INGENJÖRSHÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

ANALYS AV ANVÄNDARGRÄNSSNITT

Julia Lundström

Daniel Gustavsson

EXAMENSARBETE 2006
DATATEKNIK

INGENJÖRSHÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

ANALYS AV ANVÄNDARGRÄNSSNITT

ANALYSE OF THE USER INTERFACE

Julia Lundström

Daniel Gustavsson

Detta examensarbete är utfört vid Ingenjörshögskolan i Jönköping inom ämnesområdet datateknik. Arbetet är ett led i den treåriga högskoleingenjörsutbildningen. Författarna svarar själva för framförda åsikter, slutsatser och resultat.

Handledare: Lars-Olof Petersson

Omfattning: 10 poäng (C-nivå)

Datum: 2006-06-07

Arkiveringsnummer:

Postadress:	Besöksadress:	Telefon:
Box 1026	Gjuterigatan 5	036-10 10 00 (vx)
551 11 Jönköping		

Abstract

Who decides whether a design is user friendly or not? And what is the trick to maintain that type of design throughout the whole program? These and many more questions were asked when the work with the report were about to begin. The plan is to find out what is relevant for design today and what will be interesting to know about in a present future. Through a fundamentally analysis of the design in the program SEBS (System Economy Business System) together with a number of similar programs, several problem areas has been discovered and motivated by references to different authors and ISO standards(International Organisation for Standardization).

Since this task has an abstract nature it does not provide the authors with good case studies that easily can be presented and discussed. Despite that, a new user design has been created and new guidelines for future design have been drawn. The work gained a deep understanding for the difficulties that comes with software manufacturing and a belief that most people would back of if they knew what was waiting down the line in this type of work. As always when project includes several people problems with communications and the ability to struggle at the same direction occurs. Tools to lighten this work are presented and they are not just effective to keep the group together. They can also help to find new ways to talk about the actual user of the developing program.

Sammanfattning

Vem avgör vad som är bra användbarhet? Och hur lyckas man bibehålla det genom ett helt program? Dessa frågor och många fler ställdes när arbetet med utformningen av den nya designen startades. Meningen var att ta reda på vad som gäller idag och vad som är intressant för framtiden i frågan om användbarhet.

Genom en grundlig analys av designen i programmet SEBS (System Economy Business System) tillsammans med ett flertal andra liknande program har flera problemområden identifierats och motiverats med hjälp av teorier från författare och ISO standarder (International Organisation for Standardization).

Uppgiften har en ganska abstrakt natur och ger inte självmant upphov till gedigna praktikfall som lätt presenteras. Trots det blev ett nytt gränssnitt och nya riktlinjer framtagna och arbetssättet och tankarna bakom finns väl dokumenterade. Resultatet måste analyseras av någon som förstår större delen av innebörden i den teoretiska bakgrunden som är presenterad.

Resultaten som arbetet skapade gav en djup insikt i svårigheten att framställa programvara som är tilltalande och effektiv. Det är en utmaning många skulle backa för att ta om de visste vad den innebar. Arbetet det innebär kämpar precis som alla projekt där flera människor deltar med att få alla att dra åt samma håll. Det finns inget konkret att satsa på utan alla måste lyssna och försöka samlas kring en vision som symboliserar det kommande resultatet. Hjälpmedel för detta presenteras i rapporten, för det finns fler poänger med att låta verktygen få finnas till hands för mer än bara sammanhållningen. De kan även väcka nya sätt att prata om slutanvändare och hur man ska förhålla sig till dem.

Nyckelord

Användarfall

Scenario

Användarcentrerad systemdesign

Människa-datorinteraktion

Användargränssnitt

Användarvänligt

Användbarhet

Design

Innehållsförteckning

Figurförteckning

1 Inledning

Denna rapport behandlar hur vi har valt att utveckla och omforma användargränssnittet på det affärssystem som vår klient har skapat. Vi har fokuserat på att finna en bra metod för att skapa en funktionell design, ta reda på vad som är bra design samt definiera nya riktlinjer för designen på affärssystemet.

Rapporten är skriven som en del i utbildningen till högskoleingenjör inom datateknik. Den sammanfattar ett flertal kurser som utbildningen har omfattat.

1.1 Bakgrund

SEBS (System Economy Business System) är ett affärssystem som är utvecklat av System Technologies AB (SET) i Nässjö. Företaget grundades 1986 och då släpptes deras första program som kallades System Ekonomen. Därifrån har utveckling skett och idag är de framme vid ett komplett affärssystem, SEBS, som till skillnad från System Ekonomen som bara hanterade ekonomi. Idag riktar sig affärssystemet främst till producerande industrier och hanterar allt från tidsstämpling till orderhantering men de har även gjort specialanpassade varianter för optiker och tvätterier. Programmet är utvecklat i en miljö som heter Visual Data Flex och designen i SEBS följer dess grafikstandard. Användargränssnittet har aldrig genomgått någon designrevision vilket skapat ett program som inte är designmässigt konsekvent, varken mot Windows riktlinjer eller mot sig självt. Frågan är om det skapar några konsekvenser för användaren eller om han/hon kan vara tillfreds med affärssystemet ändå?!

1.2 Syfte och mål

Programmet som vi ska analysera är i stort sett oförändrat designmässigt de senaste tio åren. Troligtvis är inte designen enbart föråldrad utan även ineffektiv gentemot dagens nya kunskaper i människa-datorinteraktion. System Technologies anger i sin kravspecifikation att de vill att vi genom en förutsättningslös analys och en kreativ designprocess ska skapa ett användarvänligt användargränssnitt och utforma riktlinjer för en framtida design.

Målet vi satt upp för att kunna utforma designen till ett användarvänligt system innefattar vissa delmål. Dessa mål kan sammanfattas i ett antal frågor som rapporten kommer att besvara.

- Vad är användarvänlighet?
- Varför ska man ha ett användarvänligt gränssnitt?
- Hur utvecklar man ett användarvänligt och användbart system?
- Hur går en designprocess till?

1.3 Avgränsningar

Vid analys av affärssystem så finns inte tid att titta närmare på funktionaliteten hos de olika systemen utan analysen blir rent designmässig. Ur ett användarvänligt perspektiv borde analysen även innefatta enkelheten att navigera i hela systemet. Det borde också göras tester med olika användare för att få en ordentlig utvärdering men tyvärr måste vi begränsa oss här.

Hela tiden flyttas ribban uppåt för vad som är möjligt att genomföra med datorns hjälp. Det innebär också att programtillverkarna hela tiden får nya resurser att jobba med. Dels får de ännu bättre beräkningsmöjligheter och kan hantera större mängder information hela tiden men samtidigt utvecklas grafikresurserna och skapar nya möjligheter för dem att utveckla designen. Vi begränsade oss till att endast titta på dagens möjligheter och standarder eftersom nästa generation av Microsoft Windows fortfarande är lite osäker på vad som kommer att gälla. Windows XP är idag den vanligaste plattform som används av de kunder som SET har och därför sätts den som norm i detta arbete.

Vid utvecklingen av det nya användargränssnittet måste vi begränsa oss till att endast skapa en ny designvy där funktionaliteten av menyer och knappar uteblir. Utformandet av ett komplett gränssnitt skulle innebära alldeles för mycket arbete i avseende på den tidsrymd som detta projekt omfattar.

2 Teoretisk bakgrund

2.1 Iterativ utveckling

Att använda sig av iterativ utveckling innebär att man arbetar i en cyklisk process där man repeterar olika steg i utvecklingen för att kunna lösa problemen bättre. Till sist efter ett antal iterationer så har man nått de mål som man hade satt upp. För varje iteration man utför så finner man ofta nya problem och kan hitta nya sätt att hantera dem på. Iterativ utveckling grundar sig på att ingen kan klara av att uppnå de satta målen efter bara ett försök utan man behöver omarbeta det man har utvecklat ett antal gånger för att kunna uppnå målen.

Jan Gulliksen och Bengt Göransson har i sin bok Användarcentrerad systemdesign satt upp några minimikrav för att man ska få kalla sin utveckling för iterativ. Varje iteration måste innehålla:

- En analys av användarens krav och vad det ska användas till.
- En prototyputformning.
- En utvärdering där man ser på användbarheten av prototypen man har tagit fram. Utvärderingen ska även innehålla vilka förändringar man ska göra i den nästa utformningen.

[4]

2.2 Kognitiv friktion

För en tid sedan lämnade vi industrisamhället för it-samhället och har därmed fastnat i nya typer av problem. När vi enbart byggde med sten, trä och metall kunde vi alltid, oavsett hur avancerad mekaniken än var, se logiken och lita på att saker och ting kommer att följa naturlagarna. En vanlig gitarr som kan tyckas vara ganska svår att hantera kommer alltid att agera på ett visst sätt när man använder den. Oberoende av användare så kommer den aldrig att låta som någonting annat än den gitarr som den är. Det spelar ingen roll hur komplicerade eller enkla saker man spelar.

Däremot när man sitter och tittar på den enklaste hemsida som bara har några få hyperlänkar så är den mer invecklad än gitarren eftersom man inte vet vad som händer när man trycker på en utav länkarna. En länk kan ju hamna på olika ställen varje gång man trycker på den.

Detta är en av anledningarna till att man måste identifiera vilka problem som uppstår hos användaren av datorn.

Att utan anledning lägga energi och tid på utveckling av gränssnitt till ett program är det inget mjukvarutillverkande företag idag som gör. Det kanske snarare är så att man löser gränssnittet i andrahand och först koncentrerar sig på att få till funktionerna så att de fungerar. Sedan skapar man den delen som användaren hela tiden ska titta på och arbeta med. För att förstå var problemet med grafiskt gränssnitt och människans begränsningar att ta in information är, måste vi titta på vad användaren gör och där efter identifiera vad det är som händer när programmet används.

När en användare som sitter vid sin dator stöter på problem och komplikationer med programmet som används uppstår vad man kan kalla kognitiv friktion. Det är alltså ett namn på känslan som uppstår då programmet är besvärligt och inte underlättar det egentliga arbetet. Människor tenderar till att dela upp sig i två läger, ett som anser att redskapet (datorn) är ett nödvändigt ont eftersom den inte underlättar arbetet som önskat och en annan del som upplever en känsla av seger eftersom de klarat av det relativt svåra att mästra ett program. Ingen känsla är egentligen bra eftersom båda grundar sig i att programmet ofta från början är felaktigt designat. Alla som anser att programmet är svårt att lära sig bygger upp en känsla av olust inför att jobba med det. Här skapas då ett hinder för användaren att utvecklas och vidareutveckla sina kunskaper och istället söks kanske andra vägar att lösa arbetsuppgifterna.

Svårigheten med att skapa program är att på ett smidigt sätt kombinera designers och programmerare. I nästan varje steg som programmeraren tar måste designbeslut tas.

Hur funktioner skall anropas och i vilken ordning procedurer anropar varandra. Alla små beslut formar till slut hela programmet och det innebär ju att programmerarens erfarenhet och känsla för vad som är bra och dåligt skapar tillsammans miljön som användaren sedan skall arbeta i.

När ett team skall arbeta tillsammans mot ett gemensamt mål är det viktigt att man kan dela samma vision. Hur ska man då måla upp visionen för varandra så att det står klart att man jobbar mot samma mål?

Det blir lättare om teamen håller en mindre storlek då man lättare kan hålla fokus på den gemensamma visionen. Alla kan hänga med i vad de andra gör och frågor som dyker upp kan enklare besvaras eftersom man kan ana vilka som det berör.

[3]

2.2.1 Persondesign

För att fokusera och skapa förståelse för vad som ska skapas av programmerarna måste man se till att de har samma riktlinjer för programmet. Här inför Alan Cooper i sin bok The inmates are running the asylum något som han kallar Personas, vilket jag skulle vilja översätta till personer. Det är rent fiktiva människor som enbart används i utvecklingssyfte men de spelar en väldigt stor roll i utvecklingsarbetet. Man skapar personerna som ska använda programmet och anpassar programvaran till dessa människors unika behov. Vilka delar som den fiktiva användaren kommer att använda och hur den uppfattar designen och tidigare erfarenheter tas i beaktning vid skapandet av programmet.

Ingrid 53 är inte intresserad av färgglada 3D-diagram som presenterar fakta för henne utan nöjer sig med att informationen skrivs ut i rader med alla värden presenterade. Då skulle en avancerad presentationsrutin bara kosta extra pengar och vara i vägen för Ingrid när hon skall skriva ut.

Vid diskussioner mellan programmerare och deras överordnade som styr projektet saknas ibland relevanta exempel på varför man ska göra på ett visst sätt. När programmeraren kommer till skapandet av till exempel utskriftsrutinen till det projekt som han arbetar i så finns alltid massor med alternativ på sätt som man skulle kunna implementera det på. Här måste då hans chef styra så att rutinen skrivs på ett effektivt sätt för det den ska användas för. Det hjälper inte att man kan skriva ut snygga rapporter om ingen har någon nytta av dessa.

Här kan då personer användas för att styrka den väg som tidigare var bestämd. Chefen kan sonika säga att Ingrid inte är intresserad av att använda en sådan rapport och därför finns ingen mening med att skriva den. Programmeraren får en djupare förståelse för designen och inser säkert att det var ju fakta som han redan visste för han ”känner” ju Ingrid lika bra.

[3]

2.3 Användbarhet

Många gånger när man talar om användarvänliga system så pratar man om användbarheten i ett system. Man kan tycka att om ett system går att använda utan att några större fel uppstår så är det användbart men riktigt så enkelt är det inte. [4]

För att en person ska använda sig av ett program måste programmet uppfylla en viss önskad funktionalitet, men det måste också ha en viss användbarhet. Om en användare inte vill använda programmet till exempel på grund av att det är svårförståligt och svårt att hitta i så tappar användaren motivationen att lära sig programmet och kanske då ger upp och slutar använda det. Ju större användbarhet ett system har desto större blir systemets effektivitet vilket kan bidra till en ökad produktivitet. [1]

Tänk om ett företag har beställt ett program som ska utföra vissa funktioner. När de får programmet så går det att utföra dessa funktioner men användarna finner det svårt och osmidigt att utföra de önskade operationerna. Visst har de fått ett program som klarar av att utföra det som det begärde men frågan man kan diskutera är om det räcker för att kallas klar produkt. Alla tekniska ting som vi använder har någon form av användargränssnitt. Visst kan vi tycka menyerna i vår tv-apparat är väldigt osmidiga. Räcker då dessa menyer eller borde ansvarigt företag utveckla dem mer?

Kontentan blir att programmet eller menyerna inte kan anses särskilt användbara, även om de klarar av att hantera den avsedda uppgiften. [4]

Användbarhet finns definierat i ISO 9241-11, 1998 som:

”den utsträckning till vilken en specificerad användare kan använda en produkt för att uppnå specifika mål, med ändamålsenlighet, effektivitet och tillfredsställelse, i ett givet användningssammanhang”

Ändamålsenlighet definieras som:

”noggrannhet och fullständighet med vilken användarna uppnår givna mål.”

Effektivitet definieras som:

”resursåtgång i förhållande till den noggrannhet och fullständighet med vilken användarna uppnår givna mål.”

Tillfredsställelse definieras som:

”frånvaro av obehag samt positiva attityder vid användningen av en produkt.”

Användningssammanhanget definieras som:

”användare, uppgifter, utrustning (maskinvara, programvara och annan material) samt fysisk och social omgivning i vilken produkten används.”

[4]

ISO definitionen gör att användbarheten i mjukvara blir mätbar. Man kan till exempel mäta hur ofta man gör fel och den tid det tar att återhämta sig efter ett fel. Vid vidareutveckling av program kan man mäta om programmet har blivit mer användbart än det var tidigare och kunna uttrycka det i till exempel en procentsats. Olika system är med denna definition jämförbara på flera olika sätt. Tillfredställelsen som designen på det grafiska gränssnittet ger eller effektiviteten på programmets inbyggda funktioner för betalningsrutiner är två olika sätt som man kan tillämpa definitionen.

Användbarhet har också definierats av flera olika författare. Allwood beskriver användbarhet som tre olika begrepp: användaracceptans, användarkompetens och användarvänlighet. [1]

2.3.1 Användaracceptans

Om användarna till ett system har låg motivation till att till exempel lära sig systemet så ger det en låg användaracceptans. Användaracceptansen påverkas av saker som har med användarnas inställning till systemet att göra. I vissa fall kan användarna se systemet som ett hot och ibland som en tillgång. Det gäller att försöka utforma och anpassa användargränssnittet för att ge så mycket positiv inställning till att använda systemet som möjligt. Det kan man göra genom att till exempel göra det roligare, enklare, mer logiskt och även genom att utforma hjälpfunktioner som finns tillgängliga för användarna vid rätt tillfällen.

[1]

2.3.2 Användarkompetens

Användarkompetens innebär att en användare har tillräckliga kunskaper för att kunna använda systemet på ett effektivt sätt. För att uppnå en hög användarkompetens behöver man utbilda användarna av systemet.

Utbildning kan ske på många olika sätt till exempel genom föreläsningar, instruktionsmanualer eller demonstrationer av programmet. Alla sätt är bra men inte utan stöd för användaren. Forskning visar att det är bra om det finns någon tillgänglig som man kan kontakta när problem uppstår. Instruktionsmanualer är bra men det kan aldrig täcka alla specialfall som användarna hamnar i och man kan inte räkna med att de klarar av att följa dess handledning utan att komma på sidospår. Ett bra recept för att utbilda användare är en instruktionsmanual som beskriver alla enskilda funktioner och som går att leta i efter svar på de frågor som kan dyka upp i kombination med någon att fråga som har alla svaren.

[1]

2.3.3 Användarvänlighet

En grundläggande sak för att ett system ska bli användarvänligt är att det är åtkomligt för användarna. Med åtkomligt menas till exempel att den server som programmet är installerat på inte är ur funktion eller krånglar hos respektive användare. För om en användare inte kan komma åt att använda programmet när den vill så kommer man hitta ett annat sätt att genomföra sin arbetsuppgift.

En annan sak som bidrar till användarvänlighet i att programmet är att det inte får ställa krav som inte passar ihop med användarens sätt att fungera mentalt. Programmet får till exempel inte kräva att användaren ska kunna hålla för mycket information i minnet när användaren ska interagera med programmet på grund av att människans korttidsminne är begränsat. För att kunna avgöra om programmet ställer fel krav behövs det att man vet en hel del om användarna och hur de agerar i olika situationer. Detta ställer krav på utvecklarna eftersom de inte bara ska kunna hantera programutveckling utan även ha en god insikt i det arbete som programmet ska användas till.

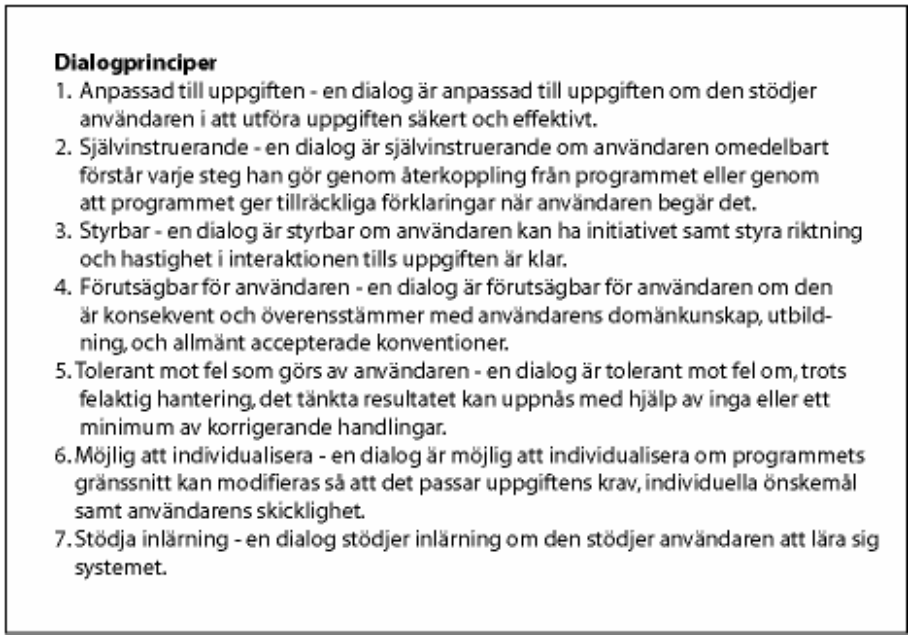
Det är viktigt att man försöker individualisera programmet för att öka möjligheten till att det ska passa till olika slags individer som exempel både nybörjare och experter. Nybörjare uppfattar sällan systemet lika. Det skiljer ofta på hur de ser systemet utifrån helhet och funktioner. Mellan experter är det sällan så stor skillnad, de tenderar till att uppfatta det lika, men istället kan de ha en bristande förståelse för själva arbetsuppgiften som programmet skall utföra medan en nybörjare har helt klart för sig vad programmet skall användas till. De vet bara inte hur. Ju fler användare det passar till desto mer användarvänligt är systemet.

Det sista som gör ett program användarvänligt är hur hjälpresurserna i programmet är utformade. Att de finns tillgängliga på ett enkelt sätt när användaren behöver det och att det är enkelt att hitta det man söker. Inbyggda hjälpfunktioner påverkar användarna olika baserat på deras erfarenhet sedan tidigare. En nybörjare kan prestera sämre tillsammans med en hjälpfunktion än tillsammans med en pappersdokumentation eftersom den ofta kräver en interaktion med datorn som de inte behärskar. Det är viktigt att även nybörjarna får stöd i hjälpfunktionen annars riskerar man att de slutar använda den. De som är lite mer erfarna utnyttjar gärna den inbyggda hjälpfunktionen och finner oftast den smidig. Hjälpfunktionen behöver kombineras med någon form av användarstöd för dem som inte hittar i hjälpen och behöver fråga sådant som kanske anses vara för simpelt för att nämnas av hjälpförfattarna.

[1]

2.3.4 Riktlinjer för användbarhet

Att använda sig av riktlinjer gör att man lättare får en konsekvent design vilket underlättar användarens arbete i systemet. Det finns olika slags riktlinjer framtagna. ISO har till exempel tagit fram dialogprinciper, ISO 9241-10. Se figur 1. [4]



Dialogprinciper

1. Anpassad till uppgiften - en dialog är anpassad till uppgiften om den stödjer användaren i att utföra uppgiften säkert och effektivt.
2. Självinstruerande - en dialog är självinstruerande om användaren omedelbart förstår varje steg han gör genom återkoppling från programmet eller genom att programmet ger tillräckliga förklaringar när användaren begär det.
3. Styrbar - en dialog är styrbar om användaren kan ha initiativet samt styra riktning och hastighet i interaktionen tills uppgiften är klar.
4. Förutsägbar för användaren - en dialog är förutsägbar för användaren om den är konsekvent och överensstämmer med användarens domänkunskap, utbildning, och allmänt accepterade konventioner.
5. Tolerant mot fel som görs av användaren - en dialog är tolerant mot fel om, trots felaktig hantering, det tänkta resultatet kan uppnås med hjälp av inga eller ett minimum av korrigerande handlingar.
6. Möjlig att individualisera - en dialog är möjlig att individualisera om programmets gränssnitt kan modifieras så att det passar uppgiftens krav, individuella önskemål samt användarens skicklighet.
7. Stödja inlärning - en dialog stödjer inlärning om den stödjer användaren att lära sig systemet.

Figur 1. Dialogprinciper enligt ISO 9241-10

Sedan finns det plattformsspecifika riktlinjer som gör att det du designar ska ha ett gemensamt utseende med den valda plattformen. De är ofta väldigt detaljrika och ger kanske mest ett stöd på detaljnivå och inte det stöd som kan behövas i helheten. Vet man inte vad man sysslar med är plattformsspecifika riktlinjer svåra att jobba utifrån eftersom man kan behöva arbeta utanför ramarna och då finns inget att luta sig mot. Designen skall inte vara låst till dessa riktlinjer men följer man dem där så är möjligt och eftersträvar att likna dem där exakt efterföljning inte är möjlig får man en bra enhet på programmet. [4]

Olika författare har tagit fram olika riktlinjer, Ben Shneiderman har tagit fram åtta gyllene regler för design:

- Sträva efter konsekvens – Man måste sträva efter att vara konsekvent hela tiden genom att till exempel göra alla hjälpfönster identiska, se till att menyer ser likadana ut överallt, använda samma typsnitt genom hela systemet, följa en viss layout på alla fönster så att hela systemet känns som en helhet. Detta gör att man känner igen sig i olika fönster i systemet.
- Gör det möjligt för frekventa användare att använda genvägar – Ju längre en användare har använt ett system desto viktigare blir det med genvägar och kortkommandon för att förkorta de steg man måste utföra för en önskad funktion.

- Erbjud informativ feedback – För varje handling en användare utför ska systemet ge feedback, men vid mindre handlingar och för handlingar som utförs ofta ska feedbacken vara måttlig.
- Designa dialoger som främjar avslut – Sammansatta funktioner bör vara organiserade så att de består av en början, mellandel och ett slut. När användaren har behandlat ett antal funktioner och kommit fram till slutet av dem bör programmet leverera en känsla av avslut, fullbordande. Användaren ska kunna släppa all inmatning som skett och inte behöva koncentrera sig mer på parametrarna som varit utan kunna gå vidare till nästa moment och där startar allt om på nytt.
- Erbjud förebygganden av fel och enkel felhantering – Försök att designa systemet så att användarna inte kan göra ett allvarligt fel. Till exempel försök att använda en lista med menyval istället för att användaren själv ska skriva in sitt val i en textruta.
- Tillåt enkla sätt att gå bakåt i programmet – Tillåt så mycket som möjligt att användaren kan ångra sin handling eftersom detta gör att användaren blir mindre spänd när han/hon vet att om man gör fel så går det att göra ogjort.
- Stöd den inre känslan av kontroll – En användare mår bra av att känna en känsla av att de bestämmer över systemet och får det att göra det de vill. Om en användare upplever svårigheter att hitta nödvändig information eller att kunna utföra de önskade handlingarna så får användaren en känsla av ångest och missnöjdhet.
- Reducera belastning på korttidsminnet – Eftersom det finns en gräns för hur mycket information en människa kan hålla aktiverat samtidigt i korttidsminnet så måste man anpassa skärmen genom att till exempel hålla den så enkel som möjligt och undvika att användaren måste sammanställa information från flera olika fönster.

[5]

2.4 Användarcentrerad systemdesign

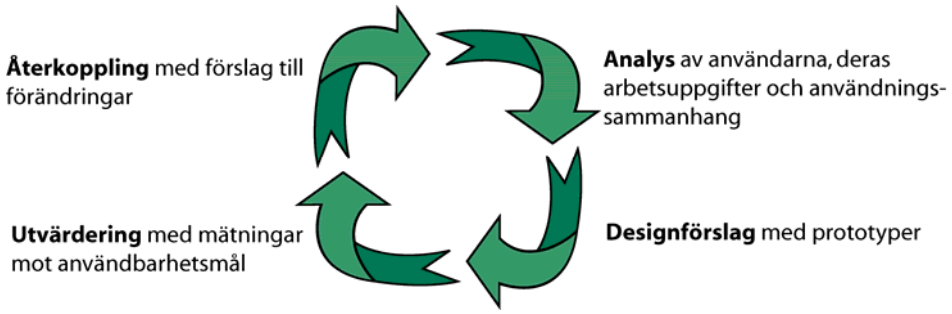
I en användarcentrerad systemdesign har man i fokus under hela utvecklingen att systemet ska betjäna användarna. Man ska fokusera på användarna och användbarheten inte bara i designen utan genom hela utvecklingsprocessen. Det viktiga i arbetet är att se till hur användarna kommer att kunna hantera systemet, hur de upplever det och om de kan utföra de uppgifter som systemet är designat för på ett tillfredställande sätt. När man pratar om användare menar man de som kommer att sitta och arbeta mot systemet, de brukar kallas slutanvändare. Slutanvändarna behöver inte vara samma människor som har beställt systemet. [7]

Design kan betyda olika saker för olika människor. Användaren ser själva layouten på skärmen som designen medan systemutvecklaren ser hela systemutvecklingsprocessen som design. Personen som designar användargränssnittet brukar tycka att design är själva utformandet av användargränssnittet. Det är bra om man diskuterar vad man menar med design så att det vid utvecklingen inte uppstår några oklarheter på grund av folks olika definitioner. [4,6]

2.4.1 Hur går en användarcentrerad systemdesign till?

När en användarcentrerad design börjar är det viktigt att alla som är med och utvecklar designen förstår verksamhetens mål och vad användaren ska använda systemet till. I ett tidigt skede bör man skapa användarprofiler där man beskriver olika slags användare. Man upprättar också olika scenarion av hur interaktionen mot systemet kan gå till. Användbarhetsmål tas fram för att sedan användas vid utvärderingen av designen. Under designutvecklingen låter man hela tiden användare av systemet tycka till och medverka för att till exempel testa en designutformning.

När man jobbar med användarcentrerad systemdesign brukar man använda ett iterativt arbetssätt. Det gör att man på ett naturligt sätt kan rätta till de fel man upptäcker vid användartesterna. Varje iteration i processen, se figur 2, måste innehålla en analys av användarens krav och i vilket sammanhang det ska användas i, ett designförslag med prototyper, en utvärdering där man gör en återkoppling till användbarhetsmålen och till sist väl beskrivna förslag på förbättringar av systemdesignen.



Figur 2. Beskrivning av användarcentrerad process

När designförslagen utformas kan man till en början rita enkla skisser på papper för att på ett enkelt sätt kunna se sina idéer. Det är viktigt att på ett konkret sätt kunna se hur det ser ut och hur man har tänkt att det ska fungera. Med hjälp av UML kan man skapa olika användningsfall och aktivitetsdiagram som beskriver användarens interaktion mot systemet, men här får man vara försiktig så att det inte blir för abstrakt för användaren utan att de fortfarande kan se en klar bild av designen. Utifrån skisserna och användningsfallen skapar man prototyper. Både prototyperna och skisserna används sedan vid utvärdering och vid tester. Det är bra om man kan jobba med flera prototyper samtidigt, det sparar tid och ger en möjlighet att se det på flera sätt.

[4]

2.5 Riktlinjer för Windows

Bra design händer inte av en slump. Precis som allt annat kräver det hårt arbete och mycket planering. Inte nog att man ska tänka igenom alla funktioner innan de skapas för att veta hur användaren skall interagera med programmet utan också när funktionen skall börja ta form på skärmen bör det finnas definierade standarder för att få alla funktioner lika. Om man kikar på ett operativsystem kan man omgående upptäcka den röda tråden i upplägget på fönster och menyer. Det är därför man känner igen sig även om funktionen man skall utföra körs för första gången.

Som programutvecklare för Microsoft Windows behöver man inte skapa sina egna riktlinjer för programmet, även om det är fullt möjligt. Det finns redan fördefinierade riktlinjer så att man kan följa standarden. Möjligtvis kan det bli svårare att skapa program om man måste följa någon annans ramar, men klarar man det får man en tydligare design eftersom programmet kommer uppfattas som integrerat. Programmet kan uppfattas som mindre professionellt om man tydligt kan se skillnaderna mellan de ramar man har försökt att följa.

Ramarna som Microsoft tillhandahåller beskriver bland annat avstånd mellan textrader, avstånd mellan knappar och storleken på dem. De ger även exempel på olika sätt att lösa placeringen av inmatningsrutor och rullistor. Lättast är att använda Visual Studio .NET eftersom där redan finns standarder avsedda för Windows.

Oavsett vilken typ av riktlinjer man väljer blir resultatet genomtänkt. Man behöver inte fundera på varför fönster är olika stora och varför knapparna på samma sida inte håller samma mått. Lättast blir det för programmeraren om det alltid finns beskrivningar för hur saker skall göras. Detta är inte nödvändigtvis det roligaste sättet att utveckla program på eftersom skaparen kan känna sig hämmad i sin design. Då är det viktigt att komplettera riktlinjerna så man kan agera fritt under kontrollerade former och där skapa något som skulle kunna vara en persons arbete och inte fem personers olika arbeten sammansatt.

Hela tiden utvecklas datorn och större skärmar och rörligare grafik blir mer tillgängligt för oss. Att behålla riktlinjer som är anpassade för en äldre version av det tänkta operativsystemet än aktuell version är inte att rekommendera. Det innebär ju att man utvecklar program för system som inte kommer att vara aktuella i framtiden.

[9, 10]

2.6 Analys och design

2.6.1 Användningsfall

Användningsfall är en beskrivning på hur en användare kan interagera mot systemet. Det beskriver det utifrån användarens synvinkel. När man till exempel använder sin Internetbank så kan man betala räkningar, föra över pengar mellan konton, köpa fonder och så vidare. Alla dessa aktiviteter är olika slags användningsfall.

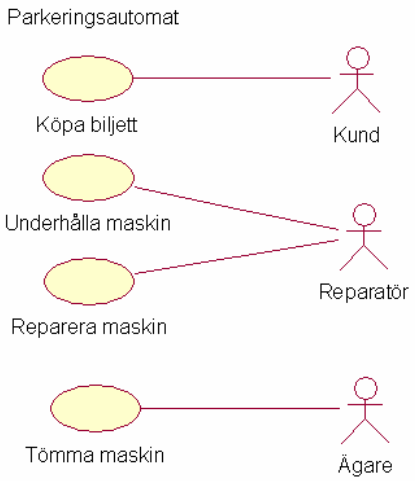
Ett användningsfall består alltid av systemet och sedan en eller flera aktörer. En aktör kan antingen vara en användare eller ett objekt som arbetar direkt mot systemet men objektet får inte vara en del av systemet. Om man ser på ett affärssystem så har de aktörer i form av användare, programmerare och till exempel en streckkodsläsare som läser av stämpelkort.

Varje användningsfall består av en sekvens av meddelanden mellan systemet och aktören. Vissa sekvenser ser alltid likadana ut medan andra kan variera beroende på om användaren får göra några egna inmatningar. Ibland kan ett agerande gentemot systemet upprepas flera gånger, som exempel inmatning av pengar i en parkeringsautomat. Användningsfallen ska även ta upp olika fel som kan uppstå och hur systemet då svarar användaren. Alla dessa beteenden mellan aktören och systemet skriver man i en användningsfallsbeskrivning.

[2]

2.6.2 Användningsfallsdiagram

Ett användningsfallsdiagram består av ett antal användningsfall och en eller flera aktörer. Uppsättningen användningsfall i diagrammet visar hela funktionaliteten i systemet med ett bestämt mått detaljer.



När man skapar diagram använder man sig av ett modelleringsspråk. Ett modelleringsspråk som används ofta inom programutveckling är UML (Unified Modeling Language). När man ritar ett diagram så ska man enligt UML:s notation använda sig av en ring som symbol för ett användningsfall och en streckgubbe för att representera en aktör. Figur 3 visar ett exempel på en parkeringsautomat där kunden är en aktör som kan interagera med automaten genom att köpa en biljett.

Figur 3. Bild av ett användningsfallsdiagram

Vi ser också att reparatören är en aktör som kan utföra underhåll på maskinen och reparera den. Sedan finns det också en tredje aktör som äger parkeringsautomaten, han tömmer de pengar som kunden har lagt i.

Just i detta exempel ingår inga aktörer i samma användningsfall men detta är fullt möjligt.

Man får inte bryta ner agerandet i för små delar. Att ha ett användarfall i parkeringsautomaten som heter lägga i pengar är alldeles för smalt, då detta är ett steg i användarfallet köpa biljett.

[2]

2.6.3 Scenario

Ett scenario beskriver den sekvens av händelser som uppstår vid en viss användning av systemet som vid ett användningsfall. När man skriver ett scenario brukar man göra det i textform liksom en berättelse av händelseförloppet. Om vi tar exemplet med parkeringsautomaten och tittar på användningsfallet köpa biljett så kan ett scenario se ut så här:

Kunden lägger i 1 krona.
Maskinen visar på en display att kunden får parkera till klockan 15.00.
Kunden lägger i ytterligare 1 krona.
Maskinen visar på en display att kunden får parkera till klockan 15.30.
Kunden trycker på knappen biljett.
Maskinen skriver ut en biljett.
Kunden tar biljetten.

Egentligen består detta scenario av olika meddelanden som skickas mellan aktören, maskinen och kanske olika enheter i maskinen.

[2]

3 Genomförande

3.1 Undersökningen

För att fastställa användningen, tankar och idéer hos dem som använder SEBS skrev vi samman ett frågeformulär, se bilaga 10, som vi skickade ut till tio stycken olika företag och bad dem titta på. Tyvärr var responsen liten och möjligtvis var tio ett för litet antal när det gällde undersökningsunderlag. Men det var vår tilldelning så vi hade inget med antalet att göra. Resultatet var växlande och de tre svar som vi fick var ändå ganska intressanta och kunde väcka lite tankar hos oss som skulle analysera SEBS.

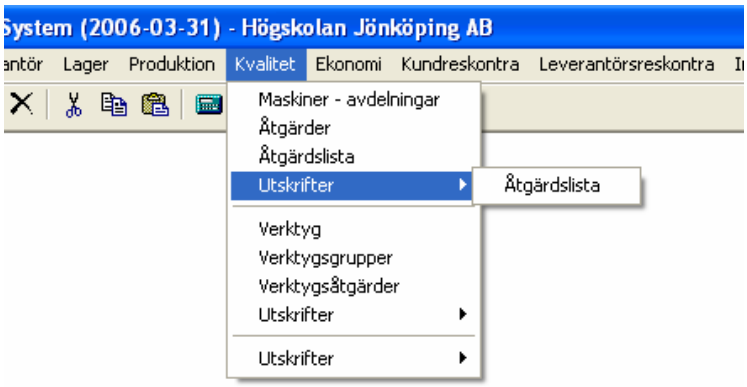
Konsekvent var hjälpavsnittet inte uppskattat och det fanns uttryck som sa att det inte var logiskt att göra saker som man normalt inte brukade göra. Även uttryck för att funktionerna var svårt utformade kom fram och vid framplockande av egna resultatlistor upplevdes rutinerna svårarbetade. Företaget får åtminstone en eloge för att de har en god telefonsupport som avhjälper alla typer av fel i programmet. För en utomstående anses nog det vara synd att användarna inte förlitar sig alls på den inbyggda hjälpen utan alltid ringer till supporten eftersom då har programmet och hjälpen inte lyckats motsvara varandra på ett bra sätt.

3.2 Analys av SEBS, och flera system

Vid analys av affärssystemen har vi efter en omfattande litteraturstudie valt att anmärka på några olika faktorer som vi finner bra respektive dåliga för programmet som helhet. Denna analys har inget med smidigheten eller funktionaliteten för programmet som affärssystem att göra eftersom endast design och upplägg av användargränssnitt har iakttagits. Vi har inte tittat på vilka funktioner affärssystemet har men vi har tittat på de funktioner som finns tillgängliga utifrån ett användarperspektiv. Det vill säga hur smidigt man navigerar i systemet och hur man uppfattar systemet grafiskt med färgval, storlek på knappar, textrutor, avstånd och dylikt samt layouten på skärmen. SEBS är det enda affärssystem som blivit grundligt synat och de övriga finns med som referenser och goda exempel på vad som är bra och dålig design.

3.2.1 SEBS

Det första som man tänker när man öppnar SEBS är att det är väldigt grått och känns föråldrat. Programmet är baserat på att man kan välja från menyer vad det är man vill jobba med, se bilaga 9. Eftersom SEBS är komplext blir menyerna många till antalet. SET har valt att separera utskrifterna för varje del i menyerna, se figur 4, vilket gör att det kan finnas flera menyval med namnet utskrift för varje meny. Detta kan röra till det för användaren då det inte alltid är självklart i vilken utskrift som man ska leta.



Namnen som beskriver menyerna är passande och gör det ändå ganska logiskt när man letar efter en viss funktion.

Figur 4. Exempel på meny med flera utskriftsval

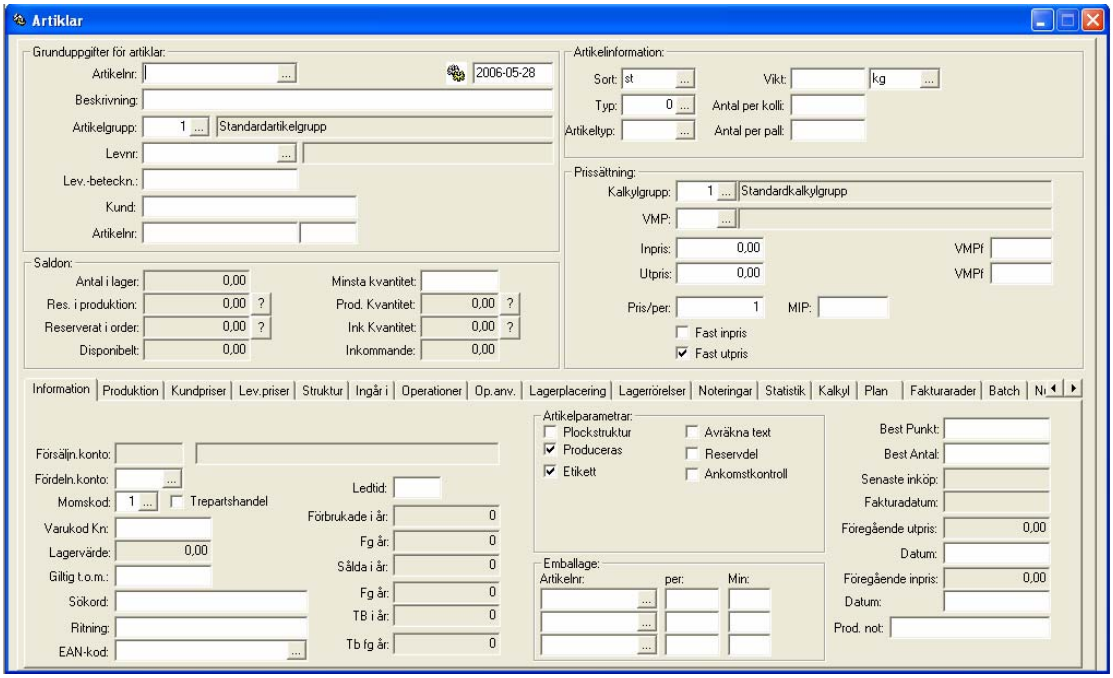
3.2.1.1 Knappar

I menyn finns det snabbvalsknappar för standardfunktioner som klippa ut, klistra in, spara och så vidare. Man har även tillgång till en knapp som öppnar upp Windowskalkylatorn. SEBS erbjuder också möjligheten att lägga till egna snabbvalsknappar för de funktioner man använder ofta. Detta uppfattar vi som väldigt bra, dels för att det snabbar upp arbetet för vana användare och dels för att det ger användaren en möjlighet att påverka användbarheten i systemet. Knapparnas svaghet ligger i att de är relativt otydliga och att det kan bli samma ikon för olika slags funktioner vilket kan bli förvirrande. Tydligare bilder vore att föredra samt inte återanvända dessa, för man kan då inte veta vilken som leder vart. När man lägger till egna snabbvalsknappar finns det vissa funktioner som inte får någon ikon alls och då inte syns uppe i menyfältet. Detta är inte acceptabelt. Vem ska kunna utnyttja den möjligheten då och det gör ju bara att frustrationen ökar hos användaren när den försöker lägga dit en ikon som genväg om det inte fungerar. Ett typiskt fall av kognitiv friktion.

3.2.1.2 Fönster

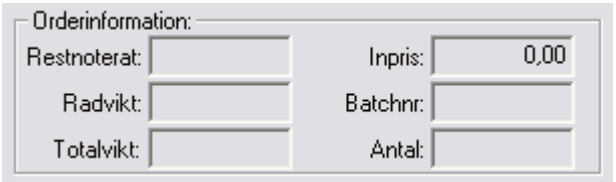
När man väljer att arbeta med en funktion i SEBS öppnas det alltid ett fönster. Detta fönster är oftast som en dialogruta där användaren ska mata in information i form av kundnamn, artikelnummer, beskrivningar och dylikt.

Många av fönstren man öppnar upp är täckta med väldigt mycket information. Vi har förståelse för att det är mycket information som måste presenteras och att det är mycket som användaren måste ha möjlighet att fylla i, men det borde gå att strukturera om informationen på ett mer lättsmält sätt. De har visserligen gjort en viss indelning genom att förlägga viss information på flikar, se figur 5, men även här är det ibland oerhört många flikar vilket gör det svårhittat och krångligt. Storleken på fönstren är ofta lika stora oavsett mängden information som skall visas, vilket innebär att man får lustiga vyer som inte ser genomtänkta ut. Man bör anpassa alla fönster till respektive informationsmängd som skall visas och alltid följa konsekventa riktlinjer för fönsterdesign, då får man alltid samma tilltalande känsla när man ser på programmet.



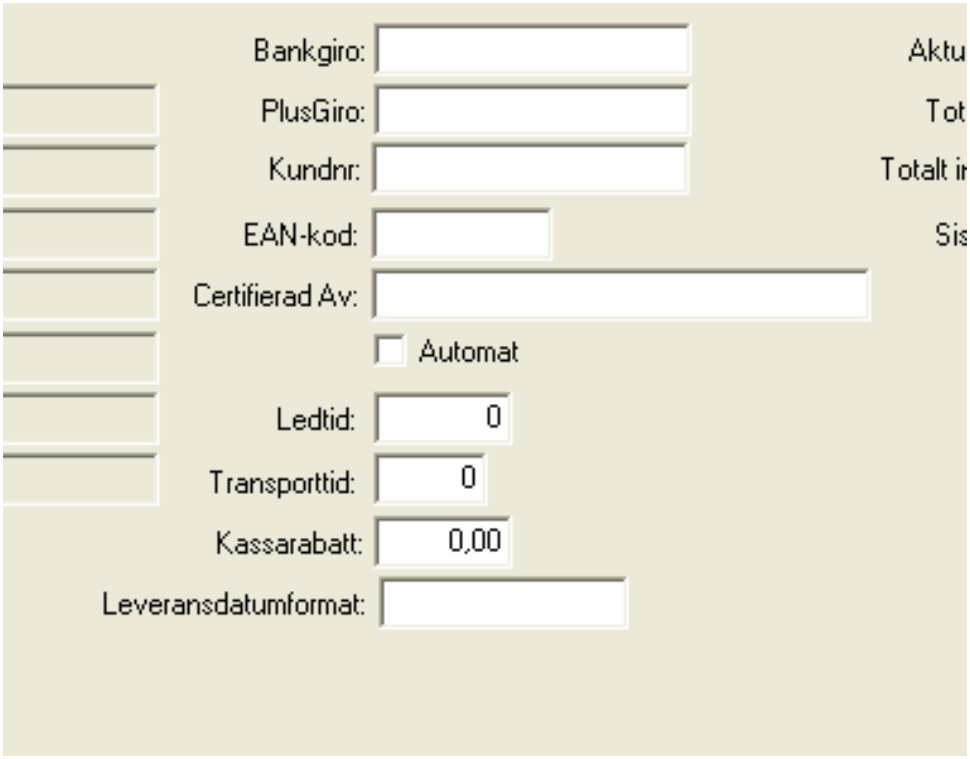
Figur 5. Bild på fönster med många flikar

Gruppering och korrekt namngivning av inmatningsrutor och information som hör ihop är bra för det skapar logik i arbetet. Se figur 6.



Figur 6. Bild av gruppering av rutor

Lika stora knappar samt lika stora inmatningsrutor med en snygg justering av tillhörande text skapar harmoni. Det finns ingen som tycker om att arbeta med något som ser rörigt ut. Som figur 7 visar bör det göras något åt gränssnittet. En konsekvent layout är tryggare att titta på.



Figur 7. Exempel på sneda rutor

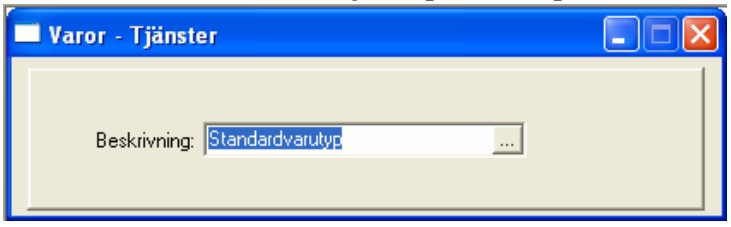
I SEBS kan man öppna fönster på fönster vilket kan vara bra för kontroll av information vid inmatning. När man ska växla till underliggande fönster så måste man antingen minimera det eller ta tag i fönstrets list och dra undan det för att nå det fönster som ligger bakom. Det finns inget naturligt sätt att växla mellan fönster. Möjligheten att maximera ett fönster finns enbart på en del funktioner vilket är svårt att förstå för att vid maximering ryms mer information och man slipper listen kring fönstret.

Längst ner finner man en statusrad med information om vem som är inloggad, vilket språk programmet körs i samt vilket lager man befinner sig i och längst ner till höger kan man se vilket fönster man arbetar i. Men när namnet på fönstret är långt försvinner informationen utanför skärmen. Se figur 8. Statusraden används för lite till nyttiga saker och man skulle kunna utnyttja den till dynamisk hjälp genom att alltid skriva ut vad som skall matas in i den ruta man står i eller vad som visas i den ruta man pekar på.

Figur 8. Bild av statusfält

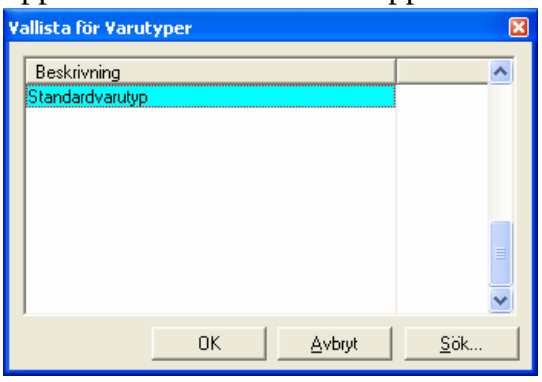
Vid inmatning av till exempel en ny kund kan det bli så att man missar att fylla i alla rutor. Programmet reagerar då på att alla rutor som behövs inte är ifyllda, men användaren får ingen hjälp med vad det är som saknas.
Felmeddelanden som direkt säger i vilken ruta felet ligger ger omedelbar respons till användaren och berättar då vad som är fel. Ändring kan ske omgående och ingen irritation kring programmet uppstår. Lika effektivt och mindre irriterande är att bara märka ut vilka rutor som saknar nödvändig information.

I kundmenyn finns det ett menyval som heter Varor-Tjänster, väljer man den får man upp fönstret som visas i figur 9. När man ser den får man ingen känsla av logik utan sitter mest undrandes hur man ska lyckas lägga till en vara. De flesta användare skulle trycka på de tre punkterna. Gör man det får man upp en vallista se figur 10. Denna hjälper dock inte till för att kunna göra den önskade funktionen nämligen lägga till en vara eller tjänst.



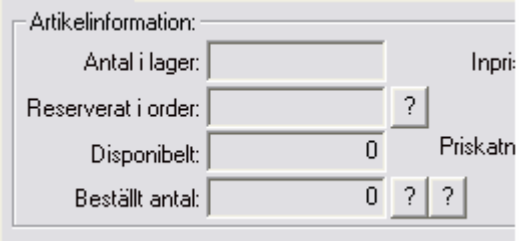
Figur 9. Bild av lägga till varor/tjänster

En van datoranvändare skulle troligtvis ge sig på att mata in en beskrivning och sedan trycka Returslag. Men detta hjälper inte. För att lösa hur det går till får man vända sig till hjälpmenyn. Däri finner man rätt information ganska enkelt vilket är att man först ska trycka Ctrl+F5 följt av inmatning av önskad beskrivning och sedan trycka F2 för att spara. I detta fall kan man lätt se att det behövs ett nytt tänkande när det gäller användargränssnittet. I fönstret man får upp borde det finnas en knapp som det står lägg till på och menyvalet borde istället heta Lägg till vara/tjänst för att förklara vad som kan utföras med denna funktion. Det som var bra med detta exempel var att det lätt gick att hitta informationen i hjälpavsnittet men det vore mer användarvänligt att ge användaren möjligheten att klara av uppgiften utan att behöva använda hjälpavsnittet.



Figur 10. Bild av vallista

Otydlighet kan ofta vara grunden till problem. Bredvid vissa inmatningsfält



finner man frågetecken på en knapp utan att förklaras varför. Ofta leder de till ett korresponderande fönster men i vissa fall finns två likadana knappar, utan text, som leder till olika ställen. Se figur 11.

Figur 11. Bild på knappar som förvirrar

Vid situationer som dessa, när man inte klart och tydligt förstår vad knapparna gör och vad som händer när man trycker på dem, skapas en känsla av osäkerhet.

3.2.2 Mamut

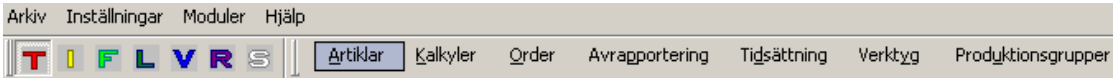
Mamut står för det mest genomarbetade affärssystemet ur en grafisk vinkel. Designen är modern och smälter väl in i Windows XP. Programmet är inte ett komplett system för tillverkande företag men jobbar istället med lösningar för Internet och man kan direkt i systemet skapa en hemsida med e-handels möjligheter. Vid varje dialogruta som man interagerar med känns designen genomarbetad och strukturerad. Vid de flesta inmatningar som användaren måste göra finns rullmeny med redan inmatade data. Alternativet till rullmeny är att en ny dialogruta kommer fram och är det inte mycket information som skall presenteras så är detta överflödigt samt kräver mer av användaren. Rullmenyn räcker att klicka på två gånger totalt om användaren kan välja direkt från menyn utan att behöva skrolla. Den nya dialogboxen däremot kräver antingen dubbelklick eller två klick med musförflyttning emellan. Tidsskillnaden är liten men känslan som skiljer i smidighet att använda programmet är stor.

3.2.3 SPCS

SPCS bygger på att det hela tiden finns en fast vy med knappar till många av de vanligaste funktionerna man använder. Idén är bra för då känner man alltid igen sig och hittar de knappar man söker. SPCS låter användaren arbeta med flera olika funktioner samtidigt och man väljer funktion via flikar som ligger precis ovanför statusraden. De är smidiga för man ser då alltid vilka fönster som man har startat och man kan smidigt växla mellan aktiviteterna genom snabbkommandon. Programmet är anpassat för att alltid arbeta med maximerade fönster och utnyttjar då till fullo ytan som finns tillgänglig. Färger och grafik är sparsamt använt men det går att anpassa programmet efter Windows eget färgschema. Ikonerna för snabbknapparna är små men det är tydliga bilder på dem och bilderna återfinns också i menyerna så att man ser vilken knapp som motsvarar raden i menyn, där så är möjligt. Ställer man muspekaren över en knapp kommer en gul skylt upp med en förklaring på vad knappen gör, samtidigt som det längst ner i statusfältet kommer en längre förklaring på samma sak. Det är alltid bra med hjälp som sker direkt när den efterfrågas. Det gör att användaren slipper använda hjälpavsnittet mer än nödvändigt.

3.2.4 Monitor

Monitor är precis som SEBS ett affärssystem för tillverkande företag. De klarar båda av ungefär samma saker men de gör det på lite olika sätt. Det är skillnad att arbeta i de olika programmen främst för att Monitor har en egen variant på vad man ska jobba med. I Monitor får man välja vilken del man ska jobba med och därifrån blir man tilldelad funktioner, se figur 12. Jobbar man med tillverkning finns artiklar med som menyval och där under finns vissa funktioner. Jobbar man med lager finns också artiklar att välja emellan men då har man andra möjligheter än i tillverkningen. Man väljer del utifrån vilken knapp man har intryckt. De är benämnda med bokstäverna T, I, F, L, V, R, S och står alla för en egen grupp aktiviteter. Monitor har lyckats skapa ett homogent affärssystem som känns bekant var man än hamnar. Layouten är inte modern men klarar sig ändå bra på att knapparna för respektive aktivitet finns där och vid varje val av funktion att arbeta med kommer snabbknappar upp som presenteras på ett eget verktygsfält anpassade för just det man ska göra.



Figur 12. Bild på Monitors menyfält

3.3 Utformande av nytt gränssnitt

För att kunna utforma ett nytt gränssnitt måste det befintliga systemet kontrolleras och analyseras noggrant. Komplexa system innebär massvis med olika variabler som bör diskuteras och utvärderas. En design måste mogna för att kunna vara till någon nytta.

Vi skaffade oss en känsla för vad vi gemensamt tyckte var bra design genom att utföra analyser på flera system och se på olika lösningar. Färger, knappar och en logisk följd har varit viktiga bitar att fokusera på för att skilja sig från den ursprungliga versionen av programmet. När man sitter och ”bara” ska konstruera en layout som man anser vara bra hinner man prata mer med varandra än om man strävar efter att hela tiden skapa lösningar som man kan visa. Det medför att iterationsantalet minskas eftersom små detaljer genast blir omhändertagna och inte följer varje version. Man kan förutom att prata färger och textjustering även diskutera kring smidigheten i det användningsfall som finns representerat i rapporten.

För att få en ökad förståelse för användaren och kunna se programmet utifrån flera synvinklar skapade vi flera olika användarprofiler, två av dessa kan ses i kapitel 3.3.1.

Vi valde att titta närmare på användningsfallet ”Registrera ny kund”. Utifrån det gjorde vi en användningsfallsbeskrivning, se kapitel 3.3.2.1. Sedan skapade vi ett scenario över hur användningsfallet används och hur man kan förenkla det, se kapitel 3.3.3. Vi ansåg inte att det handlade om att minimera antalet knapptryckningar utan att man istället skulle fokusera på vad som skulle göras och att det skulle utföras på ett sätt som får användaren att förstå vad som händer. Enkelhet handlar inte bara om få knapptryckningar och korta svar utan om tillräckligt många knapptryckningar för att skapa förståelse för hur systemet fungerar och för att användaren ska förstå hur och varför uppgiften utförs.

När scenariot var skapat och uppgiften hade blivit klar för oss började vi att skissa på olika förslag på utseendet, se bilaga 6 - 8. Genom att rita på papper har man möjlighet att på ett och samma ark testa olika varianter på design. Man ritar upp den man först tror på och kan under diskussion med kolleger ändra och trixa med bilden tills man tror sig kommit mer rätt. Sedan startar man om på nytt och när man närmar sig en färdig bild kan man använda bättre verktyg för att få mer detaljrikedom i bilderna. Vi testade oss fram genom att använda de olika metoderna som har presenterats tidigare. I början av skissandet ritade vi knapparna bara med en bild på, först trodde vi att det skulle räcka men vid ytterligare en iteration kom vi fram till att det behövdes en text för att förtydliga. Vi skapade olika personer som i systemet hade olika behov och olika sätt att se på saker. Innebörden blev att inga funktioner är enbart skapade för en mer erfaren användare men inte heller för en som är novis på datorer. Alla ska de känna att deras behov i programmet stillas men inte att de måste tvingas göra saker som de inte uppskattar.

Löpande under skapandet testade vi baklänges med vårt scenario så att det fortfarande var giltigt gentemot skissen. Tanken var ju att scenariot skulle vara dimensionerande för vad som skulle gå att göra, åtminstone ange grundkraven. För att sedan förstå användargränssnittet som användaren skall arbeta emot skapade vi ett User Interface Flow diagram. Det används till att skapa en överblick över vad användaren kan göra med systemet genom interfacet och för att kontrollera så att interfacet inte blir för rörigt. [6, 8] Alla steg som går att ta i programmet finns med i UI Flow diagrammet. Det täcker även de delar som vi inte har skapat men som måste finnas i programmet för att kundregistrering ska kunna vara möjlig.

När UI Flow diagrammet var klart startade finslipningen av designen. Vi valde att till stor del använda samma verktyg som används för tillverkning av SEBS, nämligen Visual Data Flex. Då kunde vi se vilka möjligheter som fanns för att förverkliga våra tankar för designen och rätta oss efter dem för att de nya riktlinjerna för kommande versioner skulle vara relevanta. All design gjorde vi inte i programmeringsverktyget eftersom det tog mer tid än nödvändigt. Vi valde då att montera bilderna i Adobe Photoshop som är mer flexibelt. Men inte för att göra något som inte skulle kunna gå att göra i Visual Data Flex utan bara för att snabba upp utvecklingstakten.

Vissa menyval som finns i gamla versionen av SEBS har vi plockat bort till exempel för att de ska omplaceras på en annan meny eller kanske för att kunna bli åtkomlig från ett fönster via en knapp eller en flik. Vi reserverar oss för att vi kan ha missuppfattat vissa funktioner som vi har tittat på och därför kan tillägg i menyerna krävas för ett fullständigt program.

Vi har tänkt att en bra lösning för att få systemet enkelt och logiskt att arbeta med är att man till exempel i listan för alla aktiviteter ska kunna klicka eller dubbelklicka på en aktivitet och då kommer man till denna aktivitet om man sedan dubbelklickar på den kund som är inblandad i aktiviteten så kommer man till dennes kundinformation och så vidare. Istället för att alltid starta olika funktioner/fönster genom att använda menyn så ska man kunna klicka sig vidare till de funktioner som man kan tänkas behöva. Det ställer lite högre krav på utformningen av systemet men är det sätt som vi anser vara bäst att arbeta med. En ny utformning av programmet skulle iterativt utöka sammanbindningen i programmet och ta en vy i taget. Problem skulle kunna uppstå eftersom kod som är skriven för att fungera fristående inte alltid är möjlig att bygga samman med andra funktioner.

Det är inte lätt att få designen rätt på en gång. Under utformningen av designen märker man att en idé man varit nästan säker på kanske inte alls ser bra ut när man snickrar ihop den. Det är verkligen viktigt att iterera och testa olika slags möjligheter.

3.3.1 Användarprofiler - Personer

Användarprofil för: Lars Hansson

Beskrivning av målgruppen

Ingenjörer med fokus på utveckling och produktion. Har inga direkta intressen för redovisning och bokföring. Tillhör den generation som fick datorer på gymnasiet och använder dem mer än gärna till allt. Sitter ofta med cad program och har inga problem med komplexa miljöer i form av mycket information och luriga beslut. Har en väldigt kort erfarenhet av ekonomi och finner ämnet besvärligt. Jobbar helst tillsammans med någon när det ska arbetas i ekonomi delen av programmet. Har fått ansvaret att planera produktionen tillsammans med produktionsteknikern och de sitter ofta på var sitt håll och arbetar.

Användningsmiljö och användningssammanhang

Sitter med ett eget kontor i en liten avskild kontorsdel i fabriken. Gränsar till andra kontor och bara tio meter till fabrikslokalen. Använder programmet till nästan allt som rör det administrativa arbetet. Det vill säga, kontakthantering, arbetsplanering och produktbeställning. Har en egen skrivare och tillgång till snabbare gemensam skrivare som även klarar A3.

Uppgifter

Lars får ofta förfrågningar från kollegor om läget i fabriken och hur han ser på att ta in nya jobb. Han planerar tillsammans med kollegan Eje Pärsbrandt beläggningen på maskinerna som finns och ser till att det alltid finns arbete att köra i dem. Han använder programmet till att ta fram orderlistor, körscheman, totalbeläggningstid och sådant som underlättar planeringen. Han behöver kunna utvärdera planeringen varje månad och använder även listor till det.

Mål med användningen

Lars behöver ett verktyg som skapar möjligheter att komma åt och lista all nödvändig information om fabriken som han jobbar på. Förr sparades all information i pappersformat och skickades runt till de olika ansvariga. Programmet ska kunna köra fram listor som man elektroniskt ska kunna skicka vidare till dem som behöver listorna.

Användarprofil för: Ingrid Jansson

Beskrivning av målgruppen

Medelålders ekonomer som inte utbildats med datorer. Arbetade på ekonomiavdelning i mellanstort företag när datorn introducerades.

Ingrid är 47 år gammal och är erfaren när det handlar om redovisning och ekonomi. Hon tycker att datorer är ett bra arbetsredskap och har funnit mycket glädje i Internet. Hon har inga svårigheter med att klicka runt med musen men känner sig inte hemma i situationer som man inte möter i ett vardagligt arbete vid datorn.

Användningsmiljö och användningssammanhang

Ingrid sitter och arbetar i ett kontorslandskap där hon delar skrivare med fyra stycken andra. Hon har ett övergripande ansvar när det gäller att svara i företagets växeltelefon.

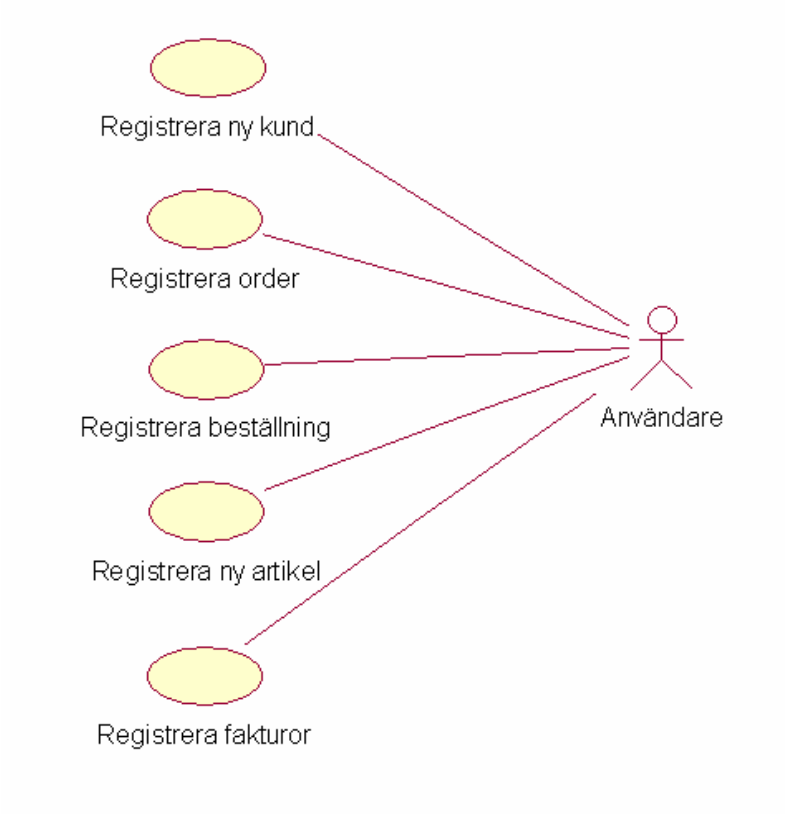
Uppgifter

Ingrid ser till att alla kunder får fakturor och att företaget där hon jobbar betalar sina räkningar i tid. Hon har ett övergripande ansvar på budget sidan där hon kontrollerar att inte olika projektkostnader springer iväg för långt.

Mål med användningen

Stöd och underlättning vid betalning och fakturering. Smidiga verktyg för att ta fram listor för verksamheten så som betalningsstatus och de senaste tre månaderna för en viss vara.

3.3.2 Användningsfall



Figur 13. Användningsfallsdiagram för en del av SEBS

3.3.2.1 Användningsfallsbeskrivning

Användarfall:	Registrera ny kund.
Summering:	Efter inmatning från användaren sparas den nya kunden i en databas.
Aktörer:	Användaren.
Förutsättningar:	Systemet är startat.
Beskrivning:	Användaren trycker på Orderknappen och väljer sedan att gå in på Kundmenyn och där väljer han kundregistret. Systemet visar kundregistret. Användaren trycker på knappen ny. Systemet visar en tom kundprofil. Användaren fyller i uppgifter om kunden. Användaren sparar sedan sin inmatning genom att trycka på knappen spara. Systemet sparar den nya kunden i databasen.

Undantag:	Avslutad: Om användaren stänger ner det aktuella fönstret kommer det upp en dialog som frågar om användaren vill spara den nya kunden. Ofullständigt ifyllt: Om användaren inte har fyllt i tillräckligt med information om kunden och försöker spara svarar systemet med ett informativt felmeddelande som talar om villka fält som användaren måste fylla i.
Villkor efter:	Inga

3.3.3 Scenario

Registrera Kund i SEBS

Användaren trycker på menyn kund.
Systemet visar kundmenyn på skärmen.
Användaren väljer menyalternativet kundregister.
Systemet visar kundregisterfönstret.
Användaren trycker på snabbknappen Ny post(kortkommando aF5).
Systemet tömmer fönstret.
Användaren skriver in 5 som nytt kundnummer.
Användaren skriver in namn på företaget, Sintermetall AB.
Användaren skriver in org nr. 333333-3333.
Användaren skriver in vat nr. 123 456.
Användaren skriver in postadress 571 31 Nässjö.
Användaren skriver in besöksadress Gårdsgatan 3.
Användaren skriver in telefonnummer 0380 12345.
Användaren skriver in e-post info@sintermetall.se.
Användaren klickar på … distr/säljare och väljer J från listan.
Användaren klickar på faktsätt och väljer skrivare från lista.
Användaren klickar på fliken kontaktpersoner.
Användaren skriver in Per Hansson, VD, 0380 12344,
per.hansson@sintermetall.se som kontaktperson.
Användaren trycker F2 för att spara.
Systemet sparar ny kund till databasen.
Användaren stänger fönstret kundregister.

Registrering av Kund i nya SEBS

Användaren trycker på knappen Order.
Systemet ändrar menyerna till dem som skall finnas vid order.
Användaren väljer alternativet kundregister.
Systemet hämtar alla kunder från databasen.
Systemet visar alla kunder.
Användaren trycker på snabbknappen ny kund.
Systemet visar ett tomt formulär med ett nytt kundnummer.
Användaren skriver in namn på företaget, Sintermetall AB.
Användaren skriver in postadress 571 31 Nässjö.
Användaren skriver in besöksadress Gårdsgatan 3.
Användaren skriver in telefonnummer 0380 12345.
Användaren skriver in e-post info@sintermetall.se.
Användaren skriver in kontaktperson Martin Persson, hans e-post martin@sintermetall.se och direkttelefonnummer 0380 12346.
Användaren väljer J i listan för säljare.
Användaren väljer faktureringssätt skrivare i listan.
Användaren skriver in org nr. 333333-3333.
Användaren skriver in vat nr. 123 456.
Användaren trycker på Ctrl+S för att spara.
Systemet sparar till databas.
Användaren stänger fönstret.

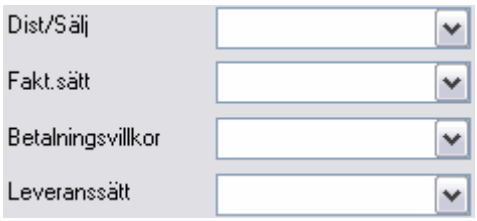
4 Resultat

4.1 Riktlinjer

Efter studerande av flera olika böcker om utformande av användargränssnitt och hur man uppnår ett användarvänligt gränssnitt har vi vid en analys av SEBS tagit fram ett antal riktlinjer som borde efterföljas vid fortsatt utveckling av SEBS användargränssnitt.

Det första och kanske viktigaste är att sträva efter att vara konsekvent och det gäller vid allt från att ha lika stora knappar och rutor till att man använder sig av en konsekvent grafisk design så att man känner igen sig överallt i programmet.

Text och inmatningsrutor ska vänsterjusteras för att öka lättläsligheten, se figur 14. Framför allt ska man inte blanda vänster och högerjustering. Inmatningsrutor och text justeras inte åt varsitt håll utan ska justeras åt samma. Längden på inmatningsrutorna är lätt att anpassa och genom att alltid ge alla rutor, som placeras tillsammans, samma storlek spelar inte justeringen någon roll.



Figur 14. Bild som visar på vänsterjustering av text och inmatningsrutor

När man har valt ett sätt att placera informationen, knapparna och inmatningsfälten på, så ska detta fortsätta konsekvent genom hela programmet. Det får absolut inte vara så att informationen på en sida är placerad längst upp i vänstra hörnet och på en annan sida placerad på en tom sida längst ner i högra hörnet. För att få flyt i användningen är det bra om man kan komma vidare i programmet genom att gå från ett fönster till ett annat som behandlar samma typ av funktionalitet. Till exempel om man kan gå från orderhanteringen och direkt komma till kunden från det fönstret. Här krävs då att man använder bilder som förklarar och att knapparna ser likadana ut i menyn som de gör när man vill gå mellan fönstren. Man känner då igen bilden och förstår att man kan fortsätta om man vill utan att behöva använda menyn.

All slags information som hör samman med varandra ska man gruppera och avskilja på ett tydligt sätt. En möjlighet är att sätta en ram runt gruppen för att tydligt markera samhörigheten. Detta ska också ske konsekvent genom hela programmet.

Snabbknappar är ett måste då detta snabbar upp arbetet för användaren. Bilderna på dessa ska vara tydliga och lättförståliga. Det ska inte heller gå att förväxla två bilder, de får vara relativt lika men något måste skilja dem åt på ett tydligt sätt. Snabbkommandon ska användas eftersom det ger vana användare en möjlighet att jobba enklare i programmet. Man ska använda standardkommandon till standardfunktioner så som klistra in, klipp ut och spara. Till andra kommandon ska man välja så logiska snabbkommandon som möjligt.

För att minska irritation hos användaren vid ej korrekt ifyllda fält ska man markera de fält som måste vara ifyllda för att man ska kunna slutföra funktionen. Detta kan man göra genom att sätta en liten röd stjärna vid de fälten. På detta sätt slipper användaren att få upp ett felmeddelande som kan skapa negativa känslor till användandet av programmet.

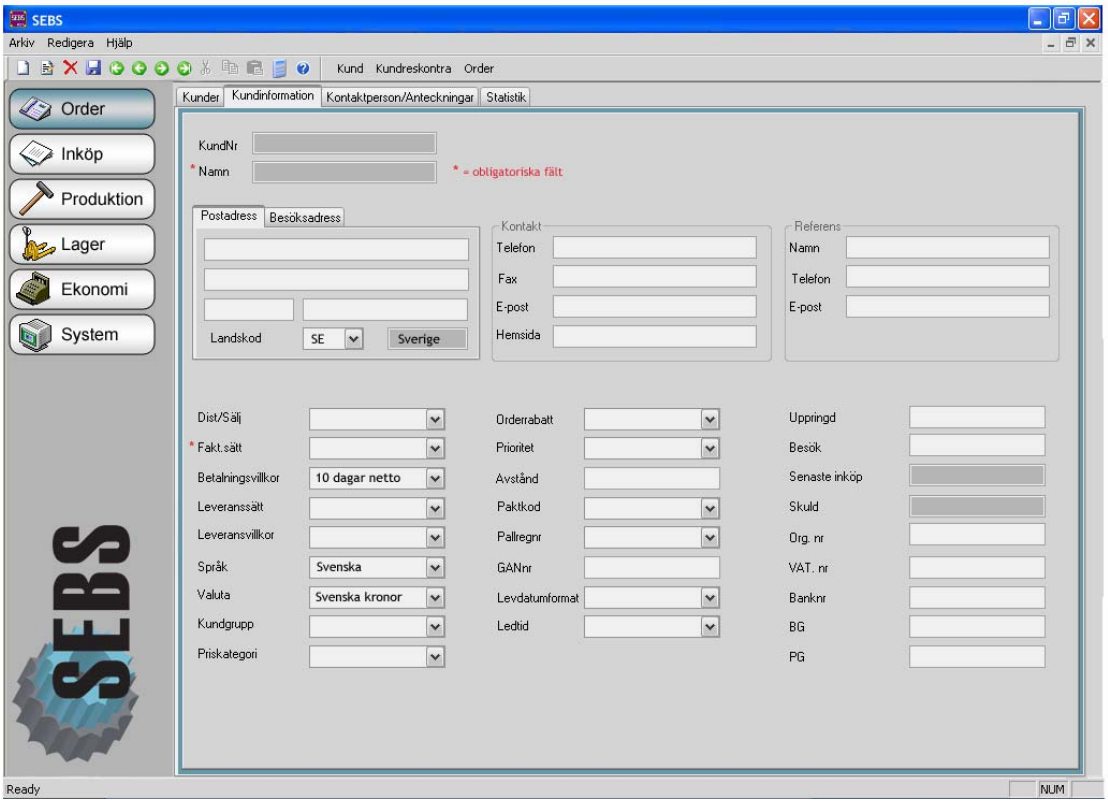
För att minska att ifyllandet i fälten blir fel eller felstavat så kan man använda sig av dropdown menyer där det är möjligt. Det är alltid bra för användaren att tydligt se vad som efterfrågas och dropdown menyer ger både exempel och förslag.

4.2 Gränssnitt

Det gränssnitt som vi skapade utifrån våra riktlinjer ger en känsla för vad som är intressant för ett program av den här typen. Tanken är inte att designen ska vara utåtriktad för att locka användare som dras till attraktiva animationer och snygga övergångar. Programmet är ett arbetsredskap för någon och personen sitter där för att jobba. Därmed inte sagt att personen måste vantrivas med sin tillvaro och tittar man på skärmen hela dagen är det av intresse att skapa en trivsam arbetsmiljö. Den kan vara nog så viktig som ytan kring personen som arbetar. Känns programmet rörigt skapas stress hos användaren och därför har vi valt att vara harmoniska i vår design genom att inte fylla alla ytor som är tillgängliga med information utan alltid eftersträva att fönstren behålls lättlästa och lättförståeliga. Fakta används olika ofta och därför bör man flytta undan den information som inte är relevant att alltid visa för användaren till en flik eller ruta som lätt kan kommas åt när så önskas.

För att få programmet att kännas nytt och fräscht bör man eftersträva att följa senaste utgåvan av det operativsystem som man tillverkar för. Det finns alltid standardikoner och riktlinjer lättillgängligt via Internet för programutvecklare och genom att använda samma typ av knappar som folk gör hemma vid sina datorer känner de igen ikonerna för att spara, nytt och miniräknaren till exempel. Då slipper man skapa nya bilder för dessa funktioner och användaren behöver då inte fundera nämnvärt på vad som är vad.

Vi har försökt omstrukturera menyerna och den information som behöver presenteras så att det inte ska uppfattas som så mycket information på en gång. Vi har valt att dela in sättet man arbetar mot SEBS i olika kategorier: order, inköp, lager, produktion, ekonomi och system, se figur 15 och även bilaga 2-5. Vi utformade designen så att det för varje kategori finns en knapp man kan trycka på. När man har tryckt på en knapp kommer det upp olika menyer i raden under menyfältet. Dessa ger en möjlighet att arbeta inom den kategori man har valt. Varje knapp är utformad med en informativ bild ihop med förklarande text.

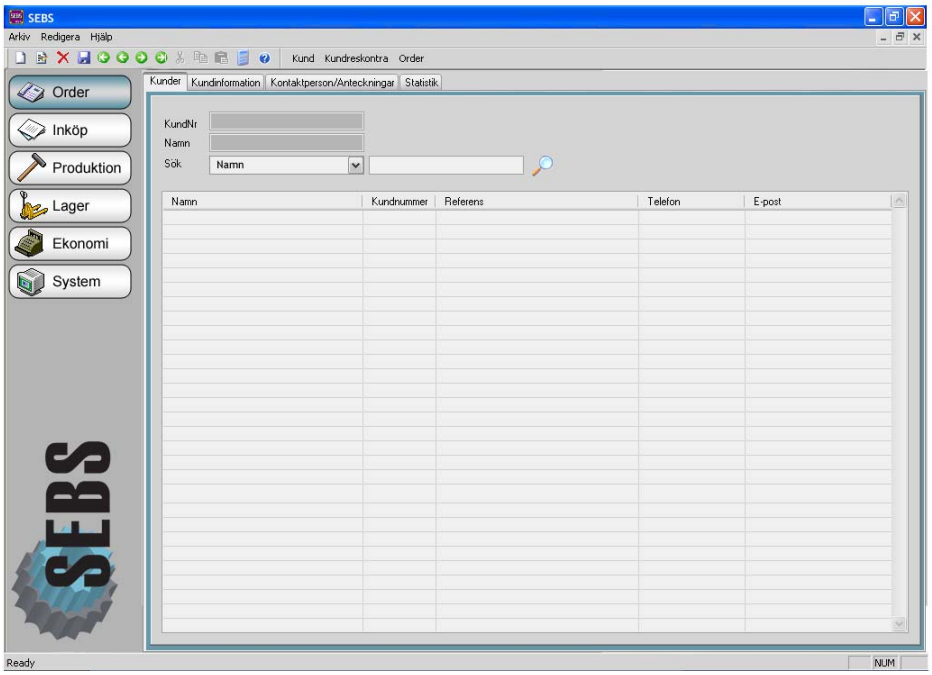


Figur 15. Bild på nya gränssnittet

För att informationen inte ska vara svåröverskådlig har vi valt att placera flikarna längst upp i fönstret istället för i mitten som det var gjort i det gamla SEBS. Vi har även här omstrukturerat hur informationen hanteras för att slippa ha så många flikar.

För att användaren inte ska bli förvillad utan hela tiden känna igen sig på ett enkelt sätt så har vi försökt hitta ett bra sätt som visar var man befinner sig någonstans i programmet. Vi valde att göra knappen man har tryckt på i en annan färg än de övriga, se figur 15. Dessutom band vi samman det med själva fönstret som öppnas genom att lägga en kant runt fönstret i samma färg.

Vid utvecklandet av gränssnittet kom vi fram till att sökfunktionen vid kundinformationen borde bli bättre. När man idag går in på kundregistret i SEBS får man upp en kundinformationssida. Vid kundnumret finns det en knapp man kan trycka på, då får man upp en lista med alla användarna. Från listan kan man antingen välja en av kunderna och då kommer kundens uppgifter fram i kundinformationsfönstret eller så kan man söka efter en kund men bara på kundnumret. Detta tyckte vi var lite dumt eftersom att det inte är kundnumret man först och främst kommer ihåg utan man borde kunna söka på



Figur 16. Bild på sökfunktion av kund i det nya gränssnittet

företagets namn eller telefonnummer också. Vi valde först när du går in i kundregistret visa en lista på alla dina kunder, se figur 16. Vill du söka efter ett företagsnamn väljer du namn från en dropdown-meny och sedan skriver du in företagets namn i rutan jämte och klickar på förstoringsglaset. Vi har också tänkt oss att man i denna lista ska kunna dubbelklicka på en kund och då få upp kundens information.

5 Slutsats och diskussion

- Vad är användarvänlighet?
- Varför ska man ha ett användarvänligt gränssnitt?
- Hur utvecklar man ett användarvänligt och användbart system?
- Hur går en designprocess till?

Vid slutet på arbetet kan man se tillbaka på det som har hänt och kan då finna svar på de frågor som vi ställde oss tidigt i rapporten. Frågorna är omfattande och skapade mycket förarbete genom litteraturstudier. Svaren är inte entydiga och lätta att summera men har man läst rapporten inser man att det handlar mycket om logik, att vara konsekvent och att lyssna på vad andra säger. Frågorna är uppdelade i olika områden men eftersom de vävs samman i varandra vid en förklaring så kan man inte bena ut dem var för sig.

Alla de undersökta programmen lider av små brister och fel som upptäcks när man gör en närmare studie av detta slag. SEBS är noggrant studerat och fler brister är framtagna där än hos de övriga programmen. Dels för att SEBS är utgångsmaterialet för rapporten men även för att det fanns fler tydliga brister i programmet som vi kände var direkt påverkande för användaren än hos de övriga. De brister som vi fann lämpliga att plocka fram hos de övriga för att diskutera lyfts upp här och mindre petitesser lämnar vi därhän för detaljnivån blir så ingående i så fall att läsaren tappar fokus på det som är viktigt.

Sättet vi har använt för att arbeta fram ett nytt användargränssnitt och riktlinjer på har fungerat bra. Använder man sig av en iterativ designprocess som vi har gjort, där man letar fram användarfall, beskriver dem med olika scenarion, ritar UI flow diagram, skapar fiktiva användare och så vidare, får man en bra insyn i hur användaren uppfattar programmet och hur användandet kan gå till. Med hjälp av alla dessa användningsfall, diagram och så vidare ser man lätt saker som kan förbättras. Om SET inser precis som vi, vikten av att se till användaren av programmet och tar till sig sättet vi har arbetat på tror vi att de kan vidareutveckla SEBS till att bli ett väldigt lättarbetat och användarvänligt program som ger en stabil och professionell känsla. De skulle dessutom ha möjligheten att involvera användarna av programmet i själva utvecklingen, till exempel vid framtagandet av användarkrav och vid tester av prototyper, vilket skulle leda till ett ännu mera utvecklat program.

6 Referenser

[1] Allwood Carl Martin(1991) *Människa-datorinteraktion – Ett psykologiskt perspektiv*.
Studentlitteratur ISBN 91-44-32671-8.

[2] Blaha Michael; Rumbaugh James (2005) *Object-oriented modeling and design with UML*.
Pearson Prentice Hall, ISBN 0-13-196859-9.

[3] Cooper Alan (2004) *The inmates are running the asylum*.
Sams publishing, ISBN 0-672-32614-0.

[4] Gulliksen, Jan; Göransson Bengt (2002) *Användarcentrerad systemdesign.*
Studentlitteratur ISBN 91-44-020295-5.

[5] Shneiderman Ben (1998) *Designing the user interface*.
Addison Wesley Longman, Inc, ISBN 0-201-69497-2.

[6] Qiyang Chen (2001) *Human Computer Interaction: Issues and Challenges.*
Idea Group Publishing, ISBN 1-878289-91-8.

[7] Användarcentrerad systemdesign (2006) http://www.acsd.se (Acc.2006-05-19).

[8] Introduction to user interface flow diagram (UI Storyboards) (2006) http://www.agilemodeling.com/artifacts/uiFlowDiagram.htm (Acc. 2006-06-06)

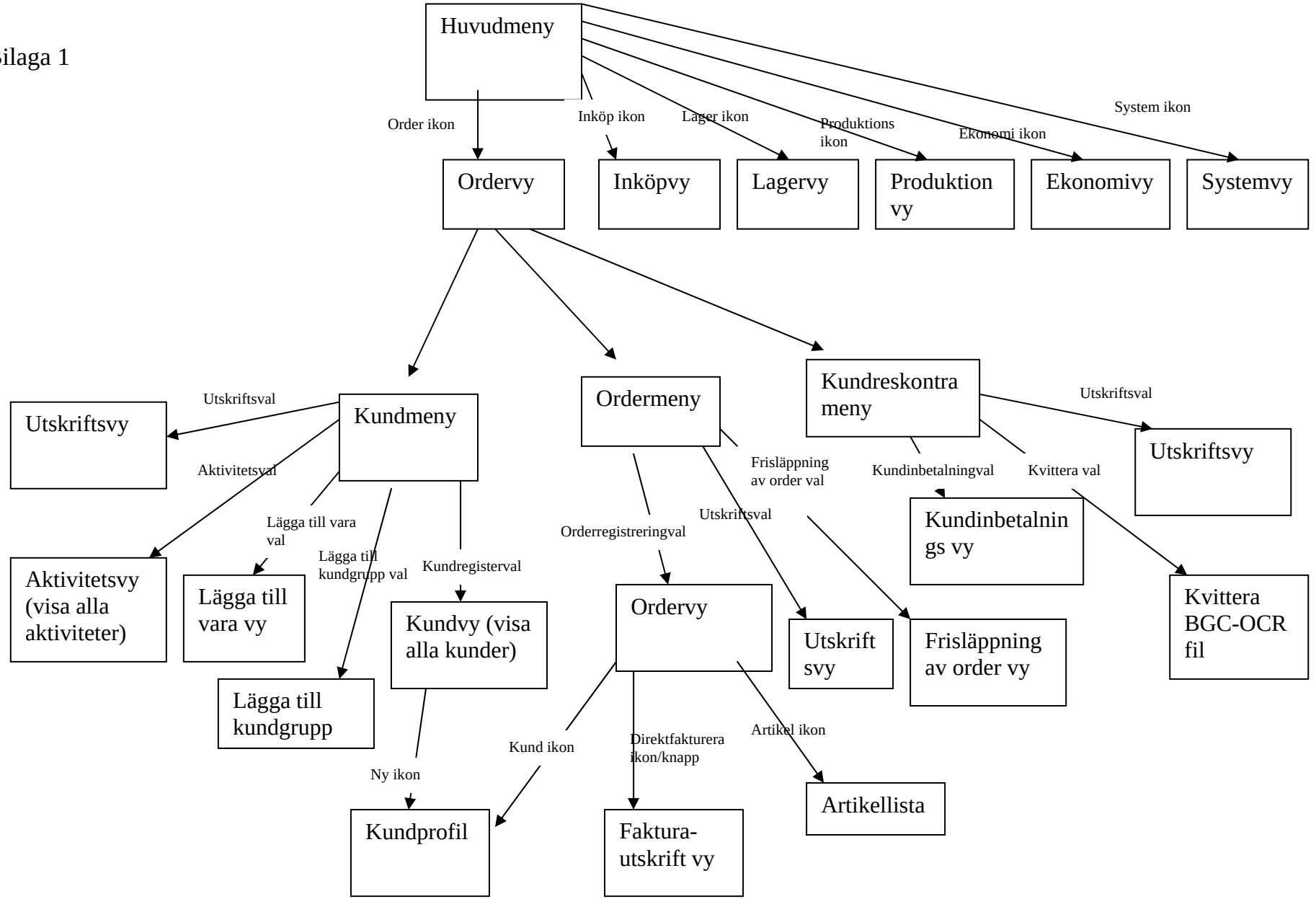
[9] Aero Aesthetics (2006) http://msdn.microsoft.com/library/default.asp?url=/library/en-us/UxGuide/UXGuide/Visuals/index.asp (Acc. 2006-06-07).

[10] Windows XP – Guidelines for Applications (2006) http://www.microsoft.com/whdc/Resources/windowsxp/default.mspx (Acc. 2006-06-07).

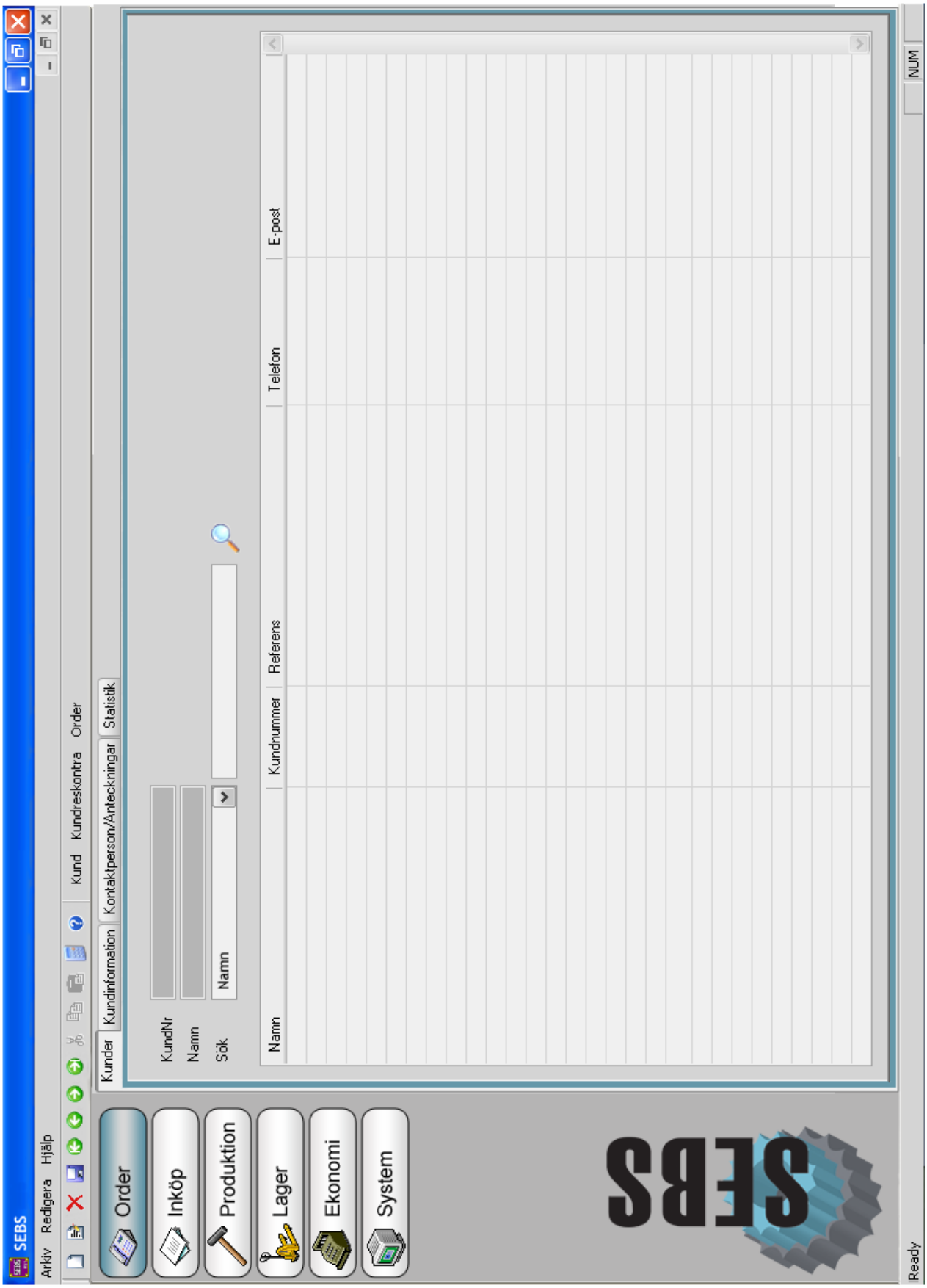
7 Bilagor

Bilaga 1 User Interface Flow Diagram
Bilaga 2 Nya gränssnittet till SEBS - kundvy
Bilaga 3 Nya gränssnittet till SEBS - kundinformationsvy
Bilaga 4 Nya gränssnittet till SEBS – kontaktpersoner/anteckningsvy
Bilaga 5 Nya gränssnittet till SEBS – statistikvy
Bilaga 6 Skisser av ny design
Bilaga 7 Skisser av kundlista och kundinformation
Bilaga 8 Skisser av statistik och anteckningar
Bilaga 9 Bild på SEBS
Bilaga 10 Frågeformulär

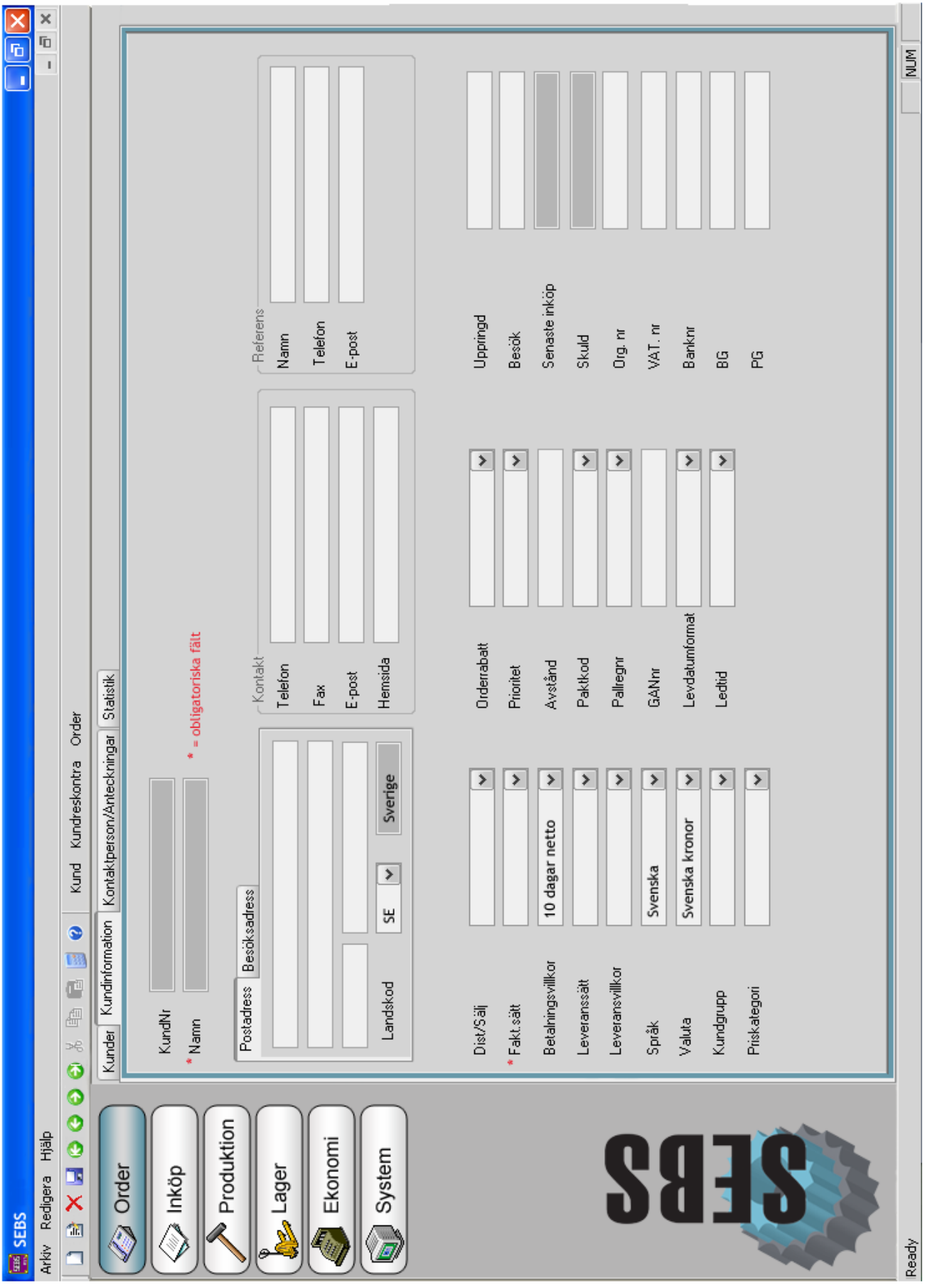
Bilaga 1


Huvudmeny
Order ikon
Inköp ikon
Lager ikon
Produktions ikon
Ekonomi ikon
System ikon
Ordervy
Inköpvy
Lagervy
Produktion vy
Ekonomivy
Systemvy
Kundmeny
Ordermeny
Kundreskontra meny
Utskriftsval
Utskriftsvy
Aktivitetsval
Lägga till vara val
Lägga till kundgrupp val
Kundregisterval
Aktivitetsvy (visa alla aktiviteter)
Lägga till vara vy
Lägga till kundgrupp
Kundvy (visa alla kunder)
Orderregistreringval
Frisläppning av order val
Utskriftsval
Ordervy
Utskrift svy
Frisläppning av order vy
Kundinbetalningval
Kvittera val
Utskriftsval
Kundinbetalnin gs vy
Kvittera BGC-OCR fil
Kund ikon
Direktfakturera ikon/knapp
Artikel ikon
Ny ikon
Kundprofil
Faktura-utskrift vy
Artikellista

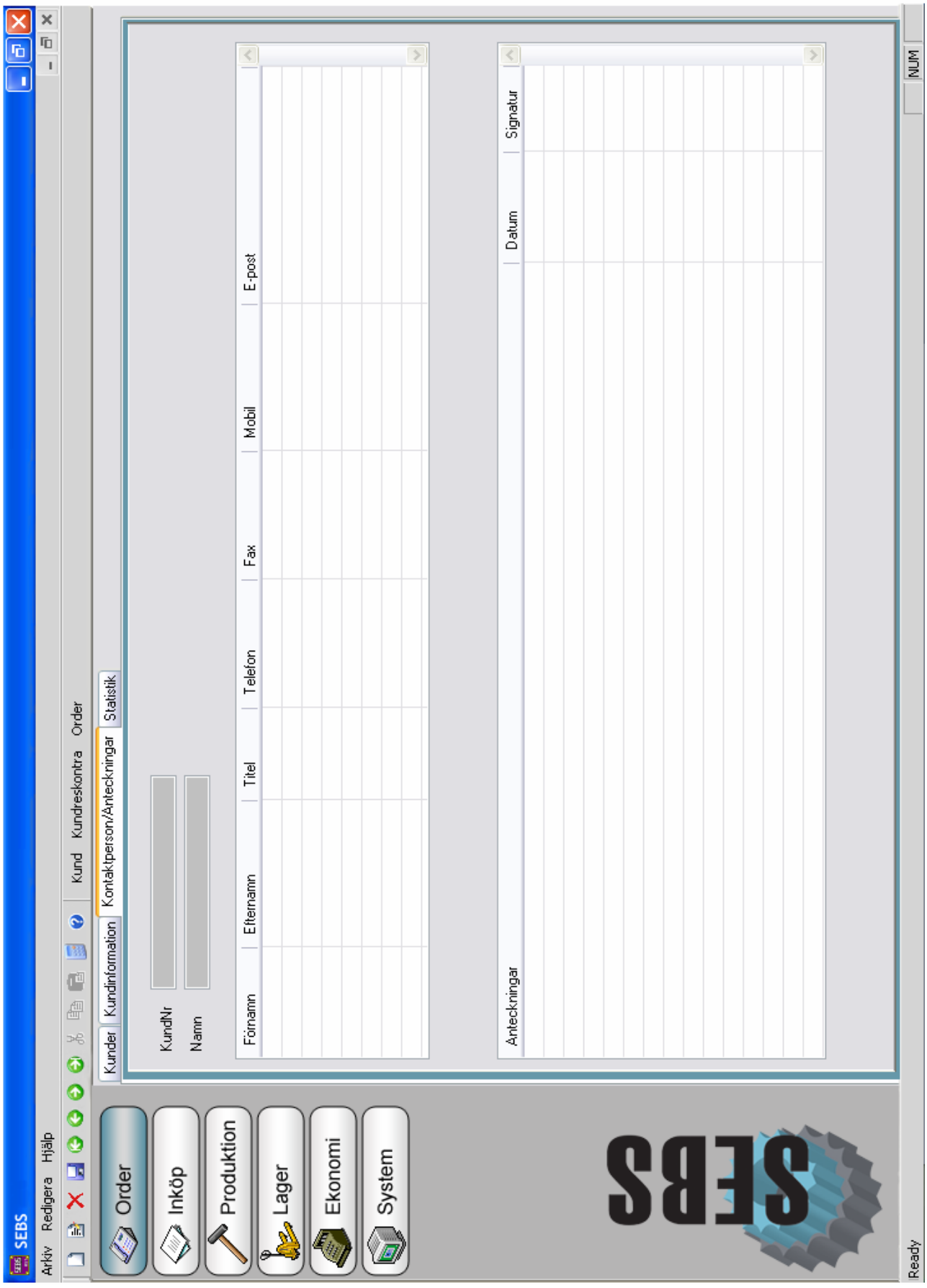
Bilaga 2


SEBS
Arkiv Redigera Hjälp
Kund Kundreskontra Order
Order
Inköp
Produktion
Lager
Ekonomi
System
Kunder
Kundinformation
Kontaktperson/Anteckningar
Statistik
KundNr
Namn
Sök
Namn
Kundnummer
Referens
Telefon
E-post
NUM
Ready

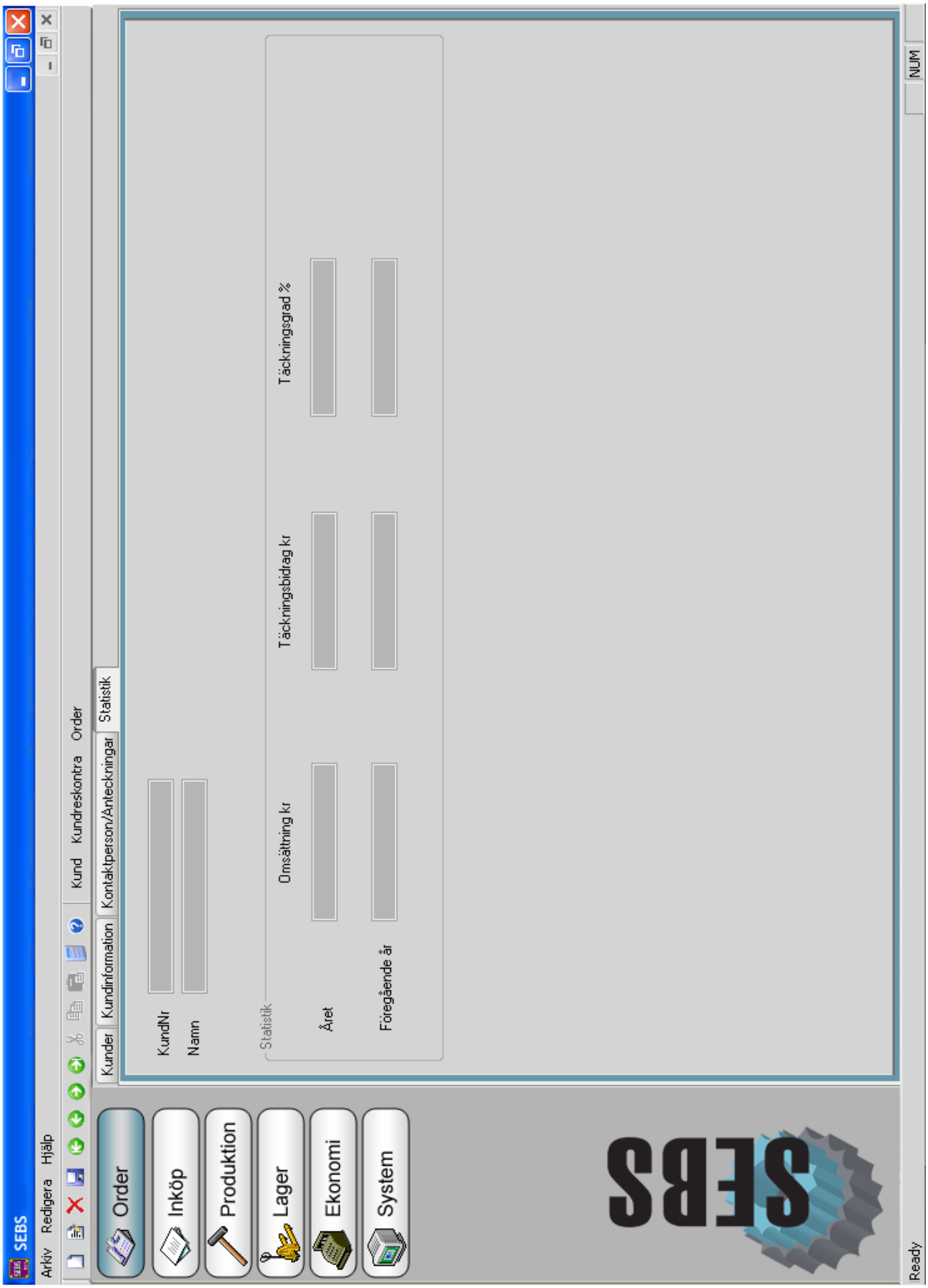
Bilaga 3


SEBS
Arkiv Redigera Hjälp
Kund Kundreskontra Order
Order
Inköp
Produktion
Lager
Ekonomi
System
Kunder Kundinformation Kontaktperson/Anteckningar Statistik
KundNr
* Namn
* = obligatoriska fält
Postadress Besöksadress
Landskod SE Sverige
Kontakt
Telefon
Fax
E-post
Hemsida
Referens
Namn
Telefon
E-post
Dist/Sälj
* Fakt.sätt
Betalningsvillkor 10 dagar netto
Leveranssätt
Leveransvillkor
Språk Svenska
Valuta Svenska kronor
Kundgrupp
Priskategori
Orderrabatt
Prioritet
Avstånd
Paktkod
Pallregnr
GANnr
Levdatumformat
Ledtid
Uppringd
Besök
Senaste inköp
Skuld
Org. nr
VAT. nr
Banknr
BG
PG
Ready
NUM

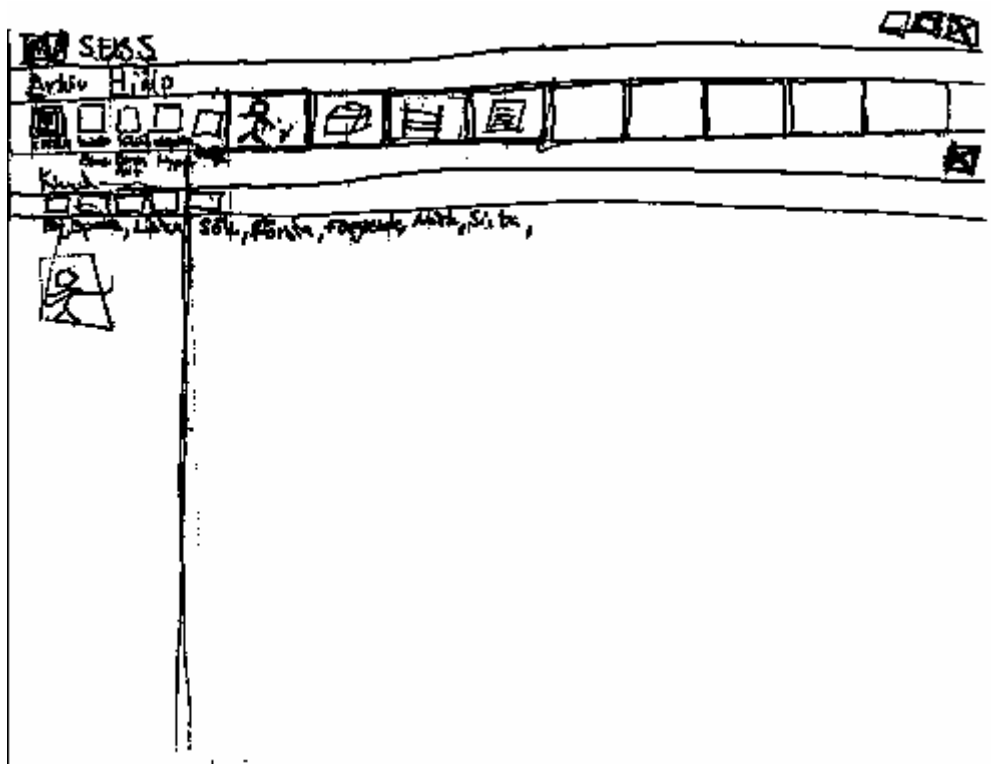
Bilaga 4

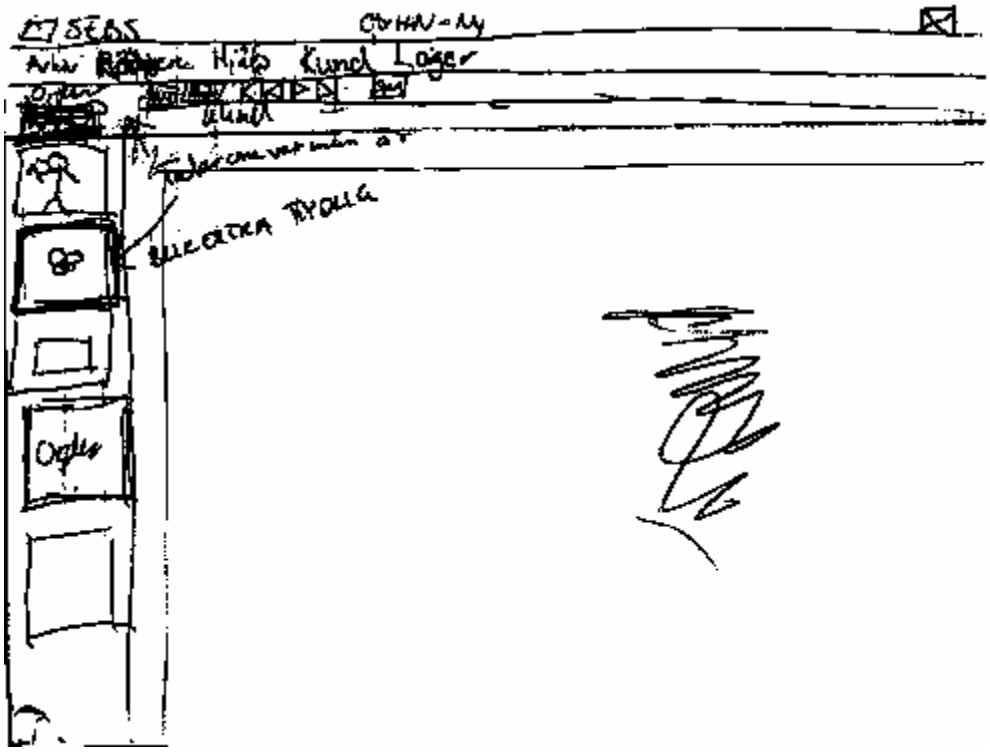

SEBS
Arkiv Redigera Hjälp
Kund Kundreskontra Order
Kunder Kundinformation Kontaktperson/Anteckningar Statistik
KundNr
Namn
Förnamn Efternamn Titel Telefon Fax Mobil E-post
Anteckningar Datum Signatur
Order
Inköp
Produktion
Lager
Ekonomi
System
SEBS
Ready
NUM

Bilaga 5

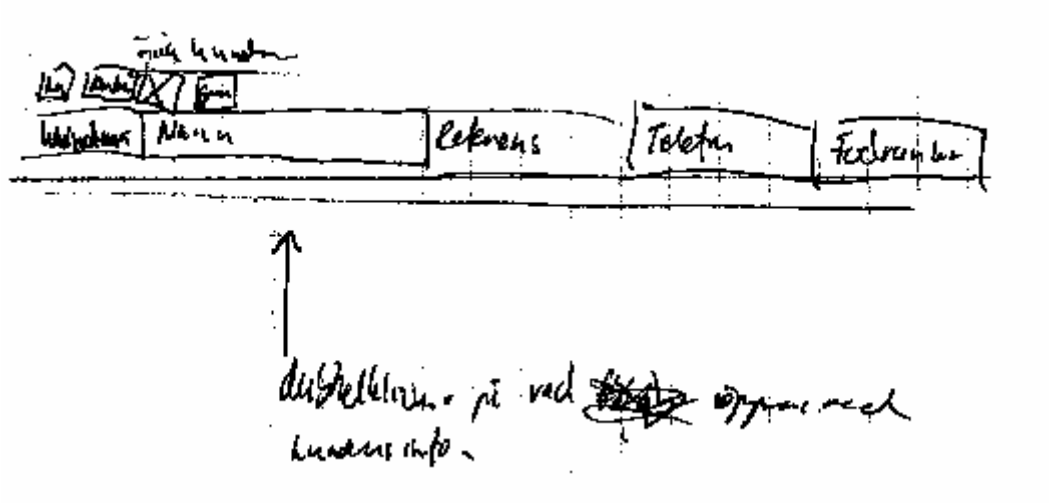

SEBS
Arkiv Redigera Hjälp
Kund Kundreskontra Order
Order
Inköp
Produktion
Lager
Ekonomi
System
Kunder Kundinformation Kontaktperson/Anteckningar Statistik
KundNr
Namn
Statistik
Omsättning kr
Täckningsbidrag kr
Täckningsgrad %
Året
Föregående år
SEBS
Ready
NUM

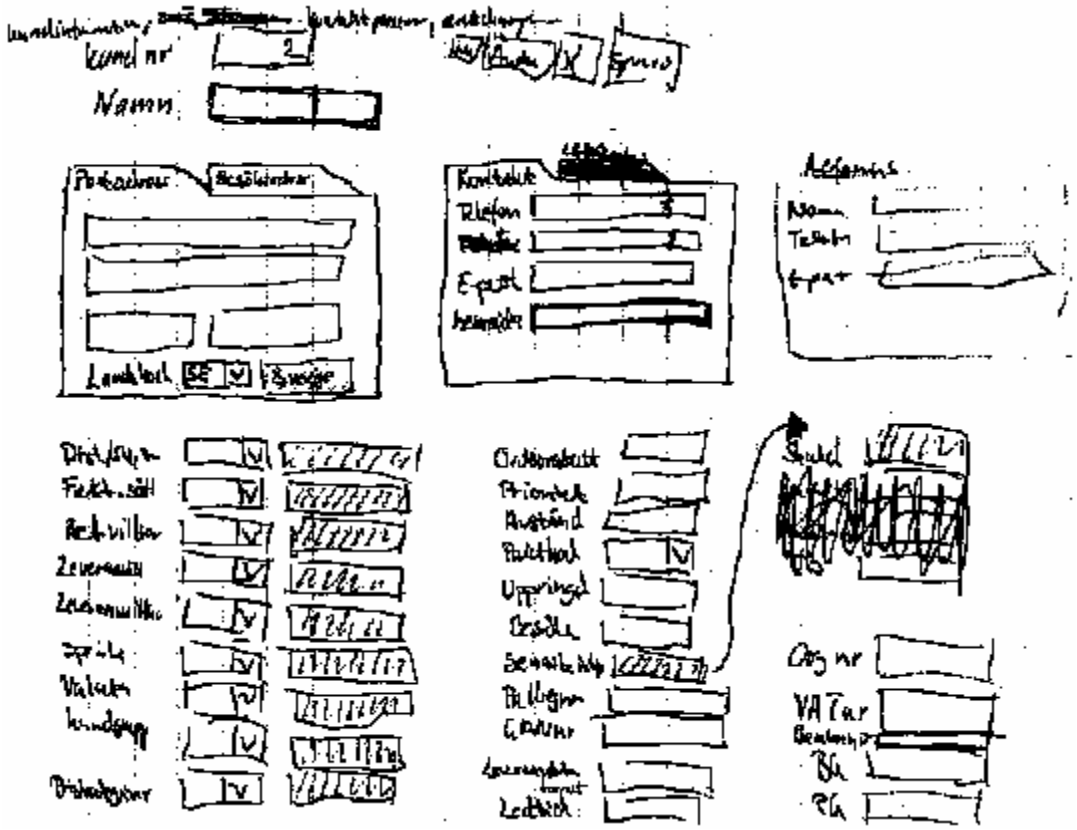
Bilaga 6



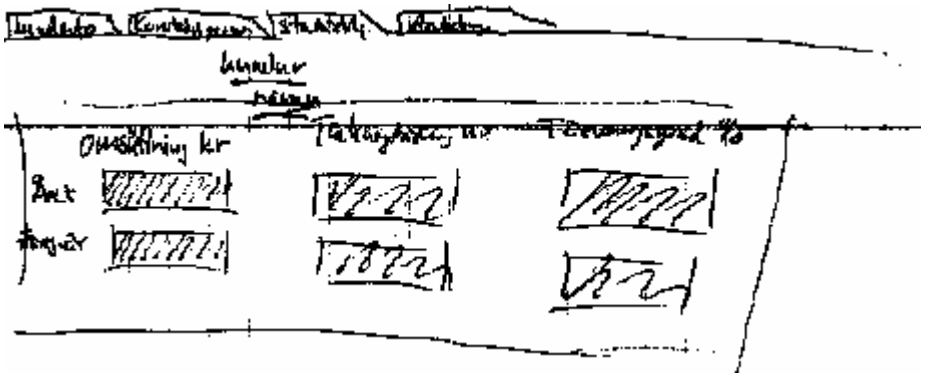


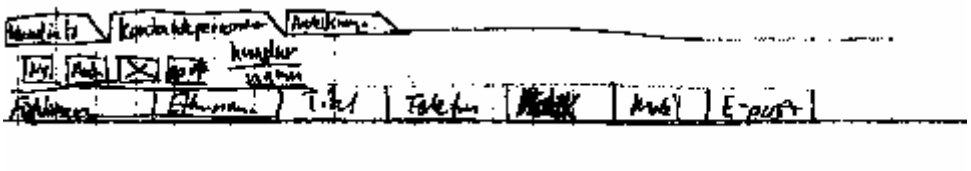
Bilaga 7


Namn
Telefon

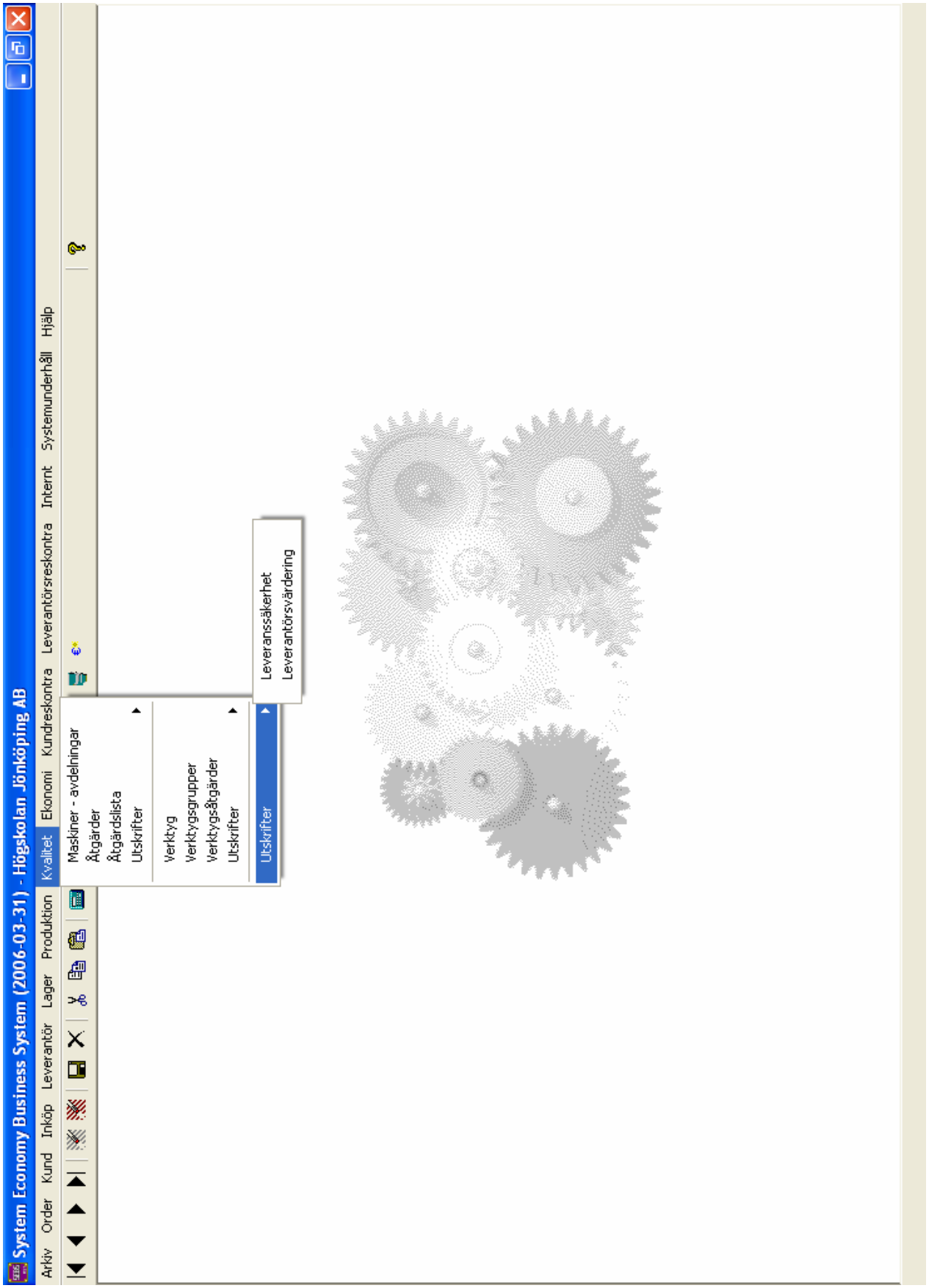

Kund nr
Namn
Land
Sverige
Kontakt
Telefon
E-post
Adress
Namn
Telefon
E-post
Org nr
VAT nr

Bilaga 8





Bilaga 9


System Economy Business System (2006-03-31) - Högskolan Jönköping AB
Arkiv Order Kund Inköp Leverantör Lager Produktion Kvalitet Ekonomi Kundreskontra Leverantörsreskontra Internt Systemunderhåll Hjälp
Maskiner - avdelningar
Åtgärder
Åtgärdslista
Utskrifter
Verktyg
Verktygsgrupper
Verktygsåtgärder
Utskrifter
Utskrifter
Leveranssäkerhet
Leverantörsvärdering

Bilaga 10

Frågeformulär

- Hur länge har ni använt SEBS?
- Vilka tidigare erfarenheter har ni av affärssystem?
- Hur många män respektive kvinnor använder systemet hos Er?
- Vilken utbildning fick ni när ni började använda programmet?
- Vad är Eran uppfattning om att lära sig arbeta i SEBS? Uttryckt på en skala 1(mycket lätt) - 10(mycket svårt).
- När du ska använda funktioner som inte används så ofta, känns det då logiskt och lätt att hitta dem eller måste du leta länge alternativt titta i hjälpavsnittet? Svara utförligt.
- När du använder hjälpavsnittet, finns då informationen du söker 1 (lättillgänglig) - 10 (svårtillgänglig)?
- Skulle du vilja att hjälpen vore tillgänglig på något annat sätt, om ja vilket?
- Till vilka funktioner använder du snabbkommandon i programmet?
- När du ska starta en funktion använder du dig av menyerna eller använder du dig av de egna inlagda snabbknapparna?
- Ange några ord som beskriver hur du uppfattar programmets utseende.
- Är ni rädda för en förändring av programmet? I så fall på vilket sätt?
- Övriga synpunkter om SEBS