Avdelningen för datavetenskap vid Institutionen för Informatik och Matematik

EXAMENSARBETE 2004:DS01

Utveckling och implementering av sekvenshanteringssystem åt leverantörer till bilindustrin.

Erik Ragnarsson

HÖGSKOLAN
TROLLHÄTTAN • UDDEVALLA

EXAMENSARBETE

Högskolan Trollhättan · Uddevalla
Institutionen för Informatik och Matematik

Uppsats för filosofie kandidat i Datavetenskap

Utveckling och implementering av sekvens hanteringssystem åt leverantörer till bilindustrin.

Erik Ragnarsson

Examinator:
Stefan Christiernin Institutionen för för Informatik och Matematik

Handledare:
Andreas Boklund Institutionen för för Informatik och Matematik

Trollhättan, 2004

EXAMENSARBETE

Utveckling och implementering av sekvens-hanteringssystem åt leverantörer till bilindustrin.

Erik Ragnarsson

Sammanfattning

Leverantörer till industrin får fler krav på sig när tillverkarna försöker optimera sina lagerutrymmen, kvalitet och produktionshastighet. När kraven ökar är det viktigt att inte belasta de anställda med ansvaret att alltid veta om de gör rätt och därför bör automatiska kontroller införas.

Detta examensarbete handlar om hur ett sådant system designas, utvecklas och sätts i bruk åt en underleverantör till Volvo Car Corporation. Projektet utfördes för ett företag i Trollhättan som heter Supplier Partner AB och de har redan ett system som inte går att använda till andra projekt inom sekvensproduktion.

Systemet som skall utvecklas har som ett fungerande system som utgångspunkt. Detta skall förändras för att kunna återanvändas i flera nya projekt om det skulle behövas i framtiden. Systemet skall vara enkelt att vidareanpassa om så skulle vara tvunget.

Publisher: University of Trollhättan · Uddevalla, Department of Informatics and Mathematics
Box 957, S-461 29 Trollhättan, SWEDEN
Phone: + 46 520 47 50 00 Fax: + 46 520 47 50 99
Examiner: Stefan Christiernin
Advisor: Andreas Boklund, HTU
Subject: Computer science **Language**: Swedish
Number: 2004:DS17 **Date**: Maj 17, 2004
Keywords Sequence production system, EDI, logistic

DEGREE PROJECT

Development and implementation of a sequence production system for suppliers to car manufactures

Erik Ragnarsson

Abstract / Summary

Suppliers to industries must answer to the higher demands when the manufactures are optimizing their stock capacity, quality and the speed of production. When the demands increase is it important to ease the workload for production workers. Automatic verifications of the goods will take some of the responsibility of them.

This exam work is about how such a system is designed, developed and started for a sub supplier to Volvo Car Corporation. The project was implemented for a company in Trollhättan, Supplier Partner AB and they already have a system that is not generic enough to adept to the new environment.

The system that will be developed has a working system to start from, the changes that must be made is for creating an adaptable system for reusing in new projects. The system must support future changes.

Utveckling och implementering av sekvens-hanteringssystem åt leverantörer till bilindustrin.

Förord

Projektgrupp

Projektet som utvecklingsprojekt har blandat in fler människor än jag själv. Dessa personer har gjort den del dom blivit ombedda att göra och det betyder allt för att nå projektets mål. Uppdelningen har varit som följer:

Erik Ragnarsson, Systemering och Utveckling
Johan Gustavsson, Supplier Partner AB, Nätverksansvarig/Hårdvara
Fredrik Dimberg, Supplier Partner AB, Systemering och rapportdesign

Tillkännagivelser

Trots den knappa tiden har vi som arbetat ihop lyckats sköta den uppgift som vi haft, de problem som upptäckts har blivit lättare att lösa genom stöd både inifrån och utifrån. Dessa personer har stött oss på olika sätt:

Veronica Johansson, Sambo, Den som får mig att göra något

Björn Golmen, Volvo Cars, EDI ansvarig Torslanda

Stig Klem, Supplier Partner AB, Logistikansvarig SP Trollhättan

Innehåll

Utveckling och implementering av sekvenshanterings system åt leverantörer till bilindustrin.

1. Inledning

Så länge som produktion av varor har funnits, har hanteringen förfinats. Desto större volymer man vill tillverka ju smidigare produktionsflöde måste man ta fram. För att kunna tillverka stora volymer kostnadseffektivt använder man inom industrin en hantering som kallas *Just-in-time (JIT). JIT är egentligen en filosofi, som definierar hur man bör styra ett produktionssystem [1].* Genom att använda sig av JIT så förflyttar tillverkarna ut mer ansvar på sina leverantörer och spar stora kostnader [2]. Bilindustrin har ett mycket välutvecklat logisktsystem. Tillverkarna har sparat pengar genom att införa JIT och detta leder till att de försöker ha så lite varor i lager som möjligt. Detta ställer stora krav på de leverantörer som har blivit utvalda att leverera materialet till produktionsmiljön. Desto mindre lager som hålls inom företaget ju viktigare blir det att materialet kommer vid rätt tidpunkt för att inte skapa en försening vid monteringen.
Sekvenshantering kallas det som utförs när underleverantörer förbereder material för en JIT industri. Med förberedelser innebär att artiklarna omarbetas, testas för kvalité och blir märkta med identifikationsetiketter för att sedan placeras i lastpallar. Dessa lastpallar har ett flertal fack för artiklarna som de skall innehålla och kallas för rack, de placeras bredvid produktionslinan där de tillhandahåller materialet för montörerna. Det är därmed viktigt att underleverantören har skött sin del av avtalet och kontrollerat allt material och placerat det i rätt fack för att snabba upp monteringen vid linan. Om något fel har uppstått t.ex. material som är fel, skadat eller av annan anledning obrukbart kan det leda till att produktionen i värsta fall måste stanna och invänta en ny leverans av ersättningsartiklar. I de fall då detta är underleverantörens fel kommer tillverkaren i sin tur låta dem betala kostnaderna för stoppet. Det är därför viktigt för underleverantören att detta fungerar bra, annars kommer det inte bli vinstgivande på något sätt. För att det skall gå att arbeta som en underleverantör till en sådan tillverkare krävs det att man inte tillåter någon typ av fel leverans, utan har ett datasystem gjort för att säkra upp eventuella felkällor innan leverans.

2. Bakgrund

Supplier Partner AB (SP) är ett logistikföretag i Trollhättan/Göteborg, och deras verksamhet bygger på att lagerhålla gods och leverera/förfina produkter främst till bilindustrin. Under december 2003 fick de två uppdrag som innebär sekvenshantering åt Volvo. SP är en sekvensleverantör för Volvo idag, och har ett system för att ta emot de elektroniska beställningarna, men systemet är inte tillräckligt anpassningsbart för att möta de nya krav som dessa projekt har.

2.1 Projekt Sekurit och Pilkington

Projektet skall sättas upp i Göteborg på Volvos fabriksområde i Torslanda. SP har blivit utvalda som mellanleverantör åt Sekurit och Pilkington för att ersätta den befintliga leverantören, Volvo Logistik(VLC). Uppdraget går ut på att sätta ihop de bakersta rutorna till Volvos V70, V70 Crosscounty och XC90. De glas som levereras till V70 skall monteras med en list lackerad i samma färg som bilen innan de skickas till Volvos monteringslina, detta behövs inte med XC90 då dessa rutor levereras färdiga till SP. Programvara för att producera samma typ av etiketter som VLC:s system Movex gör i dagsläget. Detta för att underlätta så mycket som möjligt för personalen som arbetar med det. Etiketterna beskriver vad det är för delar som skall monteras.

2.2 Projekt Inoplast

Projektet skall sättas upp i Trollhättan, och involverar Volvo Uddevalla. Leverantörer i det här fallet är Inoplast. Inoplasts nuvarande leverantör SE-DE har gått i konkurs och kommer att sluta existera den 30 Dec. SP har blivit de nya sekvensleverantörerna och skall bygga höljena som ligger och skyddar den nerfällda suffletten på Volvos C70 Cab.

3. Identifiering av problem

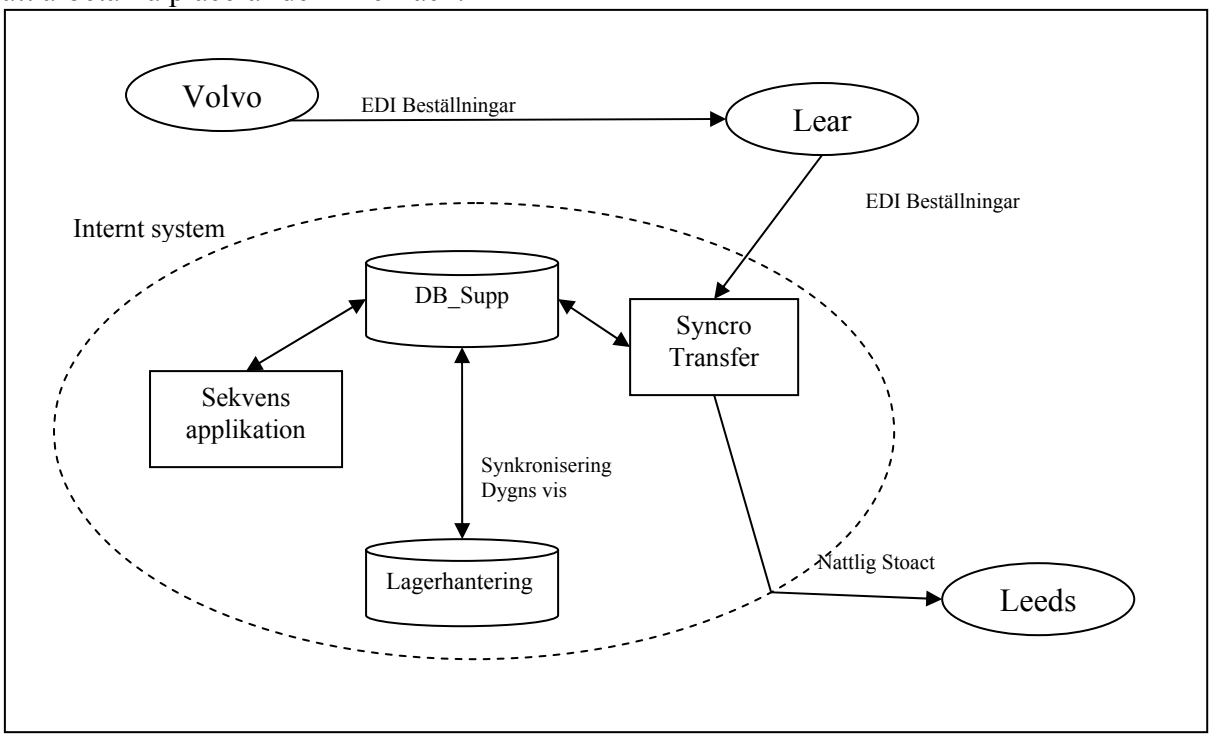
SP måste skapa ett system som gör det möjligt att koppla sig mot Volvos JIT system, en datormiljö för att kunna ta emot elektroniska beställningar ifrån Volvo. Detta system skall ta hand om beställningarna på ett säkert sätt, presentera dem för de ansvariga montörerna på SP samt skriva ut etiketterna för att kunna identifiera delarna i ett senare skede. Systemet skall kunna kopplas mot det lagersystem som används på SP, och det skall finnas möjlighet att skicka elektroniska lagersaldon till SPs leverantörer.
De elektroniska beställningarna är i *Electronic Data Interchange (EDI)* format som är en typ av textfil som följer Odette standard [3], och dessa standarder definierar hur data skall lagras i filerna. En beställning som skickas på detta sätt kallas för ett *synkro.* Varje synkro är kopplat mot ett sekvensnummer som man kan säga motsvarar en bil. Sekvensnummer är en löpande nummerserie som inte bryts, och den stämmer alltid överens med den ordning som bilarna byggs på produktionslinan. Montörernas uppgift är att ta emot beställningar, hämta den artikel som beställningen gäller och lägga in i ett *sekvensrack*. För att identifiera varje rack tilldelas den ett nummer och en streckkod. Volvo skickar parallellt med synkrona även ett bananmeddelande[1]. Dessa innehåller information om vilken bil som senast gick upp på produktionsbanan, vilken bil som går igenom den station som leverantören skall leverera delar till. Det finns ingen specifikation i Odette för detta meddelande så Volvo har själva gjort detta för att efterlikna uppbyggnaden av ett synkro.
Leverantörerna för projekten, Sekurit, Pilkington och Inoplast skall få dagliga rapporter på vad SP har i lager, hur mycket som har levererats och vad som beställts. All denna information kan sättas ihop till en elektronisk fil som kallas *Stock Activities (STOACT)* [4], denna behövs skapas och skickas en gång per dygn. Det övergripande målet för projektet är hur ett datorsystem skall sättas samman för att nå hög användbarhet och administrationsmöjlighet och samtidigt avlasta de anställda.

3.1 Befintligt System

SP har ett befintligt sekvenssystem, dessvärre är det inte tillräckligt generiskt för att kunna återanvända, så denna del kommer att behöva programmeras om. De system som finns har även delade databaser av den anledning att de är individuellt utvecklade. Sekvensapplikationen, EDI delen av detta system har tidigare skötts av EDS innan SP själva tog över hanteringen av dessa. Sekvensapplikationen är sammansätt på ett sådant sätt att den har fasta rackstorlekar och går endast att använda mot sekvensrack som innehåller 15 artiklar. Detta medför att det inte kommer att finnas något sätt att använda den ursprungliga applikationen utan att förändra den.
I *fig.1* beskrivs följande: Volvo skickar ut elektroniska beställningar (i det här fallet på motorrumskablage). Dessa beställningar skickas till Lear då dessa är den främsta leverantören av Volvos alla kablage, Lear har i sin tur hyrt in SP för att lagerhålla och packa om dessa i Sekvensrack. Lear ringer upp SP via ISDN och skickar över EDI beställningarna, dessa beställningar skapas som filer i en katalog på servern. För att få in informationen till databasen finns ett program som kallas *SyncroTranfer,* detta är ansvarigt för att leverera in

[1] Banan, som i produktionsbana inte som frukten banan

synkrona i rätt ordning så sekvensprogrammet inte läser dem i fel ordning vilket skulle göra att arbetarna placerar dem i fel fack.



Figur 1

Varje natt kl 03.00 körs en synkronisering mellan Sekvensdatabasen DB_SUPP och SP:s lagersystem, detta för att leverera ut materialet som blivit bearbetat under dagen. SyncroTransfer sätter även samman en *STOACT,* en Elektronisk inventari rapport, och skickar denna till leverantören av dessa kablage. All EDI kommunikation (beställningar, STOACT) skickas via ISDN anslutning med hjälp av ett program som heter OFTP, då detta program är ett krav ifrån Volvo.

3.2 Befintligt System, Sekurit och Pilkington

Sekurit och Pilkington (SoP) har tills nuläget hyrt in Volvo Logistik (VLC) för att utföra sekvensarbete/förberedelser av deras produkter. De använder sig av applikation som går direkt emot Volvos stordatorer för att hämta information om vilka beställningar som inkommer. De skriver sedan ut ett A4 ark med information om beställningen. SoP hanterar endast bakrutor till bilar med Kombi vilket medför att beställningar på bilar utan dessa bakrutor blir såkallade ”nollavrop”. Ett nollavrop är en beställning på en bil som inte innehåller något beställt artikelnummer. VLC skriver i dagsläget ut även dessa nollavrop och låter den ansvarige arbetaren plocka bort dessa sidor.
Arbetsytan är indelad i en höger och en vänster arbetsyta, eftersom varje bil skall ha en ruta av varje, byggs artiklarna parallellt. Det kommer in två beställningar per bil, vilket kommer att ställa till problem eftersom det skiljer sig en del ifrån den ursprungliga applikation som tar hand om synkron hos SP. Varje komplett rack skall märkas med en rackflagga enligt Volvos specifikation, numrering av rack skall löpa mellan 01 till och med 99.
Då artiklarna innehåller lackerade delar så är det viktigt att montörerna får ut en plocklista över Innhållet i racket så de kan gå och hämta allt material innan de påbörjar arbetet.

3.3 Befintlig System, Inoplast

Artiklarna som monteras innehåller lackerade delar så det är viktigt att få ut någon typ av plocklista för att veta i förväg vad som skall hämtas ifrån lagret. Artiklarna innehåller 28 delar, varav två av dem som finns i olika färger, resterande delen är muttrar och gummilister. Efter att alla delar har monterats fast på artikeln så skall en streckkods etikett fästas på den. Artikeln placeras sedan i ett sekvensrack som fylls upp till det maximala antalet av tio artiklar, sedan skall en rackflagga sättas på racket, denna skall löpa mellan nummer 1 och 9.
Eftersom detta Inoplast projektet levererar delar åt Volvo Uddevalla så är det en liten skillnad gällande EDI kommunikationen. Till skillnad ifrån att ha löpande sekvensnummer som Volvo Torslanda har man i Uddevalla ett äldre löpnummersystem istället. Sekvensnummer här består av år-vecka-veckodag och tre löpnummer (ex 04182001 för första numret 2004-04-27). Detta kommer att betyda att förändringar till programmet *SyncroTransfer* kommer krävas då det nu är sagt att endast hantera löpande ordning.

4 Metodik

Metodiken skiljer sig lite ifrån de olika delar som bygger upp projektet. Större delen av projektet skall sättas upp enligt den normala livscykelmodellen. De faser som projektet är indelat i är *krav, analys, design och implementering/test*. Eftersom projektet kommer delas in i mindre delar kommer denna modell att användas för alla dessa delar förutom sekvens-programmet som utvecklas genom prototyping[5], och kommer därmed återupprepa de fyra cyklerna tills programmet uppnår det slutgiltiga målet.
Kravspecifikationen innehåller de krav som systemet måste uppfylla för att projektet skall lyckas. Kraven som sätts upp är en del systemkrav ifrån Volvo, kommunikationskrav och systemförändringar som måste införas. För att hitta dessa krav kommer en dialog att hållas med ansvariga inom SP och eventuellt deltagare ifrån Volvo.
Under *analysfasen* skall de förutsättningar som motsvarar kraven skapas. Varje krav skall ses över och en förutsättning för att motsvara detta krav skall hittas. Detta skall tas fram och presenteras för IT avdelningen på SP, de får sedan avgöra om förutsättningarna är riktiga och genomförbara. Befintlig programvara kommer att undersökas för att hitta belägg för påståenden som IT avdelningen har framfört. Undersökning av existerande programvara sker lite annorlunda beroende på vad det är men vid färdig programvara testats programmen, och källkod undersöks. Databasen undersöks för att ta reda på vad det är för data som lagras och hur länge informationen skall sparas. Det arbete som fysiskt utförs av de anställda montörerna kommer inte förändras, det är viktigt att under analysfasen ta reda på hur de arbetar.
I d*esignfasen* skall systemets utseende bestämmas. Genom att bygga efter de förutsättningar som satts upp under analysen, kommer systemet att uppfylla de krav som tagits fram. Vid de delar som endast kommer att innebära förändringar av befintliga funktioner så skall förändringen beskrivas med ord. Förändringar kommer att beskrivas som *ny, ändring eller borttagning*. På den del i systemet som inte redan finns kommer designfasen att återupprepas eftersom programmet tas fram som en prototyp. De faser som beskrivs här återupprepas till systemets krav är uppnådda. När det gäller prototypdesignen så kommer även ett grafiskt gränssnitt att tas fram i denna fas.
När *implementerings*fasen genomförs kommer programmet att byggas enligt den design som är bestämd, vid behov av förändringar av designen skall projektet tas tillbaka till designfasen för att genomföra de förändringar till designen som är nödvändiga. Testningen kommer att köras parallellt med utveckling av programmet och testas med *Verifiering och validering* [6] för att snabbt kunna hitta eventuella fel som uppstått under skrivningen av funktionen. När programmet sedan når ett körbart stadium skall det testas för att se om det uppträder förväntat. Denna testning skall ske när en funktion som finns angiven i funktionsbeskrivningen har

skrivits färdig. Valideringen av funktionen kommer att genomföras genom att programmeraren läser igenom metoden och hittar de positioner och otillåtna parametervärden kan ställa till besvär, eventuell felhantering skall läggas till så att metoden inte kan få programmet att avbrytas. Vid funktionstester när programmet når körbart läge skall enhetstest för funktionaliteten testas med eventuellt genererad data.

5 Krav och Analys

En kravspecifikation finns inte när projekten startas, det första målet måste bli att ta fram en sådan. En sammanställning av vilka krav som kommer att ställas på systemet skall tas fram och i vilken typ av miljö det kommer att köras. IT avdelning på SP slår fast att systemet som skall tas fram bör delas in fyra huvuddelar då det kommer underlätta vid framtagningen av kravspecifikationen och vid utveckling av systemet. IT avdelningen konstaterar snabbt att logistikansvariga kommer att behöva vara inkopplade i en av dessa delar. Själva behöver de närvara i resterande delar då det är viktigt att systemet inte tillför stora förändringar för underhåll av systemet. Volvo Cars har krav som måste tas tillvara på för att kunna kommunicera med dem på ett riktigt sätt. SP har även tillfrågats att skicka inventarierapporter till de kunder som tillverkar materialet och kommer att behöva tillfråga dessa om speciella krav ifrån deras sida. Då delar av systemet redan finns kommer det ta fram krav för varje del och förändringar till dessa som behöver utföras. Diskussioner med IT avdelningen på SP hjälpte till att gränsa av systemet för att lättare kunna ta fram en specifikation för varje del som kommer kräva förändringar

Område	**Inblandade**
Lagersystem, integrering	IT
SyncroTranfer, EDI kommunikation.	IT, Volvo Cars IT, kunders IT
Sekvensprogram	IT, logistik
Övriga krav på systemet	IT, logistik, ledning.

Tabell 1,Systemet indelat, Inblandade parter vid fastställning av kravspec för varje delsystem

Kraven på systemet har tagits fram i en gemensam diskussion med de inblandade parterna som angivits i tabell 1. De krav som finns på sekvensprogrammet är de slutgiltiga kraven på applikationen, med slutgiltiga krav menas att dessa krav skall vara uppfyllda för att programmet skall anses färdigt. Varje krav skall numreras så att det går att identifiera i efterhand.

SyncroTranfer	
1	Beställningar för Volvo Uddevalla måste kunna tas emot, även då de har annorlunda löpnummerordning
2	Beställningar ifrån Volvo Torslanda innehåller både vänster och höger artikelnummer kopplade mot samma sekvensnummer, dessa måste lagras i databasen båda två, utan repeterande rader för det skulle innebära annorlunda
3	Nollavrop ifrån Volvo Torslanda skall kunna behandlas
4	Vid borttappade beställningar skall inte ”luckor” mellan beställningar kunna accepteras, då detta leder till oordning i tillverkningen. Detta fungerar idag och får absolut inte störas
5	STOACT måste anpassas så man kan skapa en rapport till varje leverantör. Stödjer idag endast en leverantör, vilket kommer att ställa till problem i Skurit och Pilkington projektet
6	Kunna ta emot och lagra information ifrån Volvos Banan meddelanden

Tabell 2, exempel på hur systemets krav ser ut

De delar som systemet har blivit indelat i kommer att utgöra den arbetsordning som vi har genom projektet. Det tänka systemet kommer modifieras för att se ut som *Fig 2*, denna modell

skiljer sig redan här ifrån den ursprungliga designen, främst genom att databaserna nu blivit endast en.

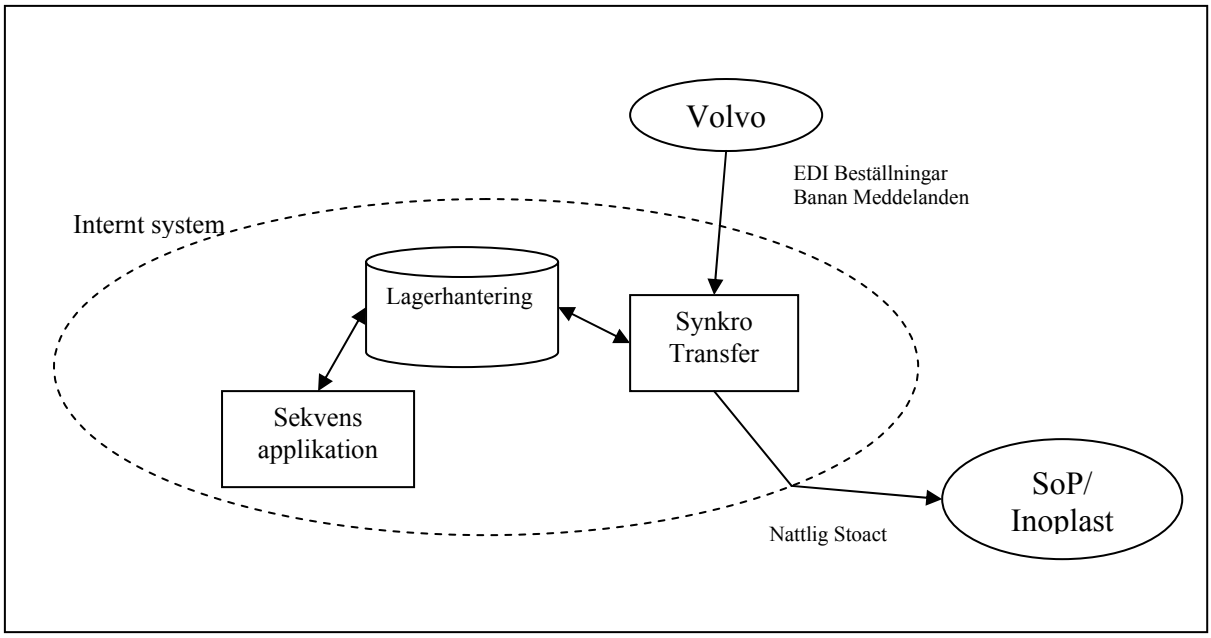


Fig 2, modell över hur det tänkta systemet skall fungera

5.1 Undersökning för modifiering av SyncroTransfer

Det är viktigt att börja med att samla in vital data för att veta vilka förändringar som måste implementeras. Efter att ha talat med ansvarig EDI personal på Volvo Uddevalla har det framkommit att klientapplikationen måste implementera funktionalitet för att klara av den annorlunda löpnummerordningen. Uddevallafabriken skickar till skillnad ifrån Torslanda inte ut lika många beställningar per dygn och skickar endast beställningarna fyra ggr per dygn.
Torslandas beställningar grundar sig i en höger- och en vänsterartikel. För att tillföra funktionalitet att lagra denna detaljinformation i en så lik version av databasen som möjligt har det framkommit att databasen i dagsläget innehåller fria textfält. Dessa heter *fritext1, fritext2* och *fritext3*. Under en diskussion med SP IT avdelning har det framkommit att dessa fält inte är under användning till något för tillfället så det beslutades att använda en av dessa kolumner till att lagra informationen. Genom att göra detta påverkas inte databasen ifrån det ursprungliga utseendet och möter kraven ifrån IT avdelningen. Denna justering av datalagring kommer däremot att medföra större förändringar till den funktion i SyncroTransfer som ansvarar för att synkrona skall vara i ordning då det krävs att ett synkro måste ta emot två beställningar för att vara komplett. En verifikationskolumn kommer att läggas till för detta syfte och fyllas med ett värde skilt ifrån noll då den andra beställningen inkommit. Detta kommer också att bistå till att nollavropen som inkommer ibland, kommer hanteras och inte störa löpnummerordningen. För att programmet skall kunna särskilja på när det skall förvänta sig dubbla beställningar per synkro så kommer en kryssruta läggas till i programmet, när denna är ikryssad så skall denna speciella ordning aktiveras.
Efter att ha undersökt databasen för sekvensprogrammet bevisas IT avdelningens, påstående om att skapa STOACT rapportering till flera olika leverantörer inte är möjlig i dagsläget är korrekt. När SyncroTransfer läser in data till STOACT:en så kommer den endast ta data ifrån den översta raden i varje tabell som är till för detta. Vilket gör det omöjligt att ha olika data för varje leverantör, vilket är ett måste. En viss förändring av den befintliga databasen måste genomföras för att krav 5 skall kunna nås. Ett förslag är att använda sig av systemets parametertabell (t_supp_sysparam) för att lagra mer data än namn på leverantören som den gör i dagsläget. En kolumn för att tilldela varje leverantör ett ”partner id” läggs till och en

liknande kolumn läggs till i alla tabeller som STOACT:en hämtar data ifrån. Nu kommer SyncroTransfers metod för att skapa STOACT:en skapa en rapport för varje leverantör i t_supp_sysparam och via den ta reda på leverantörens ”partner id” för att utvinna dess speciella data ifrån de andra tabellerna. Detta kommer att motsvara kraven för underhållning av systemet då det inte tillför någon egentligt stor förändring av systemet ur databassynpunkt. IT avdelningen på SP håller med om att förändringen är bra för deras administrativa arbete och anser den hållbar även för framtiden.
Enligt Volvo Cars IT så skall deras bananmeddelande tas emot på samma sätt som ett synkro, det är ungefär samma struktur på filen fast mindre information, Odette har inte någon information om meddelandet eftersom det inte är enligt EDI standard så en specifikation för detta skall Volvo tillhandahålla. Det är viktigt att denna specifikation finns tillgäng för SP innan projektet skall tas i bruk för annars kommer det inte gå att implementera stöd för bananmeddelandet.

SyncroTranfer	
1	Bygg om SyncroTransfer för att kunna ta emot Volvo Uddevallas nummerordning för att inte äventyra krav 4 och för att möta krav 1
2	Använda sig av befintlig databas och lagra vänster artikelinformation i normala datakolumner och högra i fritextkolumner. Krav 2 mött.
3	Ett verifieringsvärde skall sättas när varje bil fått in två beställningar, en fritextkolumn skall användas för detta med. Detta kommer att säkerhetsställa att en bil har mottagit alla artiklar som skall beställas åt den. Möter krav 3 och krav 4.
4	På varje tillhörande tabell för STOACT skapning skall en extra kolumn läggas till, denna skall ha identifiera ett partner id som i sin tur kopplas mot tabellen för system paramterar(t_supp_sysparam). En kolumn i för att tala om en leverantör skall få en STOACT rapport finns även i t_supp_sysparam. Gränssnittet skall ändras för att tillåta administration om vilka leverantörer som skall få STOACT. Detta fullfyller krav 5.
5	Lägg till stöd för att läsa Volvos egna meddelande typ bananmeddelanden. Dessa skall läsas som vanliga syncron till stor del och lagras i en tabell som skapas för detta ändamål. Täcker in krav 6.

Tabell 3, förutsättningar för Syncrotransfer

5.2 Förutsättningar för ett sekvensprogram

Ett visst arbetsflöde har uppkommit när sekvensprogrammet har diskuterats med ansvariga parter på SP, främst de logistiskt insatta. Tanken är att man i framtiden skall kunna öka säkerheten med hjälp av streckkodsläsare, men eftersom etiketterna för Sekurit och Pilkington inte har detta får en annan lösning tas fram. När arbetsflödet runt programmet togs fram var det mest Sekurit och Pilkington projektet som diskuterades, de som monterar artiklarna här har nämligen redan en arbetsform och ledningen på SP vill ogärna ställa till besvär för dem. Det fastställs att det fysiska arbetssättet skall bestämma hur programmet skall utformas. En idé som uppkommer under diskussionen är att man skall kunna ha två datorer, en uppställd på var sida av den yta avsedd för montering. Vid den första datorn skall den ansvarige skriva ut alla etiketter och rackrapport, det skall inte gå att få ut en rackflagga på den här datorn. Innan pallen skall skickas iväg måste en racketikett finnas på det, denna skall skrivas ut på den andra datorn i andra änden på lokalen. För att vara säker på att det är rätt rack som skall avslutas så skall racket vara kopplat mot en kod som står på rackrapporten. När detta är inskrivet så kommer rackflaggan ut. En idé till förbättring vid senare tillfälle är att vid detta läge låta programmet starta en streckkodsläsningsrutin här, innan rackflaggan skrivs ut. Detta är inte aktuellt i dagsläget då SP:s administrativa personal inte anser det vara nödvändigt då personalen inte kommer att göra fel. Om det ändå kommer uppstå problem här kommer det inte bli svårt att lägga till en sådan funktion i efterhand.

1	Lagledaren/Ansvarige på arbetsplatsen trycker på ”Skapa Rack”. Detta kommer att skriva ut det antal etiketter för att fylla ett rack (alternativt så många som finns kvar i systemet) och två stycken rackrapporter
2	Ansvarig tar en av rackrapporterna och lämnar den andra rapporten till en plockare. Den andra listan läggs tillsammans med etiketterna och lämnar dem till montörerna, dessa kommer inte att göra något med dessa artiklar innan material inkommit
3	Plockaren hämtar de specifika material som kommer krävas för att montera ihop de beställda artiklarna på plocklistan när detta är utfört placera materialet vid montörerna
4	Montörerna börjar plocka ihop de artiklar som de har etiketter till, och lägger dem i sekvensracket i den ordning som det står på plocklistan tills racket är komplett
5	Ansvarige utför en slutkontroll på det kompletta racket, går bort till sekvensprogrammet och trycker på ”Slutför rack”. Den ansvarige kommer att bli tillfrågad att skriva in en kod som står på rackrapporten och kommer på detta sätt att godkänna racket datamässigt, rackflaggan kommer att skrivas ut, den ansvarige användaren kommer att sätta in rackrapporten i en pärm och sätta fast rack flaggan på racket
6	Racket är färdigt för leverans och inväntar Bil/Truck för detta ändamål

Tabell 4, Arbetsflöde för sekvensprogrammet

Förändringarna av befintliga delar och implementering av ett nytt sekvensprogram kommer få följande modell. Det är denna enkelhet som skall eftersträvas för att underlätta administration av hela systemet för SP. Genom att undersöka vad det finns för aktörer i systemet underlättar när användarfallen skall tas fram.[7]

Montör	De som fysiskt sätter samman godset och klistrar fast etiketter på de artiklar som blir klara, de placerar de färdiga artiklarna i sekvensracket.
Lagerpersonal	Använder systemet Lagerhantering för att leverera in material i lagersystemet.
Plockare	De personer som samlar in materialet till varje rack så montörerna har något att bygga delarna utav
Lagledare	De huvudsakliga användarna av systemet, det är de som skriver ut etiketter, rapporter och racksflaggor. De verifierar godset innan leverans
Samarbetspartner	Den kund som tar emot de STOACT filer som systemet kommer att skicka.
Beställare	Den som skickar beställningar till vårt system. I grunden Volvos datorsystem.

Tabell 3, aktörer inom systemet

För att ta så mycket hänsyn till arbetarna som möjligt, kommer den tänkta applikationen skriva ut samma typ av pappersdata (etiketter, rapporter, rackflaggor) som idag. Detta för att arbetarna inte skall behöva ägna någon tid åt att lära sig något nytt system. Programmet skall anpassas för att kunna köras mot en uppdaterad version av den sekvensdatabas som används idag, för att underlätta för IT avdelningen på SP.

1	Flödet ifrån Volvo är lika i båda projekten, och kommer att sättas upp för att efterlikna det ursprungliga SP projektet. Detta är lika tills programmet *SyncroTransfer* skall ta vid.
2	Papper/Etiketter för att identifiera beställningar behövs
3	Plocklistor för hela racken till att bistå montörerna
4	Historik över redan körda beställningar måste finnas kvar i databasen
5	Avdragning mot lagersystemet av utlevererat gods
6	Konfigurationsfil för att snabbt kunna ändra
7	Övervakning av inkommande beställning och bananmeddelande skall visas

8	Kunna skriva ut kopior av redan utskrivna etiketter
9	Visa rackrapporten så länge racket är under produktion
10	Visa rack under produktion
11	Avsluta rack i annorlunda ordningen än de är skapade
12	Kunna stänga av programmet utan det skall bli förslut av information
13	Går något fel under en transaktion skall den inte genomföras
14	Racken som skapas måste kunna skapas med en variabel storlek (inte alltid 15 som är läget idag)
15	Ge systemet möjlighet att veta hur vissa artikelnummer består av andra, så vi kan göra rättvisa avdragningar av lagersaldo då en komplett artikel levereras

Tabell 5, likheter mellan sekvensprojekten

De två projekten som sekvensprogrammet skall skrivas för har en hel del gemensamma faktorer, dessa måste märkas ut för att de skall kunna implementeras på en så generisk nivå som möjligt. Det kommer att bli två versioner av sekvensprogrammet, ett för vart och ett av projekten men tanken är att större delen av koden skall vara gemensam för att underlätta vid uppdateringar. Innan man designar själva programmet är det viktigt att vara medveten om det som särskiljer programmen ifrån varandra för att kunna göra de gemensamma delarna av programkoden så bra som möjligt.

Sekurit/Pilkington	**Inoplast**
Fler skrivare är inkopplade till datorn, så stöd för att skriva ut på ”ej” standardskrivare måste finnas för rackflagga.	Etiketter skall skrivas ut på en seriell skrivare (Markpoint ML300)
Racken skapas alltid dubbelt. En för högersida, en för vänster sidan men de skall ha samma nummer och vara parallellt framställda. De skall innehålla beställningar till samma bil i samma rackfack. Rack flaggorna skall markeras med ett H eller ett V för att separera dem.	
Nollavrop (de avrop som inte är avsedda att ha något material ifrån Sekurit eller Pilkington) skall inte skrivas ut men ändå makuleras som om de skulle vara producerade för att löpnummerkontroll skall gå att igenom i efterhand.	
Visa senast inkomna Bananmeddelande	

Tabell 5, Skillnader mellan sekvensprojekten

Den viktigaste skillnaden mellan programmen är den att Sekurit och Pilkington synkrona har dubbla beställningar, detta är redan till hälften behandlat genom SyncroTransfer så det enda som återstår är att skapa etiketter i dubbla upplagor som är kopplade mot de olika datakolumnerna. För att på ett smidigt sätt ändra utseende på etiketterna så beslutas det att Crystal Reports 8 bör används för detta ändamål av flera anledningar. SP har redan licenser för denna programvara, de har intern kunskap om programmet och det finns bra stöd för det i .Net, detta trots att .Net använder sig av Crystal Reports 9. Dessvärre måste det i Inoplast projektet gå att skriva ut etiketter utan att använda CR då det i så fall inte skulle vara möjligt att använda den seriella skrivare som skall göra det. CR fungerar så att det färdigstället rapporterna på datorn och skickar utskriften som en bild till skrivaren, detta medför att utskrifterna kan bli ganska stora vilket ställer till problem när skrivaren kommunicerar seriellt. En utbytbar klass för utskrifter måste därför finnas för att kunna byta ut detta.

Volvo Uddevalla har inte heller någon produktionslina som Volvo Torslanda, och skickar därför inte heller ut några bananmeddelanden som behöver tas emot. Dessa behöver endast finnas i den version som skall köras på Sekurit och Pilkington.
Sekvensprogrammet skall designas för att lätt kunna anpassas för de två projekten som det har för avsikt att användas till, då det utvecklas experimentellt kommer fler och fler funktioner implementeras för varje iteration i utvecklingscyklerna. Prototypen kommer att utvecklas för att kunna användas i projektet mot Inoplast då detta är mest tidskritiskt.
Det är bara användaren *Lagledare* som kommer använda programmet och designen börjar med att ta fram de olika användarfallen[6] och beskriva dessa. Det är viktigt att veta vad för data som plockas ut ifrån programmet i pappersform så dessa är också utritade på användarfallsskissen.
När första prototypen skall tas fram sätts ett grafiskt gränssnitt ihop som skall kunna utföra de användarfall som är definierade, designen togs fram för att kunna visa vad applikationen skall göra, något att visa upp för inblandade parter (Lagledare) och bestämma om det är en givande design för dem, eller om den behöver ändras för ett mer logiskt utseende.

5.3 Lagerhanteringsdatabas

Genom att slå ihop de separerade databaserna *Lagerhantering* och *DB_SUPP* (*Bilaga_Databaser*, Fig 1, Fig 2) kommer det bli möjligt att alltid ha ett saldo som stämmer, innan har dessa två databaser synkroniserats på natten för att uppdatera utlevererat saldo. För att databasen skall kunna användas måste även ett par förändringar utföras på den.

5.4 Övriga punkter

SP har redan en licens för OFTP, däremot får denna inte finnas på flera datorer samtidigt. Två nya licenser kommer att behövas att köpas in för att kunna använda det. På samma datorer kommer även en Microsoft SQL Server 2000 att installeras eftersom licens för detta finns.
Det skall vara lätt att i framtiden kunna anpassa sekvensapplikationen vidare och att underlätta för SP att hitta resurser som klarar av detta. Det ledde till att valet blir C# då Microsoft.Net skulle göra det möjligt att även skriva komponenter i VB.Net eller Managed C++. Eftersom applikationen skall köras på Microsoft Windows plattformen anses det bättre att hålla sig till .Net gentemot Java, .Net kommer även tillåta en snabbare exekvering av programmet och tillåter direkt åtkomst till Microsoft SQL Server 2000.
SP:s övriga programvaror(*Lagerhantering, SyncroTransfer*) är däremot skrivna i Visual Basic 6, men det ses ändå som en fördel att gå ifrån VB6 när den nya programvaran skall utvecklas för att även i framtiden lätt kunna hitta kompetenta utvecklare. Däremot skall de förändringar som behöver apteras i SyncroTransfer skrivas i VB6 eftersom det redan är skrivet i det.
Rapporter som skall tas ut ur programmet skall designas i *Crystal Reports 8* då denna licens redan finns hos SP.

6 Design och Implementering

6.1 SyncroTransfer

Båda projekten strider emot en del av syncrohanteringen som idag används i det befintliga systemet. Inoplast projektet bryter den löpordning som används. Det skulle i dagsläget leda till att sätta SyncroTransfer i vänteläge när dagen upphör. Detta skall ske genom att utöka funktionaliteten när programmet flyttar över sorterade syncron, den skall även kunna jämföra om det finns chans till ett syncro som inte är i ordning fyller kriterierna för en dygnsbrytning.
Sekurit projektet kommer att använda den ursprungliga löpnummerordningen men vid varje sekvensnummer skall en vänster och en höger artikelbeställning inkomma innan det är okej att

börja skriva ut etiketter för dem. Var fjärde till femte beställning är ett nollavrop, då dessa bilar inte skall ha den typ av rutor som Sekurit och Pilkington levererar. Dessa beställningar kommer ”levereras in” i systemets databas precis som de andra syncrona och det kommer istället bli Sekvensprogrammets uppgift att behandla dessa.
STOACT rapporten som skapas måste kunna kopplas mot en speciell leverantör, när den skapas i dagsläget så samlar den in all data i de tillhörande databastabellerna (det finns bara en enda rad i de flesta tabellerna). Eftersom detta inte kommer att kunna ge möjlighet att skapa två separata STOACT-rapporter i Sekurit och Pilkington projektet måste vi implementera detta.
En extra programdel skall ta hand om och behandla bananmeddelandet, ingen direkt förändring av befintlig kod behövs här utan en extra procedur skall implementeras för detta ändamål.
Vid undersökning av de syncron som skall tas emot av Volvo Uddevalla så upptäcktes det att de skiljer sig på ännu ett sätt ifrån Volvo Torslanda. När meddelandena överförs så har man endast en *MessageHeader(UNH[1])* och det har innan varit sagt att det alltid skall finnas ett sådant huvud. Förändring av SyncroTransfer för att behålla informationen ifrån det enda huvudet och sedan använda det i varje syncro som inkommer i den beställningen måste implementeras annars kommer beställningarna inte lagras komplett i databasen.
Förändringarna som tillkommer till SyncroTranfer göra att programmet inte går att köra i den ursprungliga miljön. Det är också viktigt att det gamla systemet skall kunna uppgraderas om hastiga förändringar måste införas. Man vill ur supportsynpunkt ha samma version av programvaran överallt i företaget vid dataförlust såsom data krasch eller liknande, IT personal skall inte behöva veta vilka olika versioner det har funnits på den trasiga datorn.
Förändringar till SyncroTransfer tas fram för att motsvara de mål som tagits fram, det är funktioner som påverkas och dokumenteras som *nya* och *förändrade* funktioner. De beskrivs i följande exempel

> **Förändra funktion: ParseSingleMessage**
>
> Ändringen skall tillåta att släppa igenom Volvo Uddevallas meddelanden trots att dessa saknar *edi-headers[1]* för alla meddelanden utan upprepar endast SEQ, ARD och SDD segment för varje ny beställning. Nästa ändring skall vara aktiv om checkrutan för ”Sekuritsyncro” är ikryssad, ändringen kommer att tvinga systemet att leta efter två inkommande beställningar på varje sekvensnummer. När dessa två beställningar hittas, varav det vänstra artikelnumret sparas i fältet ”Artnr” och den högra i fältet ”Fritext1” kommer fältet ”Fritext2” sättas till ett värde som är skiljt ifrån null. Det är först nu ett synkro kommer att ses som ett komplett synkro. Denna metoden påverkar inte databasen utan det utförs av funktionen ”SaveSyncro”.

Exempel på funktionsbeskrivning för syncrotransfer

Designen skall beskriva funktionen så väl att det i efterhand skall vara möjligt läsa igenom det och veta vad förändringen innebär. Den nya kod som skrivs kommer att skrivas i Visual Basic 6 eftersom resten av programmet är skrivet med det. För att utföra enhetstesterna har ett program för att skapa fejkade syncron implementeras. Genom att skapa en mängd olika syncron kan sorteringsfunktionaliteten fastställas. När enhetstest för bananmeddelandet skall genomföras så används skarpa meddelanden direkt ifrån Volvo.

6.2 Sekvensprogram

Ett användarfallsdiagram [9] skall utformas för sekvensprogrammet, så det lätt skall synas vad som skall gå att göra i programmet och prototypen kommer att byggas för att innehålla dessa händelser. Diagrammet skall också visa vilken typ av pappersdata användarfallen skall producera. Se Bilaga_Utveckling_1, kap 3.2 för beskrivning av de användarfallen som finns.

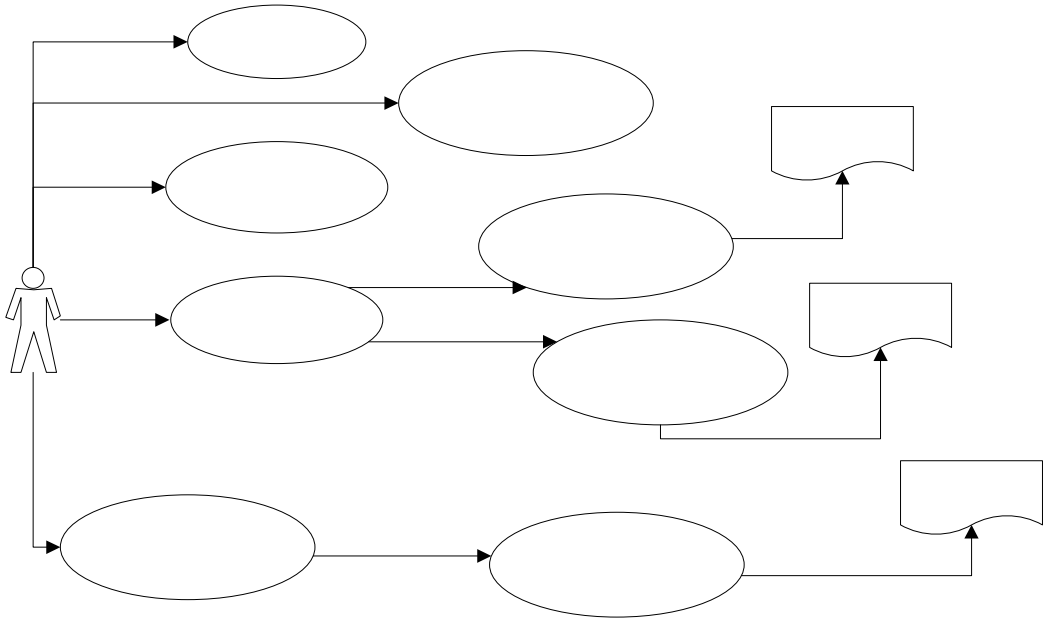


Fig 1. Händelser för sekvensprogrammet

Då ett utseende på prototypen har tagits fram tas de funktioner programmet behöver för att fungera, dessa funktioner kommer att skrivas i programkod och läggas i klasser som tillhör programmet. Det är dessa funktioner som utgör grunden för programmet och allt som det skall kunna göra. För framtagning av de funktioner som programmets första prototyp så kommer endast SP:s IT avdelning att vara inblandad, för att de har kunskap om vad programmet behöver stödja för att implementeras. Efter att den första prototypen blivit gjord kunde de logistikansvariga på SP bli inkopplade för att ta reda på vilken typ av information som behövdes göras tillgänglig och vilka funktioner som de saknades. När sekvensprogrammet har gått igenom den andra iterationen skall det anpassas för projektet mot Skurit och Pilkington och övervakningen skall förändras till att stödja Bananmeddelandet. För att definiera en funktion beskrivs funktionen enligt följande:

Namn: **Skapa Rack**
Beskrivning:
Funktionen skall skapa ett nytt rack och rader i DB genom att skicka nödvändig information till Uppdatera DB. Vidare skall funktionen skapa rackrapport genom att skicka nödvändig information till funktionen Skriv Rackrapport, och även skapa etiketter genom att skicka nödvändig information till funktionen Skriv Etiketter.

Exempel på funktionsbeskrivning för sekvensprogrammet

Dessa funktioner skall sedan läggas i de klasser som kommer tas fram för programmet, varje klass som skapas skall ha som huvudsyfte att tillföra funktionalitet till programmet och kunna vara utbytbar vid behovs. Klassen tas fram för att kunna innehålla de funktioner som redan är skapade, finns det ingen funktion att tilldela klassen finns det ingen direkt anledning att använda den, om det inte är av någon särskiljd anledning. Klasser beskrivs som i följande exempel:

Namn: **SekvensApplikation.UtskriftSupport**
Beskrivning:

> Klassen är till för att kommunicera med skrivarna. Innehåller metoder för att skiva ut rapporter, etiketter. Denna klass är skriven speciellt för varje projekt, även om de är mycket lika.

Exempel på klassbeskrivning för sekvensprogrammet

De funktioner som finns kopplas till de klasser som nu har blivit skapade. Det kommer sedan under implementeringen underlätta när programmet skall skrivas.
De etiketter som skall produceras av sekvensprogrammet skall se ut på ett sätt som redan är specificerat, eller används i dagsläget. Målet är att efterlikna dessa i den utsträckning som det finns möjlighet till. Inoplast Etikett som skall fästas på det monterade godset (*Se bilaga Utveckling*)
Etiketten går inte att designa i Crystal Reports för att den inte skulle gå att skriva ut på en seriell skrivare eftersom utskrifterna tenderar att bli ganska stora, uppåt 1-2 MB och det kommer att ta för lång tid att föra över 1-2 * 10 MB via seriekabel till skrivaren. Istället skall denna etikett skapas som ett skrivarmakro specifikt för denna skrivaren (Markpoint MP104 MKII). (Se bilaga *Utveckling, Etikett Inoplast*).
Sekurit och Pilkingtons material skiljer sig lite. Vad gäller de artiklar som kommer ifrån Pilkington så är det enkelt då endast levererar två typer av glas, standard och tonade. Detta ger totalt fyra olika artikelnummer som skall särskiljas på etiketterna med ordet *SUNDYM*. Vi väljer att läsa detta ifrån artikelregistret i lagersystemet, tabell *"ar"*. (Se bilaga *Utveckling, Etikett Sekurit, Pilkington*). Det material som kommer ifrån Sekurit är dessvärre fler, varje beställd artikel som kommer ifrån Volvo skall monteras på plats med en typ av ruta och en typ av list. Det som särskiljer rutorna är om dom är avsedda för larm, el uppvärmning och extern antenn, de finns även i standard och tonat utförande. Det måste framgå tydligt vilken typ av glas som skall användas så inte plockaren tar fram fel material åt montören. Detta gäller även för listerna fast dessa har en färgkod som identifierar dem. All denna information måste lagras i systemets artikelregister där det också skall framgå om artikeln är avsedd för höger eller vänster sida. Rackrapporten designas för att innehålla den information som kan behövas när man plockar material för de artiklar som skall byggas. Rackflaggan får enligt Volvos specifikation inte ha något annat format. Till detta kommer den flaggan ifrån det ursprungliga sekvensprogrammet användas med en modifiering för Sekurit och Pilkington som talar om ifall den hör till ett höger eller vänster rack.
De skillnader som kommer att finnas i programkoden när den anpassas till de olika projekten skall endast vara klasserna *SekvensUI*, *OvervakningSupport* och *UtskriftSupport*. Av den praktiska anledningen att *SekvensUI* måste modifieras för att ändra titelrad och stödja bananmeddelandet.

6.3 Databasförändringar

Förutom att de två databaserna skall slås ihop och bli en, kommer vi behöva förändra den del som tillhört sekvensdatabasen en hel del (Se bilaga *Databaser*). Så när själva databasen skall sättas samman så kommer den ändra utseende på samma gång. De förändringar som tas fram för att den skall kunna fungera med Syncrotransfer, Lagerhantering och den nya sekvensprogramvara kommer inte att innebära att lagerhantering (som är den enda programvaran som inte förändras) behöver anpassas speciellt för detta ändamål.
De databasförändringarna som skall införas delas in i tre grupper *skapade tabeller, förändrade tabeller* och *borttagna tabeller*.

> **T_SUPP_ArtikelJeeves**, ingen anledning att ha kvar denna då hela lagersystemet finns i samma databas numera, använd istället *ar*.

Exempel på en tabell som kommer tas bort

Index på kolumner som ofta söks, sorteras är en vanligt förekommande åtgärd för att snabba upp databasfrågorna[8].
För att SyncroTransfer skall kunna sammanställa en STOACT rapport ifrån föregående dygn kommer ett nattligt jobb köras på SQL Servern. Detta jobb kommer att sammanställa information om vad som blivit in och utlevererat i lagret under dygnet och spara detta i tabellen *t_supp_parnerarticles*. Genom att spara undan detta i en separat tabell kommer inventarierapporten kunna skapas manuellt under dagen och fortfarande ge samma värden som på natten. Detta ger säkerhet om den på något sätt skulle försvinna.

6.4 Övriga förändringar

Då EDI kommunikationensmiljö redan är färdigbestämd av Volvo är denna bit inte så speciell ur designsynpunkt. Programvaran OFTP sätts upp på servern som skall ta emot de beställningar som inkommer ifrån Volvo och sparar ner dessa beställningar i serverns filsystem. Filerna skapas med löpande numrering och behandlas därefter av SyncroTransfer. De inventarierapporter som SyncroTransfer kommer att skapa skickas till leverantörerna med hjälp av samma programvara.

7 Resultat och Diskussion

Genomförandet av projektet blev oerhört tidspressat eftersom Inoplast projektet var tvungen att fungera innan 2004-01-12. Klartecken ifrån SP att påbörja projektet arbetet kom 2003-12-08 vilket gav lite mer än en månad att sätta ihop en applikationsmiljö för detta. Då Inoplasts befintliga sekvenspartner SE-DE även gått i konkurs blev det oerhört viktigt att systemet fungerade den 12 januari.
Då det är viktigt för en leverantör till en JIT tillverkare att sköta sig är det också viktigt att man är införstådd i vad man har för ansvar. Tillverkarna ställer ofta stora krav på sina JIT leverantörer eftersom filosofin bygger på att minimera montering och lagerutrymme. Det är också viktigt att veta hur man skall möte de krav som leverantören ställer, annars kan hela affären kosta mycket pengar. Olika tillverkare har naturligtvis olika krav på sina leverantörer och i det här fallet har Volvo satt upp tydliga regler för vad som måste uppfyllas. Genom att använda det ursprungliga logistiksystemet på företaget och integrera in JIT-systemet så kommer företaget få mindre problem vid saldorapporteringar, detta naturligtvis förutsatt att integreringen fungerar korrekt. Ett datasystem som själv sköter all hantering är det mål som man vill nå, detta för att en mindre felmarginal kommer nås då den mänskliga faktorns inblandning i datasystemet minskar. Detta är vad vi eftersträvat i detta projektet och har uppnått.
Systemet sattes upp i Trollhättan i mellandagarna, EDI kommunikationen till detta var klart ungefär vid samma tidpunkt tack vare Johan Gustavsson på Supplier Partner som tagit på sig ansvaret att sköta den biten. De mål som sattes upp har nåtts, ett datorsystem för att kunna ta Volvos beställningar i form av EDI meddelanden fungerar. Dessa meddelanden kan tas emot ifrån både Volvo Torslanda och Volvo Uddevalla och lagras i systemets databas. Lagersystemet och sekvensprogrammet fungerar att samköra, lagersaldon stämmer hela tiden istället för som innan en gång per dygn. Systemet gör detta och behåller även mycket av det ursprungliga administrationssättet vilket var ett av målen.
De anställda som har hand om materialet och som packar om artiklarna har fått tillgång till smidigare listor när de hämtar in det material som de skall bygga med. Artiklarna märks sedan upp med etiketter som de alltid har gjort innan SP tog över projekten. Leverantörerna får vetskap om aktuellt lagersaldo och dagens beställningar när de tar emot den elektroniska inventarierapporten (STOACT).

Projekten har blivit lyckade tycker jag, först med tanke på att systemet tillhandahåller all den funktionalitet som förväntades av det, dels för att det förbättrar arbetssättet hos dem som använder systemet. Om man tittar på Sekurit och Pilkington projektet så bidrog det ursprungliga systemet med väldigt mycket pappersarbete för lagledaren, alla beställningar skrevs nämligen ut i en följd, tillsammans med nollavrop, och personen fick sortera ut alla dessa beställningar på manuell väg. Ingen typ av plocklistor fanns, utan en extra etikett för det ändamålet fick istället skrivas ut.
Under utveckling av sekvensprogrammet så behövdes det inte mer än tre prototyper, de varv som togs var relativt kompletta och detta var ett val för att inte behöva göra så många iterationer. Genom att vid varje ny iteration ta reda på vad som saknades kunde antalet hållas nere. SyncroTransfer tillåter synkron, oavsett om det är normala Volvo synkron eller sådana skickade ifrån fabriken i Uddevalla att gå igenom och lagras i databasen. STOACT rapporterna skapas dygnsvis vid en angiven tidpunkt.
Det finns i dagsläget ett par systembuggar, systemfel som uppkommit efter systemstarten som är viktiga att nämna, SyncroTransfer och Sekvensprogrammet kan vid vissa tillfällen göra så att databastabeller låser sig av okänd anledning. Problemet uppstår när sekvensprogrammet stängs av och startas om, men bidrar inte med något problem då låsningen inte varar speciellt länge (5-10 minuter). Det finn även en logisk bugg som uppstår när man i lagersystemet levererar in material i systemet, allt material som kommer in under dagen kommer på kvällen med på STOACT rapporten för att leverantören skall vara medveten om vad som finns i lager. Om inkommet material under dagen har blivit använt under samma dag kommer detta materialet inte att synas på rapporten.
Vidare så medför Sekurit och Pilkington projektet ett problem när nya rack skapas. Problemet uppstår när ett såkallat nollavrop ligger som nästa artikel i det nya racket. När racket då skapas kommer sekvensprogrammet att hoppa över den artikeln (vilket är riktigt då den i inte skall ta upp en plats i det fysiska racket). När racket kompletteras sedan så makuleras inte nollavropet. Detta synkro kommer att ligga kvar i databasen tills att det tas bort manuellt och detta är naturligtvis inte bra. Förbättringar som kan tillkomma till systemet är en streckkodskontroll, det skulle säkerhetsställa rackens innehåll väsentligt mer än i dagsläget. Hjälpfunktioner för sekvensprogrammet skulle vara bra för att tillåta vem som helst av arbetarna att skriva ut etiketter vid behov. En hantering för sk. extraleveranser borde finnas med. En extraleverans är när Volvos personal ringer och beställer en artikel som inte skickas ut den vanliga vägen. Detta inträffar om man vid produktionslinan stött på problem med en artikel och den av okänd anledning inte går att använda. Denna extraleverans vill man ha in i datorsystemet för att kunna räkna av lagersaldon och få med den på STOACT rapporten.

8. Slutsats

Projektet har under den knappa tid som har funnits tillgänglig nått de mål som varit uppsatta, nämligen att sätta upp den nödvändiga datormiljö för att köra sekvenshantering. Det program som skrivits för att sköta utskrifter av de pappersdata som behövts tas ut ur systemet fungerar tillfredställande. Det kraschar inte, tappar inte bort något material, räknar inte fel och uppfattas inte fel av användarna, det går även att anpassa till eventuellt fler projekt i framtiden. Genom att undersöka det befintliga logistiksystemet som finns i bruk har det program som utvecklades för att köra sekvenshanteringen kunnat kopplas in direkt mot detta logistik/lager system.
Den prototyputveckling som användes vid framtagningen av sekvensprogrammet fungerade bra, då det i den första utvecklingsrundan kunde ta fram en version som var direkt anpassad för IT insatta. När den sedan skulle anpassas för de logistiskt insatta medarbetarna bidrog förändringarna till att den blev lätt att förstå för dem som arbetar med den. Vid eventuella förändringar av de pappersdata som skall plockas ut behöver man endast förändra rapporterna

med Crystal Reports, vilket gör programmet hållbart även om en sådan förändring är tvungen att genomföras. Det tog inte heller så många utvecklingsvarv som man i början kunde tro att det skulle göra. Programmet är skrivet i C#.Net vilket medför att det inte kommer bli svårt att hitta resurser för att tillföra ändringar i framtiden.
Det har ur administrationssynpunkt varit viktigt för SP:s IT avdelning att systemet inte förändras så mycket ifrån det ursprungliga programmen för att spara tid på utbildning av personal.

9. Referenser

[1] O'Grady, PJ(1990), Just-In-Timefilosofin i praktiken

[2] Nicolaou, Andreas(2000), Adoption of just-in-time and electronic data interchange systems and perceptions of cost management systems effectiveness

[3] http://www.volvo.com/NR/rdonlyres/E01BF3B3-2FAE-4EA7-AD2D-5ADF1388105C/0/odette.pdf , 2004-05-17

[4] http://www.volvo.com/NR/rdonlyres/629CD720-E2E3-47F0-987C-CACBB028156A/0/stoact.pdf 2004-05-17

[5] Andersen, Erling S(1994), Systemutveckling, kap 17.

[6] Sommerville, Ian (2001). *Software Enginering*. (Kap. 19)

[7] Sommerville, Ian (2001). *Software Enginering*. (kap. 12 s270)

[8] Henderson, Ken(2000), The Gurus's guide to Transact-SQL (kap 16)

[9] Regnell**,** Björn(1995), Improving the Use Case Driven Approach to Requirements Engineering