätta versionshanteringen för anställda på IFS.

1.3 Målgrupp

Uppsatsen vänder sig först och främst till anställda på IFS som använder sig av Harvest men även till personer som är allmänt intresserade av problematiken kring versionshantering i samband med systemutveckling.

1.4 Problem

Man har arbetat mycket i KD-projektet med arbetsrutiner i Harvest. Det finns fortfarande en del problem att lösa och många funktioner att förbättra.
Vår hypotes är att Harvesttillägget, som vi utarbetade under vår praktikperiod, kommer att förenkla arbetet för anställda på IFS.

Frågeställningar:

- Hur har arbetssättet/flödesbeskrivningarna för versionshantering i Harvest utvecklats i KD-projektet från år 1999 till 2001?

- Vilka problem finns i dagens projektmetodik rörande versionshantering i KD-projektet för tekniker/systemutvecklare/projektledare?

- Kan det nyutvecklade verktyget användas för att underlätta versionshanteringen för tekniker/systemutvecklare/projektledare?

- Förändras arbetsmetodiken/flödet pga Harvesttillägget?

[5] ”Versionshantering med inriktning på Harvest på IFS AB”, Lovisa André och Kristina Falkenström, 1999

1.5 Avgränsning

Vi kommer att begränsa vårt arbete till *en* projektgrupp om hur versionshantering på IFS fungerar idag. Denna avgränsning gör vi pga att tidigare examensarbete[6] som behandlat detta ämne, har studerat samma grupp. Det finns därför dokumenterat hur man arbetade år 1999 och vi har möjlighet att jämföra tidigare arbetssätt/flöden av versionshantering i Harvest med dagens arbetssätt.
Vi kommer inte att göra någon allmän utvärdering av Harvesttillägget. Istället väljer vi ut ett fåtal personer för ett test. Dessa är mycket kunniga i användandet av Harvest,. Detta görs för att få en säkrare bedömning av tillägget. Av testpersonerna är det *en* person som inte ingår i KD-projektet.

1.6 Metod

Vi har gjort litteraturstudier i ämnet versionshantering och till viss del även i kvalitetssäkring.

En beskrivning av Harvest och Harvesttillägget görs. Informationen om Harvest är inhämtad från ”Detaljspecifikation Harvest Extension”[7]. Vi har även studerat tidigare examensarbete[8] och pratat med gruppledare Peter Widén för att kontrollera att uppgifterna är aktuella. Dessutom har vi studerat den information som finns på IFS intranet.

Vi gör en kort redovisning av de allmänna riktlinjerna/flödesbeskrivningen[9] för arbetsmetoderna i Harvest som utarbetades under våren 1999 på IFS samt av riktlinjer/flöde för arbetsmetoder i Harvest i KD-projektet år 1999 och år 2001. Därefter görs en jämförelse mellan dessa riktlinjer i KD-projektet år 2001 med år 1999 respektive allmänna riktlinjer år 1999. Dessa uppgifter har vi fått genom att intervjua Johan Planmo, studera tidigare examensarbete och hämta information från intranet, Harvest Guidelines.

För att få en uppfattning om hur Riktlinjer i KD-projektet år 2001 ser ut har vi gjort en modell som liknar tidigare modeller, men som beskriver de states som används i KD-projektet i dag. För att göra detta utgick vi från tidigare modeller och diskuterade dessa med gruppledare Peter Widén, som hela tiden deltagit i KD-projektet. Dessutom har han varit med och tagit fram utvecklingsrutiner i Harvest.

Vi gjorde även en mindre utvärdering av Harvesttillägget. Med hjälp av Peter Widén valde vi sex testpersoner: Ralf Johansson, Helena Nordquist, Liselotte Taxén, Peter Svensson, Johan Planmo och Stefan Litzén. Enligt Peter Widén var dessa personer mest lämpade för att testa Harvesttillägget pga att de har en gedigen kunskap om hur Harvest fungerar. Testpersonerna fick testinstruktioner där det stod vilka filer och datorinställningar de behövde ha för att kunna genomföra testet. Dessa personer kontaktades via mail. När vi fått klartecken att de var positiva till att ingå bland testpersonerna, mailade vi testinstruktionerna (bil.2).

---

[6] ”Versionshantering med inriktning på Harvest på IFS AB”, Lovisa André och Kristina Falkenström, 1999
[7] Skriven av praktikplatshandledare, Fredric Travaglia
[8] ”Versionshantering med inriktning på Harvest på IFS AB”, Lovisa André och Kristina Falkenström, 1999
[9] Harvest Guidelines, IFS intranet

Testet innehöll tre delar: testprotokoll på 11 sidor (bil.3), ett testexempel samt sju frågor som testpersonen skulle besvara. Mallen till testprotokollet togs ifrån ett tidigare test som gjorts i KD-projektet. Testexemplet gick ut på att simulera början av en leverans. Detta fick vi hjälp av tekniker Claes Håkansson att göra.
Fyra av de sex testpersonerna som hade valts ut hann lämna in sina testresultat innan deadline. Orsaken till att två testpersoner uteblev var sjukdom och tidsbrist.

1.7 Disposition

Uppsatsen inleds med en kort presentation av IFS. Därefter följer ett allmänt avsnitt om versionshantering och vikten av att versionshantera. Med detta som bakgrund görs en beskrivning av Harvest och en beskrivning av Harvesttillägget. Därefter följer en presentation av riktlinjer för användandet av Harvest och flödesbeskrivningar dels från ”allmänna riktlinjer” och dessutom från ett specifikt projekt år 1999.
I resultatdelen beskrivs riktlinjer och flöde från samma projekt år 2001. Här jämförs även de olika arbetssätten.
Därefter redovisas testresultaten från utvärderingen av Harvesttillägget.
Därefter följer en diskussion om resultat- och metoddelen.

2 Lite kort om 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 43 länder och har över 3000 kunder runt om i världen. IFS i Sverige sysselsätter ca 1017 personer och av dessa arbetar 220 på Göteborgskontoret. 2000 hade hela IFS en nettoomsättning på 2 352 Mkr.

Göteborgskontorets organisation är uppdelat i följande avdelningar: Marknad, Sälj, Utveckling och Leverans.
Leveransavdelningen sköter kontakten med kunden från det att kunden beställt IFS Applications[1] till det att alla anpassningar är gjorda och projektet är avslutat.

På leveransavdelningen arbetar projektledare, gruppledare, systemutvecklare, tekniker 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.

De flesta av ovan nämnda yrkesgrupper använder eller har använt sig av Harvest och hur detta förhåller sig kommer vi att redovisa längre fram i uppsatsen.

[1] Ett komponentbaserat program

3 Konfigurationsstyrning - Versionshantering

För att ett programutvecklingsföretag skall kunna hålla en hög kvalitet på produkten krävs att man har ett kvalitetssystem med bra rutiner för bla Konfigurationsstyrning. På IFS finns ett kvalitetssystem som heter QDS. IFS är inte certifierad enligt någon standard ännu men när man tittar på delen om konfigurationsstyrning ser man att kvalitetssystemet följer de riktlinjer som finns i exempelvis ISO 9000-3[1].
Eftersom ISO 9000-3 har fått ett stort genomslag i många länder och ligger till grund för många andra standardiserings modeller kommer vi att titta lite närmare på den delen av ISO 9000-3 som har med konfigurationsstyrning – versionshantering - att göra. Lite längre fram i uppsatsen kommer vi att beskriva versionshantering i verktyget Harvest, som ingår i IFS kvalitetssystem.

Redan på tidigt 1900-tal började man i USA ställa krav på underleverantörernas kvalitetssystem och militära beställare fortsatte utveckla en standard för hur underleverantörerna skulle utvärderas. Under 80-talet utvecklades en rad internationella standarder som ofta tog intryck av den snabba kvalitetsutvecklingen i Japan. Däribland var ISO 9000-3.

Viktigt att förstå är att dessa standarder inte talar om kvalitet på produkterna utan kvalitet på organisationerna och deras arbetssätt. Ett företags kvalitetssystem är det som styr upp deras processer, ansvar, rutiner och resurser med avseende att leverera kvalitetsprodukter.[2]

3.1 Konfigurationsstyrning ISO 9000-3

I sista delen av ISO 9000-3 behandlas kvalitetssystemets stödjande aktiviteter d v s de som inte är kopplade till någon bestämd utvecklingsfas.
Däribland behandlas konfigurationsstyrning.

Följande riktlinjer finns för Konfigurationsstyrning:
Denna aktivitet bör identifiera versioner, komponenters utvecklingsstatus och ändringar, styra samtidig produktuppdatering som utförs av flera personer samt samordna produkter som finns i flera versioner.

Dessutom bör det finnas en konfigurationsstyrningsplan som definierar organisatoriska ansvar, aktiviteter som skall utföras, verktyg, teknik och metodiker samt när konfigurationsstyrning ska sättas in.

Det som ska styras inkluderar specifikationer, verktyg, gränssnitt, dokument, datafiler och ändringar. När det gäller frisläppta produkter bör det finnas rutiner som underlättar spårbarheten.

[1] ISO 9000 i programvaruutveckling – att konstruera kvalitetsprodukter, Östen Oskarsson Robert L Glass, Studentlitteratur 1995
[2] Programkonstruktion med kvalitet, Sven Eklund, Studentlitteratur 1998

Syftet med konfigurationsstyrning är att säkerställa att den framväxande programvaruprodukten inte kan gå förlorad. Detta uppnås vanligen genom att man skapar och sparar basversioner av produkten utifrån vilka aktuella versioner kan byggas på nytt om det skulle behövas. Produkten inkluderar inte bara koden (i olika former som käll- och objektkod), utan även databaser, filer, dokumentation och andra delar av avgörande betydelse för programvaruprodukten.

Konfigurationsstyrningen måste av nödvändighet också hantera versionskontrollen. En programvara släpps normalt till användare/beställare i en ”version”, dvs en variant av produkten som är giltig vid tiden för frisläppandet för en viss leveransplattform men som kan komma att bli föråldrad vartefter ytterligare ändringar görs. Olika versioner kan alltså vara plattformsberoende (de fungerar på en viss dator eller med ett visst operativsystem) eller kronologiskt beroende (de fungerar vid en viss tidpunkt).[3]

3.2 Vikten av versionshantering

Fletcher J. Buckley[4] ger, i ett exempel, en enkel beskrivning som ger en bra förståelse för vikten av versionshantering. Han beskriver att ett dataprogram exempelvis kan innehålla 40.000 kodrader. Med ett genomsnitt av 100 rader per fil, blir det ungefär 400 filer i ett dataprogram. Alla dessa filer kommer under utvecklingen med största sannolikhet förändras minst 1 gång och en del upp till 10 gånger.
Med anledning av detta är det viktigt att varje fil får en egen unik identitet.
Mjukvara är således identifierad på filnivå. Varje fil brukar ha tre användbara fält:

- Filnamn
- Filtyp
- Versions nummer (eller datum)

Ex.HeInstallation.apy.3

3.3 Verktyg och tekniker för versionshantering

I ett tidigare examensarbete[5] beskrivs sammanfattande att man vid systemutveckling skapar 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, genom att enkelt kunna identifiera och spåra de olika versionerna och releaserna, används ofta automatiserade verktyg för versionshantering. Harvest är ett sådant verktyg.

Det finns enligt Sommerville[6] tre grundläggande tekniker som kan användas vid versionshantering:

[3] ISO 9000 i programvaruutveckling – att konstruera kvalitetsprodukter, Östen Oskarsson Robert L Glass, Studentlitteratur 1995
[4] Implementing Configuration Management
[5] ”Versionshantering med inriktning på Harvest på IFS AB”, Lovisa André och Kristina Falkenström, 1999
[6] Software Engineering, Sommerville, 2001

1. *Version numbering* Komponenten får ett unikt versionsnummer. Detta är den vanligaste identifieringsmetoden.

2. *Attribute based identification* Varje komponent har ett namn (som inte är unikt för varje version av en speciell komponent) och en uppsättning attribut som skiljer sig för varje version av komponenten. Komponenter känns därför igen på kombinationen namn och attributuppsättning.

3. *Change-oriented identification* Varje system namnges som i den attribut- baserade identifieringen, men är också associerad med en eller fler förändringar. Systemversionen känns igen genom att man förknippar namnet med förändringarna som är implementerade i komponenten.

I Harvest används version numbering.

3.4 Funktioner i ett versionshanteringsvertyg

Exempel på funktioner[7] i ett versionhanteringsverktyg är:

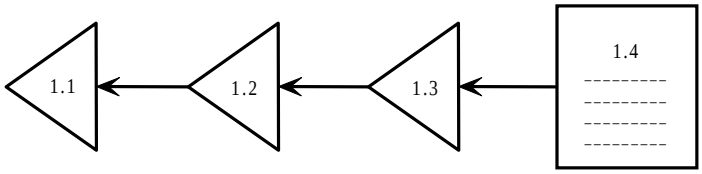
- *Identifiering av releaser och versioner*. Varje version av en systemfil tilldelas en unik beteckning för identifikation (i Harvest används ett nummer), vilket gör det möjligt att identifiera filen under hela utvecklingsprocessen.

- *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 filer 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:

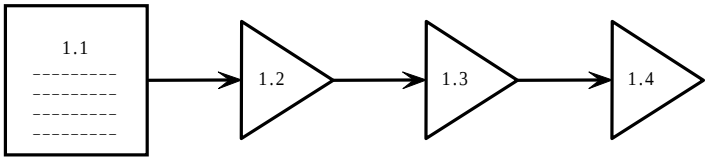
  Backward delta – Den senaste filversionen finns lagrad i sin helhet. Tidigare versioner nås genom att de ändringar som har gjorts mellan den sökta versionen och den senaste versionen tas bort.

[7] ”Versionshantering med inriktning på Harvest på IFS AB”, Lovisa André och Kristina Falkenström, 1999



Figur 3:1 Backward delta

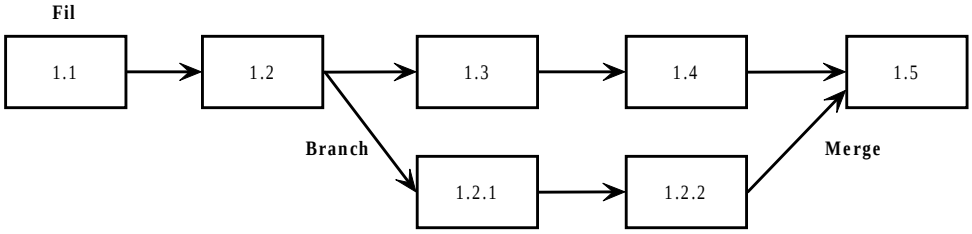
Forward delta – I Harvest används forward delta, pga att man utgår från färdiga moduler och utifrån dessa gör förändringar och tillägg Den ursprungliga filversionen finns lagrad i sin helhet. Efterkommande versioner lagras genom att ändringarna som har gjorts mellan de olika versionerna sparas. När en senare version söks, läggs ändringarna till den ursprungliga filversionen.



Figur 3:2 Forward delta

- *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. Denna funktion finns även i Harvest.



Figur 3:3 Branch och merge

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.[8]

I Software engineering(2001) kan man läsa att versionshantering innefattar hanterandet av stora mängder information och försäkrandet att systemförändringar är lagrade. Versionshanteringsverktyg kontrollerar ett förvaringsutrymme som innehåller filer och som inte kan förändras. För att kunna arbeta med en fil, måste den först checkas ut från sitt förvaringsställe och in till ett arbetande bibliotek. När man har arbetat klart med den, läggs den åter till sitt förvaringsutrymme och en ny version skapas automatiskt.

Enligt Fletcher J. Buckley skall det finnas en stegvis kontroll vid mjukvaruutveckling. Innan man börjar testa mjukvaran är det bara programmeraren som kontrollerar förändringarna. Men när programmet börjar testas av andra programmerare som gör tillägg och förändringar är det viktigt att ha ett bra kontrollsystem, annars finns det stor risk att fungerande kod blir förstörd. Ju närmare kundleverens produkten kommer, desto viktigare är kontrollen av filer för att kunden skall få rätt leverans och produkten skall hålla en hög kvalitet.

Sammanfattningsvis kan man säga att versionshantering förbättrar säkerheten vid levererans till kund. Det är dessutom tidsbesparande när man använder versionshantering rätt. Risken att systemutvecklare, utan vetskap, förstör varandras arbete minskar.

[8] Software engineering, Sommerville, 1997

4 Beskrivning av Harvest

Informationen om Harvest har vi fått genom att studera Detaljspecifikation Harvest Extension[1]. Vi har även studerat tidigare examensarbete[2] i ämnet och pratat med gruppledare Peter Widén för att kontrollera att uppgifterna är aktuella. Dessutom har vi studerat den information som finns på IFS intranet.

4.1 Kvalitetssäkringssystem på IFS – QDS

Harvest ingår i IFS kvalitetssystem QDS. Harvest är ett windowsbaserat verktyg för versionshantering av filer och paket. Harvest används på IFS sedan i december 1997, då det ersatte det teckenbaserade verktyget CMS[3].

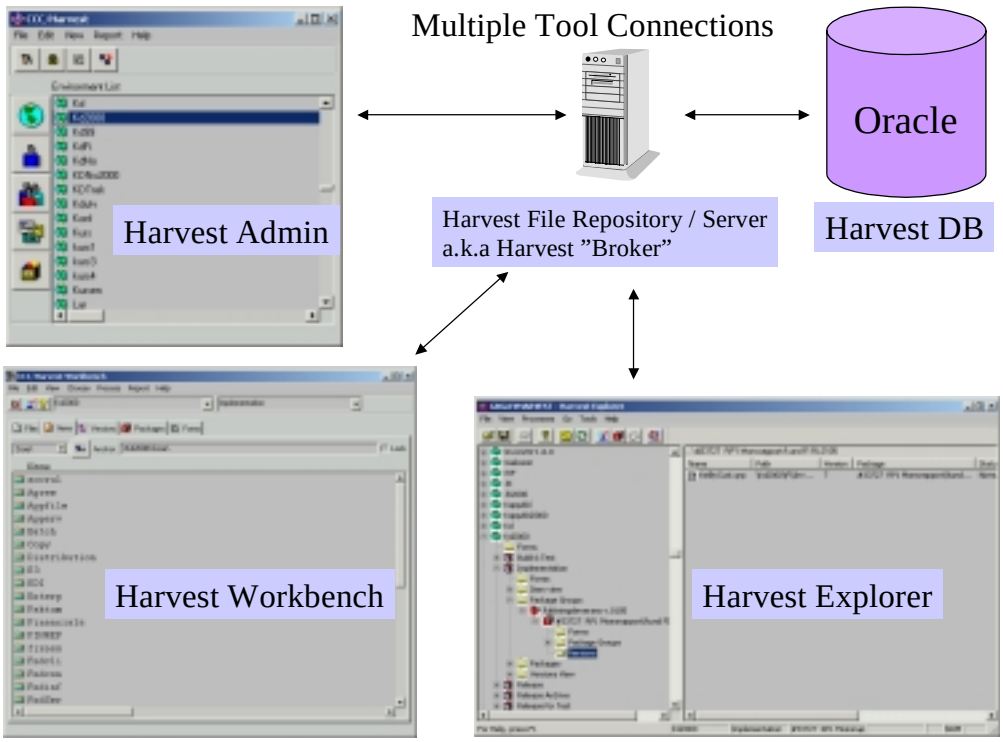
4.2 Harvest-Arkitektur

Harvestapplikationerna fungerar idag genom två (*tre*) skilda gränssnitt:

- Harvest Administration: Här arbetar tekniker med att skapa Harvestmiljön, genom att lägga upp användare, rättigheter för användare, states och vad man skall kunna göra i varje state.
- Harvest Workbench: Här kan man checka in- / ut filer, göra paket mm. Används av både Tekniker och Systemutvecklare.
- (*Harvest Explorer: Är ett senare framtaget gränssnitt av Harvest Workbench*)

Varje klient/användargränssnitt kopplar upp sig mot en serverapplikation (Harvest Broker) som hanterar alla förfrågningar. Den är i sin tur uppkopplad mot en Oracle-databas. Där lagras förändringarna i filer, versioner, snapshots, paket etc.
Det är i Harvest Workbench/Explorer själva versionshanteringen sker.

[1] Dokument skrivet av praktikplatshandledare, Fredric Travaglia
[2] ”Versionshantering med inriktning på Harvest på IFS AB”, Lovisa André och Kristina Falkenström, 1999
[3] Code Management System



Figur 4:1 Arkitektur Harvest

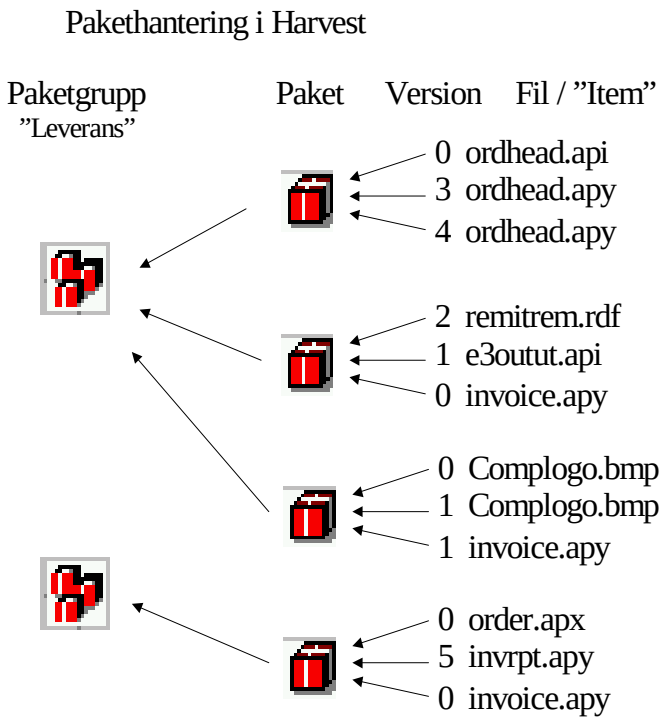
4.3 Komponenter i Harvest

Nedan följer en beskrivning av de viktigaste komponenter som ingår i Harvest:

- Repository: Alla kunders anpassade filer i olika versioner finns lagrade i ett bibliotek, *repository*, i en databas.

- Environment: 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. Under ett environment lagras alla filer/items som förändras i anpassningarna. Filer som finns i Harvest kallas items. Detta för att markera att de finns i Harvest. Filerna lagras i environmentstrukturen genom paket.

- 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 projektet. Paketgrupp är oftast synonymt med ett leveranstillfälle.



Figur 4:2 Pakethantering i Harvest, Fredric Travaglia, IFS

- 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 standard states för olika stadier i utvecklingen:

1. ToDo – Planering
2. Implementation - Utveckling
3. Build & Test – Testning
4. Release – Klar för installation hos kund
5. Release Archive – Tidigare releaser

I KD-projektet, som vi har studerat lite närmare, finns ytterligare två states:

6. Verification
7. Release for test

- Transitions: Förflyttningar, mellan olika states i livscykeln:
  Promote - förflyttning framåt
  Demote - förflyttning bakåt (i KD-projektet används inte demote)

  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.

- 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 anpassade filer. 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 anpassade filer 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.

- Harvest Workbench/Explorer: Applikationen för versionshantering i Harvest kallas för Harvest Workbench/Explorer. Vid uppstart av applikationen öppnas ett fönster som är indelat 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.

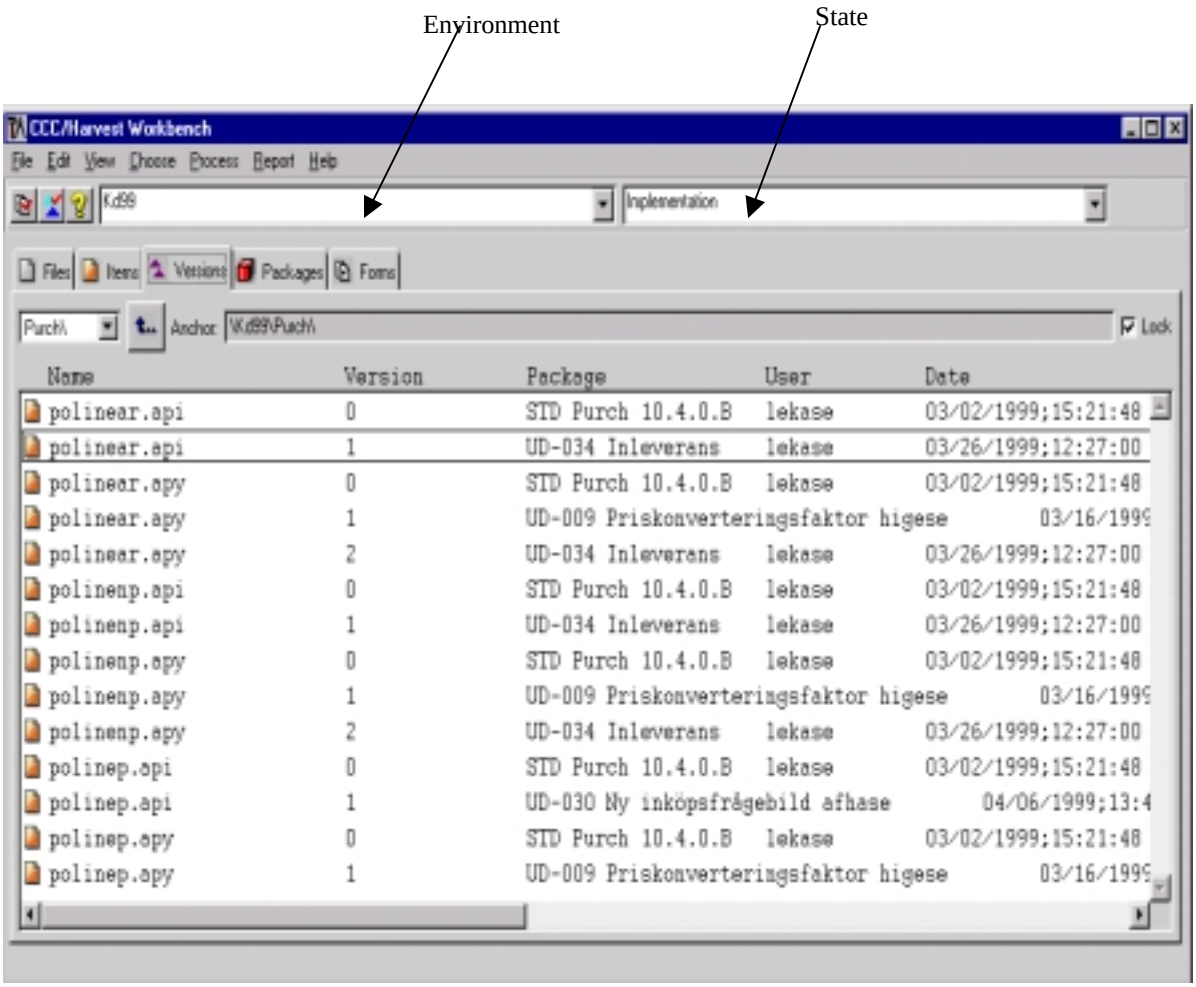
  Items är en filhanterare som visar alla filer som finns i databasen för systemfiler i Harvest. De filer som ska hämtas ut markeras och det är alltid senaste version av filen från valt environment och state som hämtas. Ett lås sätts i den position där filen hämtas. Det som finns under låset i trädstrukturen av filer följer med vid utcheckningen och placeras i den position som har angivits i fliken Files.

  I Items checkar man alltså ut filer som ligger i ett speciellt state. Vad man har som önskemål i Harvesttillägget är att man skall kunna checka ut alla filer (senaste version) från ett speciellt paket / en speciell paketgrupp. Detta går att göra under versions men då får man leta sig fram bland paketen och plocka ut en fil i taget.

  I *Versions* listas alla existerande versioner av varje fil i valt environment och state. 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 4:3 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 och får då ett nytt nummer (ny version).

4.4 Användargrupper i Harvest

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:

1. CM Administrator: Tekniskt ansvarig med fullständiga rättigheter i Harvest. Skapar environments och releaser.

2. Project Administrator: Administrativ projektledare med begränsade rättigheter. Skapar paket i state ToDo.
3. Developer: Systemutvecklare. Rättigheter främst i state Implementation.
4. Approver: Godkänner paket för uppflyttning till nästa state.

5 Beskrivning av Harvesttillägget

Under vår praktikperiod på IFS, fick vi i uppdrag att göra ett tillägg till versionshanterings- verktyget Harvest. Vi fick en utförlig Detaljspecifikation av vår handledare (Fredric Travaglia) som innehöll bakgrund, syfte, vilka problem som finns i Harvest och riktlinjer för tillvägagångssätt för att skapa Harvesttillägget.
I Detaljspecifikationen används namnet Harvest Extension. Senare har vi fått kännedom om att det redan existerar ett Harvest Extension och därför använder vi, tills vidare, namnet Harvesttillägget.
Nedan följer en beskrivning av uppgiften vi fick.

5.1 Bakgrund och syfte med Harvesttillägget

Detaljspecifikation för Harvest Extension innehåller följande bakgrund och syfte:

I Harvest finns strukturer som möjliggör association av ett antal filer (items) till paket. Det går även att koppla samman paket till paketgrupper. Strukturen är i sig mycket användbar men de funktioner som finns i de olika applikationerna, som är tillgängliga för att interagera med Harvestdatabasen, är i ett större sammanhang relativt begränsade och behöver kompletteras för att fungera väl i medelstora till stora projekt. Data om items, paket och paketgrupper samt hur dessa associerar med varandra finns alltså lagrat i Harvestdatabasen. Det finns dock inga bra funktioner i existerande applikationer som gör det möjligt att göra utsökningar på exempelvis paketgrupp och dess items i en viss kundmiljö.

Enligt Detaljspecifikationen var syftet med Harvest-tillägget att skapa ett verktyg som utökar möjligheterna att interagera med Harvestdatabasen, parallellt med övriga applikationer (Harvest Workbench, Harvest Explorer och Harvest Administrator) för att få en bättre kvalitetskontroll av leveranser och installationer i kundprojekten..

5.2 Problemställningar och metoder – Harvesttillägget

Det har även i tidigare examensarbete[1] framkommit att det finns problem i samband med Harvesthanteringen. Två av de fem punkter som togs upp i examensarbetet är:

- 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.

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

Därför fanns önskemål från IFS att tillägget i Harvest skulle underlätta sök- och sorteringsfunktionerna. Mest angeläget var dock en funktion som gjorde det möjligt att checka ut alla filer (senaste version) från samma paket/paketgrupp samtidigt.

[1] ”Versionshantering med inriktning på Harvest på IFS AB”, Lovisa André och Kristina Falkenström, 1999

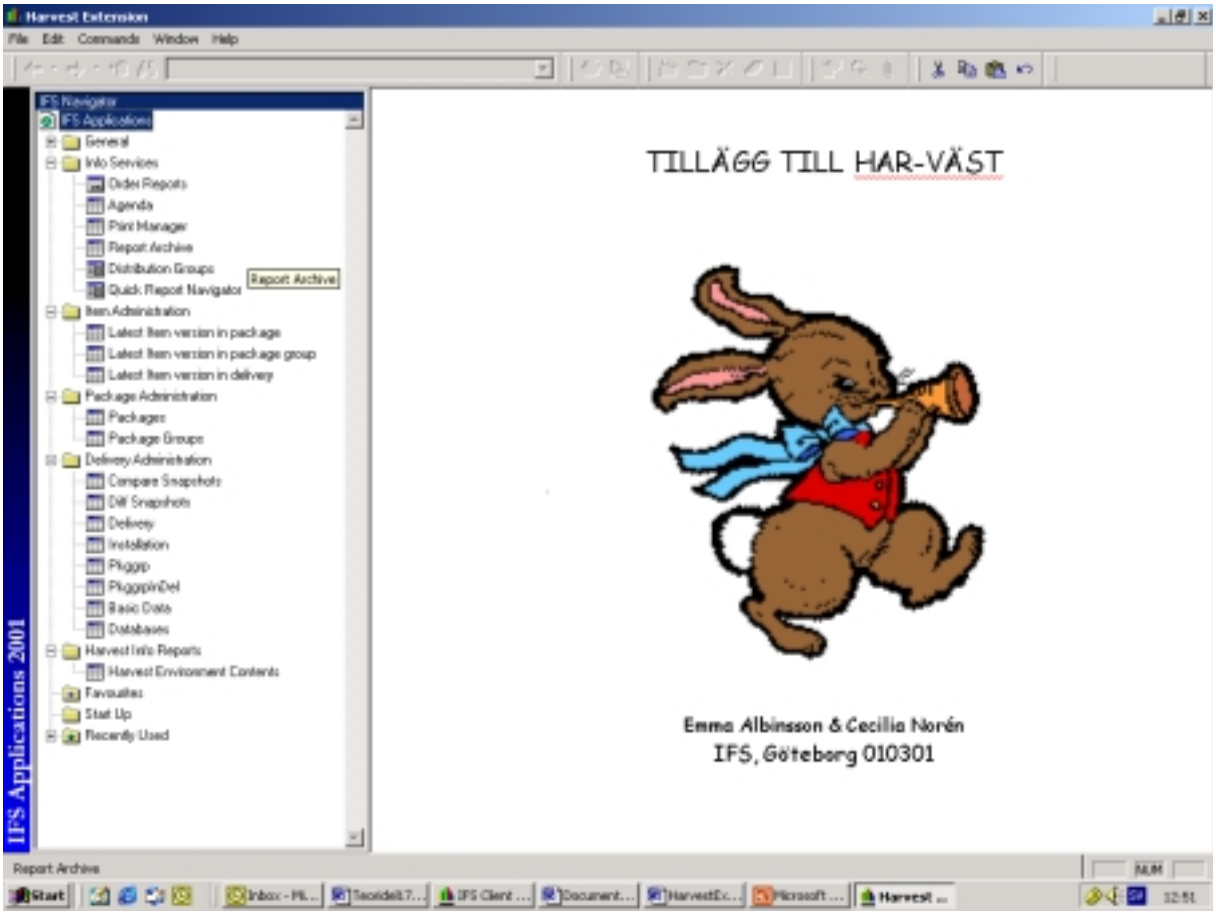
I Harvest Workbench måste man checka ut filerna manuellt eftersom man där söker på environment (kund) och state istället för environment och paket/paketgrupp.
Dessutom var det viktigt att generera en följesedel, en rapport över de paket som tillhör en viss paketgrupp/leverans, så att man lätt skulle kunna se vad som levererats vid en speciell tidpunkt.
Detta innebar att uppgiften delades i två delar som löpte parallellt.
Dels skulle vi utreda och dokumentera hur Harvest lagrar den databasinformation som används. Vi skulle identifiera tabeller och kopplingar samt göra en modell i Rational Rose, som är IFS modelleringsverktyg.
Dessutom skulle vi skapa en klient (applikation) som interagerade med Harvestdatabasen parallellt med övriga applikationer. Detta innebar även att vi fick skapa en del nya tabeller och vyer.

5.3 Disposition – Harvesttillägget

Vi kommer att redovisa klienten (applikationen) och dess funktioner eftersom det är produkten av båda delarna i uppgiften.

5.4 Resultat – Harvesttillägget

I Harvest-tillägget gjordes fjorton nya funktioner (bil.1). Nedan redogörs för de viktigaste funktionerna och varför de underlättar arbetet med Harvest.



Figur 5:1 Harvest-tillägget

5.5 Harvest-tilläggets funktionalitet

Först skapades en navigator, IFS Navigator, som innehåller fem mappar, vilka är Info Services, Item Administration, Package Administration, Delivery Administration och Harvest Info Reports. Under dessa mappar finns ett antal olika funktioner. De andra mapparna General, Favourites, Start Up och Recently Used är defaultmappar, som fanns innan vi började att lägga upp våra egna mappar.
Det har tidigare inte funnits någon typ av rapport. Med hjälp av funktionen Order Reports kan man få en utskrift på de paket som ligger i en viss paketgrupp och därmed tillhör en speciell leverans. Det blir en sorts följesedel. Övriga funktioner under Info Services är defaultfunktioner.

Under mapp Item Administration, Package Administration och Delivery Administration (gränssnitten Compare Snapshots, Diff Snapshot) kan man få fram uppgifter om paket och paketgrupper med tillhörande filer i senaste version. Man kan markera de filer man önskar checka ut och lägga på lämpligt ställe. Dessutom kan man få information om snapshots. All information tas ifrån befintliga tabeller i Harvest. Vår förhoppning är att dessa funktioner kommer att visa sig vara tidsbesparande. Man kan få fram samma information i Harvest Workbench men då måste man gå igenom många fler steg innan man kan checka ut filerna och handhavandet känns osäkrare eftersom det sker manuellt.

Övriga funktioner i Delivery Administration (Delivery, Installation, PkggrpInDel, Basic Data och Databases) är helt nya funktioner där man kan lägga in information. Denna läggs i helt nya tabeller, vilka är kopplade till befintliga Harvesttabeller via vyer. Funktionerna gör det möjligt att exempelvis hålla reda på datum för leveranser och installationer med tillhörande paket, filer osv.

Under mappen Harvest Info Reports finns en funktion som förutom information om paket, paketgrupper, filer och versioner även hämtar information om states. Denna kombination av information kan man inte få i Harvest Workbench.

6 Riktlinjer för Harvest 1999-2001

Under årens lopp har man i KD-projektet utarbetat många olika riktlinjer för hur man skall arbeta i Harvest.
Nedan följer en beskrivning, med hjälp av arbetsmetodik och en flödesmodell av det allmänna arbetssättet för arbetet i Harvest. Detta finns dokumenterat i Harvest guidelines[1]. Därefter följer en beskrivning av arbetsmetodiken i KD-projektet år 1999 samt en flödesmodell, som är hämtat från ett tidigare examensarbete[2]
I resultatdelen följs detta upp av en flödesmodell över KD-projektets arbetssättet i Harvest 2001 samt en intervju med J. Planmo[3] om arbetsmetodiken i Harvest samma år.

6.1 Allmänna Riktlinjer för Harvest 1999

6.1.1 Arbetsmetodik i Harvest

Genom att studera Harvest guidelines, har vi fått veta hur allmänna riktlinjer för arbetsmetodiken i Harvest såg ut år 1999.
Nedan följer en beskrivning över hur det var tänkt att man skulle arbeta med Harvest i hela organisationen, dvs inte i något specifikt projekt. Dessutom visas en modell över flödet.

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[4] 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 kravspecifikationen, flyttas paketet tillbaka till Implementation för rättning. När paketet har godkänts för installation i kundens produktionsmiljö flyttas det till state Release.

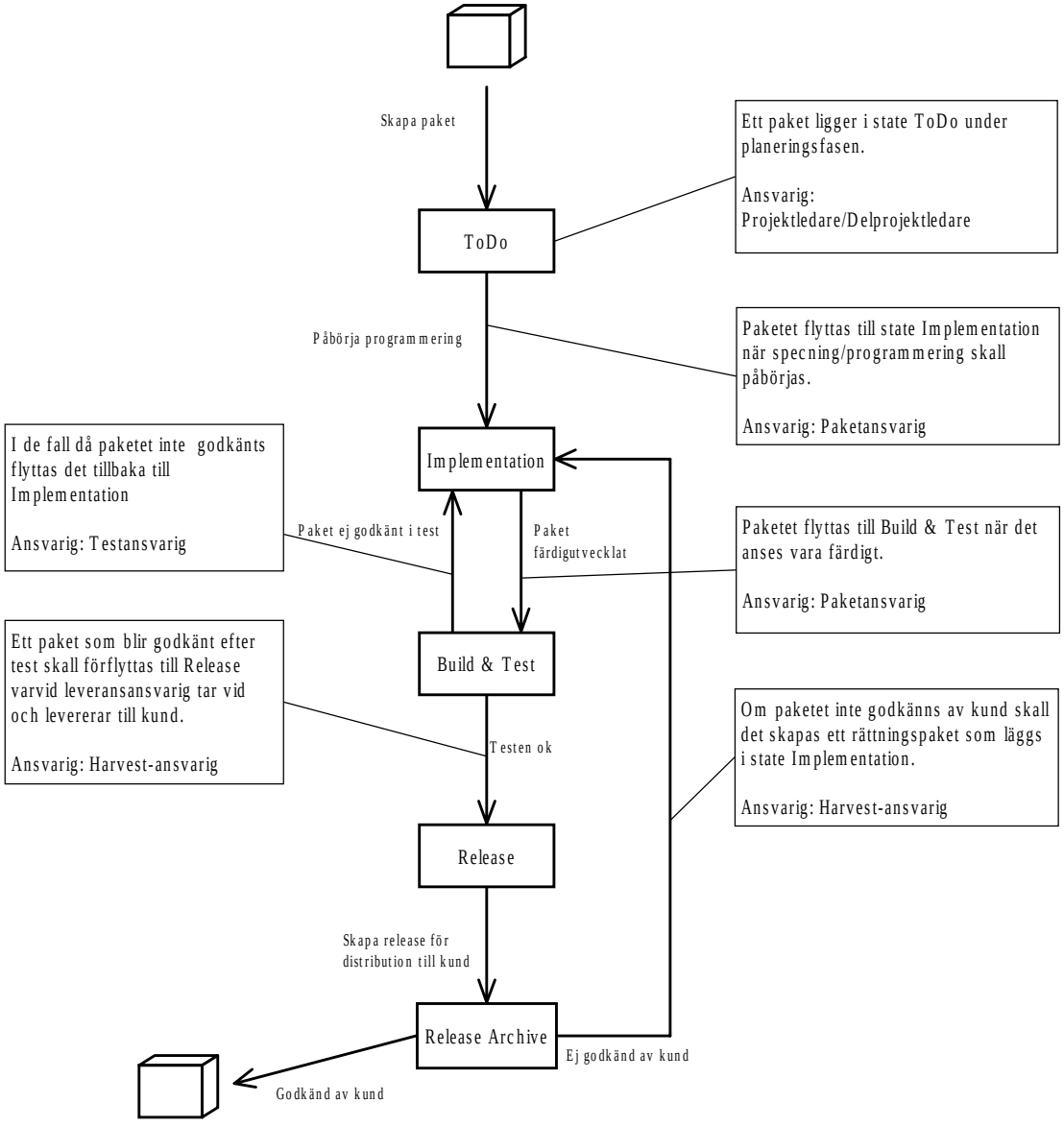
Alla paket som är godkända för leverans ligger förvarade i state Release. När en release skapas, kommer alla filversioner som är kopplade till ett paket att ingå. Vid

[1] IFS intranet
[2] ”Versionshantering med inriktning på Harvest på IFS AB”, Lovisa André och Kristina Falkenström, 1999
[3] Systemutvecklare och projektledare på IFS AB
[4] Kan vara projektledaren, systemutvecklaren eller teknikern

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.

Allmänna riktlinjer för Harvest 1999
-flödesbeskrivning.



Figur 6:1 Flödesbeskrivning för Allmänna riktlinjer 1999 (Harvest Guidelines)

7 Riktlinjer för Harvest KD-projektet 1999

7.1 Arbetsmetodik i Harvest för KD-projektet 1999

Genom tidigare examensarbete[1] har vi fått information om hur man 1999 arbetade med Harvest i KD-projektet. Nedan följer en beskrivning över detta.

I KD-projektet 1999 fick även systemutvecklare tillåtelse att skapa paket, vilket de inte hade fått göra innan.

State ToDo användes inte utan man började utveckla i state Implementation. Ett paket för varje anpassning skapades. Paketet låg sedan kvar i Implementation under hela utvecklings- och testfasen. State Build & Test användes alltså inte. Testningen utfördes av programmeraren och godkändes av den paketansvarige innan paketet flyttades till state Release.

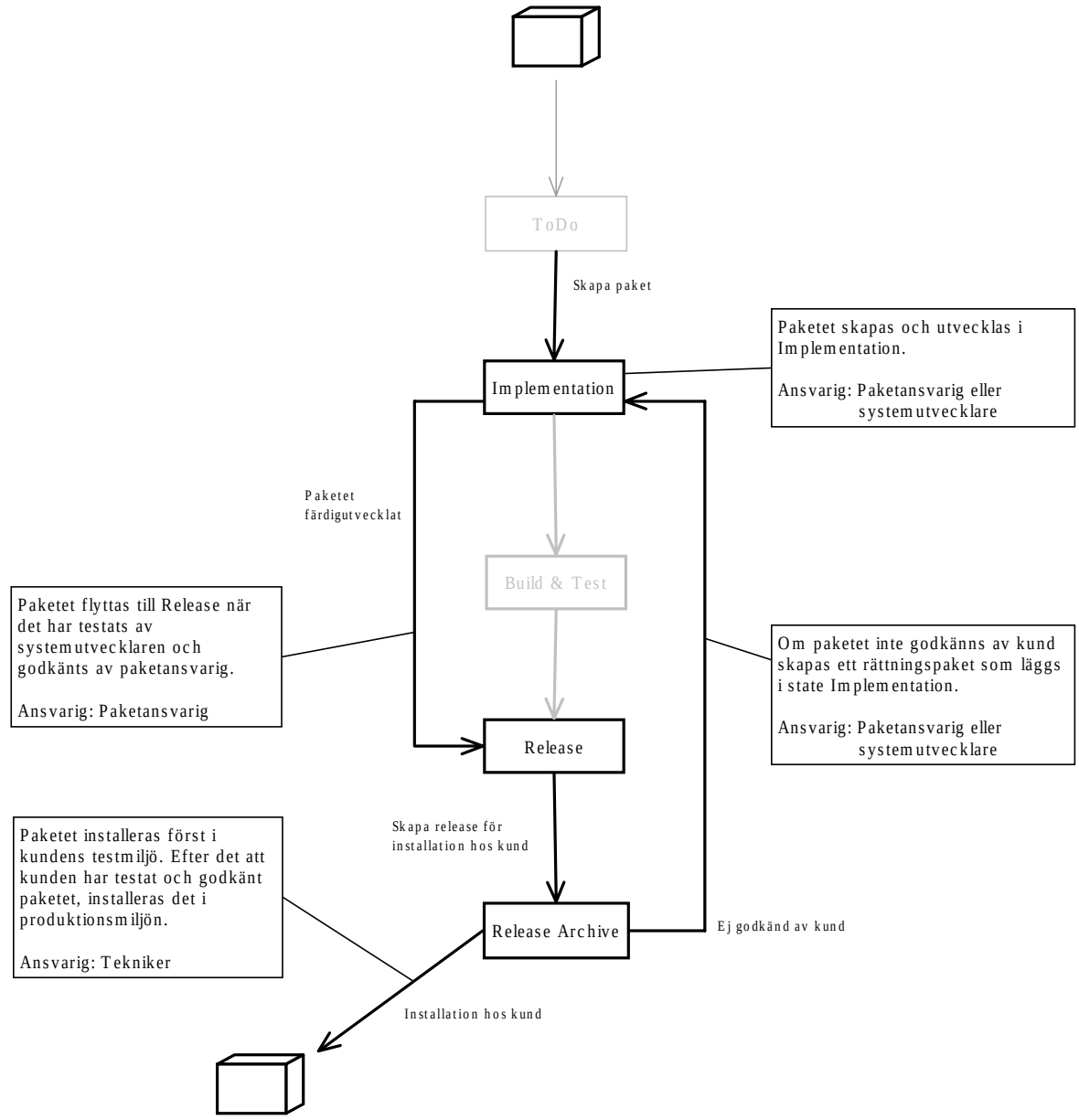
Anledningen till att Build & Test inte användes var att flera anpassningar ofta gjordes i ett fåtal, centrala filer. Om en fil med en ny anpassning checkades in i Harvest och sedan flyttades till Build & Test för att testas, kunde en annan systemutvecklare checka ut filen från Implementation för att göra ytterligare anpassningar i den. Risken var att systemutvecklaren inte upptäckte att det skapats en ny version utan fortsatte att anpassa den nyss testade filen. Det kunde bland annat resultera i att det blev svårt att spåra kompileringsfel. Problemet löstes genom att en fil flyttades från Implementation till Release, utan att state Build & Test användes.

I KD-projektet användes paketgrupper. En paketgrupp skapades för varje planerad leverans till kund. Paket kopplades sedan till aktuell paketgrupp. Genom detta så blev det lättare för teknikern att hålla reda på vilka paket och filer som ingick i en viss leverans.

I state Release Archive skapades, när det var dags för installation hos kund, en kopia av en release med utgångspunkt från de paketgrupper och paket som fanns i state Release.

[1] ”Versionshantering med inriktning på Harvest på IFS AB”, Lovisa André och Kristina Falkenström, 1999

Riktlinjer för Harvest i KD-projektet 1999
-flödesbeskrivning.



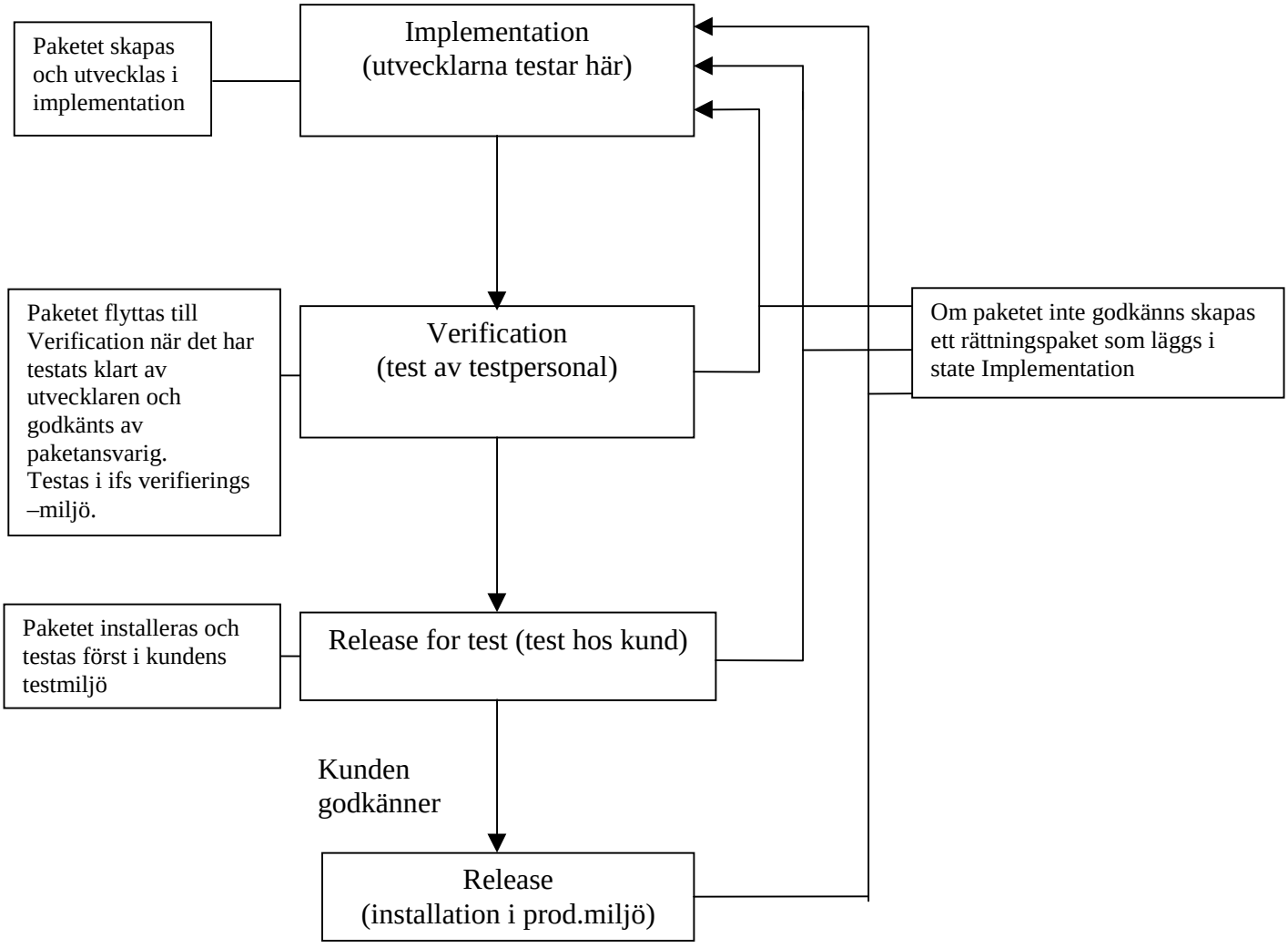
Figur 7:1 Flödesbeskrivning för KD-projektet 1999[2]

[2] ”Versionshantering med inriktning på Harvest på IFS AB”, Lovisa André och Kristina Falkenström, 1999

8 Resultat

8.1 Riktlinjer för Harvest i KD-projektet 2001 - flödesbeskrivning

För att få en uppfattning om hur Riktlinjerna ser ut år 2001 har vi gjort en modell som liknar tidigare modeller, men som beskriver de states som används i KD-projektet idag. Vi utgick ifrån tidigare modeller (se figur 6:1, 7:1) och diskuterade dessa med gruppledare Peter Widén[1], som hela tiden deltagit i KD-projektet och fick då fram följande modell.



Figur 8:1 Flödesbeskrivning för KD-projektet 2001

[1] Gruppchef i KD-projektet och systemutvecklare på IFS AB

8.2 Hur arbetet med versionshanteringsvertyget Harvest fungerar år 2001 i KD-projektet.

Intervju med Applikationskonsult Johan Planmo, Leveransavdelningen IFS AB.

Planmo beskriver Harvest som allmänt ganska bra men nämner att han bara har erfarenhet från två olika versionshanteringsverktyg, där det andra är teckenbaserat och därför tycker han att Harvest är bättre, lättare att förstå. Speciellt positivt är att man kan hålla reda på filer, man får aldrig några korrupta (förstörda/oläsliga) filer och man kan manuellt, på ett säkert sätt, utveckla parallellt, genom att checka ut filer for browse – kopiera – ändra – klistra in - checka in. Däremot är Harvest ganska krångligt att använda och därför utnyttjas inte alla funktioner av alla anställda. Dessutom fattas det en del funktioner som skulle vara tidsbesparande.

Ett av de största problemen är att man inte, på en gång, kan plocka ut alla filer i en paketgrupp. Detta sker istället genom att man manuellt går igenom 3-4 steg och plockar ut en fil i taget. Dessutom finns inget bra stöd för att dokumentera leveranser när en installation gjorts, man kan inte skriva ut följesedlar. Detta är funktioner som saknas och skulle förbättra Harvest betydligt om de fanns.
De funktioner som finns i tillägget har ännu inte satts i bruk. Johan tror inte att tillägget kommer att förändra de riktlinjer som finns för Harvest. De kommer att ligga som ett stöd genom hela utvecklingsfasen främst för teknikerna.

8.3 Hur Riktlinjerna för Harvest har förändrats i KD-projektet

I KD-projektet har riktlinjerna ändrats efter projektets förändring och projektmedarbetarnas erfarenhet. KD-projektet startade år 1997. I början gjordes många ändringar och en del nyutveckling. Under 1999 gjordes mest uppgraderingar. Därefter kom man år 2000 åter in i en nyutvecklingsfas, som år 2001 följdes av en förvaltningsfas. Under förvaltningsfasen arbetar man i huvudsak med tekniskt underhåll, kundsupport och rättning av fel.

I Riktlinjer KD-projektet år 1999 finns en modell (se figur 7:1) som man använde sig av under en period när man mest uppgraderade filer och ingen nyutveckling skedde. Därför klarade man sig bra med denna ganska enkla modell. Både uppgradering och testning skedde i Implementation. Paketet höjdes till Release av teknikern och släpptes därefter till kunden.

I Release Archive fanns paketet antingen i test- eller produktionsmiljö hos kund. Härifrån kunde ett rättningspaket[2] göras om paketet ej godkänts av kund. Att inte veta om paketet ligger i test- eller produktionsmiljö hos kunden är inte bra. När en rättning görs kan det vara risk för att man får med sig fel filer eller missar filer som skall med in i produktionsmiljö. Om för få eller fel filer installeras i produktionsmiljö kan hela systemet avstanna hos kunden, vilket resulterar i att hela produktionen stannar och stora förluster görs.

[2] Ett paket som kopieras, döps om och läggs i Implementation för rättning

8.4 Hur Riktlinjerna för KD-projektet ser ut 2001

I Riktlinjer KD-projektet år 2001 har man kommit in i en förvaltningsfas och använder sig då av ytterligare en modell. Modellen har utvecklats från de behov man hade året innan med täta leveranser och stora avancerade förändringar i standardmodulerna.

Man skapar paket, utvecklar och testar i state Implementation. ToDo-statet hade tidigare tagits bort för man ansåg att det var ett onödigt steg. Man börjar alltid utveckla direkt och skapar då paketen i Implementation. Att ta bort ToDo-statet var en tidsbesparande åtgärd.
Implementation används av systemutvecklare och paketansvarig, vilka också är systemutvecklare. När testningen är klar, höjer teknikern upp paketet till Verification.

I Verification måste man göra en rättningsleverans om teknikern upptäcker fel. Skillnaden på Build&Test och Verification är att man inte kan backa paket från Verification, vilket man kunde i Build&Test, därför är miljön alltid stabil, paketen flyttas inte och test kan göras ostört.

Därefter höjs paketet till Release for test. Detta steg är helt nytt (KD2000) och det gjordes för att det är mycket viktigt att veta huruvida paketet är i test eller i produktion hos kunden.
Om det bara finns ett state för både test och produktion, kan man inte urskilja vilken av filversionerna man skall använda sig av vid en rättningen. Alla versionerna ligger då i samma state och det enda man kan vara säker på är att de ligger i test men man vet inte om de är installerade i kundens produktionsmiljö.
Till sist höjs paketen till state Release, vilket innebär installation i kundens produktionsmiljö.

Tanken med dessa riktlinjer är att det skall finnas ett state för varje miljö (databasmiljö), så att man inte bara känner till state utan även miljö. Man försöker utveckla efter en plan och då skall alla paketen finnas i Verification efter 4 veckors utveckling (Implementation). Därefter ligger paketen 1 vecka i intern test (Verification), 2 veckor i kundtest (Release for test), 1 vecka i omrättning (Implementation) och 1 vecka sluttest av rättningar (Verification).

De användare som finns är systemutvecklare/paketansvarig i state Implementation och tekniker i övriga states.

8.5 Stora skillnader mellan allmänna Riktlinjer och KD-projektet

Sammanfattningsvis kan man säga att det som skiljer sig mellan allmänna Riktlinjer för Harvest och Riktlinjer för Harvest i KD-projektet år 2001 är:

- I KD-projektet 2001 används inte state ToDo, vilket finns med i Allmänna riktlinjer för Harvest.

- Istället för state Build&Test i Allmänna riktlinjer för Harvest används state Verification i KD-projektet 2001.

- Man gör alltid rättningspaket i KD-projektet 2001. Dessa går tillbaka till state Implementation. Man kan aldrig backa ett paket från ett state till ett annat. I Allmänna riktlinjer för Harvest kan man backa ett paket mellan state Implementation och Build&Test.

- I KD-projektet 2001 är de som ansvarar för olika states systemutvecklare eller tekniker. En systemutvecklare kan även vara paketansvarig. I Allmänna riktlinjer för Harvest har även projektledaren ett ansvarsområde i state ToDo.

- De två states Release som finns i KD-projektet 2001 ligger i två olika miljöer: test hos kund och installation i produktionsmiljö. I Allmänna riktlinjer för Harvest finns bara ett state för test och produktion hos kund.

Det som skiljer sig mellan Riktlinjer för Harvest i KD-projektet år 1999 och Riktlinjer för Harvest i KD-projektet år 2001 är:

- I KD-projektet 2001 har man återskapat en testmiljö ”hemma” Verification. Den här testmiljön är, till skillnad från tidigare testmiljö (Build&Test) i Riktlinjer för Harvest i KD-projektet, *både* lugn *och* stabil. Detta innebär att det inte sker någon nyutveckling men även att filerna inte kan backas från Verification.

- I KD-projektet 2001 är de som ansvarar för olika states systemutvecklare eller tekniker.

- De två states Release som finns i KD-projektet 2001 ligger i två olika miljöer: test hos kund och installation i produktionsmiljö. I Riktlinjer för Harvest i KD-projektet år 1999 finns bara ett state för test och produktion hos kund.

8.6 Harvesttilläggets roll i KD-projektets versionhantering

Med underlag från intervjun har vi kommit fram till följande roll och placering av tillägget i flödesbeskrivningen för KD-projektet.

Tillägget kommer troligen inte förändra själva modellen för Riktlinjer men skulle kunna ligga som en platta i botten på modellen.
Man skulle kunna använda sig av tillägget under hela utvecklingsfasen parallellt med Harvest Administration och Harvest Workbench (se figur 8:3). Det kommer att vara teknikern som använder sig av tillägget i första hand.

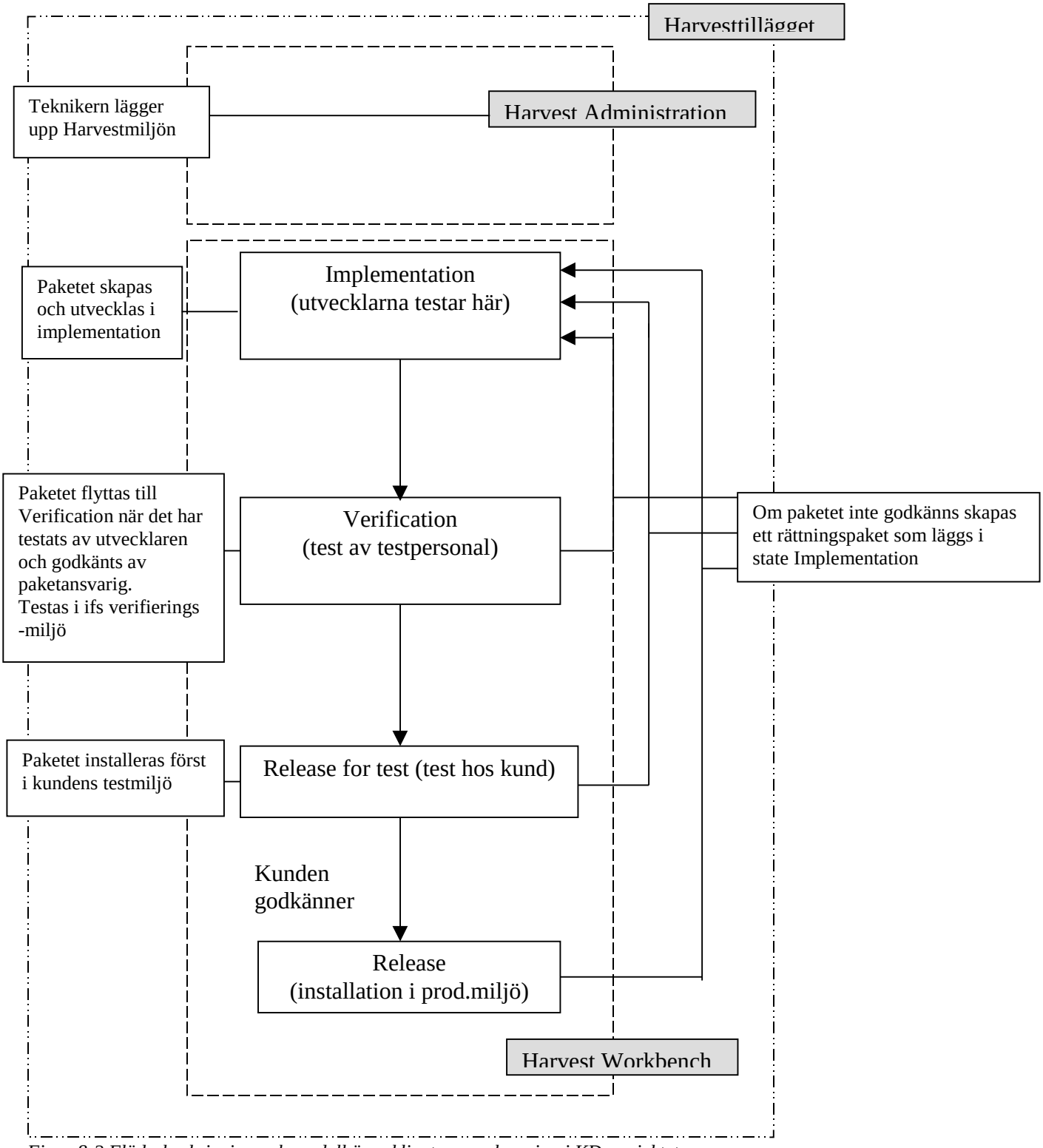
Harvest-tillägget kan främst kopplas till state verification, release for test och release, vid installationer i olika miljöer. Med hjälp av tillägget kommer man att kunna plocka ut alla filer (senaste version) ur en paketgrupp, vilken innehåller en mängd olika paket och installera i rätt miljö (databas). Detta kan göras från ett gränssnitt och teknikern slipper gå flera steg och därefter checka ut samtliga filer manuellt.

Tillägget löser alltså problemen med att man inte kan ta ut en paketgrupp på ett enkelt sätt och att Harvest inte ger något bra stöd för att dokumentera leveranser när en installation har gjorts. Som en följd av detta får man bättre kvalitet av leverans och bättre kontroll på

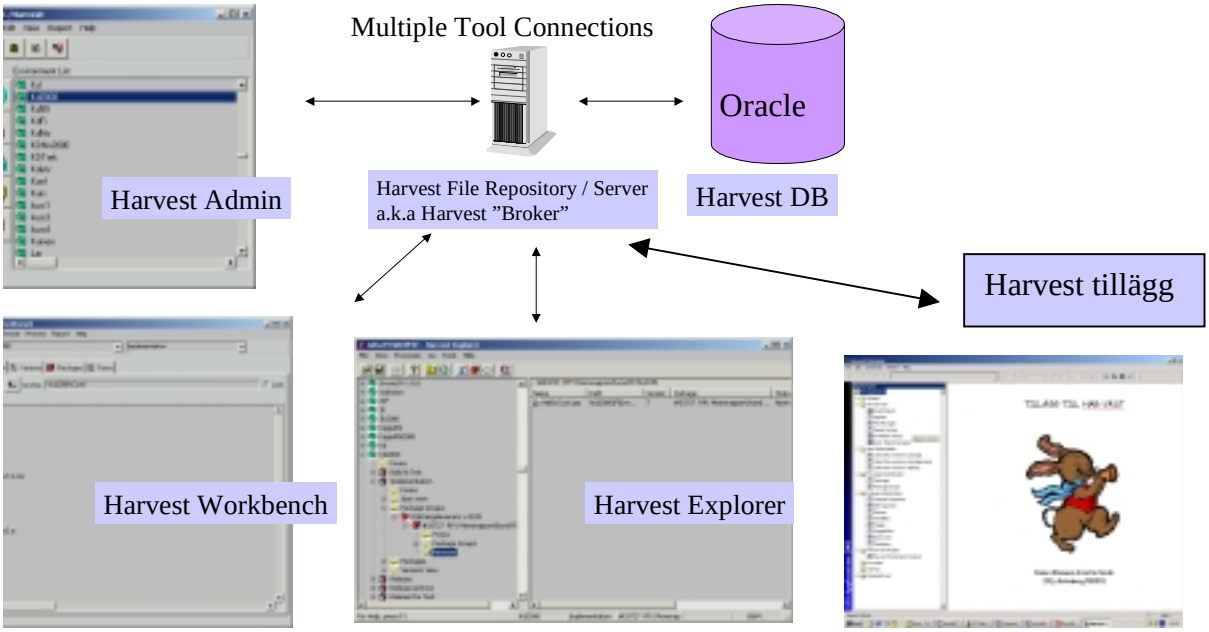
var anpassningarna finns. Det ger också bättre information till kunden genom att tillägget kan generera följesedlar. Dessutom borde det enl. Planmo vara tidsbesparande.

8.7 Riktlinjer för Harvest i KD-projektet 2001

Flödesbeskrivning och modell över klienternas placering.



Figur 8:2 Flödesbeskrivning och modell över klienternas placering i KD-projektet



Figur 8:3: Arkitektur Harvest med Harvest-tillägg

9 Testresultat Harvesttillägget

9.1 Testprotokollet (bil.4):

Det är fyra personer som har genomfört testprotokollet. Av dessa personer är två tekniker och två systemutvecklare.
Direkta fel i programmet:

- Två av testarna har svarat att programmet hänger sig om man har flera fönster öppna samtidigt och försöker populera i ett av fönsterna.
- En testare nämner att när man checkar ut filer, kommer inte .doc-filen med.
- Tre testare har upptäckt att det inte går att checka ut filer från Compare Snapshots, men detta är ett fel i testprotokollet. Man skall inte kunna checka ut filer i detta fönster.

9.2 Kommentarer

Varje kommentar har *en* källa.
Dels är det avsaknad av felmeddelanden som kommenteras. Dessutom finns önskemål om att dölja kolumner som man har valt i huvudet av fönstret. Andra kommentarer är att kikarfunktionen (query) inte fungerar, att det borde finnas funktioner för både check out for update och for browse. En testare tycker att man borde sortera på paket istället för paketgrupper i två olika fönster och att det skulle finnas en kolumn för path i Compare Snapshots. Man har även skrivit bra på funktionen Diff Snapshots.

9.3 Testexemplet

Tre av testarna har kommenterat testexemplet. Den ena personen anser att man sparar ca två timmars arbete genom att använda Harvesttillägget vid en stor leverans med många paket. Den andra testpersonen, som även han är systemutvecklare anser inte att han har tillräckligt med rutin för att uttala sig om hur fort man kan checka ut filer i Harvest. Han kan bara konstatera att tillägget är tidsbesparande pga att man utför färre steg för utcheckning och att man inte kan göra fel när man använder sig av det. En av teknikerna tycker att det är lättare att använda tillägget än Harvest vid utcheckning av filer och uppskattar tidsbesparingen med ca 50%. Den andra teknikern anser också att tilläget sparar tid men en gemensam åsikt från teknikerna är att man vid utcheckning vill ha filerna i samma katalogstruktur som i Harvest.

9.4 Testfrågor

Följande synpunkter kommer från olika testare. Om flera testare svarat samma sak nämns detta i texten.
De problem som elimineras med hjälp av Harvesttillägget är främst det tidsödande arbetet att hämta ut filer från många olika paket. Svårigheter att söka information på ett enhetligt och flexibelt sätt är också ett problem som Harvesttillägget löser. Dessutom tar man bort problemet med att få en bra översikt över en paketgrupp och att få ut den sista versionen av en fil i en paketgrupp.

Två av testarna har skrivit att de fördelar som finns med tillägget är att det är användarvänligt, att man får en bra översikt och att detta ger en bra möjlighet att hitta den information som eftersöks. Två testare anser att tillägget ger ett utmärkt stöd för teknikern som har bra dokumentation av sina leveranser. Dessutom skriver en av testarna att tillägget eliminerar de handhavandefel som kan uppkomma när man själv måste välja senaste versionen av en fil. En testare tror att Harvest-tillägget är lättare att lära sig än dagens Harvest. Han tycker att det är bra att man i ett formulär kan markera och kopiera t ex en paketgrupp, för att sedan, i t ex ”Utforskaren”, skapa en ny mapp med detta paketgruppsnamn.

De viktigaste synpunkterna om vad som inte är bra med tillägget är dels att användargränssnittet inte håller IFS-standard. Utformningen av vissa formulär kan förbättras och dessutom tappar man viss funktion som finns i Harvest. En annan aspekt är att man inte vet vilket state de olika paketen befinner sig i.

Vad man i huvudsak skulle vilja förbättra i Harvesttillägget är bland annat katalogstrukturen. Två personer tycker att det skulle vara bra om man kunde behålla katalogstrukturen när man checkar ut filerna. Dessutom skulle en testare vilja se rapporten ihop med gränssnittet ”Latest item version in delivery” för att se vilka filer som ingår i en rättningsleverans. I rapporten skulle han t ex också vilja se filer, versioner och datum. Han anser att både checkout for browse och checkout for update borde finnas i tillägget. Nu kan man endast granska en fil, uppdatera får man göra i Harvest.
Övriga kommentarer till vad testpersonerna skulle vilja förbättra (bil.4).

Alla testare tycker att Harvesttillägget gör arbetet generellt mer tidsbesparande än när man bara har Harvest att tillgå. En av dessa anser att det vid paketleveranser sparar mycket tid, men att det vid små leveranser av enstaka rättningar går fortare att arbeta direkt i Harvest. En annan testare anser att tillägget är tidsbesparande vid en installation. En testare anser att funktioner som diff snapshots och jämförelse av snapshots är mycket tidsbesparande för en tekniker, då det finns behov av att få fram denna information. Ytterligare en annan tror att tillägget är ett utmärkt stöd för teknikern som har bra dokumentation av sina leveranser. Det ger även högre leveranssäkerhet och kvalitet.

En av testpersonerna har funderingar över vilket eller vilka states i Harvest, som tillägget arbetar mot och vad som händer om man gör förändringar i arbetssättet som i dag finns i Harvest.

10 Diskussion

10.1 Resultatdiskussion

Det har varit en del svårigheter att sätta sig in i hur man generellt arbetar med Harvest på IFS.
Vi studerade de allmänna riktlinjerna men upptäckte att dessa inte följdes helt och hållet. Olika projektgrupper och yrkeskategorier hade olika förhållningssätt till Harvest. Detta känns naturligt nu när vi är bättre insatta i hur olika yrkesgrupper arbetar. Att utvecklarna är fokuserade på Implementation och teknikern har en lite mer övergripande roll och lägger mest tid i senare skeden av programvaruutvecklingen.
De olika projektgrupperna håller sig inte heller till allmänna riktlinjer. Det har vi fått reda på när vi har träffat tekniker från olika projekt. Detta beror på projektens storlek, ju större projekt desto mer krav på att ha en minutiöst noggrann kontroll på filerna. Detta kan resultera i en modell, medan det lilla projektet har en helt annan modell. Det beror även på var i själva utvecklingen projektet befinner sig. Det har framkommit när vi studerat KD-projektets modeller/riktlinjer.
Till en början kunde man tolka de olika modellerna som steg i utvecklingen av arbetsmetodiken. Det är snarare så att den Harvestutveckling som har skett i KD-projektet under de senaste två året handlar mest om att projektet har gått in i olika faser och inte så mycket om att man har utvecklat arbetsmetodiken generellt. Det som har framkommit är dock att man är i behov av att ändra arbetsrutinerna beroende på vilken fas man befinner sig i. Det handlar om en kontinuerlig process som fortgår så länge projektet finns kvar. Ett fåtal förändringar har man dock gjort som man kommer att ha med sig till andra projekt. Det är bland annat att man byter miljö beroende på vilket state man befinner sig i. Det känns som en självklarhet att man måste veta var filerna befinner sig, att man handskas med rätt filer och att de stödjande filerna finns på plats både för att kunna testas och därefter läggas in i produktion. Detta är ett svårt problem att komma till rätta med vid utveckling i stora projekt men KD-projektet har kommit långt när det gäller detta. Man arbetar flitigt för att kunna hantera ett så omfattande projekt. Det verkar även finnas ett extra stort intresse för versionshantering i gruppen och att man har insett fördelarna med att kvalitetssäkra och på så vis få nöjda kunder.

De problem som finns kvar att jobba med i KD-projektet finns alltså inte direkt i den teoretiska arbetsmetodiken eftersom den hela tiden förändras och man har bra modeller för hur utveckling skall ske. De största problemen borde ligga i den praktiska delen. Att ha bra verktyg för hur man exempelvis går till väga för att snabbt och säkert levererar en paketgrupp.

Hypotesen att Harvesttillägget kommer att förenkla arbetet i Harvest för anställda på IFS är delvis sann. Tillägget har enligt test visat sig vara ett bra komplement till övriga versionshanteringsverktyg på IFS, speciellt för teknikerna. Man har dock en del synpunkter på förbättringar som borde göras och detta kräver ytterligare några veckors utveckling. Den tidsvinst och den ökade säkerheten som tillägget ger vid leverans är dock av sådana proportioner att man borde beakta en vidareutveckling/förbättring av tillägget. En del synpunkter på förbättringar som framkom vid testet av Harvesttilläget var klara buggar. Att .doc-filerna inte checkades ut är exempelvis en bugg som beror på att dessa filer innehåller många blanksteg och därför sorterades bort vid utcheckning.

Däremot framkom en del synpunkter på förbättringar där man får en känsla av att testaren inte riktigt förstår sig på funktionerna i tillägget. En av dessa synpunkter är att man även skall kunna checka ut filen för update. Detta kan man göra, på ett bra sätt i Harvest Workbench och därför anser vi inte att det är nödvändigt att kunna göra det i tillägget. Den typen om delad uppfattning om vad som skall ingå i tillägget och delvis hur funktionerna fungerar är svåra att komma till rätta med. Vi har gjort ett Harvesttillägg som beställts av *en* tekniker och speglar *hans* sätt att arbeta i Harvest. För att komma till rätta med problemet borde teknikerna prata ihop sig och skapa gemensamma arbetsrutiner för leveranser och installationer.

De personer som inte arbetar med leveranser/installationer har inte lika stor användning av tillägget utan klarar sig bra med Harvest Workbench. Dels har man ingen användning av alla funktioner som finns i tillägget och dessutom är man oftast beroende av att checka ut filer för update, vilket man inte kan göra i tillägget. Trots att systemutvecklare inte har så mycket nytta av tillägget så är man ändå mycket positiva till det. Den uppfattning vi har fått är att det känns tryggt även för utvecklarna att få säkra leveranser. Oftast är det utvecklarna och konsulterna som får stå till svars om kunden har problem med programvaran. Med tillägget kan man kontrollera exakt, på ett smidigt sätt, vad man har levererat vid en viss tidpunkt. Detta är positivt för både tekniker och utvecklare eftersom man snabbt kan se var felet ligger. Exempelvis om det är fel fil som levererats eller om det är en bugg i koden.

Att försöka få frågan om tillägget påverkar arbetsmetodiken i Harvest besvarad, var inte helt enkelt, då det inte var någon som med säkerhet kunde svara på detta. Olyckligtvis har vår handledare, som vi hade under praktikperioden när vi utvecklade tillägget, teknikern Fredric Travaglia, vistats i Frankrike under hela tiden vi har skrivit examensarbetet. Detta har resulterat i att det har varit svårt att förklara exakt hur tillägget är tänkt att fungera och att göra relevanta test, eftersom det är byggt på ett sätt som speglar *en* teknikers sätt att arbeta i Harvest. Alla tekniker har olika sätt att arbeta på. Detta avspeglar sig i testprotokollet där testarna har ganska olika kommentarer om hur tillägget fungerar.

10.2 Metoddiskussion

Att hitta litteratur i ämnet var inga problem. När det gällde de teoretiska delarna om arbetsmetodik och flödesmodeller var det också enkelt att få ett bra material bland annat genom intervjun med J. Planmo.
Däremot hade vi stora problem att göra ett trovärdigt testexempel, då ingen av oss vet exakt hur en tekniker arbetar på IFS och då tekniker F. Travaglia inte var disponibel pga utlandarbete. Vi skulle hålla mail-kontakt men detta fungerade dock inte av olika anledningar. Vi fick istället hjälp av en tekniker Claes Håkansson som hjälpte oss mycket men tyvärr förstod inte han heller exakt hur tillägget skulle användas och därför är inte testexemplet så omfattande som vi hade hoppats på. Vi fick ändå ihop ett jämförande testexempel som gick att applicera på verkligheten och det var det viktigaste. Eftersom det finns ett begränsat antal tekniker på IFS och vi blev tvungna att stryka Claes Håkansson från vår lista så blev det en ganska liten testgrupp.
En reflektion är att fler tekniker skulle testat tillägget, men det fanns inte fler att tillgå som både hade kunskap om Harvest och tid att disponera.
Dessutom hade den optimala testningen varit ett verklighetsanknutet test enligt en mall, där testaren går igenom *alla* funktioner på tid precis som de är tänkta att användas och

därefter jämförde detta med ett liknande test av Harvest. Detta skulle gjorts med ca 10 testpersoner och hade då blivit en bättre utvärdering.
Under rådande omständigheter blev testet ganska subjektivt men ger ändå en klar indikation om att tillägget är tidsbesparande och med vissa justeringar kommer att bli ett mycket bra kompletterande verktyg för tekniker speciellt.

10.3 Efterkommande forskning

Det hade varit intressant att göra en större, uppföljande utvärdering av Harvesttillägget en tid efter alla rättningar är gjorda och Harvesttillägget satts i drift.

En annan intressant studie är att göra ett helt modellarkiv av KD-projektets alla faser. Vi har studerat KD-projektet år 1999 och år 2001 för att vi trodde att man kunde se en utveckling av arbetsmetodiken mellan dessa år. Det gjorde vi men mer intressant hade kanske varit att göra en modell av riktlinjer för Harvest, för varje fas ett projekt går igenom, hur man använder sig av livscykeln och vilka steg som är viktiga i de olika faserna.

10.4 Sammanfattande slutsatser

De förändringar som förekommit i riktlinjerna för Harvest i KD-projektet handlar till största delen om att projektet genom åren har gått in i olika faser och då kräver olika modeller för riktlinjerna.
Det har även gjorts några förändringar som är permanenta och fasoberoende. Den största förändringen är att varje state ligger i olika databasmiljöer.
Vad som kvarstår i utvecklingsarbetet är ett väl fungerande Harvesttillägg, som stöd för snabb och säker leverans av paketgrupper. Där är arbetet på god väg men kräver ytterligare några veckors utveckling efter att teknikerna har kommit överens om en kravspecifikation.

Referenser

Litteratur

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

Eklund, Sven; Fernlund, Hans, *Programkonstruktion med kvalitet*. Lund: Studentlitteratur. 1998.

Fletcher J. Buckley, *Implementing Configuration Management*, 2nd Edition. Los Alamitos, California: IEEE Computer Society Press. 1996.

Ince, Darrel, *ISO 9001 and Software Quality Assurance*. London: McGraw-Hill Book Company. 1994.

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

Oskarsson, Östen; Glass, L. Robert, *ISO 9000 i programutveckling*. Lund: Studentlitteratur. 1995.

Sommerville, Ian, *Software Engineering*, 6th Edition. Harlow, Essex: Addison Wesley Longman Limited. 2001.

Material från Internet

IFS:s internationella websida, maj 2001:
IFS Industrial & Financial Systems – IFS Home
URL: http://www.ifsab.com/

Opublicerat material: C-uppsatser

André, Lovisa; Falkenström, Kristina, *”Versionshantering med inriktning på Harvest på IFS AB”*. Göteborg: Göteborgs universitet, Institutionen för informatik, 1999.

Andersson, Marie; Bergand, Maria, *”Versionshantering – riktlinjer för anskaffning av verktyg”*. Göteborg: Göteborgs universitet, Institutionen för informatik, 1998.

Kontaktpersoner på IFS i Göteborg

Intervju:
Planmo, Johan, systemutvecklare i KD-projektet.

Testpersoner:
Johansson, Ralf, tekniker i KD-projektet.

Litzén, Stefan, tekniker.
Planmo, Johan, systemutvecklare i KD-projektet.
Svensson, Peter, systemutvecklare i KD-projektet.

Övriga:
Håkansson, Claes, tekniker.
Widén, Peter, gruppledare.

1. Mapp: Info Services

- **Funktion: Order Reports**
  Här ska användaren kunna generera en följesedel/rapport för en specifik leverans. Men först måste man välja en paketgrupp.

2. Mapp: Item Administration

- **Funktion: Latest Item version in package**
  Utsökning av de senaste item-versionerna i ett paket som finns i en viss miljö(kundprojekt).

- **Funktion: Latest Item version in package group**
  Utsökning av de senaste item-versionerna i en paketgrupp, i en viss miljö.

- **Funktion: Latest Item version in delivery**
  Utsökning av de senaste item-versionerna i en leverans. Man ska kunna välja en viss leverans, (som är detsamma som ett paketgruppsnamn). Utifrån denna leverans ska man kunna få den senaste versionen på de items den innehåller.

3. Mapp: Package Administration

- **Funktion: Packages**
  Listning av paket, som ingår i den miljö man arbetar i. Om man inte registrerar en specifik miljö, så listas *alla* paket som ingår i Harvest.

- **Funktion: Package Groups**
  Listning av paketgrupper, som ingår i den miljö man arbetar i. Om man inte registrerar en specifik miljö, så listas *alla* paketgrupper som ingår i Harvest.

4. Mapp: Delivery Administration

- **Funktion: Compare Snapshots**
  Val av en specifik miljö och två olika snapshots, som man vill jämföra. Utifrån dessa val ska alla item-versioner i Snapshot1 och Snapshot2 listas. (Även de versioner som är lika visas.)

- **Funktion: Diff Snapshots**
  Listning endast av de item-versioner som inte är lika i Snapshot1 och Snapshot2. Man vill kunna se versions- eller itemdifferensen mellan de två snapshotsen man valt i en viss miljö.

- **Funktion: Delivery**
  Registrering av leveransdatum och leveransnamn för varje leverans. Detta gör man för att hålla ordning på vid vilket datum varje leverans

har gjorts. Det går också att få veta vilka leveranser, som har gjorts, och vid vilket datum.

- **Funktion: Installation**
  Registrering av nya installationer och listning av alla installationer, inklusive installationsid, databas, installationsdatum och leverans. Teknikerna kan hålla ordning på när de har gjort en installation på en databas.

- **Funktion: PkggrpInDel**
  Registrering och listning av paketgruppsid och leveransid. Här kopplas ett antal paketgrupper till en leverans (ex. paketgrupperna 'leverans 3' och 'slutleverans 1' till leveransen 'KD010601').

- **Funktion: Basic Data**
  Registrering av miljö, databas och projektnamn. Detta ska göras en gång för varje nytt projekt. Genom registrering här har andra formulär tillgång till informationen man lagt in.

- **Funktion: Databases**
  Listning av redan registrerade databaser. Teknikern kan också registrera en ny databas, som det t ex ska göras en installation på. När man har registrerat en databas så kan man i installationsformuläret lista de databaser som är tillgängliga, för att man vill göra en installation på någon utav dessa.

5. Mapp: Harvest Info Reports

- **Funktion: Harvest Environment Contents**
  Val av en specifik miljö för att få reda på vilka paket som finns i den. Man får också reda på de paketgrupper som paketen är kopplade till, de items som hör till paketen, staten där items befinner sig och versionerna på items.

Intervjufrågor Bilaga 2

Övergripande intervju-frågor om Harvest

- Hur fungerar / används Harvest i dag?
- Vad borde förbättras?
- Vilka Harvest-förändringar har gjorts, i KD-projektet mellan år 1999 och 2001?
- Varför har dessa förändringar gjorts?
- Hur uppfattar du Harvest Extension?

Intervjufrågor:

1. Hur tycker Du, i stora drag, att Harvest fungerar?
2. Finns det några problem med Harvest och i så fall vilka?
3. Finns det något som du skulle vilja förbättra?

4. Kan du med hjälp av bilden (Riktlinjer KD-projektet år 1999) beskriva hur man arbetade i Harvest år1999?
5. Kan du med hjälp av bilden (Riktlinjer KD-projektet år 2001) beskriva hur man arbetar i Harvest år 2001?
6. Varför är detta arbetssätt bättre än tidigare Riktlinjer?
7. Varför används inte Build & Test och ToDo staten?
8. Varför är Verification och Release for Test bättre?
9. Varför används inte demote (förflyttning bakåt)?

10. Var (om det är möjligt) skulle du vilja placera Harvest Administration, Harvest Workbench/Explorer, Harvest Extension på bilden om Riktlinjer KD-projektet 2001?

11. Vilka användargrupper finns (ex. CM Adm.?, Project Adm.?, Approver? och hur använder de sig av Harvest?
12. Hur använder sig tekniker, su, projektledare, gruppledare av Harvest?

13. Hur uppfattar du Harvest Extension?
14. Vad mer skulle kunna förbättras i Harvest?

Bilagor: Riktlinjer flödesbeskrivning år 1999.
Riktlinjer KD-projektet år 1999
Riktlinjer KD-projektet år 2001

Testinstruktioner för Harvesttillägget Bilaga 3

För att kunna köra Harvesttillägget måste du ha följande filer och inställningar:

- test2.exe – körs i ett runtime-bibliotek byggt på foundation 300
- sql.ini:
- tnsnames.ora:
- HEPKGGRPINDEL.QRP
- username, password, database – PLATINUM, HARVEST, KDHEM
- använd environment KDNo2000 när du testkör

Sökväg till filerna: KD2000onGbglev9_work\HarvestExtension\work\he

Fyll i testprotokollet (ligger på samma ställe som övriga filer)

Gör följande exempel:
Exemplet går ut på att man simulerar första delen av en installation. Det är paketgrupp Rättningsleverans 25 000719, som finns i KDNo2000, som skall checkas ut.

1. Öppna Harvest Workbench.

- Arbeta från state Release (i vanliga fall Release for test). Detta måste du göra för att senare kunna jämföra resultatet från Harvest Workbench med resultatet i Harvesttillägget, vilket är kopplat mot en kopia av Harvestdatabasen.
- Checka ut filerna och räkna antalet.

2. Öppna Harvesttillägget

- Under mapp Item Administation finns formuläret Latest Item version in Package group
- Välj miljö: KDNo2000 och paketgrupp: Rättningsleverans 25 000719
- Checka ut filerna genom att markera och använda höger musknapp. Räkna antalet filer.

Vår förhoppning är att du fick ut 53 filer (senaste version) i båda fallen men att du upplevde att det gick lättare att checka ut filerna med hjälp av Harvesttillägget. Kommentera exemplet.

Svara på följande frågor:

1. Vilka tidigare problem tycker du elimineras med hjälp av Harvest-tillägget?
2. Finns det några fördelar med tillägget? Vilka?
3. Finns det något som inte är bra med tillägget?
4. Vad skulle du vilja förbättra?
5. Hur tycker du att utformningen är?
6. Är det användarvänligt eller är det svårt att förstå hur det skall användas?
7. Är tillägget tidsbesparande?

Tack för hjälpen!
Cecilia/Emma

Övriga kommentarer till vad testpersonerna skulle vilja förbättra i Harvesttillägget

- I de formulär som utgår från Environmentname i Basic Data bör environmentet presenteras i huvudet på formuläret så att man vet att man jobbar med rätt projekt.
- Man borde kunna byta environment mha av höger musknapp i formulären.
- ”Query Dialog” måste fungera i alla formulär.
- Tillägget är inkonsekvent på en del ställen, eftersom det i ungefär hälften av formulären finns en combobox där man ska välja environment, medan man i övriga arbetar mot en standardinställning i Basic Data. Det vore önskvärt att ha samma hantering i så stor utsträckning som möjligt.
- Formuläret för Basic Data borde ligga i en egen mapp, högst upp i navigatorn så att man hittar dit snabbt.
- I gränssnittet ”Packages” skulle det vara bra om man sorterade på paketnamn och i gränssnittet ”Compare Snapshots” hade det underlättat om en kolumn för sökvägen, ”viewpath” fanns med, så att man kan se var saker ligger någonstans.
- I ”Diff Snapshots” hade det varit bra, om det inte finns någon version, att det stod ”Finns ej” eller något liknande. I ”Harvest Environment Contents” är det önskvärt att man sorterar alla kolumnerna, speciellt sortering av filerna under paketnamn.

Ordlista

Approver – Godkänner paket för uppflyttning till nästa state

Backward delta – Den senaste filversionen finns lagrad i sin helhet

Branch – Skapas då en fil vidareutvecklas vid sidan om filens egentliga utveckling

Build & Test – (State i utvecklingscykeln där man testar programvara)Testningssteget

Check in existing – Process i state Implementation för att lägga in en existerande fil i Harvest

Check in new – Process i state Implementation för att lägga in en helt ny fil i Harvest

Check out for browse – Process i state Implementation för att plocka ut en fil för granskning

Check out for update – Process i state Implementation för att plocka ut en fil för uppdatering

CM Administrator – Tekniskt ansvarig vid användandet av Harvest

CMS – Teckenbaserat versionshanteringsverktyg

Demote – Förflyttning bakåt till föregående state

Developer – Systemutvecklare

Environment – En utvecklingsmiljö i Harvest

Files – Filhanterare i Harvest Workbench, speglar filstrukturen på disk och på utvecklingsservrar

Forward delta – Den ursprungliga filversionen finns lagrad i sin helhet

Följesedel – En rapport över vilka paket som ingår i en specifik leverans

Harvest – Ett verktyg för versionshantering av filer och paket

Harvest Administration – Klient/användargränssnitt, här sker skapandet av Harvestmiljön

Harvest Broker – Serverapplikation

Harvest Explorer – Senare framtaget gränssnitt av Harvest Workbench

Harvest Workbench – Klient/användargränssnitt, här sker själva versionshanteringen

IFS – Industrial & Financial Systems

Implementation – Utvecklingssteget

Item – En fil i Harvest

Items – Filhanterare i Harvest Workbench, visar alla filer som finns i databasen för systemfiler

KD – Kronans Droghandel

Merge – Sammanslagning av filversioner

Packages – En view(vy) i Harvest Workbench, listar de paket som finns inom valt state och environment

Paket – Skapas t ex för en ny funktion i produkten, förändringar i filer kopplas hit

Paketgrupper – Samlingspaket för flera paket, leveranstillfälle, paket kan kopplas hit

Parallell utveckling – Utveckling av fil i två olika riktningar oberoende av varandra

Populera – Fylla kolumner med information

Project Administrator – Administrativ projektledare vid användandet av Harvest

Promote – Förflyttning framåt till nästa state

QDS – Quality Development System, IFS:s kvalitetssystem

Release – En version av produkten, som släpps till kund, klar för installation hos kund (-steget)

Release Archive – Tidigare releaser (-steget)

Release for test – Test hos kund (-steget)

Repository – Bibliotek i en databas

Rättningspaket – Skapas om paketet inte godkänns, läggs i state Implementation

Share – Koden finns lagrad på ett ställe, tillgänglig för flera utvecklare samtidigt

Snapshot – En ögonblicksbild av filerna i state Release vid en viss tidpunkt, innehåller de versioner som releasen innehöll vid det tillfället

State – Steg i livscykeln för utvecklingsarbetet

ToDo – Planeringssteget

Transitions – Förflyttningar, mellan olika states i livscykeln

Verification – Test av testpersonal (-steget)

Version – En variant av produkten

Versions – En view(vy) i Harvest Workbench, listar alla existerande versioner av varje fil i valt environment och state

Versionsnummer – Unikt nummer på en version