Utveckling av en datamodell för lagring av meddelandedefinitioner på ett pappersbruk

Development of a data model for storing message definitions at a mill

Anders Berglund
Erik Sundström

EXAMENSARBETE
Informationsteknologi
Nr: E3357IT

2006

HÖGSKOLAN
Dalarna

EXAMENSARBETE, C-nivå
Informationsteknologi

Program	Reg nr	Omfattning
Informationsteknologi, 140 p	E3357IT	10 p
Namn	Datum	
Anders Berglund Erik Sundström	2006-01-24	
Handledare	Examinator	
Pär Douhan	Owen Eriksson	
Företag/Institution	Kontaktperson vid företaget/institutionen	
Stora Enso Kvarnsveden	Sören Lövgren	
Titel		
Utveckling av en datamodell för lagring av meddelandedefinitioner på ett pappersbruk		
Nyckelord		
datamodell, normalisering, meddelanden, systemarkitektur, meddelandesamverkan, VBS		

Sammanfattning

Uppsatsen redovisar resultatet av en kartläggning och datamodellering av definitioner för de meddelanden som sänds mellan olika stödjande informationssystem på Kvarnsvedens pappersbruk. Därtill redovisas även en analys av systemarkitekturens påverkan på problem vid felsökning av meddelandetrafiken mellan olika system.

Målet var att ta fram en datamodell för en metadatabas innehållande definitioner för samtliga meddelanden som sänds mellan informationssystemen på Kvarnsveden. Tanken med databasen är att den ska kunna kopplas till ett system för felsökning av meddelandetrafiken mellan informationssystemen. För att nå målet samlades dokumentation om meddelanden in och en datamodellering med stöd av tredje normalformen genomfördes.

Syftet med arbetet var huvudsakligen att förändra hanteringen av meddelandedefinitioner, för att möjliggöra ändrade rutiner vid felsökning i meddelandetrafiken mellan informationssystem på Kvarnsvedens pappersbruk. Syftet bestod också i att studera hur val av systemarkitekturen kan vara avgörande vid problem som uppstår vid felsökning av meddelanden som sänds mellan de olika systemen.

Resultatet visar att en gemensam datamodell för definitioner över olika typer av meddelanden är svår att åstadkomma. Meddelanden som är uppbyggda av fält med fasta längder kan inte beskrivas i samma datamodell som meddelanden med segment och dataelement avgränsade med en segmentstagg. Istället får man använda två modeller.

Resultatet från analysen av systemarkitekturens påverkan av problem vid felsökning visar, att det går att identifiera några problem där det finns ett samband. Problemen rör främst områden så som ansvar, dokumentation och systemarkitekturens uppbyggnad.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

DALARNA
University College

DEGREE PROJECT
Information technology

Programme	Reg number	Extent
Information technology 140 p	E3357IT	15 ECTS
Name of student	Year-Month-Day	
Anders Berglund Erik Sundström	2006-01-24	
Supervisor	Examiner	
Pär Douhan	Owen Eriksson	
Company/Department	Supervisor at the Company/Department	
Stora Enso Kvarnsveden	Sören Lövgren	
Title		
Development of a data model for storing message definitions at a mill		
Keywords		
data model, normalization, messages, system architecture, message passing, VBS		

Summary

This paper presents the results of a survey and data modelling of definitions for the messages sent between different supporting information systems at Kvarnsveden mill. Furthermore it presents an analysis of the effect of the system architecture on error searching the message traffic between different systems.

The goal was to present a data model for a metadatabase containing definitions for all messages being sent between the information systems at Kvarnsveden. The purpose of the database is to connect it to an error searching system for message traffic between information systems. In order to reach this result message documentation was collected and a data modelling with support of the third normal form was conducted.

The purpose of this degree project is mainly to change the handling of message definitions, to enable routine changes for error searching of message definitions between information systems at Kvarnsveden mill. The purpose was also to study how the choices of system architecture can be conclusive in problems with error searching of messages being sent between the different systems.

The result shows that a common data model for definitions of different types of messages is hard to obtain. Messages that are built on fields of a set length can not be described in the same data model as messages that consist of segments or data elements that are delimited by a segment tag. Instead you will have to use two models.

The result of the analyses of the effect of the system architecture on error searching shows that some problems can be identified where there are commonalities. The problems mainly concern areas such as responsibility, documentation and the structure of the system architecture.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

Innehåll

1 Inledning

1.1 Bakgrund

Stora Enso är en skogsindustrikoncern med tillverkning av tryck- och finpapper, förpackningskartong samt träprodukter. Inom dessa områden är koncernen marknadsledande. Idag har Stora Enso produktion i mer än 40 länder däribland Sverige. Ett av pappersbruken som tillverkar tryckpapper i Sverige är Kvarnsvedens pappersbruk i Borlänge.

Arbetssättet som man följer på Kvarnsveden är processbaserat, där en av huvudprocesserna är OTL-processen (order, tillverkning och leverans). Idag har man en mängd datasystem som samverkar för att stödja OTL-processen.

Systemen är strukturerade så att de samverkar genom att de utbyter information med varandra. Utbytet sker genom att elektroniska meddelanden skickas och tas emot av de olika systemen. Att meddelandetrafiken mellan systemen går fram som det är tänkt är väldigt viktigt för att hela produktionen av papper ska kunna fungera. Ett längre stopp i meddelandetrafiken kan leda till produktionsstopp eller försenade leveranser med relativt stora ekonomiska förluster som följd. Därför är det nödvändigt att snabbt hitta eventuella fel som kan uppstå. För att det ska vara möjligt krävs en god översikt av vilka meddelanden som skickas och en god förståelse för hur systemen samverkar.

1.2 Problem

När något går fel vid skickandet av meddelanden mellan OTL-processens system får man idag göra manuell felsökning med komplicerade loggfiler (filer med utförda aktiviteter och meddelandetrafik). Varje meddelande består av en lång teckensträng. För att förstå vad teckensträngen betyder måste man räkna tecken i den och jämföra mot en meddelandedefinition. Det är både ett tidsödande och ineffektivt arbete. Det finns idag definitioner över många meddelanden och hur de kommunicerar med systemen. Man är dock osäker på om de är helt aktuella och om alla meddelanden som finns är dokumenterade. Osäkerheten innebär att översikten och förståelsen inte alltid är klar.

Man vill därför från systemgruppens sida förenkla rutinerna för felsökning, så att man inte behöver läsa meddelandedefinitionen och manuellt översätta teckensträngen. Detta vill man åstadkomma genom att i framtiden använda ett felsökningsverktyg som hämtar meddelandedefinitioner lagrade i en databas.

1.3 Mål

Målet med arbetet är att ta fram en datamodell för en databas, som ska innehålla definitioner för strukturen över de meddelanden som skickas mellan OTL-processens system. Databasen ska i framtiden kunna användas av olika system eller program för att automatiskt hämta meddelandedefinitioner. Det kan t.ex. vara ett felsökningssystem för uppkomna fel vid skickandet av meddelanden mellan olika system.

1.4 Syfte

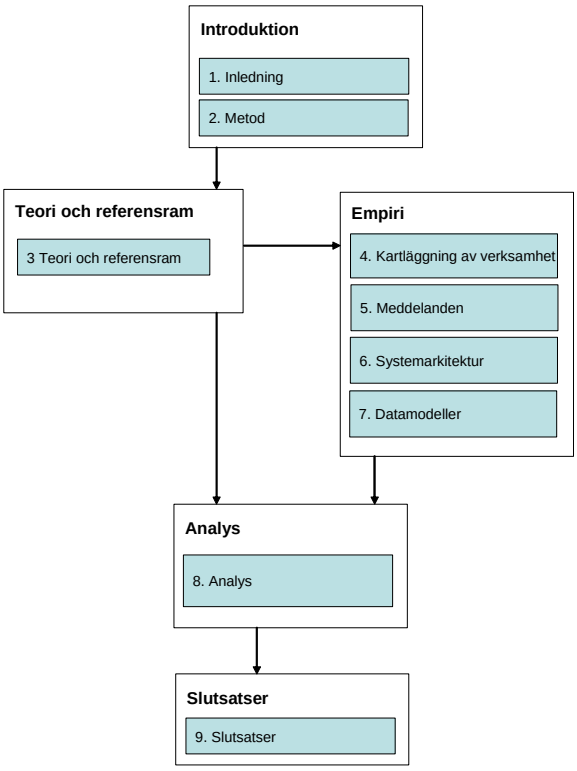
Syftet med arbetet är att förändra hanteringen av meddelandedefinitioner för att möjliggöra ändrade rutiner av felsökning i loggfiler.

Ett delsyfte är att studera om och hur val av systemarkitekturen kan vara avgörande för problematiken vid felsökningsarbete i meddelandetrafiken mellan system i en verksamhet.

1.5 Avgränsning

Studien av meddelanden kommer enbart att omfatta de som tas emot och skickas från system som verkar inom OTL-processen.

1.6 Disposition och läsanvisningar



Figur 1.1 Uppsatsens disposition

Kapitel 1 – Inledning
Beskriver bakgrunden till arbetet tillsammans med syfte och mål. Dessutom framgår det här vem som är uppdragsgivare.

Kapitel 2 – Metod
Beskriver de metoder vi har valt att använda i arbetet vid informationsinsamling och analys av insamlat material.

Kapitel 3 – Teori och referensram
Behandlar begrepp och teorier som hämtats från litteratur och andra skriftliga källor och som är väsentliga för empirin.

Kapitel 4 – Kartläggning av verksamhet
Visar hur verksamheten med utgångspunkt från informationssystemen fungerar. Systemen och miljön beskrivs utifrån den process i verksamheten de följer. Kapitel 4, 5, 6 och 7 tillhör empirin som ska ge läsaren en möjlighet att skapa sig en bild över verkligheten som den ser ut på Kvarnsveden idag.

Kapitel 5 – Meddelanden
Beskriver strukturen för de olika meddelanden som sänds mellan system.

Kapitel 6 – Systemarkitektur
Visar den systemarkitektur som finns på Kvarnsvedens pappersbruk. Arkitekturen beskrivs utifrån kriterier som t.ex. hur informationssystemen utbyter information, vem som ansvarar för information och system och synen på information.

Kapitel 7 - Datamodeller
Datamodeller som kan ligga till grund för en databas för meddelandestrukturen presenteras i detta kapitel.

Kapitel 8 – Analys
Här beskrivs de egenskaper som systemarkitekturen på Kvarnsvedens pappersbruk har och som framkommit genom en delvis egenframtagen analysmetod. Dessutom presenteras problem som finns med arkitekturen och som eventuellt går att koppla till problematiken för denna studie. Även problem som uppkommit vid framtagandet av datamodeller för de meddelandestrukturer som finns presenteras.

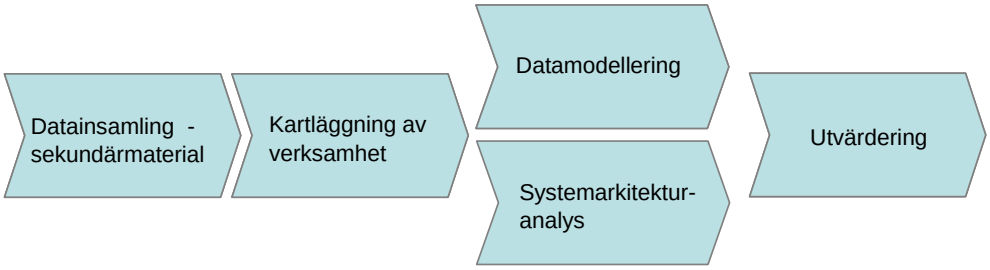
Kapitel 9 - Slutsatser
En sammanfattning av resultatet från analysdelen samt den reflektion vi gjort och de slutsatser som vi dragit.

2 Metod

För att åstadkomma en datamodell över strukturen för de meddelanden som skickas mellan informationssystemen, har det varit nödvändigt att samla information om deras uppbyggnad. Dessutom har fakta om systemarkitekturen och dess sammansättning behövt inskaffas, för att undersöka hur valet av systemarkitektur kan ha påverkan på problematiken i studien. Därefter utfördes vissa analyser på dessa fakta.

2.1 Metodöversikt

Studien delades in i ett antal faser (se figur 2.1). Den inledande fasen innefattade insamling och studier av sekundärdata. Data förekom främst i form av befintliga definitioner och dokumentationer gällande meddelanden. För att kunna kartlägga och undersöka hur systemarkitekturen är uppbyggd, utfördes en rad kvalitativa intervjuer av strategiskt viktiga personer inom verksamheten. Utifrån insamlad data utfördes därefter en analys av systemarkitekturen. Resultatet låg sedan till grund för undersökningen av arkitekturens påverkan på problematiken i studien. Det utfördes även parallellt en datamodellering över strukturerna för meddelandena som skickas mellan informationssystemen på Kvarnsvedens pappersbruk.



Figur 2.1 Illustration av metodöversikten

2.2 Datainsamling – sekundärmaterial

För att skapa sig en övergripande bild över meddelandestrukturen samt att klargöra systemarkitekturens utseende i verksamheten, är det nödvändigt att studera redan befintlig dokumentation om detta. Genom att samla in information om systemarkitekturen och de olika systemens meddelandetrafik kan man få reda på vilka olika typer av meddelandestrukturer som förekommer och se hur strukturen över systemen ser ut.

2.2.1 Genomförande

Arbetet inleddes med att samla in all befintlig dokumentation som kunde hittas rörande ämnet. Materialet fanns främst i form av textdokument lagrade på företaget. Därefter gjordes en sortering av materialet efter vilket system i verksamheten det tillhörde. Då sorteringen var genomförd studerades några meddelanden för de olika systemen samtidigt som de jämfördes med innehållet i lagrade loggfiler. Syftet var att identifiera alla typer av meddelanden och kartlägga uppbyggnaden av dem. Även grafer över hur systemen relaterar till varandra studerades.

2.3 Kartläggning av verksamhet

För att ytterligare klargöra systemarkitekturen och tanken bakom den är det väsentligt att kartlägga verksamheten. Det är också viktigt för att undersöka vilka meddelanden som förekommer. Genom en sådan kartläggning kan man även identifiera OTL-processens utseende. Man får även möjlighet att identifiera eventuella problem som kan vara kopplade till valet av systemarkitektur.

2.3.1 Kvalitativa intervjuer

Det beslutades att kvalitativa intervjuer var den mest lämpliga insamlingsmetoden att tillgå för att kartlägga verksamheten på Kvarnsvedens pappersbruk. För att på den relativt korta tid som studien genomfördes hinna göra en kartläggning var det nödvändigt att på ett flexibelt och effektivt sätt samla in data. Dessutom passar kvalitativa intervjuer för studiens syfte, då man med en kvalitativ intervju bl.a. syftar till att identifiera okända eller kända otillfredsställande företeelser (Gustafsson och Sörman 2004, s. 7).

Att utföra intervjuer med kvantitativ inriktning var inte aktuellt då syftet med sådana är att utifrån på förhand definierade företeelser formulera frågor med fasta svarsalternativ och undersöka hur svaren fördelas på en rad utvalda intervjupersoner (Starrin och Svensson 1996, s. 53-55). Syftet med kartläggningen var bl.a. att utröna för oss okända problem i samband med systemarkitekturen på pappersbruket. Vi bedömde det utifrån detta faktum omöjligt att utföra kvantitativa intervjuer för att effektivt kunna samla in data.

2.3.2 Genomförande

För insamlandet av eftersökt data valdes fem personer ut för intervju. Personerna är i denna rapport anonyma. Dessa hade alla viktig kunskap inom olika områden däribland meddelanden, systemarkitekturen och viktiga moment inom OTL-processen. Varje person intervjuades var för sig och till varje intervju togs en intervjuguide fram, vilken var anpassad efter det område personen i fråga hade kunskap inom. Syftet med en intervjuguide var att kunna anpassa intervjun beroende på hur mycket information vi hade inhämtat inom området (Starrin och Svensson 1996, s. 62). Några exempel på frågor i intervjuerna är:

- Var ska faktura in någonstans i processen?
- Kan produktionen ändra på en order?
- Anskaffas data endast en gång eller flera gånger? Dubbellagras data?
- Är magasinssystemet och TPS två olika system?
- Vad görs i TPS och av vem?

Intervjuerna pågick mellan en och två timmar. Under tiden förde de båda intervjuledarna anteckningar med papper och penna. Efter intervjun sammanställdes anteckningarna till ett gemensamt dokument. Vid ett intervjutillfälle gjordes även en ljudinspelning. Anledningen till att det bara användes vid en av fem intervjuer var att den intervjun omfattade många tekniska detaljer och berörde stora delar av verksamheten. Vi bedömde det svårt att hinna anteckna alla detaljer eller att efteråt komma ihåg vad som sagts.

2.4 Systemarkitekturanalys

När tillräckligt med data samlats in kring systemarkitekturen på Kvarnsvedens pappersbruk utfördes en systemarkitekturanalys. Genom analysen kan man identifiera om arkitekturen mer eller mindre överensstämmer med någon idealtyp. Man kan utifrån teorier om idealtyperna också påvisa eventuella problem som systemarkitekturen kan ge upphov till och möjligen går det att koppla ihop dessa till problematiken formulerad i denna studie.

2.4.1 Genomförande

Eftersom det inte finns någon vedertagen metod för att identifiera systemarkitekturer hos en organisation har vi valt att delvis konstruera en egen metod. Vi utgick från de två systemarkitekturstrategierna VBS (Verksamhetsbaserad systemstrukturering) och IRM (Information Resource Management) se punkt 3.4 och 3.5. Dessa strategier presenteras i boken Strukturering av informationssystem (Axelsson och Goldkuhl 1998, s. 35-52). Författarna gör där en idealtypisk jämförelse av de båda strategierna, i vilken man utgår från en rad kriterier för att se skillnader. Vi har valt att i vår analys använda några av dessa kriterier för att beskriva och identifiera vilken typ av arkitektur som används på Kvarnsveden idag. De kriterier som använts är följande:

- **Arkitekturprincip** – En sammanfattande beskrivning av arkitekturuppbyggnaden och hur informationsförsörjningen går till i verksamheten.
- **Syn på information** – Den syn man har på informationen och vilken roll den har i verksamheten. I VBS ses t.ex. information som en viktig resurs inom den verksamhetsfunktion som förfogar över den. Det är funktionen som bestämmer hur informationen ska samlas in, lagras och användas.
- **Informationssamverkan** – Avser sättet som olika informationssystem inom verksamheten förmedlar information mellan sig.
- **Ansvar för data, informationssystem och informationsarkitektur –** Vem i verksamheten som ansvarar för att data samlas in och lagras i ett visst informationssystem och hur dessa system förmedlar eventuell information mellan sig.
- **Förändring kontra stabilitet –** Synen man har på arkitekturen i verksamheten med utgångspunkt från om det är meningen att den ska vara stabil eller förändringsbar över tiden.
- **Tillgänglighet och spridning av data i verksamheten –** Det handlar om vem som har rättigheter att komma åt viss information inom verksamheten och vilka åtkomstbegränsningar som finns. Det handlar även om hur man sprider och kommer åt data rent tekniskt.
- **Anskaffning av data –** När och hur många gånger information i verksamheten tas fram och lagras t.ex. om man tillåter dubbellagring av data.
- **Datastrukturering och datalagring –** Det avser hur man strävar efter att data ska struktureras t.ex. med hjälp av datamodeller, men också hur man anser att data ska lagras. Inom t.ex. IRM vill man göra all information tillgänglig för alla inom verksamheten och därför struktureras data kring en gemensam databas.

2.5 Datamodellering

Datamodelleringen ska inledningsvis identifiera de viktigaste objekten (grundläggande begrepp) inom det område databasen ska lagra data om. Därefter ska relationerna mellan dessa objekt klargöras. Utifrån de identifierade objekten och deras relationer tas en datamodell fram i vilken man beskriver strukturen för hur data ska se ut och vilka tabeller som den tänkta databasen ska innehålla.

För att åstadkomma en datamodell som ska leda till en databas som är lätt att underhålla och uppdatera med nya tabeller eller attribut finns flera olika metoder att tillgå. I detta arbete har metoden normalisering använts (se punkt 3.7.1).

2.5.1 Genomförande

Efter att ha samlat in data om meddelanden identifierades ett antal olika meddelandetyper. Uppbyggnaden av dessa meddelanden studerades och utifrån resultatet utfördes datamodelleringar för alla påträffade meddelandetyper. I största möjliga mån togs modeller fram enligt tredje normalformen (se punkt 3.7). Resultatet av datamodelleringen var tänkt att bli en eller flera datamodeller som uppdragsgivaren lätt kan implementera och underhålla.

2.6 Utvärdering

I denna fas genomförs en återkoppling och utvärdering av arbetet. Hur gick det, vad gjorde vi, var resultatet väntat, vad kunde ha gjorts bättre? Resultatet av utvärderingen kan sedan ligga till grund för vad man bör tänka på vid utförandet av liknande studier.

2.6.1 Genomförande

Utvärderingen har inte planerats och utförts efter någon fastslagen metod. Det har dock förts dagbok under arbetets gång samtidigt som det skickats fortlöpande rapporter till handledaren på Kvarnsvedens pappersbruk. Detta har gett tillfällen till reflektioner och kritiska diskussioner om studien vid ett flertal tillfällen under den tid den pågått.

3 Teori och referensram

Begrepp och teorier som är väsentliga för empirin.

3.1 Systemarkitektur

Idag är ofta datorer och olika system för informationsbehandling en viktig beståndsdel i många organisationers verksamheter. Många delar har de senaste decennierna kunnat datoriseras, vilket lett till att de flesta organisationer blivit beroende av datoriserade informationssystem. Till en början rörde det sig hos de flesta bara om något enstaka system, men med tiden har de hos många kommit att växa både i storlek och i antal. Varje system uppfattas ofta som något bestämt och avgränsat innehållande information och funktioner för informationsbehandling. Hur man strukturerar och organiserar alla systemen ser olika ut för de flesta organisationer. Det finns inget enskilt vedertaget sätt för hur en systemarkitektur ska se ut. Enligt Axelsson och Goldkuhl (Axelsson och Goldkuhl 1998, s. 20) beskriver en systemarkitektur hur informationssystemen binds ihop och relaterar till varandra med utgångspunkt från verksamhets- och informationsmässiga aspekter. Det finns flera olika principer och strategier som kan tillämpas för att strukturera arkitekturen. Exempel på sådana strategier är VBS och IRM (se punkt 3.4 och 3.5).

3.2 Information

Information är något som kan finnas om i princip allt i ett oändligt antal former. I många fall är information av olika slag mycket viktig för organisationers verksamheter. Ett exempel där information är mycket central är när en kund sänder en order till en producent av något slag. Eller det motsatta, då producenten lämnar en orderbekräftelse. Man kan dock utgå ifrån att alla typer av information är något som sägs av någon till någon (Axelsson och Goldkuhl 1998, s. 17). Med andra ord utgår vi ifrån att ingen information förekommer utan kommunikation.

3.3 Informationssystem

Med informationssystem kan man avse system som behandlar data på ett datoriserat vis så väl som manuellt (Axelsson och Goldkuhl 1998, s. 18). Eftersom denna studie ska studera system som hanterar datoriserad information kommer utgångspunkten vara att informationssystem eller system endast behandlar datorbaserad information. Fortsättningsvis kommer informationssystem och system betraktas som samma sak.

I informationssystem finns det olika delar som hanterar information som skapas, skickas eller tas emot. Därför kan man anta att följande funktioner finns att tillgå i ett informationssystem:

- Insamling/inmatning
- Bearbetning
- Lagring
- Återsökning
- Överföring
- Presentation

Information kommer hela tiden att finnas i olika former, t.ex. inmatad data, bearbetad data o.s.v.

3.4 VBS (verksamhetsbaserad systemstrukturering)

En systemarkitektur som helt och hållet är strukturerad enligt VBS-strategin (verksamhetsbaserad systemstrukturering) består av autonoma informationssystem där systemen är oberoende av varandra och där de samverkar genom meddelandeutbyte. Varje system stödjer endast en avgränsad del av verksamheten. Det finns inga gemensamma system för flera funktioner i verksamheten. Informationssystemen och informationen i dessa ägs och förvaltas av den verksamhetsfunktion som systemen stödjer. Det innebär att varje funktion ansvarar för hur informationen ska samlas in, lagras och användas. När någon annan funktion behöver tillgång till informationen överförs den med hjälp av meddelanden mellan systemen (Axelsson och Goldkuhl 1998, s. 53-54).

VBS-strategin lägger stort ansvar på systemägaren som måste se till att informationsförsörjningen fungerar på bästa sätt och att man kan samverka med andra system på det sätt man bestämt. Verksamhetsfunktionerna beslutar själv om förändringar inom sitt system så länge samverkan med andra system upprätthålls. Vid en förändring ska inget annat system påverkas eftersom de ska vara avgränsade från varandra. På det viset blir informationsflödet mellan systemen relativt stabilt över tiden och oberoende av förändringar inom ett system (Axelsson och Goldkuhl 1998, s. 54-55).

I och med att informationssystemen är oberoende av varandra har varje system lokala databaser för att lagra information. Inget annat system tillåts att hämta eller lämna information i databasen, utan allt utbyte sker genom meddelandesamverkan. För att definiera vilken information som ska utbytas, upprättas sambandskontrakt mellan de berörda verksamhetsfunktionerna. Ett sambandskontrakt anger vilket system som är sändare och vilket som är mottagare, hur ofta informationen ska sändas och hur sambandsinformationen ska se ut. Det godkänns och undertecknas sedan av de systemansvariga för sändande och mottagande system. Genom kontrakten så blir de systemansvariga skyldiga att tillhandahålla aktuell information. Det är företagsledningen som är ansvarig för att sambandskontrakt upprättas (Axelsson och Goldkuhl 1998, s. 58-59).

VBS-strategin bygger på att avgränsningen av informationssystemen följer ansvarsstrukturen i organisationen. Det är viktigt att verksamhetens ansvar är tydligt definierat så att användarna i verksamhetsfunktionerna är medvetna om vad de ansvarar för och var ansvarsgränserna går. Ansvarsstrukturen måste vara tydlig redan innan ett system införs för att VBS-strategin ska fungera i praktiken. (Axelsson 1998, s. 90) När sedan verksamheten och ansvarsstrukturen förändras bör även informationssystemen förändras på motsvarande sätt. Sambandsinformationen behöver kanske inte ändras men meddelandena får andra system som sändare eller mottagare och då måste nya sambandskontrakt upprättas. Ett enstaka informationssystem kan dock förändras och anpassas utan att nya kontrakt behöver upprättas om man inte förändrar sambandsstrukturen (Axelsson och Goldkuhl 1998, s. 56-57).

Eftersom det inte förekommer några gemensamma databaser eller någon korsvis läsning mellan systemen så anskaffas lokal information inom verksamhetsfunktionen medan övrig information fås via meddelandesamverkan. Samma information kan finnas i flera olika verksamhetsfunktioner vilket innebär att man dubbellagrar information. Genom att varje informationssystem är oberoende av andra system så kan driftmiljön variera från system till system. Varje verksamhetsfunktion kan välja den miljö som lämpar sig bäst så länge man kan sända och ta emot sambandsinformationen. Oberoende system leder även till låg sårbarhet eftersom varje system kan fungera för sig själv även om andra system fallerar. Det innebär också att man lättare kan stoppa ett system för t.ex. underhåll eller uppgradering (Axelsson och Goldkuhl 1998, s. 59).

3.5 IRM (Information Resource Management)

Det finns flera olika sätt att planera och strukturera samverkan mellan informationssystem inom en verksamhet och mellan olika verksamheter. Oftast utgår man från en rad principer och synsätt vid de val man gör vid strukturingen av informationssystemarkitekturen. En annan principsamling för strukturering av system i en systemarkitektur är IRM (Information Resource Management).

Inom IRM ser man information inom en verksamhet som en resurs av samma värde som t.ex. arbetskraft och maskiner. När informationen anses som en så viktig del i verksamheten krävs det att den styrs och administreras lika mycket som andra resurser. Därför utses vanligtvis en funktion i verksamheten med ansvar för detta (Axelsson och Goldkuhl 1998, s. 35).

Informationsförsörjningen inom en verksamhet som har en informationsarkitektur baserad på IRM-strategin bygger på en eller ett fåtal centrala integrerande databaser. Databaserna innehåller verksamhetens hela samlade information och är nåbar av alla användare inom verksamheten. Strukturen innebär också att databaserna och den samlade informationen skiljs från lokala applikationer eller informationssystem i verksamheten (Axelsson och Goldkuhl 1998, s. 53). Informationen utbyts mellan informationssystemen via registersamverkan. Det innebär att kommunikation sker via den lagrade informationen i databasen. Ett informationssystem kan läsa eller uppdatera information som andra informationssystem skapat i databasen, dock med eventuella behörighetsbegränsningar kopplat till t.ex. användartyper (Axelsson och Goldkuhl 1998, s. 54).

Ansvaret för informationen och att administrera informationen inom IRM ligger på en central dataadministrationsfunktion. Men på ledningen i verksamheten ligger också ett visst ansvar. De ska dels definiera gemensamma begrepp i verksamheten samt se till att data bara anskaffas en gång och det vid källan. Samma data hämtas sedan alltid vid samma ställe. Detta gör man för att undvika dubbellagring och inkonsistent information i databasen. Dataadministrationsfunktionen har i övrigt ansvaret för alla enskilda informationssystem och för hela informationssystemarkitekturen (Axelsson och Goldkuhl 1998, s. 54-55).

Det man vill uppnå med IRM är att data och själva datastrukturen ska vara stabil men att informationsbehovet i verksamheten ska kunna förändras. Det innebär att data struktureras så att användningen blir oberoende av data- eller organisationsstrukturen. Objekten som datastrukturen bygger på blir det som utgör kärnan av informationssystemen och skapar stabilitet. Även om själva objekten blir stabila inom IRM kan informationsbehovet och typer av informationsuttag ur databasen ändras. Man uppnår ett data- och programoberoende som ger en robusthet åt arkitekturen. Det kan dock innebära problem om det skulle uppkomma behov av nya objekt inom informationssystemarkitekturen (Axelsson och Goldkuhl 1998, s. 55).

För att nå målen med IRM-strategin vill man ta fram en gemensam databas för all information inom verksamheten. Ansatsen man använder för att ta fram en sådan databas är datadriven systemutveckling. Det centrala i den systemutvecklingen är datamodelleringen, vilken bygger på att man gör en struktur över organisationens viktigaste dataobjekt. Modellen blir en bild över verksamheten där objekten motsvarar verkliga företeelser och ligger till grund för databasen. Modellen ligger också till grund för så väl enskilda informationssystem som hela informationssystemarkitekturen (Axelsson och Goldkuhl 1998, s. 58).

3.6 Meddelanden

Definitionen av ett meddelande är i den här studien en samling av data som i elektronisk form skickas från ett system till ett annat. Strukturen på ett meddelande kan variera men sändande och mottagande system måste vara överens för att utbytet ska vara meningsfullt. Det finns olika typer av meddelanden och några vanliga benämningar är datagram, förfrågan, svars- och statusmeddelande. Ett datagram behöver inget svar från mottagande system att det kommit fram. Datagram används t.ex. när det inte är helt nödvändigt att meddelandet kommer fram därför att det snart kommer att skickas ett nytt meddelande med uppdaterad information. En förfrågan skickas från ett system som vill ha en viss information av ett annat system. Det system som fått en förfrågan skickar då tillbaka informationen i form av ett svarsmeddelande (IBM 2005a). Statusmeddelanden kan skickas av ett system för att meddela sin status för det andra systemet. Det kan t.ex. ha uppstått ett fel när ett system försöker bearbeta ett mottaget meddelande. För att underrätta det sändande systemet kan ett statusmeddelande skickas av mottagaren (IBM 2005b).

Det finns ett antal olika standarder för meddelanden men många företag använder egenutvecklade meddelanden. Några av de vanligaste standarderna är EDIFACT (ISO 9735), ANSI X12 och XML (en del av SGML ISO 8879) (Fredholm 2002, s. 139). Gemensamt för många meddelandetyper, både egenutvecklade och standardiserade, är hur de är strukturerade. Det är vanligt att de börjar med ett meddelandehuvud kallat header och avslutas med en meddelandefot kallad trailer. En header och trailer kan bl.a. innehålla information om sändande och mottagande system. De kan liknas vid ett kuvert med avsändarens och mottagarens adress och som omsluter meddelandets innehåll. Om innehållet är uppdelat på flera poster omsluter man dessa med posthuvud och postfot som innehåller gemensam information för alla poster. I mitten av meddelandet finns den information man vill föra över. På det här sättet får man en hierarkisk struktur på meddelandet som gör att innehållet enkelt kan separeras från övrig information (Hendry 1993, s. 68-70).

3.6.1 EDIFACT

EDIFACT är en standard som används inom EDI, Electronic Data Interchange. EDI är ett koncept för att via elektroniska meddelanden, utbyta information som annars skulle ha utbytts i skriven form mellan olika system (Hendry 1993, s. 3). EDIFACT är en förkortning för EDI For Administration, Commerce and Transport och standardiseringsarbetet drivs av FN-organisationen UNECE, United Nations Economic Commision for Europe. Även organisationerna ANSI och ISO står bakom standarden (Fredholm 2002, s. 250).

Meddelande enligt EDIFACT är uppbyggda av segment och dataelement. Varje segment inleds med en segmentstagg bestående av tre bokstäver, som fungerar som en identifierare. Därefter kommer dataelementen med den information som ska överföras och segmentet avslutas med ett gränstecken (Fredholm 2002, s. 259).

Segmenten i EDIFACT är generiska vilket innebär att samma segment kan användas för olika syften. Genom att använda en kvalificerare i form av en kod direkt efter segmentstaggen specificerar man vilken betydelse segmentet ska ha. För segmentet NAD (Name and Address) kan man t.ex. använda kvalificeraren SU (Supplier) för leverantör eller BY (Buyer) för köpare (Fredholm 2002, s. 259-260).

Dataelementen innehåller den information som man vill överföra med EDIFACT. Ofta används koder som specifikation av datavärden i elementen. Det kan vara alfabetiska, numeriska eller alfanumeriska tecken. EDIFACT:s dataelement finns samlade i en katalog, UNTDED (United Nations Trade Data Elements Directory) som finns utgiven av FN i bokform och som en ISO-standard med beteckningen ISO 7372 (Fredholm 2002, s. 260).

För EDIFACT finns speciella syntaxregler som talar om hur ett meddelande ska struktureras, vilka tecken som får användas etc. Syntaxreglerna finns definierade i ISO-standarden ISO 9735 som även har översatts till svenska med beteckningen SS-ISO 9735. Viktiga kriterier för ett EDIFACT-meddelande är enligt syntaxreglerna hierarkisk strukturering, separation av segment och dataelement genom speciella tecken, variabla längder för data och att de ingående enheterna i ett meddelande antingen är obligatoriska (mandatory) eller villkorliga (conditional). En enhet som är obligatorisk måste ingå i meddelandet och får endast förekomma en gång. Villkorliga enheter behöver inte ingå och när de ingår kan de förekomma flera gånger i samma meddelande (Fredholm 2002, s. 260-262).

3.6.2 XML

XML är förkortning för extensible markup language. Det är ett metaspråk, ett språk som beskriver ett språk. Ett meddelande i XML kan innehålla både information och en definition av vad det är för information. XML används för att strukturera data som sedan kan hanteras olika av olika system. Man kan t.ex. använda HTML för att visa XML-meddelandet i en webbläsare samtidigt som ett annat system kan använda samma meddelande till att läsa in det i en databas (Fredholm 2002, s. 265).

XML är ett märkspråk vilket innebär att ett meddelande byggs upp med hjälp av märkord som även kallas taggar. Det finns inga på förhand definierade taggar utan det är fritt att skapa sina egna. Det viktiga är att mottagaren vet vad taggarna betyder. För att lösa det använder man dokumenttypsdefinitioner DTD eller schema som anger hur taggarna ska tolkas. Ett schema är lite mer avancerat än en DTD eftersom man med hjälp av ett sådant inte endast anger vad taggarna betyder utan också kan ange t.ex. hur många gånger en post får förekomma eller hur många tecken den får innehålla (Fredholm 2002, s. 265-266).

Det som ligger till grund för XML är SGML vilket finns standardiserat som ISO 8879. Samma standard ligger till grund för HTML. Ett XML-meddelande är en textfil och kodningen är baserad på standarden ISO 10646 även kallad Unicode. Det innebär att alla alfabetiska tecken som finns i världen stöds. XML har utvecklats till en familj av standarder med XML som grundstandard. Bland övriga standarder finns t.ex. XSL som definierar hur en XML datafil ska definieras, XML Signature som kopplar digitala signaturer till ett formulär som är strukturerat med XML och SOAP som på ett säkert sätt paketerar och transporterar XML-transaktioner (Fredholm 2002, s. 264, 267).

För att ett system ska kunna hantera ett XML-meddelande behövs en parser. Det är en programvara som läser meddelandet och validerar att det följer regelverket i XML och är syntaktiskt korrekt. För att kunna bygga gränssnitt mot andra system har standarden DOM, Document Object Model tagits fram som ett stöd. DOM är en delstandard som tillhör XML-familjen (Fredholm 2002, s. 273-274).

Ett meddelande i XML struktureras hierarkiskt där varje nivå börjar med en starttagg och avslutas med en stopptagg. Innanför taggarna kan flera nivåer skapas där den innersta nivån innehåller den information man vill överföra. Ett meddelande med kundinformation skulle kunna se ut enligt följande:

```
<Kund>
        <Fnamn>Nils</Fnamn>
        <Enamn>Nilsson</Enamn>
        <Adress>Sveagatan 10</Adress>
</Kund>
```

Fördelen med att själv definiera taggar i XML är också en nackdel eftersom det kräver att sändare och mottagare kommer överens om taggarnas betydelse vid varje transaktion. Man riskerar då att få många olika varianter på samma typ av transaktioner. Trots det så har XML accepterats av många. Flera stora affärssystem och databaser har idag stöd för XML (Fredholm 2002, s. 268-273).

3.7 Databasstrukturer och datamodellering

Det man vill åstadkomma med en datamodelleringsprocess är en datamodell. En datamodell är till för att beskriva de data man vill göra en databas över, t.ex. en varudatabas för en matvarubutik. Modellen ska beskriva koncepten för hur data som används inom det valda området relaterar till varandra. Den kan sedan användas i skapandet av en relationsdatabas. Det finns en rad olika metoder att använda vid datamodellering, t.ex. ER-modellering (Entity - Relationship), normalisering o.s.v. Modellen som skapas utifrån modelleringen dokumenteras vanligen med någon typ av grafisk notation. Syftet med att skapa en datamodell är att göra en konceptuell bild över data som används, vilket gör att den blir lättförståelig (Begg och Connolly 2000, s. 19).

3.7.1 Normalisering

För att designa en datamodell utifrån en datamodellering av de data som ska lagras finns många metoder att tillgå. En vanlig metod är normalisering. Metoden togs fram 1972 av Dr E.F. Codd (Begg och Connolly 2000, s. 103). Avsikten var att normalisering ska användas som ett stöd vid framtagandet av s.k. relationsdatabaser. Användandet går vanligtvis till på så sätt att man utför ett antal tester på de tabeller man skapat i en datamodell för att se om de följer reglerna för en viss s.k. normalform.

Det finns flera olika normalformer t.ex. första normalformen (1NF), andra normalformen (2NF) eller tredje normalformen (3NF). Alla normalformer har det gemensamt att de innehåller regler för hur man skapar en relationsdatadesign som är väl strukturerad (Begg och Connolly 2000, s. 103). Syftet är att man ska få en databas som är lätt att underhålla, uppdatera och i vilken man undviker problem så som att det lagras redundant data i databasen. Redundant datalagring innebär att samma data lagras på flera ställen i samma databas, vilket får konsekvensen att vid ändring av data måste man göra uppdateringar på flera ställen. Det kostar både i form av minnesutrymme och i processorkraft.

Att en databas lagrar redundant data kan också få följden att det uppstår s.k. uppdateringsanomalier. Det kan t.ex. uppstå då man vill göra någon typ av ändring i databasen för data som förekommer på flera ställen, vilket innebär att man måste göra exakt samma uppdatering på alla dessa rader för att inte få inkonsistent data. Man vill undvika sådana problem genom att man bara utför ändringen på en plats i databasen för data som är gemensam. Detta löser man genom att plocka ut just den information som förekommer på flera rader i en tabell och skapar en egen tabell för den.

Som nämnts är några av de vanligaste normalformerna, som man utgår ifrån när man skapar en normaliserad relationsdatabasdesign, första- till tredjenormalformen (1NF – 3NF). De är uppbyggda likt en hierarki. Den första normalformen innehåller grundläggande regler för utformandet av databasdesignen. Samma regler återkommer även i andra normalformen, som dessutom har ytterligare designregler. I sin tur innehåller tredje normalformen samma regler som andra normalformen samt ytterligare några regler. För att skapa en databas som uppfyller reglerna för en relationsdatabas räcker det i teorin att första normalformen är uppfylld, men man rekommenderar normalt att man går så långt att tredje normalformen uppfylls då man också undviker eventuella uppdateringsanomalier som kan uppstå (Begg och Connolly 2000, s. 106).

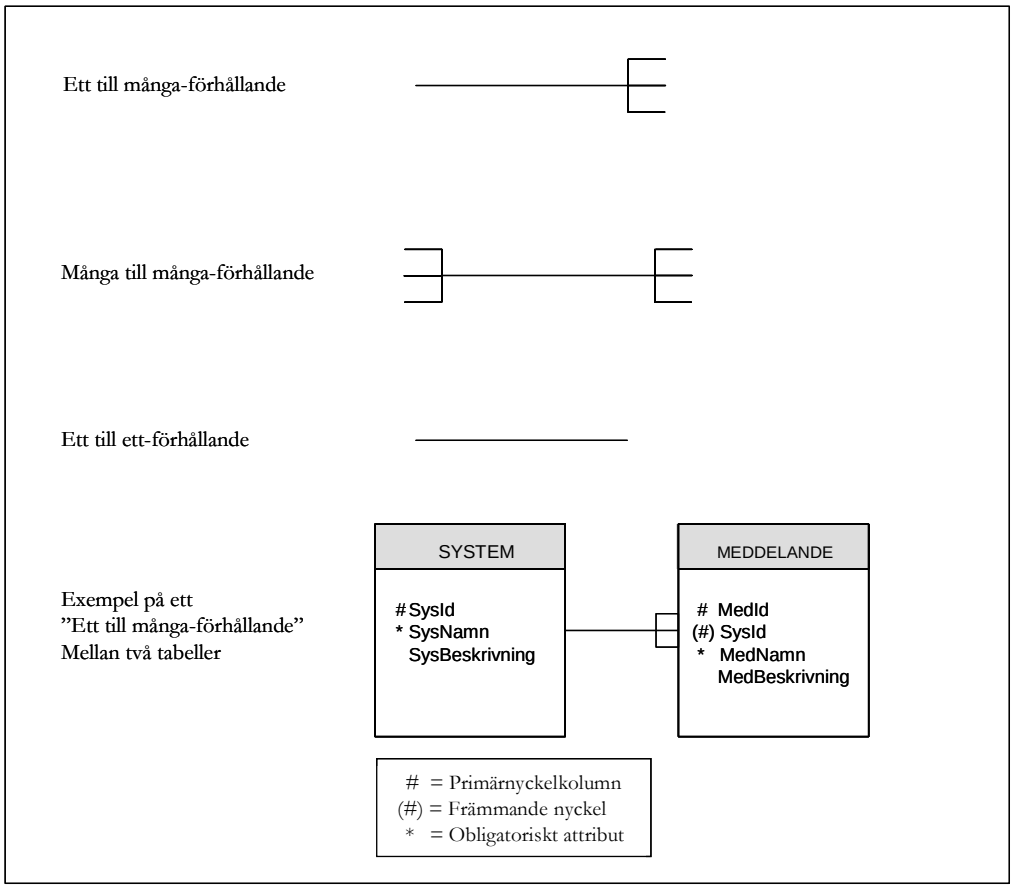
Definitionen för den första normalformen innebär att varje fält i databasen bara får innehålla ett värde. Om det finns behov av fler värden för ett visst ändamål ska detta lyftas ut i en egen tabell i databasen. För en databas som uppfyller andra normalformen krävs det att varje tabell har en primärnyckel. Alla icke-nyckelkolumner i tabellen är i sin tur helt beroende av primärnyckeln. Det betyder att endast attribut som är unikt ihopkopplade med tabellens primärnyckel får förekomma. Alla andra attribut som kan plockas bort ur tabellen utan att beroendet dem emellan ändras ska bort och placeras i en egen tabell. Tredje normalformen uppfylls då alla icke-nycklar i en tabell endast har ett enskilt beroende till primärnyckeln. Det får inte förekomma några inbördes beroenden, d.v.s. att icke-nyckelkolumnerna är beroende av varandra (Begg och Connolly 1999 s. 198-208).

3.7.2 Notation

Det finns flera olika notationstyper man kan använda vid objekt- och datamodellering samt konceptuell design av databaser. Exempel på sådana typer är chen notation, UML och Crow's feet (kråkfötter) notation (Begg och Connolly 2000 s. 343). Det tvistas ibland om vilken som är mest lämplig att använda. I detta arbete har vi valt att använda en något modifierad variant av Crow's feet notation. Det som i stora drag ändrats från den ursprungliga Crow's feet notationen är att kråkfötterna har bytts ut mot gafflar. I övrigt fungerar notationstypen på samma sätt.

Enligt Crow's feet notation finns det tre olika sätt att markera hur objekten (grundläggande begrepp eller entiteter) som tabellerna i en databas representerar relaterar till varandra. Dessa tre sätt är ”många till många”-förhållanden, ”många till en”-förhållanden och ”en till en”-förhållanden (Connolly och Begg 2000 s. 347). För modellen som vi valt markeras detta grafiskt enligt figur 3.1.

Notationen bygger även till stora delar på den s.k. maxregeln. Maxregeln innebär att man beskriver hur många gånger en post av en viss objekttyp kan förekomma relaterat till de andra objekten det är kopplat till. Det innebär t.ex. att ett objekt (symboliseras med en rektangel innehållande namn på nycklar och tabellattribut) kan kopplas ihop med ett annat objekt med hjälp av ett streck som symboliserar en relation objekten emellan. I streckets ändar som kopplar ihop objekten kan det förekomma antingen ett rakt streck eller en gaffel. Ett objekt på vilket det pekar en gaffel indikerar att det kan ha flera poster i förhållande till det andra objektet det relaterar till. Om strecket i andra änden saknar gaffel kan det objektet bara förkomma en gång, vilket innebär att objekten tillsammans har ett många-till-en-förhållande. Se exempel i figur 3.1. Om även den andra sidan av strecket har en gaffel betyder de att objekten relaterar till varandra med ett många- till-många-förhållande. Om strecket som binder ihop objekten inte har en gaffel i någon ände förekommer ett-till-ett-förhållande mellan objekten (Axelsson & Hidefjäll 1993, s. 25).



Figur 3.1 Exempel på notation

4 Kartläggning av verksamhet

En kartläggning har gjorts genom studier av dokumentation och intervjuer av personer med god kunskap om verksamheten. Se bilaga 1 för en översikt över hur olika system samverkar.

4.1 Order

Vid all försäljning hos Kvarnsvedens pappersbruk sker registreringen av kundorder i ordersystemet Fenix. Till största delen sker de direkta kundkontakterna via koncernens säljbolag, som finns i stora delar av världen. Kvarnsvedens roll är då att förse säljbolagen med information om produktionskapacitet, ta emot order, tillverka papper och leverera. Där Stora Enso inte har egna säljbolag finns det agenter och handelshus som säljer företagets produkter. Vid de tillfällena agerar Kvarnsvedens marknadsavdelning säljbolag (Källa C 2005).

4.1.1 Centrala ordersystemet Fenix

Fenix är ett gemensamt order- fakturerings- och planeringssystem för divisionen Publication paper inom Stora Enso och har utvecklats av Stora Enso och TietoEnator tillsammans. Kvarnsvedens pappersbruk har valt att använda den del av systemet som hanterar order och fakturering. Fenix körs på centralt placerade servrar och kan nås av koncernens olika enheter via det gemensamma nätverket. Alla användarapplikationer är installerade på servrar med ett system från företaget Citrix som gör att användarna inte behöver installera någon Fenix-applikation lokalt på sin dator. Varje användare av Fenix har istället en Citrix klientprogramvara och genom att tilldela användarna olika rättigheter så får man tillgång till de applikationer man behöver.

4.1.2 Order från säljbolag

När en kund gör en beställning kontrollerar säljbolaget i Fenix om det finns tillräckligt med produktionskapacitet för att producera ordern i rätt tid. Innan ordern kan registreras i Fenix måste även en kontroll göras så att inte säljbolagets tilldelning av kapacitet överskrids. Kvarnsvedens pappersbruk tilldelar varje säljbolag en viss volym som de kan sälja. Om en order överskrider tilldelningen så måste säljbolaget kontakta Kvarnsveden. Om något annat säljbolag inte har utnyttjat sin tilldelning kan man göra en omfördelning så att ordern kan registreras. När alla kontroller är genomförda registreras ordern i Fenix och kunden får en orderbekräftelse. Vid Kvarnsvedens pappersbruk kontrollerar säljkoordinatorer ordern och om allt är riktigt bekräftas den (Källa C 2005).

4.1.3 Order från agenter och handelshus

Vid order som kommer från agenter och handelshus sköter Kvarnsvedens pappersbruk försäljningen och registreringen. Själva affärerna genomförs av försäljningschefer och säljkoordinatorer kontrollerar produktionskapaciteten och registrerar order i Fenix (Källa C 2005).

4.1.4 Lokala ordersystemet Tips

För att minska sårbarheten och säkerställa att det går att hantera order även om kontakten med Fenix är bruten, finns ett lokalt ordersystem kallat Tips (TietoEnator Integrated Paper Solution). Tips är tillverkat av TietoEnator och uppbyggt med olika moduler för t.ex. order, planering och lagerhantering (TietoEnator 2005). Kvarnsvedens pappersbruk använder Tips för order och grovplanering av produktion. All orderinformation som finns i Fenix och som berör Kvarnsveden speglas till Tips och vice versa. Speglingen görs genom meddelandesamverkan mellan systemen. Tips används inte för att registrera order utan det görs alltid i Fenix, men det är möjligt att göra det i Tips vid behov (Källa D 2005).

4.1.5 Överföring av orderinformation

När en order har registrerats i Fenix överförs orderinformationen från Fenix direkt till Tips med hjälp av meddelanden. Samma information överförs även till en meddelandecentral (MC) som lagrar orderinformationen i väntan på en orderbekräftelse. När en säljkoordinator bekräftar en order skickas en orderbekräftelse från Fenix till Tips och MC. Meddelandecentralen skickar därefter orderinformationen vidare till produktionssystemet och systemet för transportplanering, lager och distribution (Källa D 2005).

4.2 Produktionsplanering

Planeringen av produktionen görs av produktionsplaneringen som är en del av marknadsavdelningen. Produktionsplaneringen gör en grovplanering för att beräkna vilken produktionskapacitet man har den närmaste tiden. Grovplaneringen resulterar i en plan med tomma produktionsblock i Fenix. Säljbolagen fyller sedan de tomma blocken med kundorder. Det går att ta emot kundorder och registrera dessa i Fenix även innan det finns lediga produktionsblock, men man kan inte bekräfta order till kunden. När en order kommer in från säljbolagen gör man en finplanering för att optimera utnyttjandet av maskiner och se till att ordern blir producerad i tid (Källa C och D 2005).

4.2.1 Produktionsplaneringssystemen RPP och PPS

Vid planeringen används flera applikationer. Grovplanering görs i en modul i Tips kallad RPP (Rough Production Planning). Vid en grovplanering delas tillverkningen av varje papperskvalitet på en pappersmaskin in i tomma produktionsblock. Blocken förs över till Fenix och säljbolagen eller marknadsavdelningen kan sedan lägga in order i blocken efter kundernas önskemål. Finplaneringen görs i applikationen PPS. Även en applikation kallad Trim används för att optimera användningen av rullmaskinerna. Trim är utvecklat av WM-data för Kvarnsvedens pappersbruk. Resultatet av finplaneringen blir körplaner och körplansordningar som överförs till produktionssystemet Prins. Produktionsplaneringen går även ut med papperskopior av körplanerna till varje pappersmaskin, rullmaskin och hylskap (Källa C och D 2005).

4.3 Transportplanering

Parallellt med produktionsplaneringen pågår planeringen för transport av den färdigproducerade ordern till kund. Planeringen görs av marknadsavdelningens transportplanerare, lagrets utlastningskontor och lastningsledare. Transportplaneringen gör först en prognos för lastbärarkapacitet med hjälp av orderinformationen. Utifrån prognosen beställer man järnvägsvagnar och dragkapacitet av transportleverantören. Vid transport med båt bokas även plats på båtarna. Efter prognosen gör utlastningskontoret en planering av lastningen kallad resa. En resa beskriver volymen, adressen och leveranstidpunkten för ordern. Utlastningskontoret beställer även järnvägsvagnar för de närmaste dagarna. Med hjälp av resan görs sedan en utplan som är en detaljerad planering för lastningen av ordern. För alla order som ska lastas gör lastningsledaren en kajplan som beskriver hur och när ordern ska lastas (Källa B och C 2005).

4.3.1 Transportplaneringssystemet TPS

Planeringen av utlastning och transport görs i applikationen TPS som är installerad på en Citrix-server så att användarna kan använda den från vilken dator som helst som har en Citrix-klient. TPS är en del av magasinssystemet som förutom transportplanering hanterar lager och distribution. Det finns två varianter av applikationen, TPS och TPSBilexp. TPSBilexp används enbart av utlastningskontoret medan TPS används av alla som planerar transporter. TPS och TPSBilexp arbetar mot samma databas som övriga applikationer i magasinssystemet men kommunicerar även med applikationerna via meddelanden.

4.4 Produktion

Tillverkningen av papper och tillskärning av rullar görs enligt de körplaner och den körplansordning som produktionsplaneringen tagit fram. Om körplansordningen måste ändras kontaktas produktionsplaneringen. Samtidigt måste lastningsledaren i magasinet, som ansvarar för utlastning av färdiga rullar meddelas att ordningen har ändrats. Ändringen av körplansordning ska göras i produktionssystemet Prins och det är endast produktionsplanerarna som får göra det. Det är dock möjligt att producera rullarna i en annan ordning än den som är angiven i körplansordningen utan att ändra i systemet. Det finns ingen inbyggd kontroll som hindrar detta. När rullarna är producerade transporteras de till emballeringen där de förses med omslag och märks med etiketter och streckkoder (Källa D 2005).

4.4.1 Produktionssytemet Prins

Produktionssystemet Prins är ett system som togs i drift 1996 och som utvecklats för Kvarnsvedens pappersbruk av företaget Carelcomp Papersoft. Systemet hanterar all information om de pappersrullar som tillverkas, från att papperet kommer ut från pappersmaskinen tills att rullen är färdigpackad och redo att gå till kund.

När en order har bekräftats av en säljkoordinator får Prins orderinformationen i form av meddelanden från meddelandecentralen. Orderinformationen kopplas sedan till en körplan och en körplansordning. Informationen används för att producera ordern, emballera rullarna med rätt emballage och för att märka dem med etiketter och streckkoder. Under produktionen skickar Prins meddelanden med information till Fenix och Tips för att säljbolagen och marknadsavdelningen ska kunna se hur långt en order har kommit i produktionen.

4.5 Lager och distribution

När pappersrullarna har gått igenom hela processen och fått ett emballage sorteras de så att alla rullar som hör till samma order placeras tillsammans. De transporteras sedan med truck för att lagras eller lastas direkt för vidare distribution. Vid lagring så har varje orderposition ett eget fack i lagret där alla rullar på den positionen förvaras. I truckarna i lagret finns datorer med Citrix-klienter som via ett trådlöst nätverk kommunicerar med lagersystemet magasin. En kamera på trucken läser av en streckkod på rullen och föraren får fram den information som gäller för rullen. När en rulle placeras i lagret eller lastas på ett fordon registrerar föraren detta genom att läsa av streckkoden och bekräfta i systemet.

När en order ska distribueras måste en kajplan skapas som talar om hur och när rullarna ska lastas. Lastningsledaren skapar kajplaner i applikationen LOK (LogistikOptimering Kvarnsveden) och dessa används sedan i truckarna när det är dags att lasta. När alla rullarna på en kajplan är lastade bekräftar truckföraren kajplanen. Magasinssystemet skickar då en fraktsedel (waybill) i form av ett meddelande till Fenix.

4.5.1 Externa meddelanden

Det finns ett antal externa aktörer som behöver ha olika information om färdigproducerade order som ska skickas från Kvarnsvedens pappersbruk. Tullen behöver information för order som ska till kunder utanför Sverige. Likaså behöver transportföretagen och olika hamnar information för att kunna frakta rullarna och lasta dem på båtar. För de order som fraktas med Stora Ensos speciella boxar till Zeebrugge i Belgien skickas distributionsorder (DO) när en resa sparas i systemet och viktspecifikation (WS) när rullarna är lastade. Vissa kunder får också information när ordern lämnar Kvarnsveden. Till de flesta externa aktörer skickas informationen som meddelanden i formatet EDIFACT. Till vissa kunder skickas informationen i form av e-post när en kajplan har bekräftats.

EDIFACT-meddelanden skickas från systemet AMTrix från företaget Frontec. AMTrix är installerat på en server som tar emot filer med information från magasinssystemet via filöverföring (FTP). På samma server finns klientprogramvara för att skicka e-post till kunder.

4.6 Faktura

Fakturor hanteras av order- och faktureringssystemet Fenix. Det finns olika fakturapunkter som anger när det är dags för Fenix att skicka en faktura beroende på vilket distributionssätt som används för ordern. För order till kunder inom Sverige faktureras kunden när Fenix får besked om att rullarna lämnar Kvarnsvedens pappersbruk. För order till kunder utanför Sverige faktureras inte kunderna förrän rullarna är på väg till kunden efter sista omlastningen. Detta för att man ska undvika att fakturera rullar som blivit skadade vid omlastning och som aldrig levereras till kunden (Källa C 2005). När Fenix fakturerar en kund skickas ett meddelande från Fenix till det lokala ekonomisystemet. Ekonomisystemet skickar sedan meddelanden med fakturastatistik till Tips.

4.7 System för stödprocesser

Stödprocesser för huvudprocessen order, tillverkning och leverans använder olika system som kommunicerar med systemen i huvudprocessen genom meddelandesamverkan. Systemet för inköp, förråd och underhåll, IFU får t.ex. meddelanden om produktionsstopp från produktionssystemet Prins. Kvalitets- och processdatasystemet MIS får flera meddelanden från Prins och magasinssystemet för olika funktioner och skickar information om papperets egenskaper till Prins.

4.7.1 IFU, Inköp, Förråd, Underhåll

IFU är ett system för att hantera inköp, lagerhantering av reservdelar och förbrukningsmaterial och hantering av underhåll och reparationer. Systemet bygger på ett affärssystem från företaget IFS och består av olika moduler eller komponenter för inköp, förråd och underhåll. Inköpsdelen av systemet hanteras av ekonomiavdelningen, medan förråd och underhåll hanteras av underhållsavdelningen. En viss del av förrådsfunktionerna används av personalavdelningen för att hantera lagret av kontorsmaterial.

4.7.2 Informationssystem för kvalitets- och processdata, MIS

MIS, Mill Information System är ett system som hanterar kvalitets- och processdata för tillverkningen av papper och massa. Det är ett omfattande system som hämtar in processdata från många olika delar av fabriken. Processdata används bl.a. för att analysera produktionen. Systemet tillhör utvecklingsavdelningen men används även av produktionsavdelningarna. Utvecklingsavdelningen tar prover på det tillverkade papperet och registrerar dessa i MIS. Produktionsavdelningarna kan sedan se resultatet av proverna och upptäcka om produktionen håller sig inom godkända värden. MIS fungerar även som en länk mellan pappersmaskinernas kvalitetskontrollssystem och produktionssystemet och skickar en tambourrapport till Prins när en tambour med papper har producerats vid pappersmaskinen. Tambourrapporten innehåller information om papperets egenskaper och skickas till Prins som ett meddelande via IPC (för beskrivning av IPC se punkt 4.8.1 Centrala ordersystemet Fenix).

4.7.3 Ekonomisystemet

Ekonomisystemets funktion i processen order, tillverkning, leverans är att stödja budgetarbetet, hantera bokslut och rapportering och bevaka betalningen av fakturor. Systemet som används heter Raindance och är utvecklat av företaget WM-Data. Det kommunicerar med andra system genom meddelandesamverkan och får bl.a. faktureringsinformation från det centrala ordersystemet Fenix.

4.8 Driftmiljö

4.8.1 Centrala ordersystemet Fenix

Systemet Fenix körs på servrar som är placerade hos Stora Enso i Finland. Även användarnas applikationer för systemet finns på externa servrar med programvara från företaget Citrix. Användarna behöver endast en Citrix-klient installerad på sin dator för att komma åt applikationerna. Fenix kommunicerar med lokala system vid Kvarnsvedens pappersbruk via IPC och mailbox.

IPC är ett kommunikationssystem för meddelandeöverföring mellan olika system. Det är utvecklat av företaget Carelcomp men tillhör nu TietoEnator och kallas TietoIPC. IPC fungerar bl.a. som ett kösystem som gör att om mottagaren inte är tillgänglig kan meddelanden läggas på kö och kommer att skickas i rätt ordning när mottagaren blir tillgänglig igen.

Att meddelanden kommer i rätt ordning är viktigt. Därför finns en lokal IPC-instans för Fenix-meddelanden vid Kvarnsveden. All kommunikation mellan lokala system och Fenix går via den lokala IPC-instansen som ser till att meddelanden till och från Fenix kommer i rätt ordning. Den lokala instansen av IPC benämns ibland Fenix KP men är inte en Fenix-applikation utan en del av kommunikationssystemet för Fenix-meddelanden.

Mailbox är en programvara som antingen packar ihop information från en databas till ett meddelande eller som packar upp ett meddelande och lägger informationen i en databas. När information ska packas till ett meddelande anpassar mailbox strukturen på meddelandet efter vilket system som ska ta emot det.

4.8.2 Lokala ordersystemet Tips

Tips består av applikationer för order och grovplanering (RPP) samt en databas. Applikationerna är installerade på en Windows-server med programvara från Citrix. Användarna har en Citrix-klient installerad på sin dator för att komma åt applikationerna. Databasen är installerad på en Unix-server med ett databassystem från Oracle. Databasservern är gemensam för flera olika system, men de olika systemen använder sina egna instanser av databasen och kommer inte åt varandras databaser. Kommunikationen med andra system sker genom meddelanden via en instans av kommunikationssystemet IPC. IPC-instansen är installerad på en annan Windows-server än applikationerna.

4.8.3 Produktionsplaneringssystemen RPP och PPS

Planeringsmodulen RPP är en del av systemet Tips och finns installerad på samma Windows-server som Tips. Användarna kommer åt applikationen via en Citrix-klient på sin dator.

4.8.4 Transportplaneringssystemet TPS

Applikationerna för transportplaneringssystemet TPS är installerade på samma Windows-server som Tips. Det enda användarna behöver ha installerat på sina datorer för att använda applikationerna är Citrix-klienter. TPS arbetar mot databasen för lager- och distributionssystemet.

4.8.5 Produktionssystemet Prins

Prins består av klientapplikationer som arbetar mot en server. Klientapplikationerna är installerade på servrar med programvara från företaget Citrix. För att köra Prins-applikationerna används tunna klienter eller datorer med Citrix klientprogramvara installerad. En tunn klient är en enhet som saknar inbyggt lagringsutrymme för operativsystem och program utan som laddar större delen av sin programvara från en server vid uppstart.

4.8.6 Lager- och distributionssystemet

Lager- och distributionssystemet även kallat magasinssystemet består av klientapplikationen LOK som arbetar mot en databas. LOK är installerat på en Windows-server med Citrix. Databasen från Oracle körs på en Unix-server. Varje användare av systemet har en citrix-klient installerad på sin dator och via den kommer man åt de applikationer man behöver. I truckarna som hanterar rullarna i lagret och som lastar det som ska distribueras finns datorer med Windows och citrix-klienter som kommunicerar med systemet via ett trådlöst nätverk. Meddelandesamverkan med andra system sker med hjälp av IPC och mailbox (för beskrivning av IPC och mailbox se punkt 4.8.1 Centrala ordersystemet Fenix).

4.8.7 Informationssystem för kvalitets- och processdata, MIS

Det är många servrar som samverkar i MIS-systemet. För att lagra realtidsdata från produktionen används fyra Windows-servrar med databasen InfoPlus 21 från företaget Aspen Tech. Viss behandling av realtidsdata görs även i dessa servrar.

4.8.8 Ekonomisystemet

Ekonomisystemet körs på en server med UNIX. Varje användare har en klientapplikation installerad lokalt på sin dator som arbetar mot ekonomisystemet.

4.9 Systemägare och ansvar

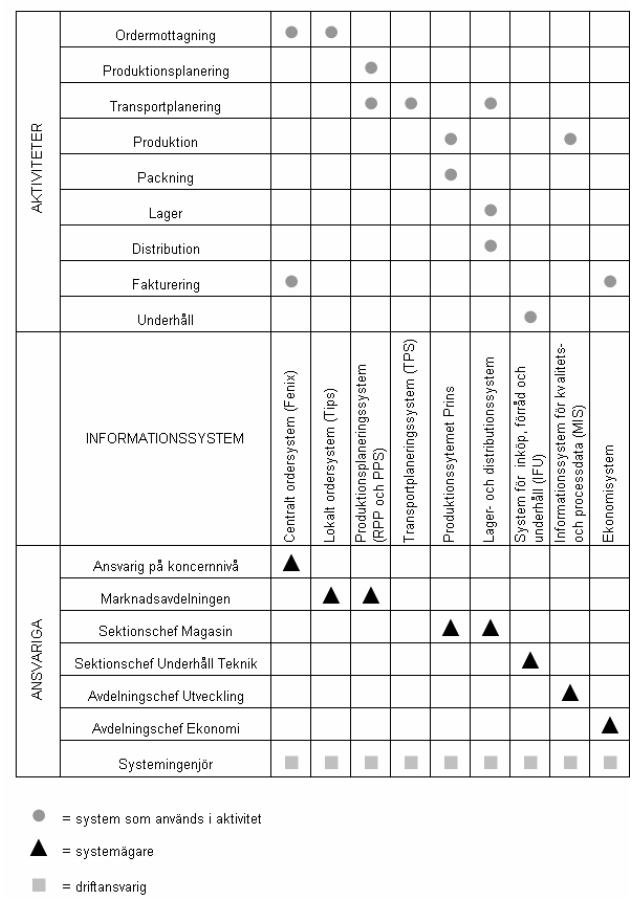
För alla större datasystem vid Kvarnsvedens pappersbruk skapar man förvaltningsråd som följer upp kostnader för systemet, skapar budget och diskuterar systemets status och aktuella åtgärder. Råden består av systemägare, driftansvarig, IT-chef, funktionsansvarig och eventuell leverantörsrepresentant.

Systemägare är oftast avdelningschefen för den avdelning som systemet i huvudsak stödjer. Systemägaren har ansvar för kostnader, förvaltning och utveckling, systemets funktionalitet, att tillräcklig datorkapacitet finns tillgänglig och att systemets tillgänglighet överensstämmer med det man specificerat. Ansvaret för vilka system personalen ska ha tillgång till har avdelningschefen för den avdelning som respektive person tillhör. Avdelningschefen kontaktar den driftansvarige för respektive system vid IT-avdelningen, som i sin tur tilldelar tillräcklig behörighet till systemen. Avdelningschefen har även ansvar för att fånga upp behov av förändringar och tillägg i systemen och föra de vidare till förvaltningsråden.

För varje system finns även en driftansvarig vid IT-avdelningen. Den driftansvarige har förutom ansvar för att tilldela behörighet även ansvar för att regelbundet mäta och redovisa belastningen på systemet. I förvaltningsråden finns en eller flera representanter för användarna av systemet. Dessa är vana användare som fungerar som funktionsansvariga. De ska ha god kunskap om både verksamheten och hur systemen fungerar och ska bevaka användarnas intressen när det gäller förvaltningen av systemen.

4.9.1 System- och ansvarsmatris

Hur ansvarsområdena för olika system är fördelade kan beskrivas i en system- och ansvarsmatris (se figur 4.1).



Figur 4.1 System- och ansvarsmatris

5 Meddelanden

Informationssystemen man har på Kvarnsvedens pappersbruk idag är uppdelade i vad man kallar autonoma delsystem. Det innebär att systemen är helt avskiljda från varandra. För att de ska kunna utbyta information skickas det därför ett stort antal meddelanden mellan dem.

5.1 Meddelandetyper

Gemensamt för alla meddelanden är att de skickas i form av långa teckensträngar. De flesta i form av vanliga ASCII-tecken medan en del också sänds i form av hexadecimal kod som tas emot och omvandlas av mottagande system. Alla meddelanden på Kvarnsveden går dock inte enbart till andra system inom verksamheten. Det skickas viss meddelandetrafik till Stora Ensos centrala ordersystem som finns beläget i Finland, men också information till den svenska tullen och SJ för rullar som ska transporteras utomlands.

Efter att ha studerat befintlig dokumentation och definitioner av meddelanden som skickas mellan systemen och jämfört dessa med loggfiler över meddelandetrafiken, kunde tre stycken meddelandetyper identifieras. De olika typerna visade sig vara tydligt kopplade till ett eller flera specifika system. Meddelandena delades därför in efter vilket system de sändes från, varefter strukturerna för de olika typerna studerades.

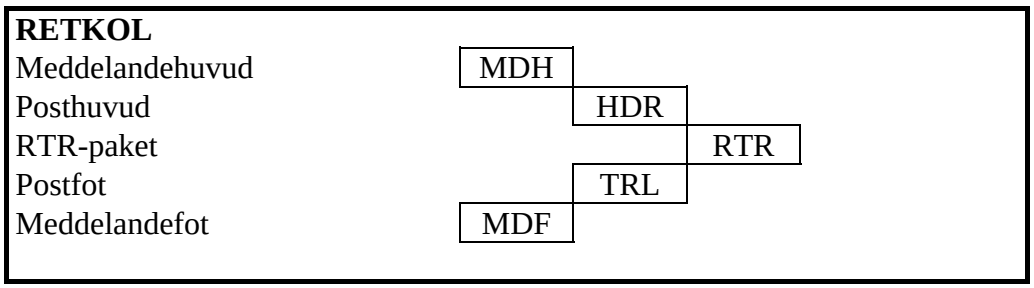
5.1.1 Meddelandetyp 1

Större delen av de meddelanden som påträffats skickas i form av långa strängar av ASCII-tecken. Meddelandenas sammansättning byggde på en struktur där meddelandet delas upp i ett antal nivåer i vilka meddelandeinformationen placeras inom. Denna meddelandetyp gick att koppla till flera av systemen bl.a. Magasin och TIPS. Utifrån intervjuer av personer med kunskap i verksamheten och inom området framkom det också att alla dessa system implementerats ungefär samtidigt och att man valt att använda en gemensam egenutvecklad meddelandestruktur för dessa system.

Meddelandestrukturen innebär att själva meddelandeinnehållet på en första nivå omsluts av ett meddelandehuvud och en meddelandefot (identifieras med MDH och MDF). Dessa fungerar dels som enkla avskiljare för att tala om var meddelandet börjar och slutar men de innehåller också metadata om meddelandet som t.ex. meddelandenamn och meddelandelängd.

Meddelandeinnehållet placeras sedan på nästa nivå inom en eller flera poster, där varje post avskiljs med ett posthuvud och en postfot (identifieras med HDR och TRL). De har enbart funktionen som avskiljare varför de inte innehåller någon ytterligare information. Inom en post delas data i sin tur upp ännu en gång i en eller flera delposter.

Varje delpost identifieras med en tre tecken lång bokstavskombination som beror av vilket meddelande som skickas. Delposterna innehåller sedan själva informationen som man vill skicka. Man har valt att beskriva den som en mängd datafält, där varje fält är av en specificerad längd och typ. Hela meddelandestrukturen i nivåer kan på ett förenklat sätt se ut enligt fig. 5.1och definitionen för innehållet på varje nivå beskrivs i fig. 5.2.



Figur 5.1 Exempel på meddelandestruktur för meddelandetyp 1

Som nämnts tidigare har varje fält i meddelandet på förhand en specificerad längd och typ, men i vissa fall innehåller inte alla fält någon information. Då detta inträffar fylls dessa fält ut med mellanslag, vilket innebär att meddelandet alltid får samma längd och fälten finns alltid belägna på samma position i meddelandet. Ett exempel på detta syns i fig. 5.3 som är ett utdrag ur en loggfil för meddelandet RETKOL.

Field name	Type	Position	Length	Description
MEDDBET	Char	1	3	Meddelande huvud, MDH
MEDDTYP	Char	4	6	Meddelande typ, RETKOL
POSTANTAL	Num	10	3	Antal poster i meddelandet, 1
POSTLANGD	Num	13	5	Post längd (HDR-TRL)
SYS	Char	18	2	Retur kod (acknowledge) 00,
PROGRAMNAMN	Char	20	8	Tömningsprogramnamn
MEDDNR	Num	28	5	Meddelande nr, 1
POSTHUVUD	Char	33	3	Post huvud, HDR
DPID1	Char	36	3	Delpost id (retur kolli), RTR
KOLLINR	Char	39	16	Kolli nr (rull nr i första änden)
RETUR_DATUM	Num	55	6	Rullens returdatum
RETUR_TID	Num	61	6	Rullens returtid
POSTFOT	Char	67	3	Post fot, TRL
MEDDBETF	Char	70	3	Meddelande fot, MDF
MEDDTYPF	Char	73	6	Meddelande typ, RETKOL
POSTANTALF	Num	79	3	Antal poster i meddelandet, 1
POSTLANGDF	Num	82	5	Post längd
SYSF (används ?)	Char	87	2	Retur kod (acknowledge), 00,
PROGRAMNAMNF	Char	89	8	Tömningsprogramnamn
MEDDNRF	Num	97	5	Meddelande nr, 1

Figur 5.2 Meddelandedefinition för meddelandetyp 1

```
MDHRETKOL00100056        98727HDRRTR12832051       040810
120113000000000000000000TRLMDFRETKOL00100056       98727
```

Figur 5.3 Utdrag ur loggfil för meddelandetyp 1

5.1.2 Meddelandetyp 2

Den andra typen av meddelanden som hittats är kopplad till Kvarnsvedens pappersbruks order- och faktureringssystem, Fenix. Systemet är placerat i Finland och är gemensamt för divisionen Publication paper. Alla ordrar läggs i Fenix centralt men för att kommunicera bl.a. orderinformation och orderbekräftelser till Kvarnsveden sänds meddelanden i form av hexadecimal kod till lokala system. Fenix utvecklades senare än övriga system som man använder på Kvarnsveden och man har delvis därför valt en annorlunda struktur för dessa meddelanden än för meddelandetyp 1.

Strukturmässigt har Fenix-meddelanden stora likheter med EDIFACT-meddelanden (se punkt 3.6.1 EDIFACT). Alla meddelanden är uppbyggda av ett antal segment innehållande datafält eller dataelement som det heter i EDIFACT. Varje segment i meddelandet inleds med en tre tecken lång segmentstagg, vilken fungerar som en identifierare. I EDIFACT finns det ett på förhand definierat antal segmentnamn som går att använda, men i Fenix har man valt att använda egendefinierade namn på segmenten. Dessutom har man i alla meddelanden tre segment som alltid finns med. Ett meddelande inleds alltid med ett adressegment (ADR) med information om sändare och mottagare. Därefter följer ett meddelandehuvud (MHD) innehållande metadata om meddelandet t.ex. meddelandenamn. Sist i meddelandet finns alltid ett slutelement (antingen EFP eller ECM) som avgör om det är slutet på ett löpande meddelande (EFP) eller om hela meddelandet i sig avslutas (ECM). Se fig. 5.4.

Datafälten som segmenten består av innehåller de data man vill sända i meddelandet. Ett datafält kan innehålla tre typer av datavärden: numeriska, alfabetiska och alfanumeriska värden. Förutom detta följer Fenix-meddelandena i stora drag syntaxreglerna för EDIFACT. Alla datafält och segment skiljs från varandra med speciella tecken. I Fenix har man valt att ASCII-tecken nummer tre ska representera slut på datafält och ASCII-tecken nummer fyra representera slut på segment.

Eftersom alla fält delas med ett tecken tillåter det att de kan vara av variabel längd. Man kan därför inte tala om positioner i den här typen meddelande. Även om datafält i ett meddelande inte innehåller något värde kommer ändå ett sluttecken för dessa att skrivas ut i meddelandet, vilket underlättar utläsandet av data (se fig. 5.5). Dessutom kan fälten vara obligatoriska (Mandatory), villkorliga (Conditional) eller frivilliga (Optional). Det innebär att vissa fält alltid finns med i ett meddelande och endast en gång, medan andra endast finns med under vissa förutsättningar. Samma sak gäller för segmenten. Hela segment kan vara obligatoriska, villkorliga eller frivilliga, men till skillnad från datafälten kan ett segment i vissa fall förekomma mer än en gång.

FOCONF

Segment	Field name	Type	Length	M/C/O
ADR:	SEGID	an	3	M
	SEGLENGTH	n	5	M
	MSGLENGTH	n	8	M
	TIMESTAMP	an	14	M
	SEQNO	n	3	M
	RECEIVER	an	10	M
	SENDER	an	10	M
	RECEIVERQ	an..	≤ 40	C
	SENDERQ	an..	≤ 40	C

Segment	Field name	Type	Length	M/C/O
MHD:	SEGID	an	3	M
	SEGLENGTH	N	5	M
	MSGID	an	10	M
	MSGVERS	n	2	M
	MSGFUNC	an	1	M
	MSGQUIT	an	1	C
	MSGKEY	an..	≤ 40	C

Segment	Field name	Type	Length	M/C/O
MOC:	SEGID	an	3	M
	SEGLENGTH	n	5	M
	MSGFUNC	an	1	M
	MILLORDER	an..	≤ 12	M
	VERSIONNO	n	2	M
	CONFDATE	n	8	M
	CONFIRMER	an..	≤ 16	M
	AMENDNO	n..	≤ 2	C
	AMENDDATE	n	8	C
	CANCELDATE	n	8	C
	CANCELCODE	an..	≤ 8	C
	UPDATOR	an..	≤ 16	C

Segment	Field name	Type	Length	M/C/O
ECM/EFP:	SEGID	an	3	M
	SEGLENGTH	n	5	M
	SEGCOUNT	n	5	M

Figur 5.4 Meddelandedefinition för meddeladetyp 2

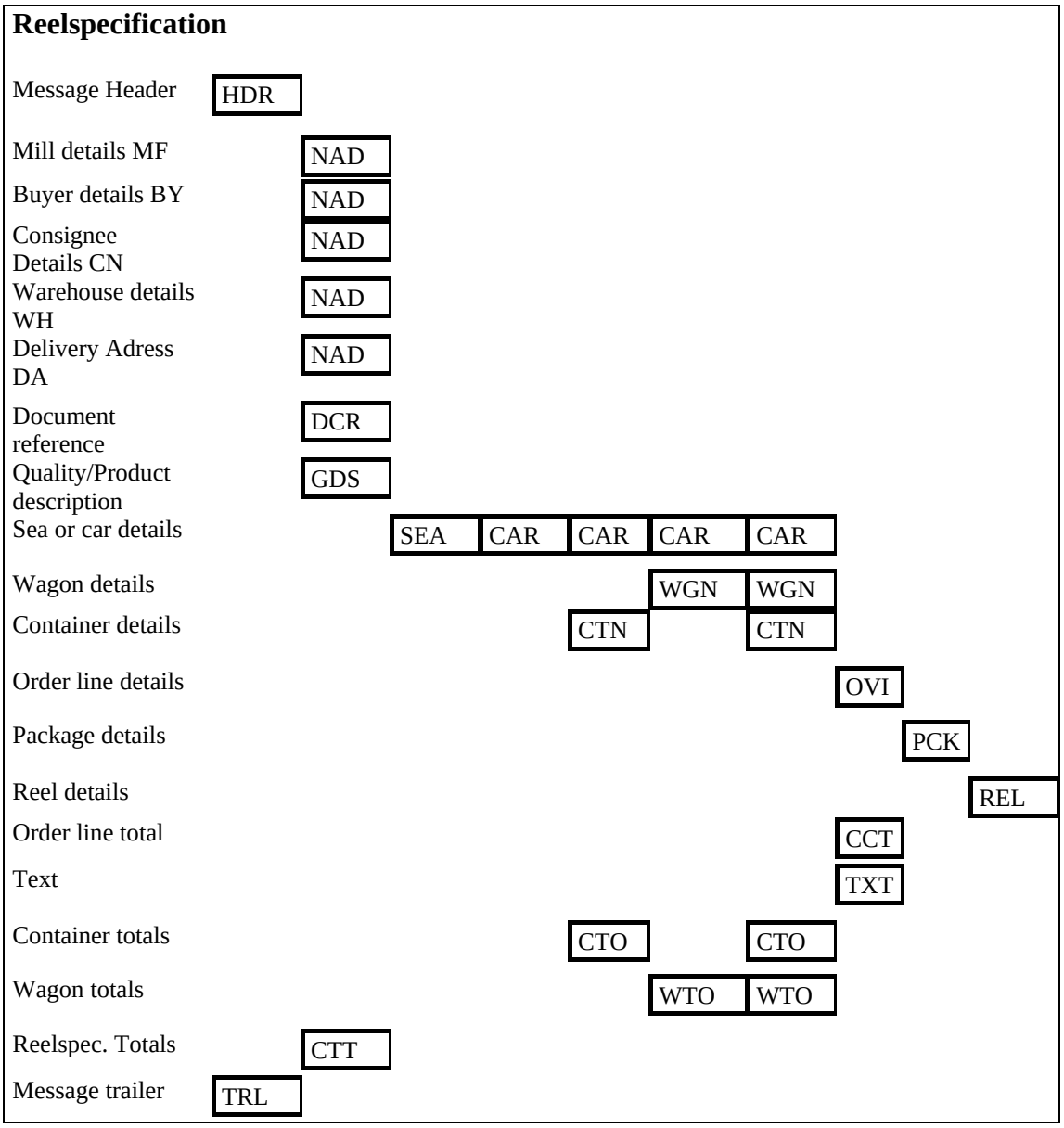
```
ADR 00078 00000188 FX532011045228 001 KVARNSVEDE FX
    Fenix Production MHD 00029 FOCONF
  01 R Q MOC 00065 R KNGB-
508467 03 20051005 blomqch 2 20051116 blomqch ECM 00
016 00004
```

Figur 5.5 Utdrag ur loggfil för meddelandetyp 2

5.1.3 EDIFACT

Som nämnts tidigare sänder man inte enbart information inom verksamheten på Kvarnsvedens pappersbruk och externt till Fenix i Finland. Det har också påträffats några meddelanden som skickas till den svenska tullen och SJ. Dessa meddelanden innehåller huvudsakligen transportinformation och specifikationer över lasten. Innan meddelandena skickas iväg av systemet Amtrix hämtas nödvändig information från lager- och distributionssystemet Magasin. Detta sker via filöverföring med FTP. Innehållet sätts sedan ihop och sänds iväg i form av EDIFACT-meddelanden.

Meddelandestrukturen finns fastslagen på förhand och följer vedertagen EDIFACT-syntax (Se punkt 3.6.1) med segment och dataelement. Strukturen för ett sådant meddelande kan se ut enligt fig. 5.6 och liknar innehållsmässigt Fenix-meddelanden. Till skillnad från Fenix-meddelanden använder man dock inte samma typ av avskiljare.



Figur 5.6 Exempel på meddelandestruktur för EDIFACT

6 Systemarkitektur

Utifrån den information som inhämtats genom studier av dokumentation och genom intervjuer, analyserades och identifierades systemarkitekturen. För att identifiera arkitekturen användes åtta idealtypskriterier för jämförelse mellan VBS- och IRM-strategin (se punkt 2.4).

6.1 Arkitekturprincip

På Kvarnsvedens pappersbruk är systemarkitekturen uppbyggd av autonoma delsystem. Det innebär att det förekommer en rad olika informationssystem som är helt avgränsade och oberoende av varandra. Systemen samverkar dock genom att information i form av meddelanden sänds mellan dem som gör att behovet av information tillgodoses i olika delar av OTL-processen. Till varje system finns en eller flera egna databaser kopplade. Det är enbart systemet som äger databasen som har rättigheter att läsa data ur den. Det är lätt att se stora likheter i arkitekturen med VBS (se punkt 3.4), men det förekommer även omständigheter som skiljer sig från den typen av arkitektur. Inom VBS förordar man att ett system enbart ska vara kopplat till en avgränsad del av verksamheten (Axelsson och Goldkuhl, 1998, s. 53). De flesta systemen på Kvarnsveden är också kopplade till enbart en funktion eller avdelning inom processen, men några system sträcker sig över flera avdelningar och delar av OTL-processen.

Systemet för inköp, förråd och underhåll har underhållsavdelningen som systemägare, men systemet är en viktig del även för inköpssektionen som hör till ekonomiavdelningen. Delar av transportplaneringen tillhör organisatoriskt marknadsavdelningen, men applikationerna som man arbetar med är kopplade till databasen för lager och distribution. Lager och distribution tillhör produktionsavdelningen. Funktionsmässigt hör all transportplanering till samma system och använder samma databas. När det gäller ordersystemet Fenix som marknadsavdelningen är beroende av tillhör systemet inte ens Kvarnsveden utan är ett centralt system med en systemägare på koncernnivå.

6.2 Syn på information

Informationen har en viktig roll i systemarkitekturen. Verksamheten bygger på att information förmedlas och tas emot av systemen på ett riktigt sätt för att pappersproduktionen ska kunna fortgå. Även om informationen har en så pass central roll, anses den inte vara en gemensam resurs som alla inom verksamheten har rätt att ta del av. Det är varje enskilt system som har ansvar för sina data. Behövs delar av informationen i andra system eller delar av OTL-processen skapas ett meddelande som förmedlar data mellan de berörda systemen. Då ett system tar emot information från ett annat system är det dess ansvar att ta tillvara informationen och lagrar det i sin egen databas.

6.3 Informationssamverkan

Som nämnts förmedlar systemen information mellan sig. Detta görs genom att det sänds en form av datoriserade meddelanden mellan systemen. Detta kallas meddelandesamverkan (Axelsson och Goldkuhl 1998, s. 54). Meddelandena ser olika ut beroende på vilket system som är avsändare och även ibland beroende på mottagare. Men principen för alla typer av meddelanden är att de sänds i form av långa teckensträngar. För utförligare beskrivning av meddelandestrukturen se punkt 5 Meddelanden.

6.4 Ansvar för data, informationssystem och informationsarkitektur

Till alla större system på Kvarnsvedens pappersbruk har man definierat vem som är systemägare. Dessa personer är i huvudsak avdelningschefer för den avdelning som systemet stödjer. Systemägaren har i huvudsak ansvar för kostnader, förvaltning och utveckling av systemet. Denna person har också ansvaret för vem i verksamheten som ska ha tillgång till systemet och dess data. Till de större systemen finns också förvaltningsråd. Det består av användarrepresentanter, systemägare och representanter från IT-avdelningen. Råden fungerar som ett forum där man diskuterar förvaltningen av systemen och ger möjlighet att ta till vara både användarnas och de driftansvarigas intressen. Ansvaret för driften av varje system ligger på en ansvarig person vid IT-avdelningen. Den driftansvarige tilldelar rättigheter till systemet och övervakar även belastningen. Själva ansvaret för hela systemarkitekturen finns inte definierat, men de närmaste ansvariga som går att finna är IT-avdelningen.

6.5 Förändring kontra stabilitet

Arkitekturen av system är byggd på att systemen är avskiljda och autonoma. Inga databaskopplingar mellan olika system får förekomma. Genom att information utbyts med meddelanden möjliggör det att man kan lyfta ut ett system och ersätta det med ett annat. Så länge övriga system får rätt information kommer de att fortsätta att fungera. Ett exempel på detta är bytet av order- och faktureringssystemet som skedde under 2003. Före bytet hade man ett lokalt ordersystem vid namn 4S. Man beslutade att byta ut systemet och istället använda ett gemensamt order- och faktureringssystem för koncernen, nämligen Fenix. Fenix är beläget i Finland och vid orderläggningar skickas information från Fenix till Kvarnsveden där meddelanden omvandlas och sänds ut till de övriga systemen inom verksamheten. Vid systembytet togs 4S bort ur arkitekturen och ersattes av Fenix och de övriga systemen kunde fortsätta köra utan större problem.

6.6 Tillgänglighet och spridning av data i verksamheten

Som tidigare nämnts är systemen autonoma och deras data avgränsade från andra systems data. För att data ska bli tillgängligt krävs att meddelanden skickas mellan systemen. Inom VBS-strategin upprättas sambandskontrakt mellan de verksamhetsfunktioner som ska utbyta meddelanden. I dessa kontrakt beskrivs hur meddelandena ska se ut. Kontrakten undertecknas av ansvariga för respektive funktion och innebär att dessa funktioner är skyldiga att tillhandahålla aktuell information (Axelsson och Goldkuhl 1998, s. 49). I dagsläget finns det inte några sådana kontrakt definierade på Kvarnsvedens pappersbruk. Beslut om vilken information och vilka meddelanden som måste skickas tas i många fall informellt av IT-avdelningen när behov uppstår. Oftast har behoven framkommit i samband med förvaltningsrådsmöten.

6.7 Anskaffning av data

Principen man använder på Kvarnsvedens pappersbruk är att data bara ska anskaffas en gång av det system som ansvarar för det. Därefter sprids det vidare genom meddelandesamverkan. Det innebär att data dubbellagras och kan förekomma på flera ställen i verksamheten. Dubbellagring kan innebära inkonsistent data om det uppstår fel i meddelandetrafiken.

6.8 Datastrukturering och datalagring

Datastrukturen på Kvarnsveden anpassas för varje system för sig. Mycket är beroende av vad som beslutades i samband med implementeringen av systemen. Ansvaret för strukturerna ligger idag mycket hos ansvariga på IT-avdelningen.

När det gäller lagring av data förekommer det ingen gemensamt lagrad datainformation för systemen. Alla system lagrar sitt eget data. Flera system använder dock samma databasserver för att minska licenskostnaderna, men varje system har en egen databasinstans (oberoende parallellt löpande versioner av samma databasprogramvara) på den servern. På det sättet skiljer man databaserna åt logiskt även om de fysiskt lagras på samma plats. Dock blir alla databaser beroende av samma hårdvara. Skulle servern av någon anledning vara ur funktion drabbas flera system.

7 Datamodeller

De tre typer av meddelanden som påträffades vid datainsamling skiljde sig mycket åt i sin struktur. Det innebär att det är svårt att skapa en gemensam normaliserad datamodell enligt tredje normalformen där alla tre typer ryms. Istället har två olika datamodeller skapats.

7.1 Datamodell för meddelandetyp 1

Den första datamodellen är anpassad för meddelandetyp 1 som beskrivs i punkt 5.1.1. Modellens utseende presenteras i bilaga 2. Vid datamodelleringen togs hänsyn till meddelandetypens olika nivåer av omslutande avskiljare, metadata och meddelandedata. Modellen bygger därför på att alla meddelanden har en arkitektur byggd på dessa nivåer som till viss del är unika för meddelandet. Omslutande data så som meddelandehuvud finns alltid med och har alltid samma struktur, men delposter innehållande meddelandedata ser olika ut beroende på meddelandet.

Utifrån det förda resonemanget om meddelandetypens struktur skapades en tabell, MEDDELANDE. Tabellen har en central roll i datamodellen för att beskriva meddelandearkitekturen som binder samman meddelandets olika nivåer. Till tabellen MEDDELANDE binds en annan tabell vid namn ORDNING som beskriver förväntad ordning av de nivåer arkitekturen består av. Speciellt för tabellen är att nivåerna identifieras vid namn och inte med en unik nyckel.
Innehållet i meddelandenas nivåer och ordningen för dessa beskrivs i sin tur i ett antal enskilda tabeller som är framtagna för att uppfylla tredje normalformen. Ett exempel är meddelandehuvud i ett meddelande som delas upp och beskrivs i tre tabeller MDH, MDH_INNEHALL och MDH_FALT. Syftet med uppdelningen är att beskriva utseendet för ett meddelandehuvud, ordningen av innehållet samt att undvika många-till-många-förhållanden. Principen är densamma även för MDF, POSTHUVUD och POSTFOT. Posthuvud och Postfot innehåller normalt bara ett fält av data och skulle enkelt kunna beskrivas med en tabell, men delas också in i tre tabeller för att möjliggöra eventuell utbyggnad av meddelandet i framtiden

Datainnehållet i meddelandet delas även det in i flera tabeller. Nämligen MEDDELANDEDATA, DELPOST_FALT, DATA_INNEHALL och DATAFALT. Tabellerna MEDDELANDEDATA och DATA_INNEHALL är till för att undvika många-till-många-förhållanden då ett meddelande kan innehålla flera delposter vilka kan förekomma i flera meddelanden. På samma sätt kan en delpost innehålla flera datafält som kan förekomma i flera delposter.

Utifrån datamodellen kan man generera definitionen för varje enskild del i ett meddelande av meddelandetyp 1. Datamodellens uppbyggnad blir på det här sättet starkt bunden vid den första meddelandetypens struktur av nivåer vilket gör att den är svår att bygga ut eller att förändra.

7.2 Datamodell för meddelandetyp 2 och EDIFACT

Som nämnts tidigare har meddelandetyp 2 och EDIFACT-meddelanden stora likheter strukturmässigt. De båda meddelandetyperna ser dock väsentligt annorlunda ut i jämförelse med meddelandetyp 1 (se punkt 5.1.1, 5.1.2 och 5.1.3). Därför var det nödvändigt att ta fram en egen datamodell för dessa typer av meddelanden. Modellen finns beskriven i bilaga 3. Strukturen bygger på EDIFACT-uppbyggnaden. Data delas in i ett antal segment som innehåller fält med meddelandedata. Ett segment identifieras med en segmentstagg i form av en tre tecken lång identifierare. Därefter följer direkt meddelandedata tillhörande segmentet.

I datamodellen skapades tabeller som beskiver strukturen som förekommer i ett EDIFACT-meddelande eller ett meddelande av meddelandetyp 2. Principen var att alla tabeller ska vara anpassade enligt tredje normalformen. Meddelandena bröts först ned i dess viktigaste beståndsdelar, segmenten och datafälten (SEGMENT och FALT). Dessa tabeller definierar längd, typ och namn av varje fält och segment. Den ordning i vilken segmenten och fälten förekommer finns definierat i tabellerna MEDDELANDERAD och SEGMENTRAD. Dessa tabeller är också framtagna med syftet att undvika många-till-många-förhållanden i databasen. Det förekommer också en tabell, AVGRANSARE som avgör vilket tecken som skiljer segment och datafält åt. Med datamodellen är det möjligt att ta fram alla datafält och presentera dem i den ordning de är placerade i meddelandets uppbyggnad. Modellen är lätt att hantera och bygga ut med ett oändligt antal segment och datafält som kan placeras i önskad ordning.

8 Analys

Analys av systemarkitekturen och de problem som eventuellt går att koppla till problematiken för denna studie. Även problem som uppkommit vid framtagande av datamodeller analyseras.

8.1 Analys av systemarkitekturen

Utifrån de uppgifter som framkommit om systemarkitekturens utseende på Kvarnsvedens pappersbruk (se punkt 6) har det varit möjligt att se stora likheter med VBS-strategin. Man talar både på Kvarnsveden och i VBS strategin om autonoma system, som är avgränsade och självständiga. Systemen samverkar med varandra genom att sända information i form av meddelanden mellan sig (Se punkt 3.4 och 6.1). Det förekommer alltså ingen registersamverkan eller gemensam data mellan systemen, som förordas inom IRM-strategin (se punkt 3.5). Arkitekturen är byggd för att den ska kunna vara förändringsbar över tiden. Systemen ska vara avskiljda från varandra och om ett system byts ut ska de andra fungera så länge det nya levererar samma information.

Det går också att se likheter med VBS-strategin på andra punkter. Synen på information är t.ex. densamma. Varje system äger och ansvarar för sitt eget data. Behövs information i något annat system förmedlas det genom att meddelanden sänds mellan systemen. Systemen har med andra ord inte rätten att läsa ur varandras databaser. Ansvaret för systemen är också något som är viktigt inom VBS. På Kvarnsveden finns ansvaret för varje system definierat i form av en systemägare i verksamhetsfunktionen som systemet stödjer. Systemägaren hanterar frågor om kostnader, förvaltning och utveckling av systemet. Dessutom finns förvaltningsråd för de större systemen där viktiga frågor diskuteras av systemägare, ansvariga på IT-avdelningen och användare i verksamheten.

Även om systemarkitekturen har stora likheter med VBS-strategin skiljer de sig dock åt på några viktiga punkter. I VBS-strategin ska systemen endast stödja en funktion i verksamheten. De uppgifter som har kommit fram om systemarkitekturen på Kvarnsveden visar att de flesta system endast stödjer en verksamhetsfunktion, men att det finns några undantag. Ett exempel är systemet för inköp, förråd och underhåll (IFU), som sträcker sig över två verksamhetsfunktioner inom OTL-processen. Systemets inköpsdel hanteras av ekonomiavdelningen medan förråds- och underhållsdelen sköts av underhållsavdelningen (se punkt 4.7.1).

Ansvaret för vilka meddelanden som skickas och tas emot av olika system i en arkitektur konstruerad efter VBS-strategin, ska finnas dokumenterat i vad som kallas sambandskontrakt (se punkt 3.4). Idag använder man sig på Kvarnsveden inte av några skriftliga kontrakt över vad som sänds iväg och tas emot i olika system. Meddelanden upprättas när det finns behov.

8.2 Problem med systemarkitekturen kopplat till felsökning av meddelandesamverkan

De problem som vi identifierat och som även varit möjliga att koppla till problematiken med felsökningsarbete vid meddelandesamverkan går att dela upp i fyra problemområden: Problem med arkitekturen, felsökning av meddelandetrafik, problem med ansvar och problem med dokumentation.

8.2.1 Problem med arkitekturen

Man har på Kvarnsveden valt en systemarkitektur som man vill ska vara lätt att förändra. Det innebär att systemen enligt VBS-strategin måste vara autonoma och avskiljda från varandra. Den formen av arkitektur fordrar att meddelandesamverkan mellan systemen förekommer för att information ska kunna utbytas (Axelsson och Goldkuhl 1998, s. 47). Eftersom OTL-processen är beroende av att systemens meddelandesamverkan fungerar är det nödvändigt med övervakning av meddelandetrafiken. Loggfiler är ett bra sätt att utföra den övervakningen på då all information som skickats sparas under en tid efteråt.

Problem med loggfiler uppstår när det är människor som måste tyda innehållet i dem. Informationen i filerna består av förkortningar, fältavgränsare och alfanumeriskt data, som är anpassat för att läsas av ett system. För att en människa ska kunna förstå informationen behövs en tolk eller översättare. En sådan översättare kan vara en meddelandedefinition. Men om meddelandedefinitioner används av en människa för att översätta loggfiler innebär det ett mycket tidsödande arbete i form av räknande av tecken och jämförande mot definitioner. Det kan liknas med att översätta en engelsk text till svenska genom att slå upp varje enskilt ord i ett lexikon. Det går bra endast om texten är mycket kort.

8.2.2 Felsökning av meddelandetrafik

Då man upptäcker fel vid skickandet av meddelanden mellan två system genomför man vanligen en felsökning för att lokalisera problemet. En viktig del i felsökningen är att studera loggfiler över meddelandetrafiken för att försöka finna felaktiga värden o.s.v. Problemet som man eventuellt hittar kan ursprungligen bero på ett fel i ett system som det mottagande systemet inte har direkt meddelandesamverkan med. Ett system kan t.ex. sända iväg felaktig information till ett annat som i sin tur sänder det vidare. Problemet kan dock först visa sig i det tredje systemet. Eftersom systemarkitekturen innebär att systemen är avskiljda och har olika driftansvariga blir helheten svår att överblicka (se punkt 6.4). Situationen kräver att man utför en felsökning som sträcker sig över flera system för att finna källan till felet. Det innebär att man kan bli tvungen att kontakta flera driftansvariga för de olika systemen som är inblandade för att spåra hela kedjan för det felaktiga meddelandet. Det innebär ett extraarbete i felsökningsarbetet.

8.2.3 Problem med ansvar

Ansvaret för systemen, data och systemarkitekturen på Kvarnsveden finns i en del fall definierat skiftligt. Systemägaren har t.ex. ansvar för kostnader, förvaltning och utveckling av ett specifikt system. Trots att vissa definitioner för ansvar existerar finns det ändå oklarheter.

Det förekommer t.ex. inte några fastställda kontrakt över vilka meddelanden som ska skickas mellan olika system. Det gör det svårt att få en bra överblick över alla meddelanden och vem som har ansvar för att de skickas. Inte heller vid eventuella ändringar av innehållet i ett befintligt meddelande eller vid framtagandet av nya meddelanden skapas några kontrakt. Beslut om ändringar av innehållet i befintliga meddelanden tas informellt av IT-avdelningen. Besluten baseras oftast på ett behov presenterat vid förvaltningsrådsmöten för olika system.

Eftersom beslut om ändringar som ska genomföras i meddelanden tas internt på IT-avdelningen kan det få konsekvenser i form av att alla iblandade som påverkas av ändringen inte får reda på det. Då ett behov som innebär att något i meddelandetrafiken behöver ändras presenterats vid ett förvaltningsrådsmöte tar driftansvariga för olika system kontakt med varandra. Det är oklart om ändringar meddelas på alla förvaltningsrådsmöten för berörda system.

8.2.4 Problem med dokumentation

Om en ändring som genomförts varken meddelas till alla berörda eller dokumenteras kan det innebära att definitioner för dessa meddelanden blir inaktuella. Att arbeta med inaktuella meddelandedefinitioner innebär ytterligare problem vid felsökning med t.ex. loggfiler. Arbetssättet med beslut om meddelanden internt på IT-avdelningen fungerar så länge driftansvariga arbetar nära varandra, men det skulle innebära problem om något system skulle skötas av en extern aktör. Då blir det viktigare att man har bra och aktuell information som man kan lämna ifrån sig.

Att kontrakt över meddelandetrafiken inte tas fram innebär också att det blir svårt att göra en prioritering av vilka meddelanden som är viktiga för verksamheten. Om det inte finns tydligt dokumenterat vilken information ett system ska leverera, när och varför, kan problem uppstå. Det är inte möjligt att läsa ut den informationen ur själva loggfilen. Om man inte känner till ett systems meddelandetrafik finns risken att man inte lägger ned tillräckligt med resurser för att lösa ett problem innan det blir kritiskt.

8.2.5 Styrkor

Vi har inte enbart identifierat problem med systemarkitekturen utan vi har även kunnat se vissa styrkor. Trots att loggfilerna över meddelandetrafiken medför problem genom att de är svårtydda, fyller de sitt syfte. Det är möjligt att övervaka och spåra fel i meddelandetrafiken mellan de avskiljda system som existerar i systemarkitekturen. Utan loggfiler skulle det vara svårt att spåra varifrån ett fel härstammar.

8.3 Framtagande av datamodeller

Som nämnts tidigare innebär systemarkitekturen på Kvarnsvedens pappersbruk att det sänds ett stort antal meddelanden mellan de olika systemen för att de ska kunna utbyta information. Som bakgrunden till studien anger uppstår ibland problem vid felsökning av meddelandetrafiken. Då något går fel vid skickandet av meddelanden mellan systemen, innebär det bl.a. att man gör en manuell felsökning med jämförelse av loggfiler och meddelandedefinitioner (se punkt 1.2). Det är ett arbete som är ineffektivt och tar lång tid. Man vill därför förändra hanteringen av meddelandedefinitionerna och lagra dessa i en databas. Genom att meddelandedefinitioner lagras i en databas kan man koppla definitionerna till ett felsökningssystem. Om ett system utför översättningen av loggfilerna effektiviseras felsökningen och man undviker feltolkningar. Genom att effektivisera felsökningen minskar risken för produktionsstopp och onödiga kostnader.

För att skapa en databas som är lätt att underhålla och uppdatera, måste en eller flera datamodeller över meddelandedefinitionerna tas fram. Databasen kan sedan skapas utifrån datamodellen.

8.4 Svårigheter vid datamodelleringen

Efter att ha studerat befintlig dokumentation om meddelanden och utfört intervjuer framkom det att det existerar tre typer av meddelanden som sänds mellan system på Kvarnsveden. Vid modellerandet av definitionerna för de meddelanden som skickas har det största problemet varit att meddelandetyperna varit så pass olika i sina strukturer, att det inte varit möjligt att få alla att rymmas i en och samma datamodell. Vi var därför tvungna att konstruera två modeller. En enbart för meddelandetyp 1 och ytterligare en där både meddelandetyp 2 och EDIFACT-meddelanden kunde inrymmas.

Framtagandet av datamodellen för meddelandetyp 2 och EDIFACT-meddelanden orsakade inte några större problem. Det gick att på ett enkelt sätt komma fram till en normaliserad datamodell enligt tredje normalformen. Modellen är lätt att underhålla och bygga ut så länge meddelanden följer grundstrukturen.

Modellerandet av datamodellen för meddelandetyp 1 orsakade istället större problem p.g.a. meddelandetypens komplicerade uppbyggnad. Själva meddelandestrukturen var till en början inte speciellt komplicerad och vi

trodde att en relativt enkel datamodell kunde tas fram. Men allt eftersom modelleringen pågick blev det tydligt att meddelandetypen innehåller flera nivåer av data och att dessa inte kan betraktas som enkla datafält. Inom varje nivå kan bara viss data förekomma och dess innehåll måste därför avdelas och beskrivas var för sig i datamodellen. Data i meddelandehuvudet kan t.ex. inte förekomma i en delpost, trots att fälten på båda ställen i sin uppbyggnad ser lika ut.

Med den givna meddelandestrukturen var det väldigt svårt att åstadkomma en modell som är lätt att underhålla och bygga ut. Det fick till följd att datamodellen blev väldigt komplex. Därför anpassades den bara för meddelandetypen så som den ser ut idag. Det är inte särskilt troligt att man bygger vidare på den här typen av meddelanden vid införandet av nya system. Många tillverkare av system väljer idag att använda XML som standard för meddelanden. Stöd för XML byggs också in direkt i applikationer och databaser vilket gör meddelandetypen till ett bra alternativ (Fredholm 2002, s. 271-273). Ett XML-meddelande liknar i sin struktur meddelandetyp 2 (se punkt 3.6.2). Det gör att ingen ytterligare datamodell behöver tas fram för meddelandedefinitioner för XML-meddelanden.

9 Slutsatser

De viktigaste slutsatserna som har kunnat dras av resultatet från rapporten samt en utvärdering av arbetet.

9.1 Systemarkitekturen

Slutsatsen som kan dras av analysen av systemarkitekturen är att arkitekturen med autonoma delsystem i stora drag liknar VBS-strategin. Varje system är avskiljt och självständigt från de andra, ansvarar för sina data och har inte rättigheter att läsa i andra systems databaser. För att de avgränsade systemen ska kunna kommunicera information sänds meddelanden mellan dem i form av textsträngar av data (se punkt 5.1).

Studien visar att valet av systemarkitektur på Kvarnsveden kan innebära vissa problem kopplat till felsökningsarbetet av meddelandetrafiken. Eftersom man valt en arkitektur som bygger på att information sänds mellan systemen via meddelanden, krävs det att meddelandetrafiken övervakas för att säkerställa att den fungerar korrekt. Den övervakningen sker med loggfiler. Problemet med användandet av loggfiler ligger i att filerna innehåller komplicerade data som är anpassat för att läsas in av system. Människans begränsningar att tolka informationen i loggfilerna innebär ett krångligt och tidskrävande felsökningsarbete (se punkt 8.2.1). Om resultatet av felet är att system som stödjer OTL-processen inte fungerar, kan det innebära ofrivilliga produktionsavbrott som blir väldigt kostsamma. Vi tror därför att en åtgärd för att snabba upp felsökningsarbetet är att låta en applikation utföra felsökningen utifrån meddelandedefinitioner lagrade i en databas.

Användandet av autonoma och avskiljda delsystem som samverkar i en process kan också ge problem vid felsökning. Det blir svårt att få en helhetsbild över processen då ansvaret för olika system ligger på flera personer. Konsekvensen blir att man blir tvungen att kontakta fler driftansvariga för att lokalisera ett fel (se punkt 8.2.2).

Det går också att koppla möjliga problem i felsökningen till ansvaret för systemen. Det finns inte definierat vad ett system har ansvar för att skicka och ta emot. Om ett meddelande ska ändras eller om ett nytt ska införas tas beslut om detta informellt på IT-avdelningen (se punkt 8.2.3). Det kan innebära problem om inte alla inblandade meddelas om detta och om det inte dokumenteras ordentligt. Inaktuell dokumentation försvårar vid felsökning och det skapar en osäkerhet som innebär att översikten och förståelsen inte alltid är klar. Informellt beslutsfattande om meddelanden fungerar relativt bra så länge man arbetar nära varandra, men inte om en extern aktör tar över vissa delar av förvaltningen av systemen. Då är det viktigt att kontrakt upprättas för att det ska vara möjligt att sätta sig in i hur systemen samverkar (se punkt 8.2.4).

Ännu en slutsats från studien av systemarkitekturen är att den inte bara har negativa sidor. Det går även att identifiera styrkor. Att arkitekturen består av avskiljda autonoma system innebär att det går att lokalisera var källan till ett fel är belägen. Eftersom man lagrar meddelandetrafiken i loggfiler är det möjligt att spåra var information ursprungligen skickats ifrån. För att utföra en sådan felsökning krävs det att man har en bra överblick över hela systemarkitekturen och har tillgång till alla loggfiler (se punkt 8.2.5).

9.2 Datamodellering

En viktig slutsats man kan dra av de problem med felsökningsarbete av meddelandesamverkan man har på Kvarnsveden är att arbetet behöver förenklas. Man vill på Kvarnsveden ta fram en databas där man kan lagra definitioner för de meddelande som sänds. Det är en bra lösning för att slippa hantera definitionerna manuellt. Den komplicerade felsökningen med loggfiler och meddelandedefinitioner är både ineffektiv och tidskrävande (se punkt 1.2). Databasen kan sedan kopplas till ett system som sköter felsökningen. Genom att effektivisera felsökningen kan man undvika problem och ökade kostnader till följd av problem i meddelandesamverkan mellan olika system.

Studien av befintlig dokumentation och definitioner av meddelanden som skickas mellan system visade att tre stycken meddelandetyper kunde identifieras. Modelleringen för strukturerna av de olika typerna resulterade i två datamodeller, som kan användas för att skapa databaser med meddelandedefinitioner. Databaserna kan sedan användas av ett system för felsökning av meddelandetrafiken. På grund av att meddelandetyperna var så pass olika i sin uppbyggnad var det inte möjligt att skapa en gemensam modell där alla tre typerna kunde inrymmas (se punkt 8.4). En slutsats som kan dras är att det generellt är svårt att skapa en gemensam datamodell över ett flertal typer av meddelanden inom en systemarkitektur om de skiljer sig alltför mycket åt i sin uppbyggnad.

Den första av de två datamodeller som togs fram var anpassad efter strukturen för meddelanden av meddelandetyp 2 och EDIFACT-meddelanden. Meddelandetyp 2 byggde i stora drag på EDIFACT. Därför kunde båda typerna beskrivas i en och samma datamodell. Konstruerandet av modellen innebar inte några större problem. Det togs fram en enkel och normaliserad datamodell enligt tredje normalformen (se punkt 8.4). Av det kan man dra slutsatsen att meddelandedefinitioner för meddelanden som är uppbyggda med segment och dataelement och där varje segment avskiljs med en segmentstagg, gör det enkelt att skapa en bra datamodell.

Den andra framtagna datamodellen anpassad för meddelandetyp 1 orsakade istället större problem. Efter att ha studerat meddelandetyperna kunde vi dra slutsatsen att meddelandetypen hade en uppbyggnad som såg relativt enkel ut men som var väldigt komplex. Meddelandena är uppbyggda av fält med fasta längder och olika nivåer av data som måste beskrivas, varför resultatet av modelleringen blev en komplicerad datamodell innehållande ett stort antal tabeller. Det var svårt att utforma en datamodell som är normaliserad och lätt att underhålla. Därför anpassades datamodellen enbart efter hur meddelandestrukturen ser ut idag. Vi drog också slutsatsen att det i framtiden inte är troligt att man utvecklar nya system som bygger på den givna typen av meddelanden. Trenden idag är att många tillverkare av system använder sig av XML som standard för meddelanden (se punkt 8.4).

9.3 Utvärdering av metod

Utgångspunkten för studien var att information krävdes för att kunna studera systemarkitekturen och för att åstadkomma en datamodell över strukturen för olika meddelanden, som sänds mellan system på Kvarnsveden. Informationen samlades in genom studier av dokumentation och intervjuer med personer med viktiga kunskaper om verksamheten. Utifrån insamlade data utfördes datamodellering och en analys av systemarkitekturen. Datamodelleringen byggde på normalisering enligt tredje normalformen, men för analysen av systemarkitekturen existerar ingen vedertagen metod att använda sig av. Vi såg att det fanns ett behov av detta. Därför konstruerades en egen metod utifrån kriterier för egenskaper hos en systemarkitektur som presenteras i litteraturen i ämnet (se punkt 2.4).

Totalt sett har metoden fungerat som det var tänkt. Det har varit möjligt att utifrån insamlad information genomföra en datamodellering och göra en analys enligt de givna metoderna. Man kan dock diskutera om den framtagna metoden för analysen av systemarkitekturen var tillräcklig för att kunna bedöma vad som karaktäriserar den. Vi anser att kriterierna som studeras genom metoden var tillräckliga för detta arbete, särskilt då det gick att se stora likheter med en av idealtyperna för systemarkitekturer, som presenteras i teori och referensramen för arbetet (se punkt 3.4 och 3.5). Det var möjligt att på ett enkelt sätt göra jämförelser mellan resultat och teori. Det kan dock vara nödvändigt att vid studier av systemarkitekturer som inte liknar en idealtyp, anpassa metoden. Andra kriterier kan då vara mer lämpliga för att analysera och beskriva systemarkitekturen.

9.4 Utvärdering av arbetet

Arbetet har följande mål och syfte (punkt 1.3 och 1.4)

> **Mål:** Målet med arbetet är att ta fram en datamodell för en databas, som ska innehålla definitioner för strukturen över de meddelanden som skickas mellan OTL-processens system. Databasen ska i framtiden kunna användas av olika system eller program för att automatiskt hämta meddelandedefinitioner. Det kan t.ex. vara ett felsökningssystem för uppkomna fel vid skickandet av meddelanden mellan olika system.
>
> **Syfte:** Syftet med arbetet är att förändra hanteringen av meddelandedefinitioner för att möjliggöra ändrade rutiner av felsökning i loggfiler.
>
> Ett delsyfte är att studera om och hur val av systemarkitekturen kan vara avgörande för problematiken vid felsökningsarbete i meddelandetrafiken mellan system i en verksamhet.

Från början var målet för arbetet att ta fram en datamodell för en databas som skulle innehålla alla definitioner över strukturen för de meddelanden som sänds mellan system inom OTL-processen. Databasen skulle sedan kunna kopplas ihop med ett felsökningssystem. Det visade sig svårt att åstadkomma en gemensam databas där alla typer av meddelanden kan beskrivas, varför två datamodeller skapades. Man kan därför diskutera om arbetet helt och hållet uppfyller målet. Det hade kanske inledningsvis varit mer lämpligt att ställa upp ett mer öppet mål, där det inte förväntas att alla meddelanden ska kunna beskrivas och lagras inom en och samma databas.

Man kan också ifrågasätta om och hur väl de framtagna datamodellerna kommer att fungera tillsammans med ett felsökningssystem, då det inte finns något verkligt befintligt system att testa lösningen mot. Det kan ses som en brist i arbetet.

Vi anser dock att det genom datamodellerna är möjligt skapa databaser som kan hantera meddelandedefinitionerna. Om det går att koppla dessa till ett system för felsökning, kan man i sin tur förändra rutiner vid felsökningsarbete i framtiden. På det sättet har huvudsyftet med arbetet uppfyllts. Vi anser också att delsyftet uppfyllts då det varit möjligt att studera systemarkitekturen och vad den har för betydelse för problemen vid felsökningsarbete.

Källförteckning

Böcker

Axelsson, Karin (1998) *Metodisk systemstrukturering – att skapa samstämmighet mellan informationssystemarkitektur och verksamhet.* Linköping. Linköpings universitet.

Axelsson, Karin och Goldkuhl, Göran (1998) *Strukturering av informationssystem – arkitekturstrategier i teori och praktik*. Lund. Studentlitteratur.

Axelsson, Lars och Hidefjäll, Martin (1993) *Praktisk datamodellering – ta greppet om begreppen.* Lund. Studentlitteratur.

Begg, Carolyn E. och Connolly, Thomas M. (1999) *Database system – a practical approach to design, implementation and management.* Harlow. Addison-Wesley.

– (2000) *Database solutions – a step by step guide to bulding databases.* Harlow. Addison-Wesley.

Hendry, Mike (1993) *Implementing EDI.* Norwood. Artech House Inc.

Fredholm, Peter (2002) *Elektroniska affärer.* Lund. Studentlitteratur.

Starrin, Bengt och Svensson, Per-Gunnar (red). (1996) *Kvalitativa studier i teori och praktik.* Lund. Studentlitteratur.

Elektroniska källor

IBM (2005a), *IBM WebSphere MQ – What is a message?* [www] Hämtat från <http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=/com.ibm.mq.csqzal.doc/csq0334.htm>. Hämtat den 28 december 2005.

IBM (2005b), *IBM WebSphere MQ – Type of message* [www] Hämtat från <http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=/com.ibm.mq.csqzal.doc/msgtype.htm>. Hämtat den 28 december 2005.

TietoEnator (2005), TietoEnator Integrated Paper Solution [www]. Hämtat från <http://www.tietoenator.com/default.asp?path=1;93;16080;164;242;12311>. Hämtat 29 december 2005.

Uppsatser

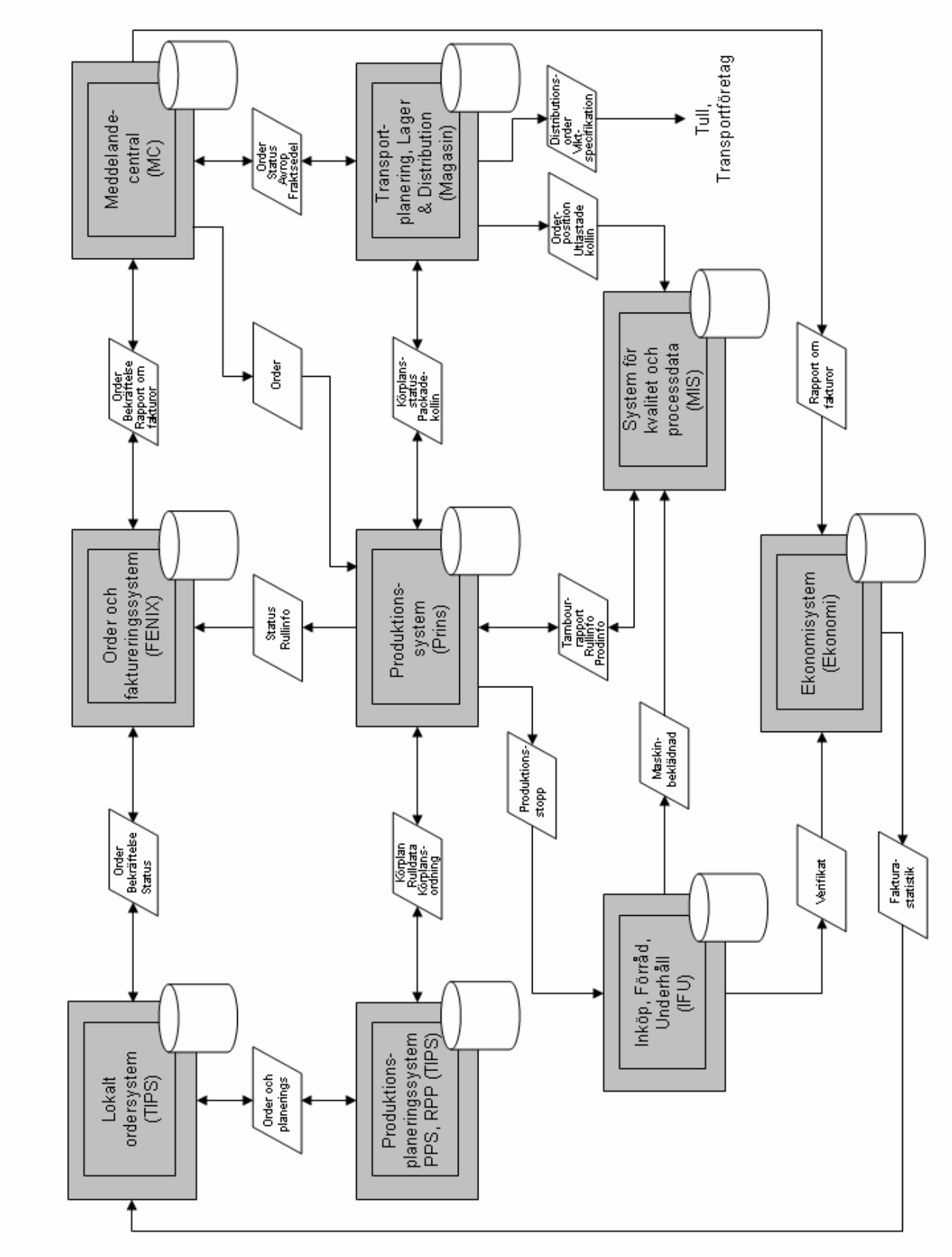
Gustafsson, Andreas och Sörman, Ulrika (2004) *Datakvalitet – Ett måste för en kostnadsmedveten organisation.* Examensarbete. Institution för informatik, Högskolan Dalarna, Borlänge.

Opublicerade källor

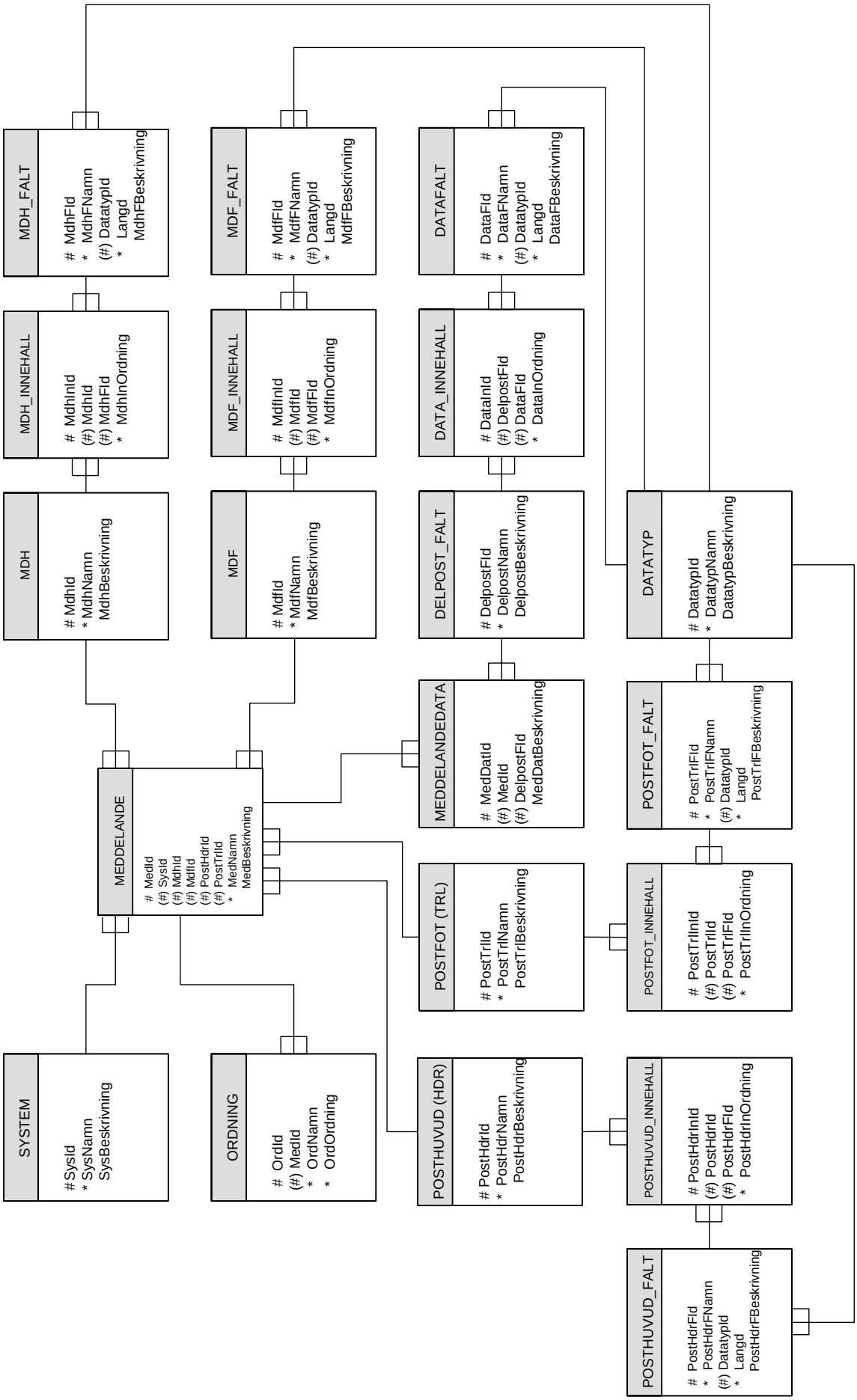
Anonyma källor:

A. 2005-12-02 Intervju. Meddelanden
B. 2005-12-05 Intervju. Lager och distribution
C. 2005-12-06 Intervju. Order och planering
D. 2005-12-07 Intervju. Systemarkitektur och meddelanden

Bilaga 1 – Översikt av systemarkitektur



Bilaga 2 - Datamodell för meddelandetyp 1



Bilaga 3 - Datamodell för meddelandetyp 2