Avdelning för datavetenskap

Christer Oscarsson, Jonas Larsson

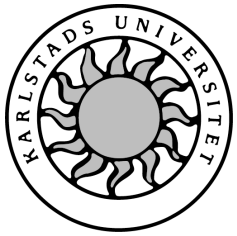
Administrationsverkyg för marinvåg

Administration tool for marine scales

Datavetenskap
C-uppsats

Datum: 10-06-08
Handledare: Thijs Holleboom
Examinator: Martin Blom
Löpnummer: C2010:03

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se



Datavetenskap

Christer Oscarsson, Jonas Larsson

Administrationsverktyg för marinvåg

Examensarbete, C-nivå

C2010:03

Administrationsverktyg för marinvåg

Christer Oscarsson, Jonas Larsson

Denna uppsats är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Christer Oscarsson, Jonas Larsson

Godkänd, 2010-06-08

Opponent: Ewelina Helmersson, Mollin Widegren

Handledare: Thijs Holleboom

Examinator: Martin Blom

Sammanfattning

Detta examensarbete är gjort på uppdrag av UniSystem AB och Zoff Com AB i Torsby. Uppdraget gick ut på att skapa ett program för administration av en flyttbar lagringsenhet som används tillsammans med en våg som är avsedd för yrkesfiske med mindre fiskebåtar. Vågar av denna typ kallas för marinvågar, vilket innebär att de är utvecklade för att kunna väga korrekt ombord på ett skepp i gungning. Vågsystemet har stöd för att skriva ut etiketter till de lådor med fisk som man vägt.

Lagringsenheten agerar mellanhand mellan vågsystemet ombord på fiskebåten och en Windows-baserad dator som vanligtvis finns i land. Lagringsenhetens uppgift är att flytta inställningsfiler till vågsystemet, samt att lagra en logg över uppmätta värden som sedan kan användas tillsammans med programmet för att skapa en rapport över fångad fisk.

Programmets uppgift är att ge användarna möjlighet att hantera de inställningsfiler som finns på lagringsenheten, samt att kunna skapa rapporter över fångsten utifrån det logg-data som finns sparat i lagringsenheten. Inställningsfilerna innehåller information som används av vågen för att anpassa vissa menyalternativ i vågen, samt för att styra utseendet på etiketterna.

Abstract

This dissertation describes a project that was done for UniSystem AB and Zoff Com AB in Torsby.

The assignment was to create a program for management of a removable storage device that is part of a new model in Unisystems series of scales intended for the fishing industry. Scales of this kind are called marine scales, they are designed to give a correct result even on a ship at sea. The scales system has support for printing labels to the boxes of fish that's been weighed.

The storage device is acting as a messenger between the scales system on board the fishing boat, and a Windows-based computer usually stationed on land. The storage device's task is to transport the configuration files to the scales system, and to store a log of measured values which can be brought back and be used by the program to create a report on the amount of caught fish.

The programs mission is to enable its users to manage the configuration files on the storage device, and to generate reports from the log data that is stored on the storage device. The configuration files contain information used by the scales system to modify parts of its menu system, as well as to control the look of the printed labels.

Innehåll

Figurer

Kapitel 1

Bakgrund

Detta kapitel går igenom bakgrunden till hur projektet kom till. Därefter ges korta beskrivningar av de företag som är knutna till projektet. Vi ger därefter en snabb beskrivning av den vågmodell som hör till projektet och slutligen förklaras hur vägning av fisk går till och hur vågsystemet används ombord på fiskebåtarna.

1.1 Projektets bakgrund

Unisystem AB, ett företag som bland annat tillverkar industriella vågar, höll på att utveckla en ny marinvågsmodell. En marinvåg är en våg som är utvecklad för att kunna användas ombord på ett skepp i rörelse.

Vågen är tillverkad för fiskeindustrin och riktar sig funktionsmässigt främst till användning ombord på mindre fiskebåtar. Vågen används till att väga fisken när den packas i lådor ombord på fiskebåtarna.

Funktionsmässigt är skillnaden på den nya vågen och dess föregångare att den nya modellen klarar etikettutskrifter och loggning av mätvärden. Med dessa funktioner hoppades Unisystem kunna skapa en vågmodell som var tillräckligt avancerad för att täcka in behovet hos mindre fiskebåtar, fast samtidigt enkel nog för att inte bli för dyr eller

svårhanterlig.

För att en loggning av uppmätta värden ska bli användbar, behövs mer information kring vägningen än bara vikt och tidpunkt, till exempel art på fisken och var fisken fångades. Sådana uppgifter utskrivna på etiketterna förenklar också arbetet jämfört med om lådorna märks upp med informationen i ett separat moment.

För att kunna lagra en loggfil och dessutom kunna flytta den från vågen till en dator behövs en flyttbar lagringsenhet. Denna enhet benämns som 'loggerenhet' eller 'logger', eftersom den behöver vara kopplad till vågsystemet för att spara loggfilen vid vägningarna.

Förutom till att lagra loggdata används loggern även för att uppdatera vågens information kring bland annat fisksorter och etikettutskrifter. Denna information finns sparad som filer på loggern och laddas in till vågen varje gång en loggerenhet ansluts.

För att ge sina kunder möjligheten att enkelt kunna hantera den information som finns på loggern och dessutom få ut något användbart av loggfilen behövde Unisystem ett nytt verktyg i form av ett Windows-program. Projektet som beskrivs i denna uppsats är utvecklingen av detta verktyg.

1.2 Företagsbeskrivningar

1.2.1 Unisystem AB

Unisystem AB[1] är ett svenskt företag med fabrik i Torsby vars huvudinriktning är industrivågar, marinvågar samt termometrar med hög precision. Företaget bildades 1979 och fabriken i Torsby öppnades 1986. För närvarande har företaget sju anställda i Torsby.

Några av företagets specialområden är vägning i explosionsfarliga miljöer, vägning i fordon och båtar samt termometrar med precision på 0.001° Celcius.

1.2.2 Zoff Com AB

Zoff Com AB[2] står för utvecklingen av loggerenheten och programmeringen av vågens kontrollerenhet. Företaget, som är beläget i Torsby, grundades 1997 och har för närvarande tre anställda.

Zoff Coms sysselsättning är bland annat utveckling av elektronikprodukter för industriella tillämpningar, programutveckling samt utbildning inom elektronik, data och IT.

1.3 Marinvåg U350

Vågen bygger på en teknik som gör att den med hjälp av en referensvikt kompenserar den analoga mätsignalen för förändringar i rörelse och lutning innan signalen omvandlas till ett digitalt mätvärde. Detta gör att vågen har mycket god precision även ombord på ett skepp i rörelse, med ett fel på under fem gram vid mätningar på upp till fem kg.

1.4 Vägning av fisk

Den centrala funktionen hos marinvågen är vägning av fisk, men för att beskriva processen vid vägningen behöver vi först se närmare på begreppen viktklass och fiskeområden.

1.4.1 Klassindelning av fiskar

Fisken delas in i olika grupper baserade på deras art och viktklass. Viktklassen anger ett viktintervall på fisken där en den lägsta klassen används för de största fiskarna. Till exempel så har Torsk fem viktklasser, från ‘Torsk 5’ som är den lättaste, till ‘Torsk 1’ som är den tyngsta. Viktklassen uppskattas av fiskemännen under tiden de packar fisken.

1.4.2 Fiskeområden

Fiskevattnen är indelade i olika områden där varje område är uppdelat i ett antal delområden. Ett exempel på ett område med ett delområde är: ZON 27111 D Östersjön 22-24 Västra, där ZON27111 D Östersjön är ett område och ett av dess delområden är 22-24 Västra. Enligt reglerna för yrkesfiske får en fiskebåt bara fiska inom ett och samma fiskeområde på samma resa, vill de byta område måste de alltså först gå i hamn.

1.4.3 Vägningsprocessen

Innan man börjar väga fisk så ställs vågen in på rätt fiskeområde. Detta sker endast en gång per resa. Man ställer in vågen på den art och viktklass man ämnar väga och placerar en låda på vågplattan. Därefter nollställs vågen och fisk av den art och viktklass man angett lastas i lådan.

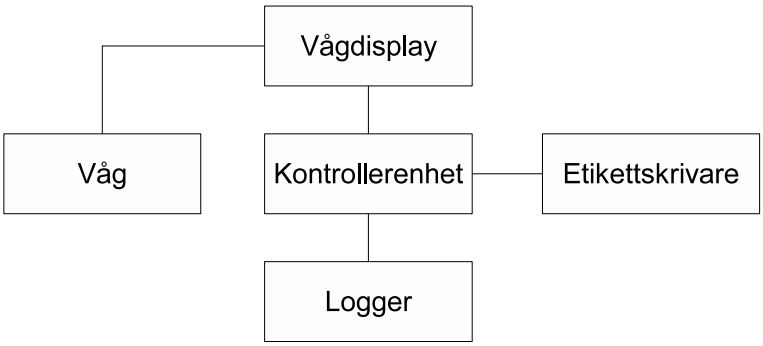
När lådan blivit full anges detta på vågen. Beroende på vilket arbetssätt som föredras kan man nu skriva ut en etikett och märka upp lådan, eller vänta med etikettutskriften och börja på nästa låda. Om man väljer att väga flera lådor innan man skriver ut, så kommer etiketterna för alla lådor som vägts sedan förra utskriften att skrivas ut i den ordning man vägde klart lådorna.

Kapitel 2

Vågsystemet

I detta kapitel beskrivs de delar som bygger upp vågsystemet, samt hur de olika delarna är kopplade till varandra.

2.1 Vågsystemets uppbyggnad



Figur 2.1: Ovan visas hur vågsystemets olika delar är sammankopplade.

Det som tidigare, för enkelhetens skull, har benämnts som 'vågen' är egentligen ett system som består av flera delar. Dessa är: vågen, vågdisplayen, kontrollerenheten, loggerenheten och etikettskrivaren. Figur 2.1 visar hur de olika delarna i vågsystemet är kopplade till varandra.

2.2 Vågen

Figur 2.2: Föregångaren till marinvåg u350. Bilden visar 1) Vågplattformen, 2) Vågdisplayen. (Bilden är tagen av UniSystem AB)

Vågen består av en viktindikator och en vågplattform. Viktindikatorn är vågens mätenhet och befinner sig under vågplattformen. Figur 2.2 visar den våg och display som u350 bygger på. Vågen som visas i figuren klarar av att väga last upp till 30kg.

Matningsspänningen ombord på båtar är oftast 230V AC eller 24V DC. Till skillnad från sin föregångare har denna modell av vågen ett nätaggregat för 230V AC - 24V DC byggts in i den rostfria inkapslingen, vilket gör att vågen enklare kan flyttas mellan båtar med olika matningsspänning.

2.3 Vågdisplayen

Vågdisplayen visar aktuellt mätvärde på vågen. Vågdisplayen är oförändrad i utseende jämfört med föregående vågmodell (se figur 2.2 punkt 2).

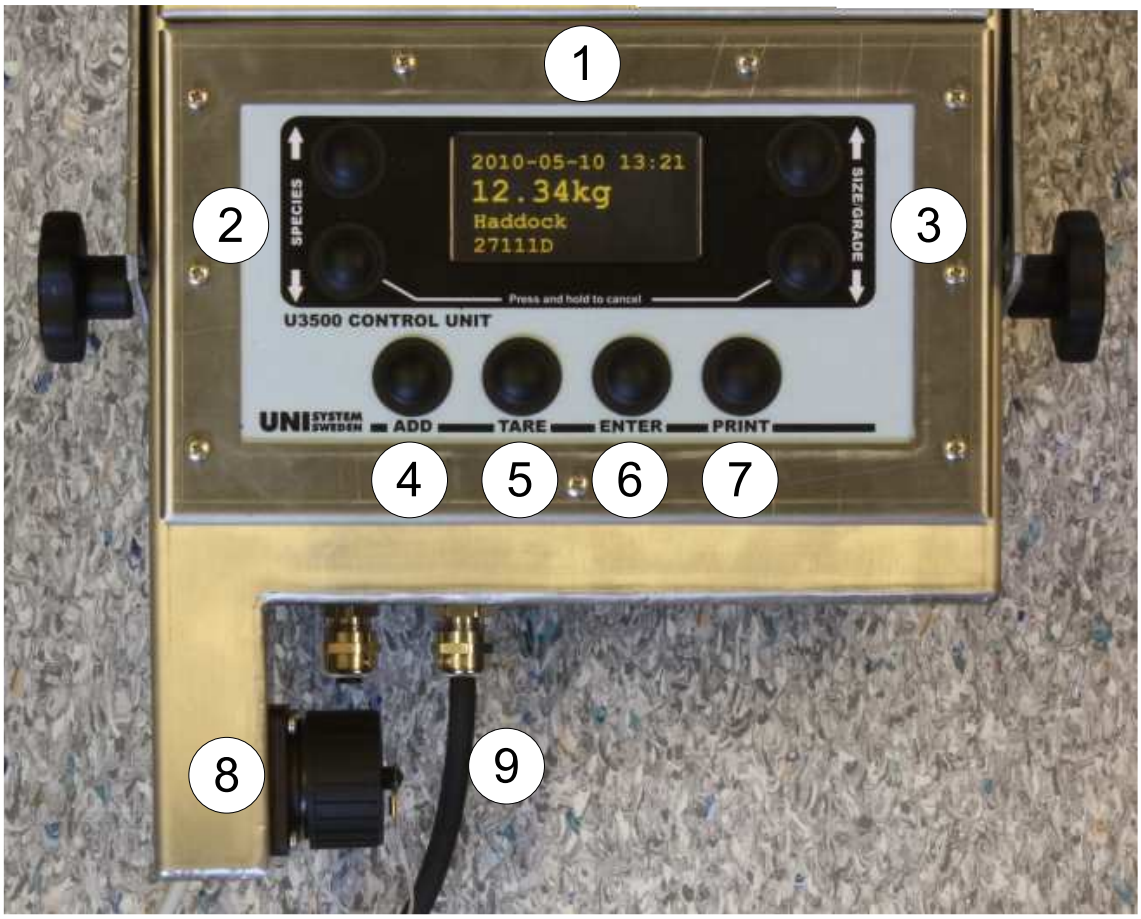
2.4 Loggerenheten

Figur 2.3: Loggerenheten. 1) 14-pinnars hankontakt för anslutning till kontrollenheten, 2) Miniusb-kontakt för anslutning till dator.

Loggerenheten är en lagringsenhet som används för att flytta data mellan kontrollerenheten och en dator. Man kan tänka på loggern som ett USB-minne som är robust byggt för att klara av stötar och diverse stänk.

Loggern har två anslutningar. Ena sidan har en 14 pinnars anslutning för koppling till kontrollerenheten och kan skruvas fast för att säkra att den inte lossnar under användning. Den andra sidan av loggern har en anslutning av typen miniusb-hona för anslutning till dator. Denna kontakt kan skyddas med en hatt medan loggern är kopplad till kontrollerenheten vilket visas i figur 2.3.

Loggern ansluts till en dator med hjälp av en usb-kabel med en mini-usb hane i ena änden som kopplas till loggern och en vanlig usb-hane i andra änden som kopplas till datorn. För datorn ser loggern ut och beter sig som ett vanligt usb-minne med 2GB lagringsutrymme. Loggern strömförsörjs antingen av kontrollerdelen eller av datorns usb-kabel beroende på var den är inkopplad.



Figur 2.4: Närbild på kontrollerenheten. 1) Display, 2) Spieces, 3) Size/Grade, 4) Add, 5) Tare, 6) Enter, 7) Print, 8) Kontakt för loggerenhet, 9) Anslutning till vågen.

2.5 Kontrollerenheten

Kontrollerenheten är den del av vågsystemet som agerar som mellanhand mellan viktindikatorn, loggern och etikettskrivaren. Kontrollerenheten tar ständigt emot viktvärdet från viktindikatorn och visar det aktuella värdet i sin egen display. Det är med kontrollerenheten som man väljer vilket fiskeområde, fiskart och viktklass som ska skrivas ut på etiketterna. I samband med etikettutskriften sparas även all information kring vägningen ner i en logg-fil om loggern är ansluten.

Figur 2.4 visar kontrollerenheten i närbild. I figuren visar nio punkter:

1. **Display** Displayen visar olika information beroende på vilken operation man utför. I figur 2.4 visar displayen datum, tid, vikt, fiskklass och fiskzon.

2. **Species** Knappar för att bläddra mellan de fiskarter som är sparade.

3. **Size/Grade** Knappar för att bläddra mellan klasserna för angiven fiskart.

4. **Add** Används för att ange att man vill lägga till mer emballage. Efter att man lagt till det man vill i lådan, till exempel ett lager is, använder man knapp 5 (Tare) för att fortsätta vägningen.

5. **Tare** *Tare weight* är den engelska beteckningen på vikten av en tom behållare. Knappen används för att nollställa vågen efter att man ställt på en låda och på så vis räkna bort vikten på lådan+emballage.

6. **Enter** Knappen används för att ange att man är klar med vägningen av den nuvarande lådan.

7. **Print** Knapp för utskrift av etiketter för alla lådor som vägts färdigt sedan man senast skrev ut etiketter.

8. **Loggerkontakt** Den 14-poliga kontakt som används för att ansluta loggerenheten. Kontakten kan skyddas av ett lock när ingen logger är ansluten, vilket visas i figur 2.4.

9. **Anslutning till vågen** Den kabel som kopplar kontrollerenheten till vågens viktindikator.

2.6 Etikettskrivaren

Etikettskrivaren är av märket Zebra och modellen är GK420d. Den är av typen termoskrivare vilket innebär att utskrifterna sker genom att skrivaren värmer upp de punkter

Figur 2.5: Etikettskrivare GK420d i sin inkapsling.

som skall bli svarta på etiketten. Förutom etikettrullarna finns det inget behov av andra förbrukningsvaror för skrivaren, som exempelvis toner eller bläck, vilket gör den enkel att underhålla.

Skrivaren kopplas till kontrollerenheten med en seriell kabel. Skrivaren monteras in i en inkapsling som är tillverkad av rostfritt stål för att ge bra skydd mot vatten och annat stänk. För att hantera vatten som ändå lyckas ta sig in i inkapslingen så är denna försedd med ett dräneringshål i botten.

Skrivaren klarar av flera programmeringsspråk: EPL2, Line Mode, ZPL I samt ZPL II. För projektet valdes språket ZPL II enligt uppdragsgivarens önskemål. En introduktion till ZPL II ges i bilaga A.

Kapitel 3

Kravspecifikation

Kapitlet går igenom de önskemål som företaget hade på produkten, samt den begränsade kravspecifikation som projektet jobbade mot.

3.1 Kravspecifikation

Vid projektets start så var vågsystemets funktionalitet något obestämd. Unisystem hade en vision om vad själva vågsystemet skulle kunna göra, men hade ännu inte tagit fram några riktlinjer för programvaran som skulle hantera loggerenhetens filer. Den första uppgiften i projektet blev därför att gemensamt ta fram en lista över de egenskaper programmet skulle ha.

3.1.1 Ursprunglig kravspecifikation

Under de första mötena med UniSystem skapade vi en lista över programmets önskade funktioner och egenskaper. Dessa var följande:

Användarvänligt Programmet skulle vara enkelt att hantera även för personer som har liten datorvana.

Rapporthantering Programmet skulle kunna sammanställa och presentera loggdata i en rapport. Denna rapport skulle kunna skrivas ut.

Design av etiketter Programmet skulle låta användaren själv designa etiketternas utseende genom ett grafiskt användargränssnitt.

Uppdatera fiskesorter Programmet skulle kunna uppdatera den lista över fiskesorter som finns på loggerenheten. Denna lista skulle vara begränsad till max 10 arter med högst 10 klasser per art.

Uppdatera fiskeområden Programmet skulle kunna uppdatera den lista över fiskeområden som finns på loggerenheten. Listan skulle vara begränsad till 10 områden med upp till 10 delområden i varje område.

3.1.2 Tillkomna önskemål

Under projektets gång tillkom ytterligare önskemål:

Stöd för fler språk Programmets huvudspråk är engelska. I mån av tid ville Unisystem att programmet skulle översättas till ytterligare två språk, först till svenska och därefter till norska.

Återställning Programmet skulle kunna återställa loggerenhetens filer till grundinställningarna om användaren så önskade.

Informationsfält Programmet skulle ge möjlighet för användaren att spara information kring fiskebåten eller användaren i tre fält. Namnen på informationsfälten skulle gå att ändra enligt användarens önskemål.

3.1.3 Avgränsad kravspecifikation

På grund av projektets tidsram så behövdes kravspecifikationen begränsas. Den slutliga kravspecifikationen som vi jobbade mot är följande:

Användarvänligt Programmet ska vara enkelt att hantera även för personer som har liten datorvana.

Rapporthantering Programmet ska kunna sammanställa och presentera loggdata i en rapport. Denna rapport ska kunna skrivas ut.

Etikettutseende I stället för att användaren kan designa sina egna etiketter så får denne ett urval fördefinierade etikettmallar att välja bland. Detta gör även programmet enklare att använda vilket var ett av de första kraven.

Uppdatera fiskesorter Programmet ska kunna uppdatera den lista över fiskesorter som finns på loggerenheten. Antalet poster i listan ska vara obegränsat.

Uppdatera fiskeområden Programmet ska kunna uppdatera den lista över fiskeområden som finns på loggerenheten. Antalet poster i listan ska vara obegränsat.

Informationsfält Programmet ska ge möjlighet för användaren att spara information kring fiskebåten eller användaren i tre fält. Namnen på informationsfälten ska gå att ändra enligt användarens önskemål.

Kapitel 4

Användargränssnittet

I detta kapitel ges en beskrivning av programmets användargränssnitt. Först ges en översikt till gränssnittets startfönster, därefter beskrivs gränssnittet till programmets olika funktioner.

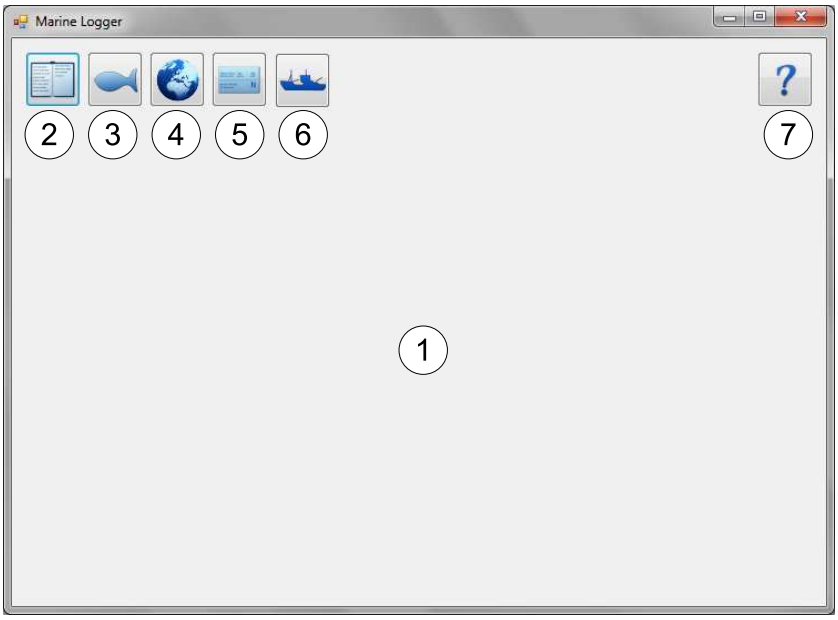
4.1 Översikt

Användargränssnittet gjordes med målet att bli lättanvänt för en person som har liten datorvana. Programmets gränssnitt består av ett enda fönster där ett flertal olika vyer kan visas en åt gången i den undre delen av fönstret (fig.4.1 punkt 1).

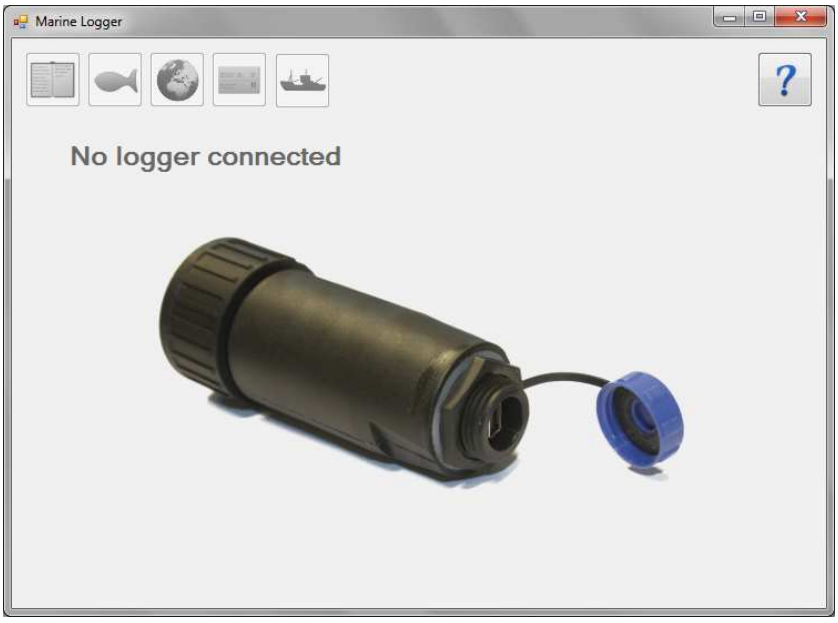
I fönstrets övre vänstra del finns fem knappar där användaren kan välja vilken funktion i programmet som ska användas. Varje knapp är försedd med en bild som representerar dess funktion. När användaren klickar på en av dessa knappar visas motsvarande vy. När någon av vyerna visas blir motsvarande knapp svagt blåfärgad för att indikera att funktionen är aktiv.

Längst uppe till höger i fönstret finns hjälpknappen som är märkt med ett frågetecken (fig.4.1 punkt 7). När man klickat på denna öppnas online-manualen i systemets webbläsare.

När programmet startats söker den efter en ansluten loggerenhet. Skulle en sådan ej



Figur 4.1: Programmets huvudfönster. 1) Område där vald vy visas. Knappar för: 2) Rapport, 3) Fisksorter, 4) Fiskeområden, 5) Etikettval, 6) Statisk information, 7) Onlinehjälp.



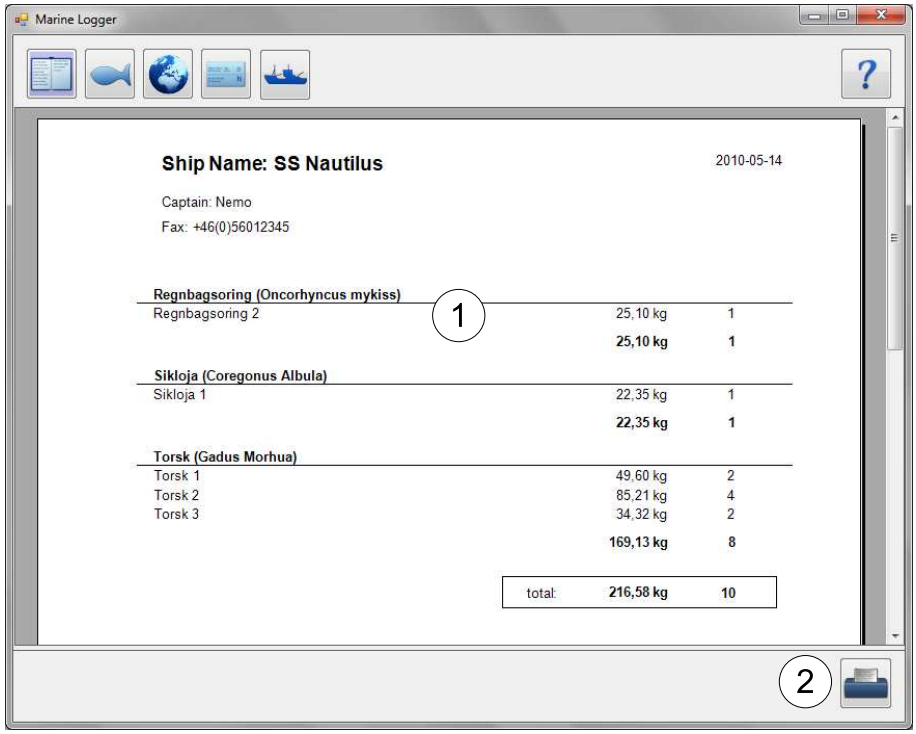
Figur 4.2: Programmets huvudfönster utan ansluten loggerenhet.

hittas, eller om loggern kopplas ur datorn medan programmet är igång avaktiveras alla knappar utom hjälp-knappen och en ny vy öppnas där programmet beskriver att ingen logger är ansluten (se figur 4.2).

4.2 Vyer

De olika vyerna fungerar som vanliga fönster med skillnaden att de visas som en del av huvudfönstret i stället för att hamna som egna fönster. I denna del kommer vi att gå igenom de olika vyerna och deras funktioner.

4.2.1 Rapport-vyn

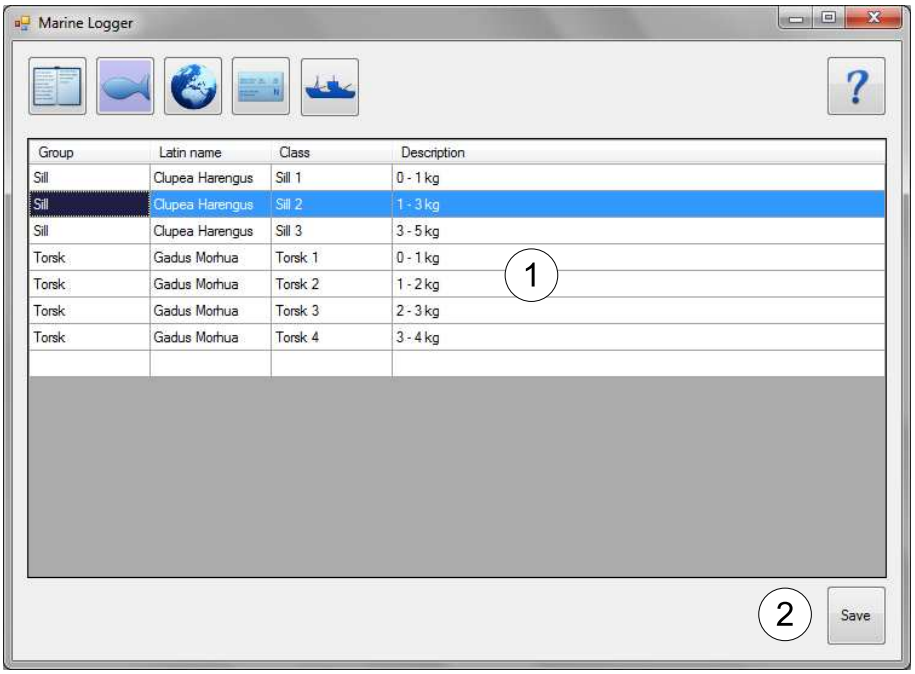


Figur 4.3: Rapportvyn. 1) Rapporten, 2) Knapp för utskrift

Rapportvyn (fig. 4.3) sammanställer en rapport över fångsten, baserad på loggfilens innehåll och den information som användaren sparat på loggern via informationsvyn (sektion 4.2.5).

Rapporten presenteras som den skulle vara skriven på ett A4-papper. I vyns nedre högra hörn finns en knapp för att skriva ut rapporten (fig. 4.3 punkt 2). När man klickar på denna får man upp systemets standarddialog för utskrift.

4.2.2 Fisksorts-vyn



Figur 4.4: Vyn för editering av fisksorter. 1) Tabell över fisksorter, 2) Knapp för att spara ändringar.

I denna vy kan användaren editera den tabell som innehåller informationen om de fiskarter användaren planerat att fånga. Eftersom olika fiskarter fångas i olika områden eller med viss utrustning, har användaren här möjlighet att anpassa vilka arter som det ska gå att välja på via kontrollerenheten. Detta gör det snabbare för användaren att använda

kontrollerenheten eftersom han slipper bläddra förbi arter som är oväsentliga. Här kan användaren lägga till nya arter och viktklasser om så önskas.

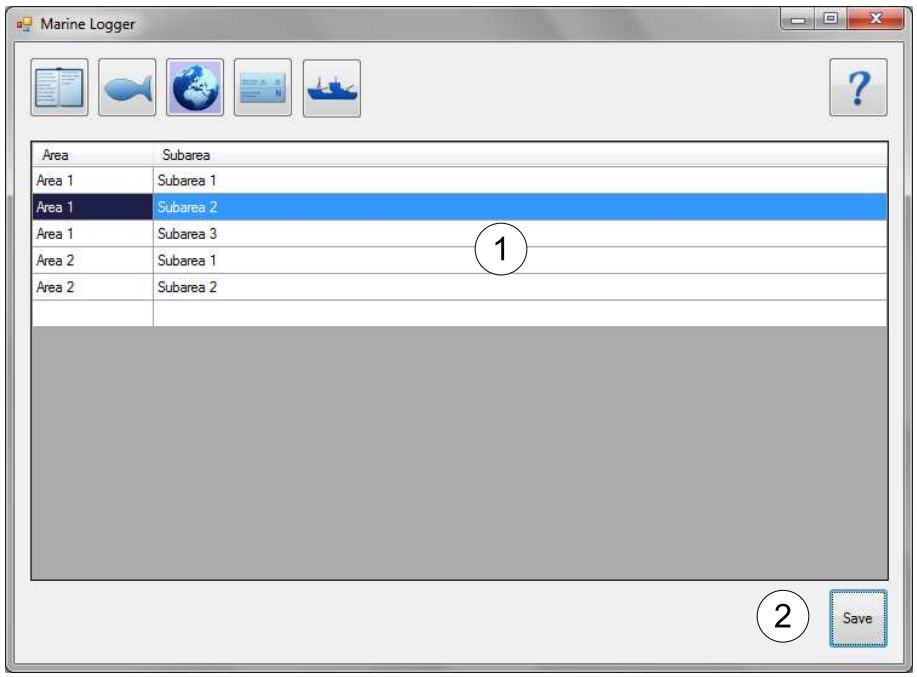
Tabellen (figur 4.4 punkt 1) innehåller fyra kolumner: Den första kolumnen är fiskartens namn på användarens egna språk. Den andra kolumnen är fiskartens latinska namn. Den tredje kolumnen innehåller arternas olika viktklasser. Den fjärde och sista kolumnen innehåller beskrivningen av viktklassen, till exempel '3-5 kg'.

I tabellen kan man bara välja en hel rad i taget. Den cell som är aktiv i den valda raden visas i mörkare blått. För att redigera en cells värde markerar man först den cell man vill ändra genom att klicka på cellen eller flytta sig till cellen med piltangenterna, därefter kan man börja skriva in det nya innehållet. För att ta bort vald rad trycker man på delete-knappen på tangentbordet. För att lägga till en rad i tabellen skriver man in nya värden på den tomma raden och trycker på 'Save'-knappen (figur 4.4 punkt 2).

Tabellens rader sorteras automatiskt när man gör en förändring eller sparar tabellen. Sorteringen sker med stigande värden på alla fyra kolumnerna i ordningen grupp, latinskt namn, gruppnamn och gruppbeskrivning.

Man sparar ändringar som gjorts i tabellen genom att klicka på knappen 'Save' som finns längst ner till höger i vyn (figur 4.4 punkt 2).

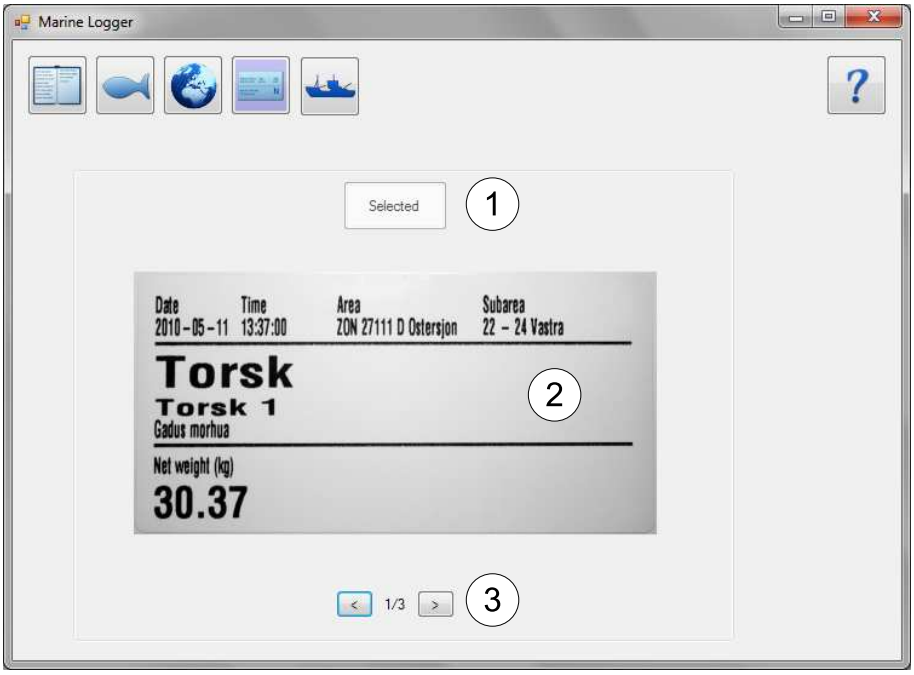
4.2.3 Områdes-vyn



Figur 4.5: Vyn för editering av fiskeområden. 1) Tabell över fiskeområden, 2) Knapp för att spara ändringar.

I denna vy så kan användaren editera den tabell som innehåller informationen om de områden och deras respektive delområden som finns inlagt på loggern. Tabellen innehåller två kolumner: Area och subarea, eller område och delområde som det kommer benämnas i en framtida svensk version. Arbetssätt och utseende är i övrigt identiskt med fisksorts-vyn som beskrevs i föregående sektion [4.2.2].

4.2.4 Etikett-vyn



Figur 4.6: Vyn för etikettval. 1) Knapp för att välja etikettutseende, 2) Bild på etiketten, 3) Bläddringsknappar.

Olika användare kan tänkas ha behov av olika utseenden på sina etiketter. Därför ger programmet användaren möjlighet att välja utseende på hur etiketterna ska se ut vid utskrift. Detta sker på vyn för etikettval.

I denna vy får användaren möjlighet att ställa in vilket utseende som etiketterna ska ha när den aktuella loggerenheten används i kontrollerenheten. Den fil som styr utseendet på etiketterna kallas för en etikettmall. Valet av etikettmall sker från ett antal fördefinierade mallar som försetts med en bildrepresentation.

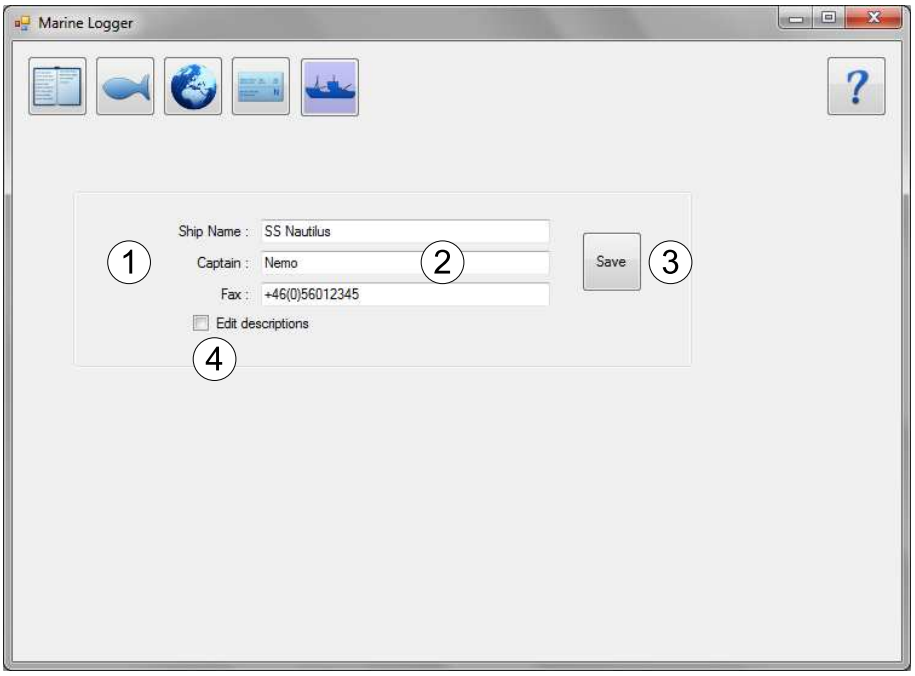
Som visas i figur 4.6 består etikettvyn av tre delar.

1. Knapp för att välja etikettutseende. Om den etikettmall som visas är den som är vald som på den anslutna loggern är knappen vit och har texten ‘Selected’, annars

är knappen grå och har texten 'Select'. När användaren klickar på knappen sparas etikettmallen som representeras av bilden ner till loggern.

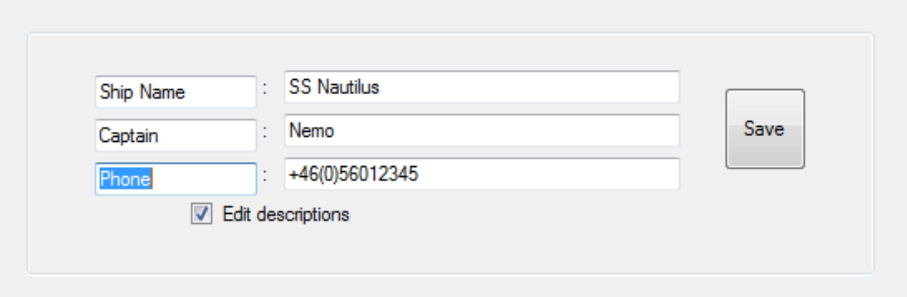
2. **Bild på etiketten** I mitten av vyn presenteras en bild på den etikett som man bläddrat till. När vyn öppnas för första gången visas den nuvarande valda etiketten först.
3. **Bläddringsknappar** Längst ner i vyn ser vi två knappar för att bläddra mellan etiketterna, samt en text som visar vilken etikett i turordningen som nu visas och hur många etiketter som går att välja på totalt.

4.2.5 Informations-vyn



Figur 4.7: Informationsvyn. 1) Fältbeskrivningar, 2) informationsfält, 3) Spara-knapp, 4) Checkruta för editering av fältbeskrivningarna.

Användarna har behov av att kunna spara information på loggern som gör den personlig. Av den anledningen har programmet en informations-vy för detta ändamål. Eftersom användarna kan tänkas ha behov av att spara olika sorters information på loggern har



Figur 4.8: Informationsvyn. Här visas hur fältbeskrivningarna kan editeras.

möjligheten lagts till för att kunna editera inte bara fälten, utan även beskrivningen av fälten. Denna information används inte ombord på fiskebåtarna, utan är till för att kunna identifiera en logger när man är i land. Informationen visas även i sidhuvudet på rapporterna (se figur 4.3).

Som vi kan se i figur 4.7 har vyn fyra huvuddelar: 1.Fältbeskrivningarna, som i bilden ej är ändringsbara. 2. Informationsfälten. 3. Knappen för att spara ändringar, 4. Checkruta för att tillåta att man uppdaterar fältbeskrivningarna.

När användaren kryssar i checkrutan blir fältbeskrivningarna ändringsbara och texten i dessa vänsterställs (se figur 4.8). Användaren kan då mata in en beskrivning på textfältet till höger. Beskrivningen kan innehålla alla tecken utom kolon, semikolon och enter. När användaren är klar med ändringen kan han klicka i checkrutan en gång till för att åter igen göra fälten skrivskyddade.

För att spara ändringar i beskrivningarna eller i informationsfälten klickar man på knappen ‘Save’ (punkt 3 i figur 4.7). Om checkrutan är ikryssad när man sparar kommer denna att göras okryssad igen.

Kapitel 5

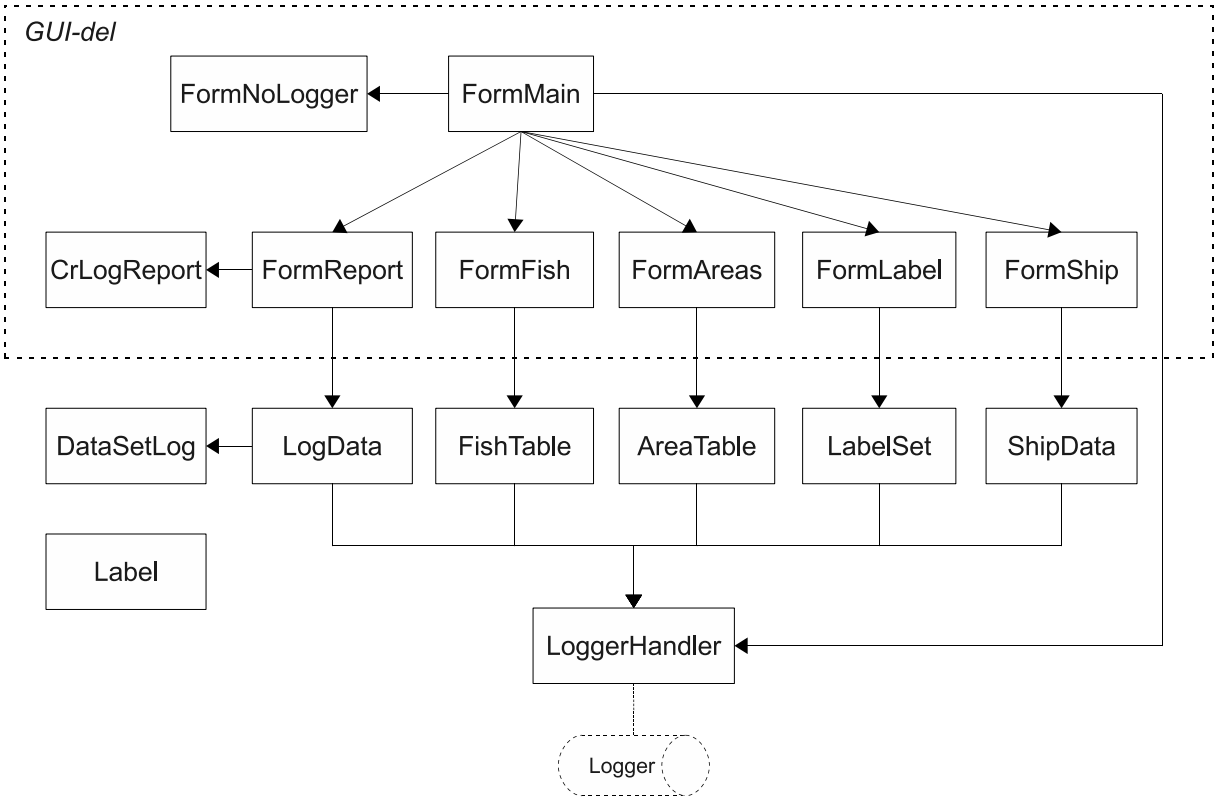
Programbeskrivning

I detta kapitel ser vi närmare på hur programmet fungerar bakom användargränssnittet. Först ges en översikt till programmets programstruktur och klassernas olika funktionsområden, därefter följer en översikt på hur informationen färdas i systemet. Slutligen ges en mer ingående beskrivning på hur programmet hanterar rapporter, tabellerna för fisk och fiskeområden, etikettmallarna samt datat i informations-vyn.

5.1 Programöversikt

Programmet är uppdelat i tre funktionsområden. Vi har det grafiska gränssnittet, de klasser som hanterar behandlingen av informationen som finns på loggern och slutligen loggerhanteraren som sköter all kontakt med loggern. Figur 5.1 visar ett förenklat diagram över programmets struktur.

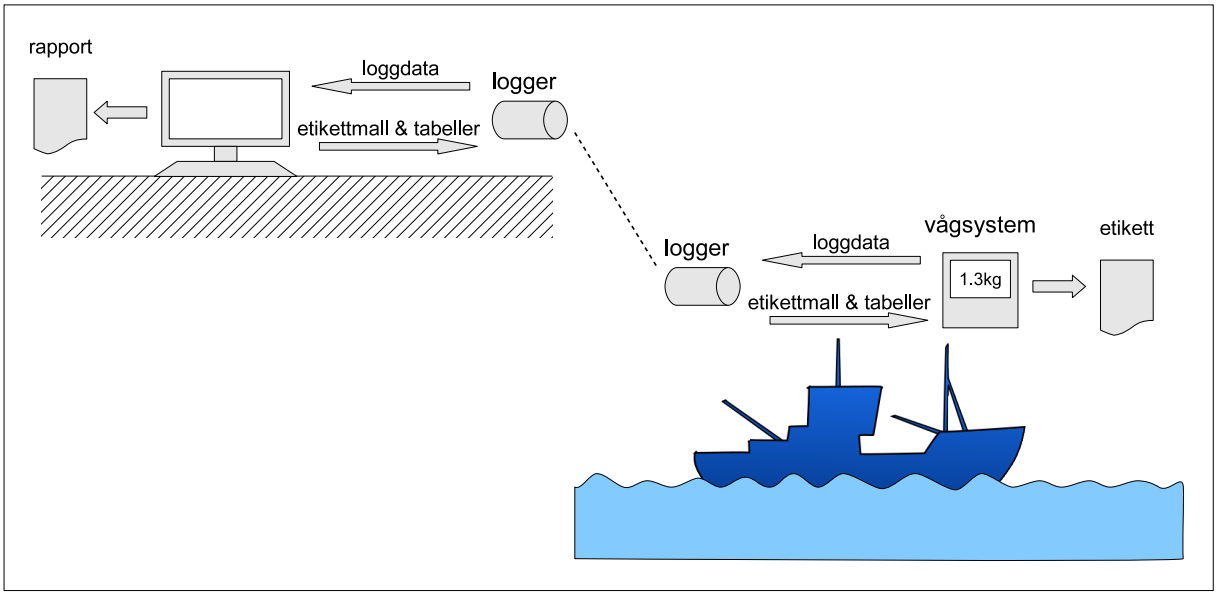
Loggerhanteraren (LoggerHandler) har, förutom all filhantering, även till uppgift att söka rätt på loggern och rapportera förändringar kring anslutna loggerenheter till de klasser som är intresserade, vilket i detta fall innebär själva huvudfönstret för programmet. Denna koppling visas i figur5.1 av pilen mellan *FormMain* och *LoggerHandler*. Klassen är statisk vilket innebär att klassens funktioner anropas utan att man skapar en instans av klassen.



Figur 5.1: Programstruktur. Den övre sektionen inramad med en streckad linje visar de delar som är direkt knutna till användargränssnittet.

Klassen Label används enbart som en grundläggande datastruktur i klasserna LoggerHandler och LabelSet. Klassen används för att lagra information om etiketterna.

5.2 Informationsflöde



Figur 5.2: Informationsflödet.

Figur 5.2 ger en översikt på hur informationen rör sig i i systemet. Informationen flyttas mellan dator och vågsystemet via loggerenheten som fysiskt flyttas fram och åter.

I land

När loggern är ansluten till datorn med programmet, kan loggfilen användas till att skapa en rapport och tabellerna på loggerenheten kan anpassas efter användarens önskemål. Där kan även etikettmallen väljas.

Till havs

När loggern kopplats till vågsystemet ombord på fiskebåten förs tabellerna och etikettmallen över till kontrollenheten. Etikettmallen, tillsammans med värden som man har valt från tabellerna, används för att skriva ut etiketter. Vid varje vägning sparas sedan loggdatat ner till loggerenheten.

5.3 Rapporthantering

5.3.1 Crystal Reports

För att skapa rapporten i rapportvyn användes *Crystal Reports*[4], ett verktyg som kan användas för att skapa rapporter utifrån en mängd olika datakällor. En version av verktyget ingår i utvecklingsmiljön *Microsoft Visual Studio .NET 2008*[3] och det är denna version som användes i projektarbetet.

Rapportfilen, den fil som hanterar utseendet på rapporten, kan antingen hanteras som en vanlig fil, som i så fall kan uppdateras utan att göra ändringar i programmet. Alternativt kan rapportfilen hanteras som en resurs, vilket innebär att den då kommer att sparas som en del av programmet. Vi valde den senare lösningen eftersom rapporten då alltid följer med programmet och rapportutseendet är skyddat för förändringar.

5.3.2 Klassbeskrivningar

För rapporthanteringen används fem klasser: det grafiska gränssnittet (FormReport), ett dataset (DataSetLog) som innehåller en datatabell, en klass (CrLogReport) som hanterar användningen av Crystal Reports dokumentet, loggdata-hanteraren (LogData) och slutligen loggerhanteraren (LoggerHandler) som sköter filhanteringen på loggerenheten. Figur 5.1 visar hur de olika klasserna hänger samman i systemet.

FormReport är klassen för det grafiska gränssnittet för rapportvyn. Utseendet på rapportvyn beskrevs i sektion 4.2.1.

CrLogReport är den klass som har hand om själva Crystal Reports dokumentet. Detta dokument presenterar tabellen i DataSetLog.

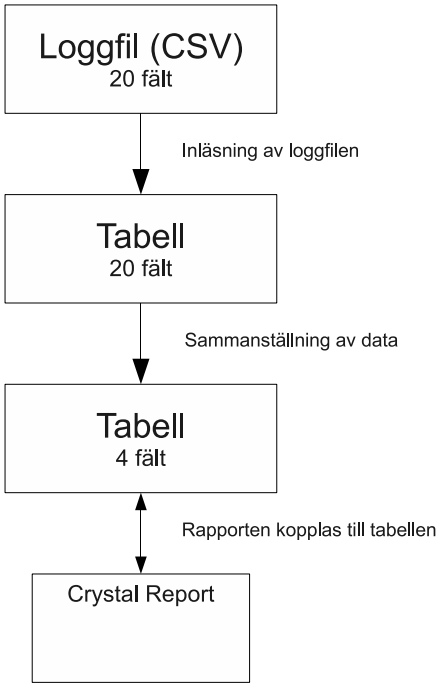
DataSetLog är ett dataset, en klass som innehåller en eller flera datatabeller. I detta fall innehåller DataSetLog bara en enda tabell. Denna datatabell används för att hålla de värden som ska visas i rapportdokumentet.

LogData är den klass som agerar mellanhand till LoggerHandler. Klassen har till uppgift att hantera inläsningen av loggfilen och sammanställa dess värden till tabellen i DataSetLog. Denna tabell kan sedan hämtas av FormReport och användas som datakälla till rapporten.

LoggerHandler Loggerhandler är den klass som har hand om filhanteringen på loggerenheten. All läsning (och skrivning) av filer går via denna klass.

5.3.3 Processer

Skapande av rapport från logg-data



Figur 5.3: En förenklad modell över hur sammanställningen av rapporten går till.

Rapporten som presenteras i rapport-vyn visar en sammanställning av datat som sparats till loggerenheten. När kontrollerenheten sparar viktdata till loggern, sparar den även en mängd information som inte används av programmet, till exempel vågens lutning.

Anledningen till all extra information är att kunna utöka programmet utan att förändra kontrollerenhetens funktion. Kontrollerenheten sparar loggfilen som en semikolon-separerad fil (CSV) med 20 kolumner. Semikolon-separerade filer används ofta för att spara tabeller som text. Varje rad i tabellen motsvarar en rad i textfilen, och varje kolumn avgränsas med ett semikolon. Benämningen CSV används både för filer som använder semikolon och de som använder komma-tecken som fältavgränsare.

Rapporten skapas i flera steg. I steg ett läses loggfilen in till en tabell, där varje rad i loggfilen hamnar som en rad i tabellen. I denna tabell finns all information från varje vägning som loggats till filen. I steg två sammanställs informationen i från den första tabellen till en ny tabell. Sammanställningen sker genom att alla förekomster av en art tillsammans med gruppering slås samman till en rad i den nya tabellen tillsammans med summan av viktvärdena för gruppen, samt antalet rader som användes för framställningen av viktsumman. Artnamnet slås samman med det latinska namnet i den nya tabellen. I steg tre används den nya tabellen därefter som datakälla till det Crystal Reports-dokument som används på rapport-vyn. De fältnamn och värden som användaren har sparat till loggern via informations-vyn skrivs även de in i rapportens sidhuvud. Efter detta är rapporten klar att presentera för användaren.

5.4 Fisktabellshantering

5.4.1 Klassbeskrivningar

Fisktabellen hanteras i tre olika klasser. Vi har en klass för det grafiska gränssnittet (FormFish), vi har tabellhanteraren (FishTable) och slutligen loggerhanteraren (LoggerHandler) som läser och skriver filen.

FormFish är klassen som har hand om det grafiska gränssnittet för fisktabellen. Utseendet på denna beskrevs i sektion 4.2.2.

FishTable Denna klass agerar som en mellanhand mellan det grafiska gränssnittet och loggerhanteraren. Denna klass tillåter att man kommer åt tabellen och kan spara eventuella förändringar.

LoggerHandler Loggerhandler är den klass som har hand om filhanteringen på loggerenheten. All skrivning och läsning av filer går via denna klass.

Figur 5.1 visar hur de olika klasserna hänger samman i systemet.

5.4.2 Beskrivning av fisktabellen

Tabellen över fisk består av fyra kolumner som innehåller:

1. Fiskens artnamn på användarens lokala språk.
2. Fiskens latinska namn.
3. Viktgruppens namn.
4. Viktgruppens beskrivande text.

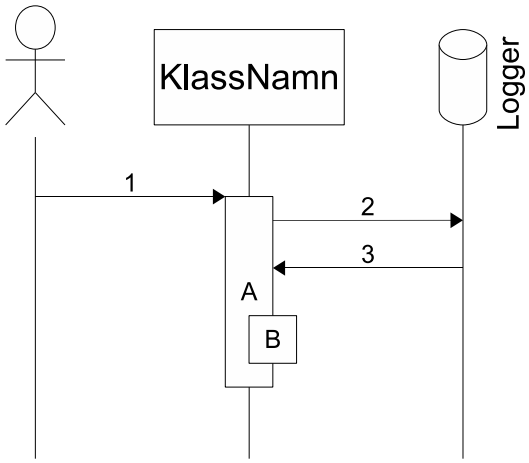
Tabellen sparas på loggerenheten som en semikolon-separerad fil med namnet *fish.csv*. Varje rad i tabellen motsvarar en rad i textfilen, och kolumnerna avgränsas med ett semikolon. Ett utdrag på några rader i filen för fisktabellen kan se ut såhär:

```
Sill;Clupea Harengus;Sill 1;3 - 5 kg
Sill;Clupea Harengus;Sill 2;1 - 3 kg
Sill;Clupea Harengus;Sill 3;0 - 1 kg
Torsk;Gadus Morhua;Torsk 1;5 - 8 kg
Torsk;Gadus Morhua;Torsk 2;3 - 4 kg
Torsk;Gadus Morhua;Torsk 3;1 - 2 kg
```

5.4.3 Processer

Beskrivning av sekvensdiagrammen

I denna sektion kommer vi att beskriva vissa processer med hjälp av sekvensdiagram. Som namnet antyder så beskriver ett sekvensdiagram en sekvens av händelser.



Figur 5.4: Exempel på hur ett sekvensdiagram kan se ut.

I figur 5.4 ger vi ett exempel på hur våra sekvensdiagram ser ut. I exemplet ser vi följande delar:

Roller Överst i sekvensdiagrammen visas symboler för de aktiva rollerna i diagrammet. I figur 5.4 består dessa av: en användare som representeras av en streckgubbe, en instans av klassen *KlassNamn* som beskrivs med en rektangel och slutligen log-

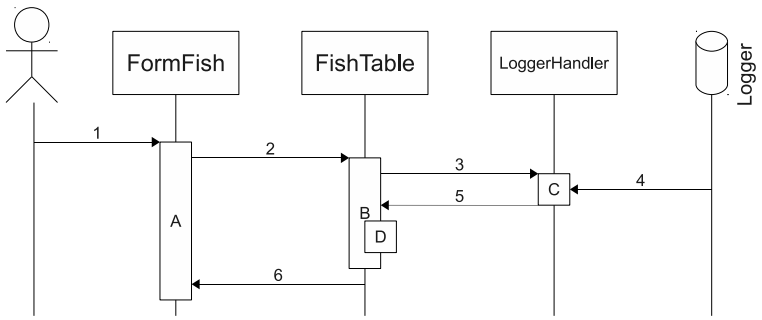
gerenheten som representeras av en stående cylinder vilket innebär att den är en lagringsenhet.

Vertikala linjer Under varje part i diagrammet finns en linje. Denna linje representerar tidsflödet, från starten högst upp, till slutet längst ner.

Pilar Pilarna som går mellan de olika rollerna i diagrammet representerar meddelanden eller informationsflöde. Pil 1 i figuren går från användaren till KlassNamn, vilket indikerar att användaren utför en handling vilket i detta fall får processen A att starta. Pil 2 går mellan KlassNamn och Logger, vilket representerar en skrivning av data till Logger eftersom denna är en fysisk lagringsenhet. Pil 3 går åt motsatt håll, från Logger till KlassNamn vilket innebär att KlassNamn läser data ifrån Logger.

Rutor Rutorna på linjerna representerar en funktion. *A* är den funktion som startas på grund av meddelandet i pil 1, medan *B* som befinner sig på rutan *A* är en intern funktion eller annan händelse av intresse i klassen som startats av A.

Öppnande av fisksorts-vyn



Figur 5.5: Sekvensdiagram över öppnandet av fisksorts-vyn.

Innan programmet kan visa vyn över fisksorter måste ett antal steg genomföras:

1. Användaren klickar på knappen för att öppna fisksorts-vyn vilket anropar funktionen FormFish_Load.

A. Funktionen FormFish_Load startas.

2. FormFish_Load skapar en instans av klassen FishTable.

B. Konstruktorn i FishTable startas. En tom datatabell med fyra kolumner skapas.

3. FishTable anropar ReadFish i LoggerHandler.

C. Funktionen ReadFish startar.

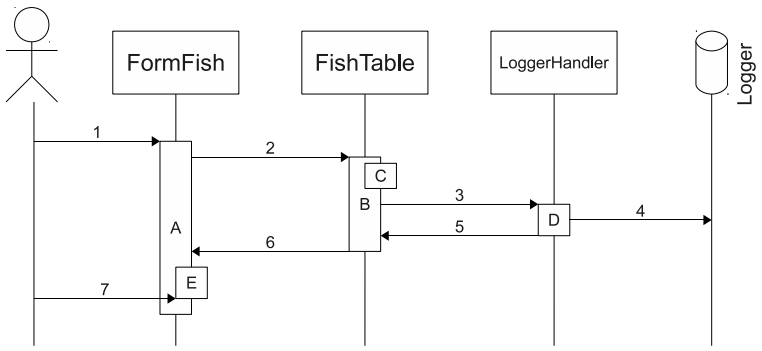
4. ReadFish hämtar innehållet från fisktabells-filen på loggerenheten.

5. Innehållet i fisktabells-filen returneras som en textsträng.

D. Strängen delas upp i rader och kolumner och sparas till datatabellen som skapades i B. Därefter är Konstruktorn för FishTable klar.

6. FormFish hämtar datatabellen i FishTable och länkar denna till datatabellen i användargränssnittet (se punkt 1 i sektion 4.4). När detta är klart är fisksorts vyn färdig att visas. Alla förändringar som användaren därefter gör på tabellen via det grafiska gränssnittet, återspeglas i datatabellen i FishTable eftersom de är länkade.

Spara fisksorts-tabellen



Figur 5.6: Sekvensdiagram över sparandet av fisksorts-tabellen.

När användaren vill spara sina ändringar sker även detta i flera steg:

1. Användaren klickar på knappen för att spara tabellen, funktionen BtnSave_Click i FormFish anropas.

A. Funktionen BtnSave_Click i FormFish startas.

2. BtnSave_Click anropar funktionen SaveTable i FishTable.

B. Funktionen SaveTable i FishTable startar.

C. SaveTable skapar en sorterad vy av datatabellen och bygger sedan, rad för rad, upp en semikolonseparerad textsträng som representerar den sorterade tabellen.

3. SaveTable anropar funktionen WriteFish i LoggerHandler.

D. Funktionen WriteFish startar.

4. LoggerHandler ersätter innehållet i filen *fish.csv* på loggerenheten, med den csv-formaterade textsträngen.

5. WriteFish returnerar om skrivningen lyckades till SaveTable.

6. SaveTable returnerar om skrivningen lyckades till BtnSave_Click.

E. BtnSave_Click visar en dialogruta som talar om det gick bra att spara tabellen.

7. Användaren bekräftar dialogrutan.

5.5 Områdeshantering

5.5.1 Klassbeskrivningar

Tabellen över fiskeområden hanteras på samma sätt som fisksorts-tabellen, det vill säga i tre olika klasser. Vi har en klass för det grafiska gränssnittet (FormAreas), vi har tabellhanteraren (AreaTable) och slutligen loggerhanteraren (LoggerHandler) som läser och skriver filen.

FormArea är klassen som har hand om det grafiska gränssnittet för områdestabellen. Utseendet på denna beskrevs i sektion 4.2.3.

AreaTable Denna klass agerar som en mellanhand mellan det grafiska gränssnittet och loggerhanteraren. Denna klass tillåter att man kommer åt tabellen och kan spara eventuella förändringar. Klassen hanterar även omvandling av textsträng i CSV-format till datatabell och vice versa.

LoggerHandler Loggerhandler är den klass som har hand om filhanteringen på loggerenheten. All skrivning och läsning av filer går via denna klass.

Figur 5.1 visar hur de olika klasserna hänger samman i systemet.

5.5.2 Beskrivning av områdestabellen

Tabellen över fisk består av två kolumner som innehåller:

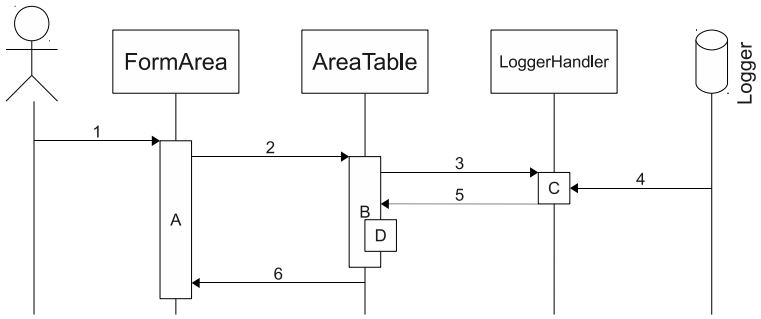
1. Namnet på huvudområdet
2. Namnet på delområdet

Tabellen sparas på loggerenheten som en semikolon-separerad fil med namnet *area.csv*. Namnen på områden och delområden kan innehålla alla tecken utom semikolon. Filen har följande format:

```
Område 1;Delområde 1
Område 1;Delområde 2
Område 2;Delområde A
Område 2;Delområde B
Område 2;Delområde C
```

5.5.3 Processer

Öppnande av områdes-vyn



Figur 5.7: Sekvensdiagram över öppnandet av områdes-vyn.

Hanteringen av fiskeområden är i stort sett identisk med hanteringen av fisksorter. Innan programmet kan visa vyn över fiskeområden måste ett antal steg genomföras:

1. Användaren klickar på knappen för att öppna områdes-vyn, funktionen FormArea_Load anropas.

A. Funktionen FormArea_Load startas.

2. FormArea_Load skapar en instans av klassen AreaTable.

B. Konstruktorn i AreaTable startas. En tom datatabell med två kolumner skapas.

3. AreaTable anropar ReadAreas i LoggerHandler.

C. Funktionen ReadAreas startar.

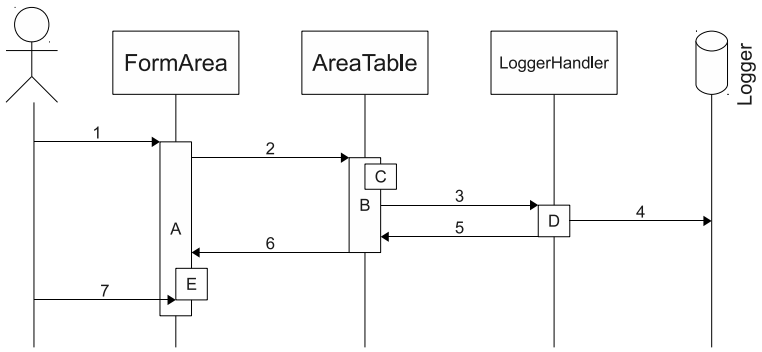
4. ReadAreas hämtar innehållet ifrån områdes-filen på loggerenheten.

5. Innehållet i områdes-filen returneras som en textsträng.

D. Strängen delas upp i rader och kolumner och sparas till datatabellen som skapades i B. Därefter är konstruktorn för AreaTable klar.

6. FormArea hämtar datatabellen i AreaTable och länkar denna till datatabellen i användargränssnittet (se punkt 1 i figur 4.5). När detta är klart är områdes-vyn färdig att visas.

Spara områdes-tabellen



Figur 5.8: Sekvensdiagram över sparandet av fiskeområdes-tabellen.

1. Användaren klickar på knappen för att spara tabellen, funktionen BtnSave_Click anropas.

A. Funktionen BtnSave_Click i FormArea startas.

2. BtnSave_Click anropar funktionen SaveTable i AreaTable.

B. Funktionen SaveTable i AreaTable startar.

C. SaveTable skapar en sorterad vy av datatabellen och bygger sedan, rad för rad, upp en semikolonseparerad textsträng som representerar den sorterade tabellen.

3. SaveTable anropar funktionen WriteAreas i LoggerHandler.

D. Funktionen WriteAreas startar.

4. LoggerHandler skriver över filen *areas.csv* på loggerenheten, med den csv-formaterade textsträngen.

5. WriteAreas returnerar om skrivningen lyckades till SaveTable.

6. SaveTable returnerar om skrivningen lyckades till BtnSave_Click.

E. BtnSave_Click visar en dialogruta som visar om det gick bra att spara tabellen eller inte.

7. Användaren bekräftar dialogrutan.

5.6 Etiketthantering

5.6.1 Klassbeskrivningar

Etiketthantering sker i fyra olika klasser. Klassen för det grafiska gränssnittet (FormLabel), klassen för hantering av etikettdata (LabelSet), klassen som representerar etiketterna (Label) och slutligen klassen som arbetar mot loggerenheten (LoggerHandler).

FormLabel - FormLabel är den grafiska klassen som tar hand om uppvisning av etiketterna, samt låter användaren bläddra igenom alla tillgängliga etikettmallar och välja vilken av dessa som ska användas vid utskrift. Beskrivning av gränssnittet finns i sektion 4.2.4.

LabelSet - LabelSet är den klass som fungerar som en mellanhand mellan LoggerHandler och användargränssnittet. Den har hand om hantering av data som skickas mellan LoggerHandler och användargränssittet.

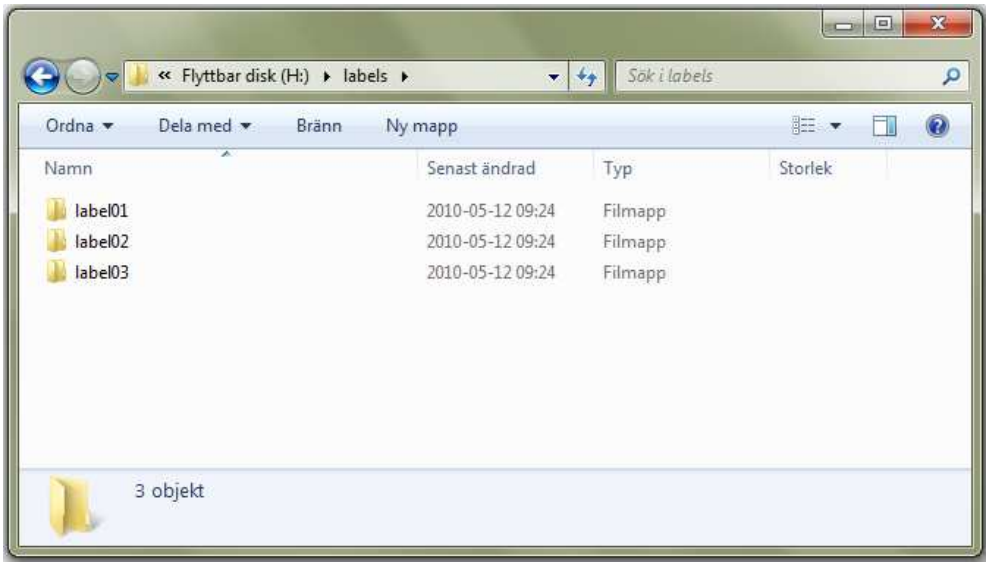
Label - Label är den klass som representerar etiketter. Den har två egenskaper, en som lagrar en bild och en som lagrar en textsträng.

LoggerHandler - LoggerHandler är den klass som arbetar direkt mot loggerenheten, den både skriver och läser bilder samt ZPL II-kod emot loggerenheten.

Hur klasserna hänger ihop kan man se i figur 5.1.

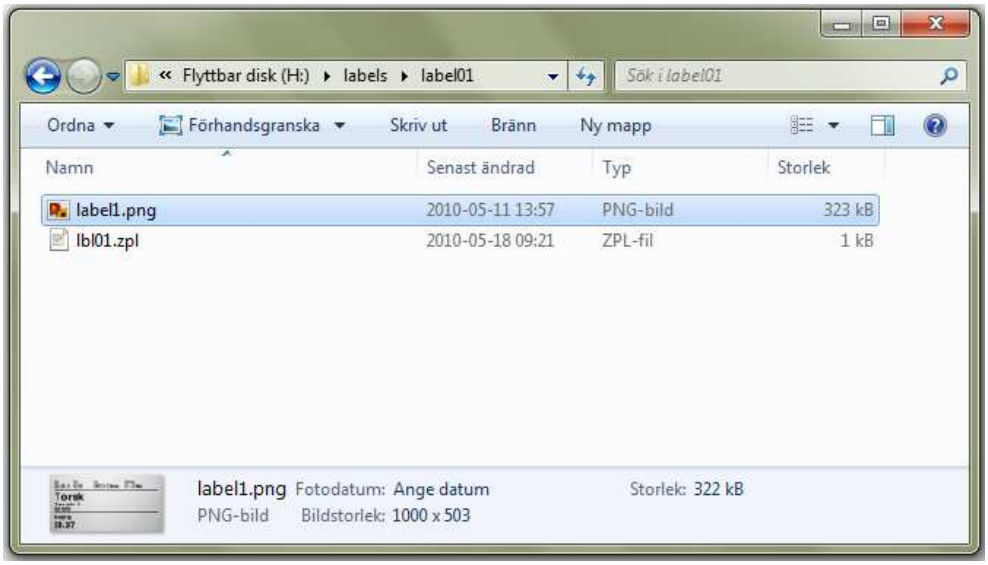
5.6.2 Beskrivning av filer och kataloger

De filer som styr etiketternas utseende kallas för etikettmallar. Dessa är textfiler som innehåller ZPL II-kod och har filändelsen *.zpl*. I loggerenhetens rotkatalog finns en fil med namnet *selectedlabel.zpl*, det är denna fil som innehåller koden för den etikett som är vald som den aktuella etikettmallen. Katalogen *labels*, som också ligger i loggerenhetens rotkatalog, innehåller ett flertal kataloger. Var och en av dessa kataloger representerar en etikett och kommer att hänvisas som etikettkataloger hädanefter.



Figur 5.9: Filstrukturen i katalogen *labels*.

Varje etikettkatalog innehåller en bild och en textfil. Textfilen innehåller ZPL II-koden för etikettmallen och bilden visar ett exempel på en utskriven etikett. Bilden är i formaten PNG (Portable Network Graphics).

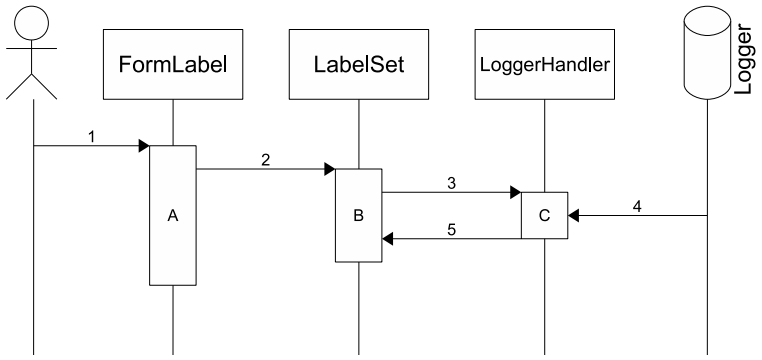


Figur 5.10: Filstrukturen i en av etikettkatalogerna.

5.6.3 Processer

Öppnande av etikett-vyn

Öppnandet av etikett-vyn startar funktionen FormLabel_Load, i denna funktion genomförs tre steg innan vyn kan visas. Dessa beskrivs i de tre följande sekvensdiagrammen.



Figur 5.11: Sekvensdiagram då användaren klickar på knappen för att visa etikettvy.

Följande lista beskriver sekvensdiagrammet i figur 5.11

1. Användaren klickar på knappen för att visa upp etikettvyn och funktionen FormLabel_Load i klassen FormLabel anropas.

A. Funktionen FormLabel_Load startas.

2. FormLabel_Load skapar en instans av klassen LabelSet.

B. Konstruktorn i LabelSet startar och en tom lista skapas som kommer innehålla objekt av klassen Label. Denna lista kommer att refereras till med namnet listOfLabel i följande sekvensdiagram.

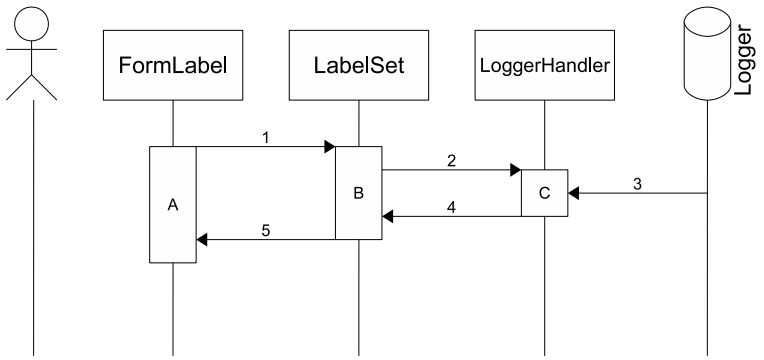
3. LabelSet anropar funktionen ReadLabels i LoggerHandler.

C. Funktionen ReadLabels i LoggerHandler startar och skapar en lista som ska innehålla objekt av klassen Label.

4. ReadLabels läser in etikettbilden och etikettmallen en etikettkatalog i taget. För varje etikettkatalog som har fått sina filer inlästa skapas ett objekt av klassen Label som

innnehåller etikettbilden och etikettmallen. Därefter läggs objektet till i listan. Detta upprepas för varje etikettkatalog.

5. LoggerHandler returnerar listan med objekten till klassen LabelSet där listan listOfLabel blir tilldelad denna lista.



Figur 5.12: Sekvensdiagram då FormShip vill läsa in index för den valda etikettmallen.

Finna index för den valda etikettmallen

Följande lista beskriver sekvensdiagrammet i figur 5.12 som är en fortsättning på föregående sekvensdiagram.

1. FormLabel_Load anropar funktionen ReadSelectedLabel i LabelSet.

B. Funktionen ReadSelectedLabel i LabelSet startas.

2. ReadSelectedLabel anropar i sin tur funktionen ReadSelectedLabel i LoggerHandler.

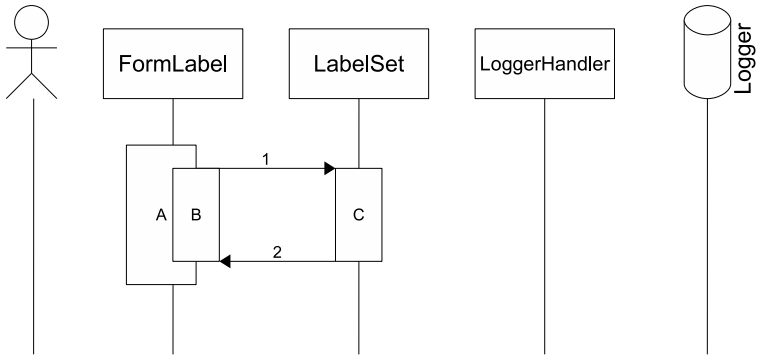
C. ReadSelectedLabel startas i LoggerHandler.

3. ReadSelectedLabel läser in innehållet från den aktuella filen *selectedlabel.zpl* i loggerenheten.

4. ReadSelectedLabel returnerar innehållet till ReadSelectedLabel i LabelSet.

5. ReadSelectedLabel returnerar indexet för det objekt i listOfLabels som matchar filen *selectedlabel.zpl*.

Visning av bild för vald etikettmall



Figur 5.13: Sekvensdiagram då FormShip vill visa den motsvarande bilden till indexet.

Följande lista beskriver sekvensdiagramet i figur 5.13 som är en fortsättning på föregående sekvensdiagram.

A. Efter att det aktuella indexet hämtats från LabelSet, anropar FormLabel_Load den interna funktionen ShowLabel.

B. ShowLabel startar.

1. ShowLabel anropar funktionen ReturnLabel i LabelSet och skickar det aktuella indexet.

C. ReturnLabel startas och tar emot indexet.

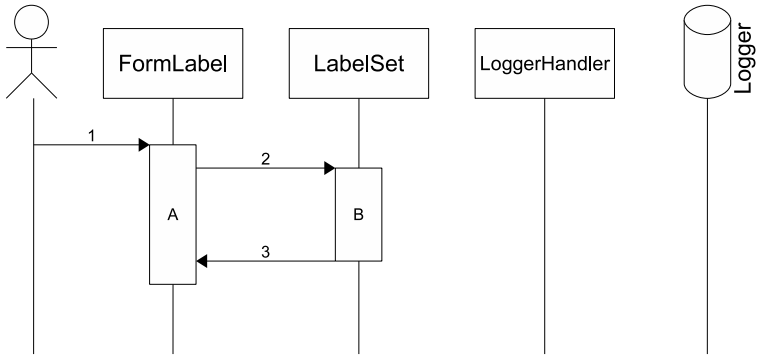
2. ReturnLabels returnerar den bild från det objekt som finns på indexet den tog emot i listan listOfLabels.

B. ShowLabel tar emot bilden och visar upp den i användargränssnittet. Efter detta är klart visas vyn för etiketthantering.

Bläddrande bland etiketter

Följande lista beskriver sekvensdiagramet i figur 5.14. Detta sekvensdiagram visar vad som sker då användaren klickar på någon av de knappar som finns i gränssnittet för etikettval.

1. Användaren klickar antingen på knappen för att gå framåt eller bakåt i listan med



Figur 5.14: Sekvensdiagram då användaren klickat på någon av knapparna för att gå vänster eller höger bland bilderna av etiketterna.

bilder. Beroende på vilken knapp man klicka på anropas funktionen GoRight eller GoLeft.

A. Funktionen GoRight eller GoLeft startas och det aktuella indexet blir uppdaterad, därefter anropas funktionen ShowLabel.

B. Funktionen ShowLabel i FormLabel startas.

2. ShowLabel anropar funktionen ReturnLabel i klassen LabelSet och skickar det aktuella indexet.

B. ReturnLabel startas och tar emot indexet.

3. ReturnLabels returnerar den bild från det objekt som finns på indexet den tog emot i listan listOfLabels.

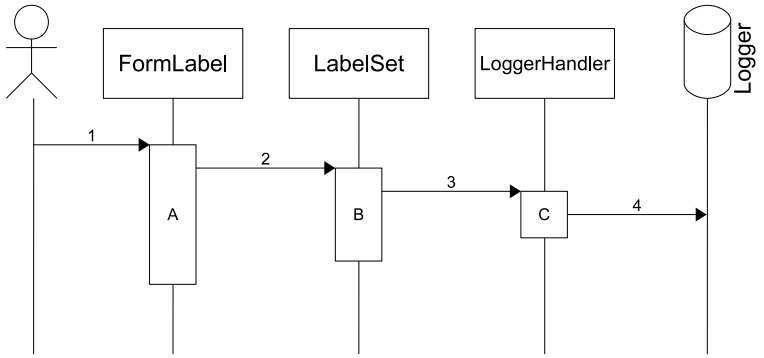
B. ShowLabel tar emot bilden och visar upp den i användargränssnittet.

Spara den nya etikettmallen

Följande lista beskriver sekvensdiagrammet i figur 5.15:

1. Användaren klickar på knappen för att välja den etikett som visas till den aktuella etikettmallen och funktionen SetLabel anropas.

A. Funktionen SetLabel startas.



Figur 5.15: Sekvensdiagram då användaren klickat på knappen för att välja ny etikettmall.

2. SetLabel anropar funktionen SaveLabel i klassen LabelSet och skickar det aktuella indexet.

B. SaveLabel startas och med hjälp av indexet letar den upp rätt objektet i listan listOfLabels.

3. SaveLabel anropar funktionen SaveLabel i klassen LoggerHandler och skickar till den objektets textsträng.

4. SaveLabel i LoggerHandler skriver textsträngen till filen *selectedlabel.zpl* som finns på loggerenheten.

5.7 Hantering av informationsvydata

5.7.1 Klassbeskrivningar

Informationshanteringen sker i tre olika klasser. Klassen för det grafiska gränssnittet (FormShip), klassen för hantering av datan som ska visas upp i användargränssnittet (ShipData), samt klassen som arbetar direkt mot loggerenheten (LoggerHandler).

FormShip - FormShip är den grafiska klassen som tar hand om uppvisning av informationen som användaren har sparat på loggern i filen *shipdata.csv*.

ShipData - ShipData är den klass som agerar mellanhand och hanterar datan som skickas mellan FormShip och LoggerHandler.

LoggerHandler - LoggerHandler är en klass som arbetar direkt mot loggerenheten, all skrivning och läsning till filen *shipdata.csv* går via LoggerHandler.

Figur 5.1 visar hur de olika klasserna hänger samman i systemet.

5.7.2 Beskrivning av filen *shipdata.csv*

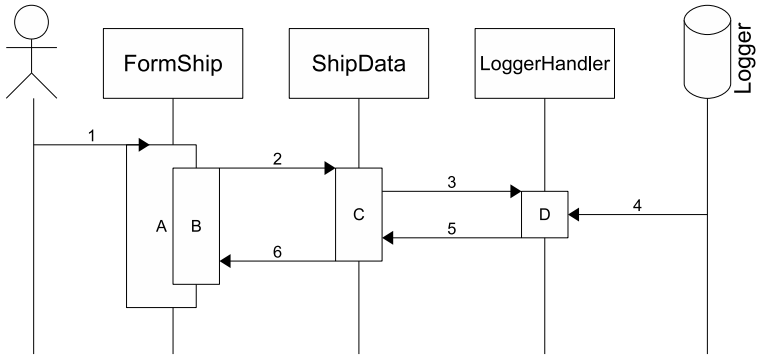
Filen som innehåller informationen som visas upp i informations-vyn finns i loggerenhetens rotkatalog och har namnet *shipdata.csv*. Ett exempel på strukturen i filen kan ses här:

```
Ship Name;SS Nautilus
Captain;Nemo
Fax;+46(0)56012345
```

Det som är skrivet före semikolonet på varje rad är texten som kommer att visas i fältbeskrivningarna (se punkt 1 i figur 4.7). Texten efter semikolonet är den som kommer att visas i informationsbeskrivningarna (se punkt 2 i figur 4.7).

5.7.3 Processer

Innan programmet kan visa upp vyn måste först ett antal steg genomföras:



Figur 5.16: Sekvensdiagram då användaren klickar på knappen för att visa informationsvyn.

Följande lista beskriver sekvensdiagramet i figur 5.16:

1. Användaren klickar på knappen för att visa upp informationsvyn och funktionen FormShip_Load i klassen FormShip anropas.

A. Funktionen FormShip_Load startas och en instans av ShipData skapas. Därefter anropas funktionen LoadShipData som finns i klassen FormShip.

B. Funktionen LoadShipData i FormShip startar.

2. LoadShipData anropar funktionen LoadShipData i klassen ShipData.

C. Funktionen LoadShipData i klassen ShipData startas.

3. LoadShipData kallar på funktionen LoadShipData i klassen LoggerHandler.

D. LoadShipData i LoggerHandler startas.

4. LoadShipData läser in datan från filen *shipdata.csv* som finns på loggerenheten och lägger värdena i en lista.

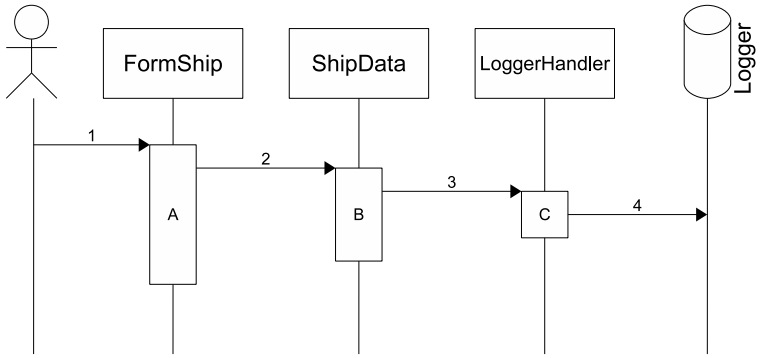
5. Listan returneras till funktionen LoadShipData i LabelSet

6. LoadShipData i LabelSet returnerar listan vidare till funktionen LoadShipData i FormShip.

B. LoadShipData i FormShip hanterar innehållet i listan och visar upp det i användargränssnittet.

Följande lista beskriver sekvensdiagramet i figur 5.17:

1 Användaren klickar på knappen för att spara informationen och funktionen SaveShipInfo



Figur 5.17: Sekvensdiagram då användaren sparar informationen som visas på informations-vyn.

i klassen FormShip anropas.

A. Funktionen SaveShipInfo startar och lägger in all information som ska sparas i en lista.

2 SaveShipInfo anropar funktionen SaveShipInfo i ShipData och skickar listan.

B. Funktionen SaveShipInfo i ShipData startar och tar emot listan.

3 Funktionen SaveShipInfo i LoggerHandler anropas och listan skickas till funktionen.

C. SaveShipInfo i LoggerHandler startar och tar emot listan.

4 SaveShipInfo skriver ner informationen från listan till filen *shipdata.csv* som finns på loggerenheten.

Kapitel 6

Slutsats

I detta kapitel sammanfattar vi vad vi har gjort i projektet, vi går även igenom de problem vi har haft och ger slutligen förslag på framtida förbättringar

6.1 Slutsats

Detta projekt har resulterat i ett fullt fungerande program för att administrera en flyttbar lagringsenhet. Programmet är lätt att använda även för personer med liten datorvana. Med programmet kan man sammanställa och presentera en rapport baserad på loggdatat från loggerenheten. Man kan välja den mall som kommer användas vid utskrift från ett urval av fördefinierade etikettmallar. Programmet låter användaren spara information kring fiskebåten eller användaren, samt uppdatera listorna med fiskeområden och fisksorter.

Programmet kommer att agera som grund för vidareutveckling till den slutliga produkten som kommer skeppas tillsammans med vågsystemet.

Enligt kravspecifikation (sektion 3.1.3) var vi tvungna att utesluta vissa funktioner från det slutliga programmet på grund av tidsbrist. Alla funktioner som den reviderade kravspecifikationen angav finns dock med i programmet.

6.2 Problem

Projektet fick en något försenad start, eftersom den uppsats som vi ursprungligen skulle ha arbetat på visade sig vara allt för likt en uppsats som skrivits tidigare. Detta ledde till att vi ej fick med all funktionalitet som vi hade önskat i programmet, vilket främst var filåterställning på loggerenheten och mer inställningar i samband med rapportgenereringen.

Det har varit svårt att få fram information kring fiskarnas arbetsrutiner och hur yrkesfisket fungerar i allmänhet. De beslut som har gjorts under projektet har därför alltid gjorts i samråd med Unisystem.

6.3 Framtida tillägg och förbättringar

Programmet utgör en bas för vidareutveckling. De funktioner som kan tänkas tillkomma innan programmet är färdigt för användning av slutkund är bland annat: stöd för fler språk, möjlighet att återställa loggerenhetens filer samt förbättra rapportfunktionen.

Referenser

[1] *Unisystem AB*
http://www.unisystem.se/
2010-06-07

[2] *Zoff Com AB*
http://www.zoff.se/
2010-06-07

[3] *An overview of Microsoft Visual Studio 2008*, Microsoft
http://www.microsoft.com/downloads/details.aspx?FamilyId=17319EB4-299C-43B8-A360-A1C2BD6A421B
2010-06-07

[4] *What is Crystal Reports for Visual Studio?*, Microsoft
http://msdn.microsoft.com/en-us/library/ms225592(v=VS.90).aspx
2010-06-07

[5] *ZPL II Programming Guide Volume One*, Zebra Technologies Corporation
Dokumentets namn: 45541LB-R3.pdf
http://www.zebra.com/id/zebra/na/en/index/resource_library/manuals.results.html
Utgivningsår: 2004

[6] *ZPL II Programming Guide Volume Two*, Zebra Technologies Corporation
URL: http://www.zebra.com/id/zebra/na/en/index/resource_library/manuals.results.html
Dokumentets namn: 45542L-002.pdf
Utgivningsår: 2004

Bilaga A

En introduktion till ZPL II

A.1 Skrivarspråket ZPL II

Skrivaren som används tillsammans med vågen klarar av skrivarspråken: EPL2, Line Mode, ZPL samt ZPL II. Efter önskemål från uppdragsgivaren valdes språket ZPL II för projektet.

A.2 Bakgrund

ZPL II är en förkortning och står för Zebra Programming Language II. Språket utvecklades av Zebra Technologies Corporation. Det är ett kraftfullt högnivå-språk som används till att skapa etiketter, det ger även möjligheten till att kontrollera skrivaren. För att antingen skriva ut etiketter eller kontrollera en skrivare så måste kommandon skickas till en skrivare.

ZPL II är en vidareutveckling av språket ZPL (Zebra Programming Language). Målet med att utveckla ZPL II var att kunna sänka tiden från att en skrivare börjar ta emot data för skapandet av en etikett, tills etiketten har skrivits ut.

Det finns ett antal skillnader mellan ZPL och ZPL II, som exempel har många nya kommandon lagts till i ZPL II och gamla kommandon har förbättrats. Den mest markanta skillnaden är att med ZPL börjar skrivaren först hantera kommandon när kommandot som

visar att det är slut på den aktuella etiketten dyker upp. Med ZPL II så hanteras alla kommandofält direkt de tas emot.

A.3 Språköversikt

En skrivare tar emot och tolkar kommandon Kommandon skickas till en skrivare som tolkar dessa och agerar enligt vilken typ av kommando som togs emot. Man kan även gruppera ett antal kommandon i ett så kallat kommandofält. Det finns två olika typer av kommandon, formateringskommandon och kontrollkommandon. För att skilja på dessa används två olika prefixtecken. Formateringskommandon har insättningstecknet (^) som standard och kontrollkommandon har tildetecknet (~). Prefixtecken är speciella eftersom de aldrig kan skrivas ut på en etikett, utan enbart används i samband med att man vill använda ett nytt kommando. Man har möjligheten att byta ut prefixtecken till vilket ASCII-tecken som helst, men då blir det nya prefixet obrukbart vid utskrifter.

Många formateringskommandon och kontrollkommandon används tillsammans med en eller flera parametrar. Parametrar fungerar som platshållare för värden och alla kommandon agerar olika utifrån vilka värden parametrarna har tilldelats. Man har även möjligheten att inte ange ett värde för en eller flera parametrar men då kommer parameterarnas standardvärden att användas. I kommande exempel på olika kommandon har kursiv stil använts för att visa vad som är parametrar.

Formateringskommando - Formateringskommandon används för att skapa etiketter. Ett formateringskommando för en etikett kan liknas med vad en ritning är för ett hus. Det är med denna sorts kommando som man bestämmer hur upplägget ska vara på en etikett. Ett exempel på ett formateringskommando är '^GB*w*,*h*,*t*,*c*,*r*'. Detta är ett kommando som ritar upp en fyrkantig ram. Parametern *w* anger bredden på ramen, *h* anger höjden, *t* sätter tjockleken på kanterna, *c* anger färgen på linjerna och *r* sätter hur mycket hörnen ska vara avrundade. Om en skrivare skulle läsa

kommandot ‘^GB*20,203,10*’ kommer en ram skrivas ut med bredden 20 punkter, höjden 203 punkter och 10 punkter som tjocklek på kanterna. I kommandot ovan utelämnades två parametrar, c och r, dessa två kommer därför att använda sig av sina standardvärden. I detta fall skulle skulle c och r få värdena B (svart färg på kanterna) respektive 0 (ingen avrundning på hörnen).

Parametrarna till kommandon är positionsberoende, vilket innebär att om man inte vill ange något värde för en parameter som inte ligger sist bland alla parametrar (som till exempel med höjdparametern h i kommandot: ‘^GBw,h,t,c,r’), måste man med hjälp av kommatecken visa vilken parameter man hoppade över.

Ett exempel på hur det skulle se ut om man inte vill ange ett värde på höjdparametern h och hörnavrundnings-parametern r är ‘^GB*20,,10*,W’. Detta skriver ut en ram med bredden 20 punkter, höjden 1 punkt (vilket är standardvärdet för höjdparametern h), tjockleken 10 punkter på kanterna, vit färg på linjerna samt ingen avrundning på hörnen (vilket är standardvärde för sista parametern r).

Kontrollkommando - Kontrollkommandon används för att kontrollera skrivare. Det som skiljer sig åt då en skrivare tar emot ett kontrollkommando jämfört med då den tar emot ett formateringskommando, är att ett kontrollkommando hanteras direkt. Det finns dock några få undantag då detta inte gäller. Om det finns formateringskommandon i skrivarens buffer som väntar på att bli hanterade så hoppar skrivaren över dessa kommandon och hanterar kontrollkommandot direkt. Ett exempel på ett kontrollkommando är ‘~JA’, detta kommando avbryter och raderar alla utskrifter i skrivarens buffer som väntar på att bli utskrivna.

A.4 Exempel på kommandofält

Här följer några exempel på kommandofält som beskrivs med förklaringar:

```
^XA
^FX-DATUM-^FS
^FO25,35^A0,36,20^FDDate^FS
^FO25,71^A0,36,20^FN1^FS
^FN1^FD2010-05-11^FS
^XZ
```

Först kommer vi förklara vad kommandona '^FS' (Field separator) och '^FD*a*' (Field data) är då dessa används på flera ställen. Som det tidigare i kapitlet nämndes kan kommandon delas in i så kallade kommandofält, för att separera dessa anges var ett kommandofält har avslutats med ett '^FS'-kommando.

Kommandot '^FD*a*' används när man vill skriva ut en textsträng, parametern *a* är den text som ska skrivas ut. All text som anges efter '^FD', det vill säga vad parameter *a* har som värde, kommer att skrivas ut tills ett prefixtecken stöts på som anger att ett nytt kommando ska börja. Hädanefter, i förklaringarna, kommer kommandot '^FD*a*' att refereras som datafältet.

^XA - Anger att här är början på en etikett och efterföljande kommandon används för att skapa en etikett.

^FX-DATUM-^FS - ^FX - '^FX' (Comment) är ett kommando som anger att efterföljande text fram till ett prefixtecken, då ett nytt kommando börjar, är kommentarer och kommer inte att skrivas ut på etiketten.

^FO*25,35*^A*0,36,20*^FDDate^FS - Detta är ett kommandofält där det första kommandot '^FO*x,y*' (Field origin) anger var på etiketten följande datafält ska ha sin placering, i förhållande till vad skrivaren har inställt som origo. Parametern *x* anger hur långt till vänster från origo datafältet ska hamna och parametern *y* anger hur långt nedåt från origo datafältet ska hamna. Som standard är den övre vänstra hörnet inställt som origo. Med kommandot '^FO*25,35*' bestäms placeringen för kommande datafält till 25 punkter åt höger och 35 punkter nedåt, i förhållande till origo. Nästa

kommando i fältet är ‘^A*f,h,w*’ (Bitmapped font), detta är ett kommando för att ställa in utseende för följande datafält som ska skrivas ut på etiketten. Parametern *f* bestämmer vilken typsnitt som ska användas, *h* anger höjden och *w* anger bredden. I kommandofältet ovan där parametrarna *f*, *h* och *w* har fått värdena 0, 36 och 20 betyder det att efterföljande datafält kommer ha typsnittet 0, höjden 36 punkter och bredden 20 punkter. Och till sist kommer kommandona ‘^FD’ och ‘^FS’ vars funktioner förklarades tidigare i delkapitlet. Resultatet av denna kommandofält kommer vara en text, *Date*, som skrivs ut på en etikett 25 punkter åt vänster och 35 punkter åt höger från origo.

^FO*25,71*^A*0,36,20*^FN*1*^FS - Detta kommandofält har samma struktur som kommandofältet beskrivet ovan, med två skillnader. Kommandot ^FO har andra värden på en av sina parametrar, 71 jämfört med 35. Detta kommer resultera i att den text som ska skrivas ut kommer hamna 36 punkter längre ner från origo jämfört med texten *Date* som skrevs ut av förra kommandofältet. Den andra skillnaden är att ^FD-kommandot har blivit utbytt mot ‘^FN’. ‘^FN*#*’ (Field number) är ett kommando som används för att referera till och namnge ett kommandofält. Parametern *#* används för att skilja olika namngivna kommandofält åt. I kommandofältet ovan där ‘^FN*1*’ skrivits, refererar man till ett kommandofält kallat ‘^FN*1*’. Det näst sista kommandofältet, ‘^FN*1*^FD*2010-05-11*^FS’, från urvalet av kommandonfält ovan, är ett exempel där man har namngivit ett kommandofält. Först skrivs namnet på fältet, vilket kommer vara det som används när man refererar till ‘^FN*1*’, därefter följer innehållet och avslutningsvis skrivs fältavslutaren ‘^FS’. Om man refererar till kommandofältet ‘^FN*1*’ och kommandofältet som heter ‘^FN*1*’ ser ut så här ‘^FN*1*^FD*2010-05-11*^FS’ kommer kommandot ‘^FD’ användas. Den text som skrivs ut av kommandofältet ‘^FO*25,71*^A*0,36,20*^FN*1*^FS’ från urvalet ovan är ‘2010-05-11’.

^XZ - Slutligen används kommandot '^XZ' (End format). Detta anger att det är slut på kommandon för pågående etikett och skrivaren skriver nu ut etiketten.