Design and Implementation of Export Shipping System with a Study of the Requirements and Life-Cycle

Goudong Zhou
Miami University, commons-admin@lib.muohio.edu

MIAMI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & SYSTEMS ANALYSIS

TECHNICAL REPORT: MU-SEAS-CSA-1992-005

Design and Implementation of Export Shipping System with a Study of the Requirements and Life Cycle

Goudong Zhou

School of Engineering & Applied Science | Oxford, Ohio 45056 | 513-529-5928

Design and Implementation of Export Shipping System with a Study of the Requirements and Life-Cycle

by

Goudong Zhou
Systems Analysis Department
Miami University
Oxford, Ohio 45056

Working Paper #92-005 07/92

Design and Implementation of Export Shipping System with a Study of the Requirements and Life-Cycle

Presented in Partial Fulfillment of the Requirements
for the Degree of
Master of Systems Analysis
Graduate School of Miami University

By

Guodong Zhou

Miami University

July 1, 1992

SYSTEMS ANALYSIS DEPARTMENT

MASTER'S PROJECT FINAL REPORT

Presented in Partial Fulfillment of the Requirements
for the Degree of
Master of Systems Analysis
in the
Graduate School of Miami University

TITLE: Design and Implementation of Export Shipping System with a Study of the Requirements and Life-Cycle

PRESENTED BY: Guodong Zhou

DATE: July 1, 1992

COMMITTEE MEMBERS:

James D. Kiper, Advisor	James D. Kiper
David C. Haddad	D C Haddad
Alton F. Sanders	Alton F Sanders

Design and Implementation of Export Shipping System with a study of the Requirements and Life-Cycle

Guodong Zhou

Dept. of Systems Analysis
Miami University

Abstract

The Export Shipping System is focused primarily on upgrading the manual system currently used for tracking export orders and inventory in the shipping department at Square D Company. It consists of a group of program modules which facilitate the receipt, shipment, and associated paperwork involved in the processing of export orders from the time they reach the shipping department through the invoice and subsequent accounting reports. This paper describes the project and the associated software implemented. This description analyzes the project development life-cycle and compares it with the alternative software development life-cycles, as described in the literature in order to identify strengths and weaknesses of the method used in the Export Shipping System. Finally, it will give some afterthoughts about the project development to guide future project development.

Table of Contents

1. Introduction

For the past year I have worked at the Square D Company, Oxford, Ohio as a programmer. I have participated in the design and development of a large-scale software system called the Export Shipping System. I worked under the direction of a system analyst who is my primary source for the system requirements.

This paper will present an overview of the Export Shipping System, including the form of the requirements, system proposal and the life-cycle used in the development of the project. Next, it will review alternative software development life-cycles described in the literature. Finally, it will conclude with a comparison of the life-cycle used in Export Shipping System with the alternatives, and will identify ways that the methods used at Square D Company can be improved.

2. Project Background

Square D Company is a worldwide electric products company. Its industrial control and electrical distribution products, systems and services are used in industrial facilities and machinery, in commercial and residential construction, and in original equipment manufactures' products. The Oxford plant is one of its 58 plants, the products of Oxford plant are mainly busway and wire management systems. The organization chart of Oxford plant can be shown in Figure 2.1.

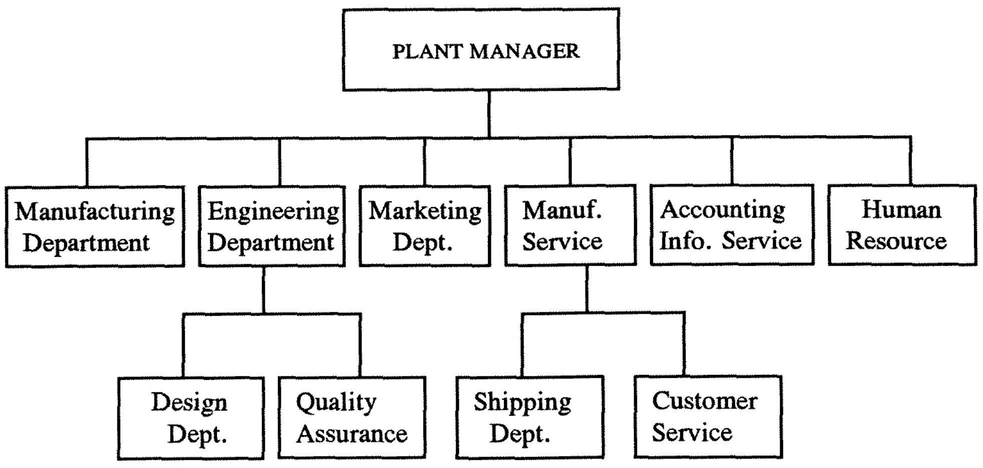


Figure 2.1 Organization Chart of Square D, Oxford Plant

2.1 Overview Current Export Shipping Methods

It is estimated that over 30% of annual revenue of the Oxford plant comes from exporting. The current working model in the shipping department is inadequate for the shipper to keep track of the orders and consequently for the persons in the accounting department to generate accounting reports and invoices. Furthermore, there are no control methods to guide the shipper with properly stuffing a crate. For example, the same order may have two different shipping methods - by air or by ocean, items shipped by air should not be put into the crate which will be shipped by ocean. Also there is no good effective way to undo a shipping order if some errors occur or customers change their orders. If the problems are not fixed, the company profits will be effected, and customers will switch to other sources. The present working model can be showed in Figure 2.2 below.

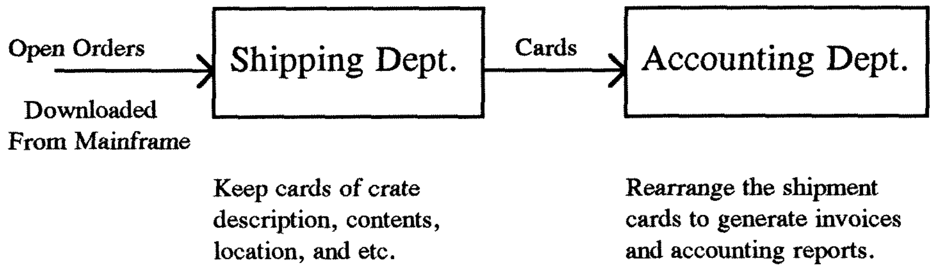

Open Orders
Downloaded From Mainframe
Shipping Dept.
Cards
Accounting Dept.
Keep cards of crate description, contents, location, and etc.
Rearrange the shipment cards to generate invoices and accounting reports.

Figure 2.2 Current Export Shipping Working Method

2.2 Findings and Recommendations

Currently, for each order that arrives at the shipping department, the shipper needs to pick items to put into the crate and maintains a list of the crate specifications, individual catalog numbers of each item, and item quantities. As crates are stacked for shipping, each item must be checked against a copy of the shipping list. Often the shippers have many page long lists and the same catalog number may be listed many times on different pages.

The physical location of each shipping site of each crate is logged on index cards. When the shipper needs to ship a crate out of plant, he needs to look up the index cards to fill the packing list for a specific container (such as truck). For large orders, the determination of when there are enough crates to fill a container is based on frequent scans of all the order sheets. The shipper must mentally estimate the total volume of the order based on length, width, and height for each crate and also need to add up the weights to determine whether they can make a load.

When a load of export orders comes to the accounting department, the following information is recorded on the packing list (sheets) for each order: Sequential number, type of crate (Pallet, Crate, etc.), catalog number and its description, item quantities. These data will be rearranged to generate an invoice and sub-sequential accounting reports.

The study of the current system used at the shipping and accounting departments revealed that the initial solution for the Export Shipping System should include the following functions:

1). Receive export open orders and their items;
2). Put items into crates and temporally store before shipping out;
3). Ship crates, print out crate labels and generate picking list for each crate; if needed, unpack a crate;
4). Pack crates for a container and generate packing list which needs to pass to the accounting department in order to generate invoice and accounting report;
5). Let coordinators add shipment control comment to each item to guide the shipper to properly pack the crates;

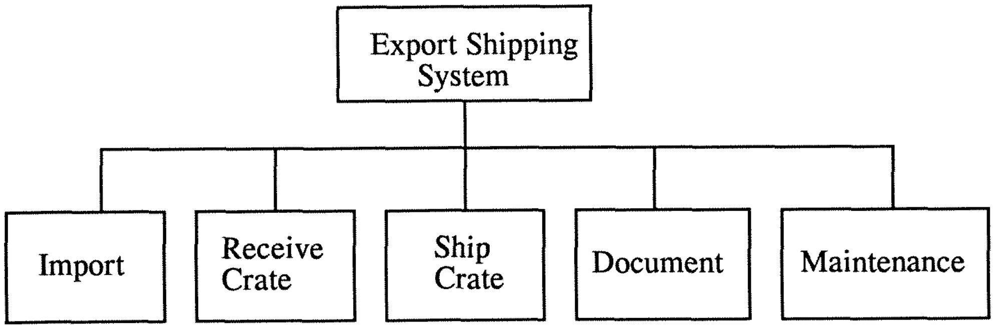
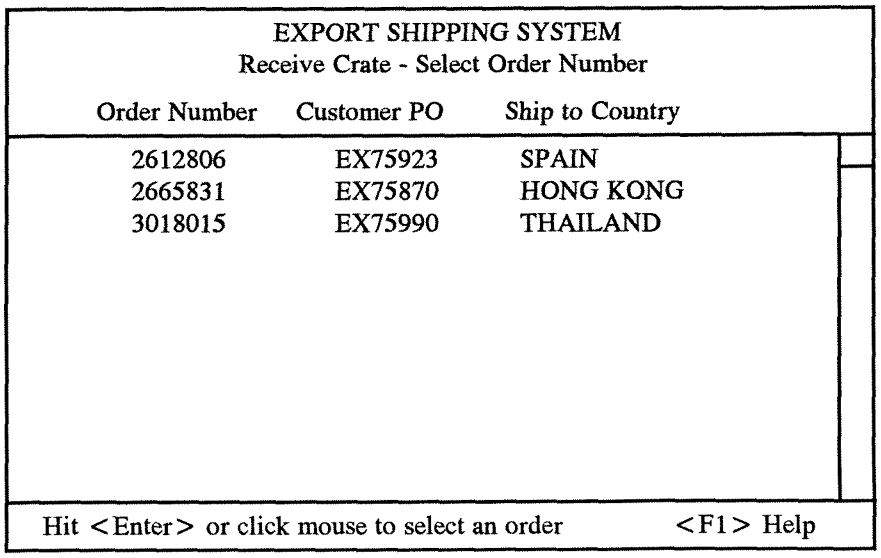


Figure 2.3 Export Shipping System

2.3 Export Shipping System Proposal

After analyzing the initial user requirements, the decision was made that the following functions and associated screens would be implemented, as shown in Figure 2.3. The system was considered to be a set of functions with data flowing from one to another. Additionally, the staff decided to take advantage of in-house equipment and software development tools in building the system.

1). **Import function**: a list of open export orders and backorder quantities for each item will be downloaded nightly to a Btrieve database on the network server from the mainframe;


EXPORT SHIPPING SYSTEM
Receive Crate - Select Order Number

Order Number	Customer PO	Ship to Country
2612806	EX75923	SPAIN
2665831	EX75870	HONG KONG
3018015	EX75990	THAILAND

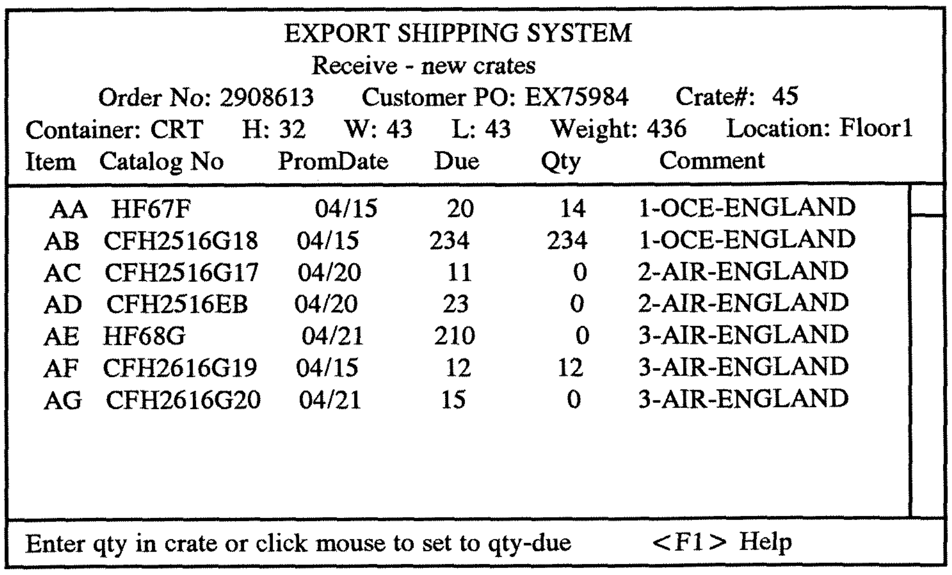
Hit <Enter> or click mouse to select an order <F1> Help

Figure 2.4 Receive Order Screen

2). **Receive function**: allows the user to define a new crate and to assign items to that crate; if needed, the user can change crate descriptions, or item quantity. He can also add or delete some items from that crate.

The screens of the Receive function consist of the following. When the user selects the Receive function from main menu, the first screen, as shown in Figure 2.4 above, will display a list of SQD order numbers for all open export orders. The export customer purchase order number and shipping to country name will also be listed as cross-reference. The screen should be able to scroll up or down to hold all open orders. The user will select an order by moving the cursor to the appropriate position then hit <Enter> or click with a mouse.

After the user hits <Enter> from the Receive Order Screen, a second screen, as shown in Figure 2.5, will pop up. This screen consists of two separate parts. The user will key in the crate specifications, such as container type, dimensions, weight, and storage location, into the upper portion of the screen. The lower portion of the screen will display a list of all open items for the current order, the quantity due, shipment control comments, and a place for the user to enter the quantity being received. The crate number will be assigned to the next available number (the last crate number plus one for this order), unless the user specifies a crate number to view or edit an existing crate.



EXPORT SHIPPING SYSTEM
Receive - new crates

Order No: 2908613 Customer PO: EX75984 Crate#: 45

Container: CRT H: 32 W: 43 L: 43 Weight: 436 Location: Floor1

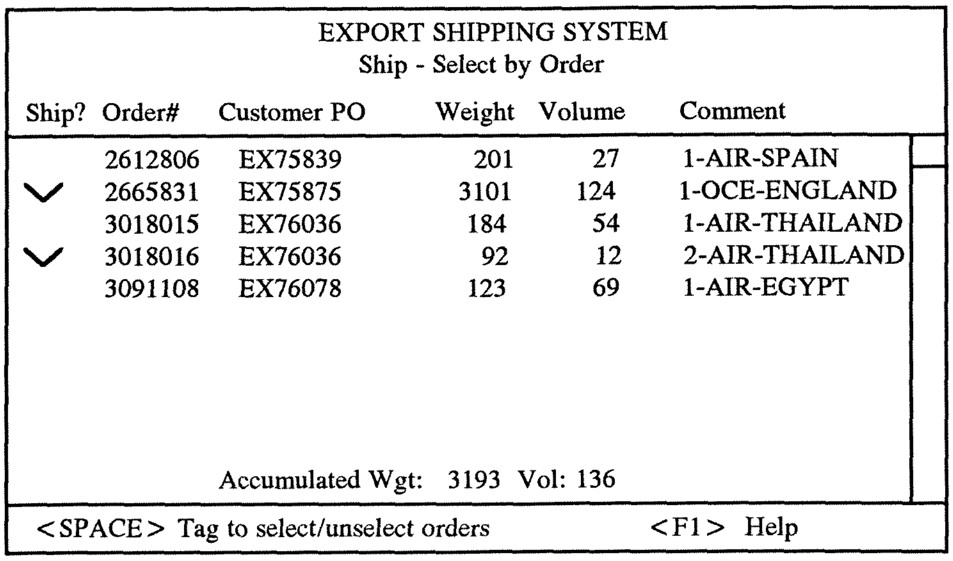
Item	Catalog No	PromDate	Due	Qty	Comment
AA	HF67F	04/15	20	14	1-OCE-ENGLAND
AB	CFH2516G18	04/15	234	234	1-OCE-ENGLAND
AC	CFH2516G17	04/20	11	0	2-AIR-ENGLAND
AD	CFH2516EB	04/20	23	0	2-AIR-ENGLAND
AE	HF68G	04/21	210	0	3-AIR-ENGLAND
AF	CFH2616G19	04/15	12	12	3-AIR-ENGLAND
AG	CFH2616G20	04/21	15	0	3-AIR-ENGLAND

Enter qty in crate or click mouse to set to qty-due <F1> Help

Figure 2.5 Create Crate Screen

3). **Ship function**: allows the user to select crates to be shipped. The printing of picking lists and crate labels is available as well.

The first screen of the Ship function will display a list of all orders for existing crates in the shipping area. The total weight, volume, and status (complete or incomplete), and shipping comment will be displayed as well. The user can use <Space bar> to tag selected orders to ship all crates associated to the orders. The required prototype screen is shown in Figure 2.6. A printed picking list shows each crate item, and information for selected orders will be offered.



EXPORT SHIPPING SYSTEM
Ship - Select by Order

Ship?	Order#	Customer PO	Weight	Volume	Comment
	2612806	EX75839	201	27	1-AIR-SPAIN
✓	2665831	EX75875	3101	124	1-OCE-ENGLAND
	3018015	EX76036	184	54	1-AIR-THAILAND
✓	3018016	EX76036	92	12	2-AIR-THAILAND
	3091108	EX76078	123	69	1-AIR-EGYPT

Accumulated Wgt: 3193 Vol: 136

<SPACE> Tag to select/unselect orders <F1> Help

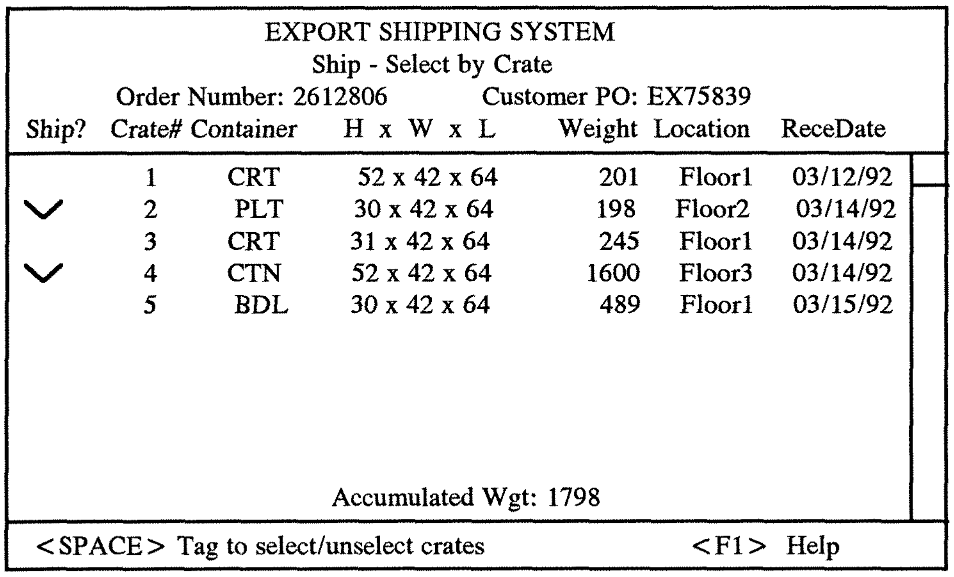
Figure 2.6 Ship Order Screen

The user may "zoom" to the next screen which displays a list of the crates accumulated for the selected order. The container type, dimensions, weight, physical location and date received will be displayed for each crate. The user may select individual crates by tagging them for shipment. As the user tags the crates, the accumulated weight and volume data is displayed as an aid to determine the container requirements, as shown in Figure 2.7.

The user can also "zoom" to next screen, as shown in Figure 2.8, which displays the individual items in the select crate. This screen is for viewing only; an editing function is not provided.

4). **Document functions**: this function has three major purposes: first to let the coordinator add shipment comments before the shipper generates the crate, and to preload some invoice header information which he knows as far; second, to generate a packing list after the shipper receives crates and ships them out of the plant; third, to generate an invoice and subsequent accounting reports. The Document functions have been divided into three sub-menu choices.

- **Template**: allows user to add shipment comments to each item before the shipper packs a crate, as shown in Figure 2.9; and to create, edit and assign invoice header templates, shown in Figure 2.10. One template can be used for multi-invoice headers for the same order number with a slight modification. For a specific order, the user can create several templates. The system will brings up the next available control number when the user wants to create a new template.



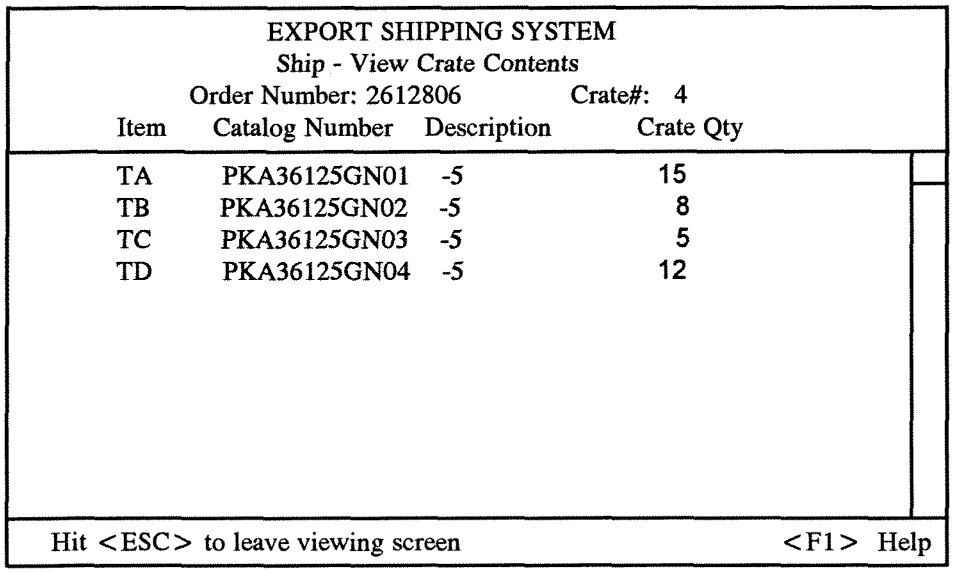
EXPORT SHIPPING SYSTEM
Ship - Select by Crate
Order Number: 2612806 Customer PO: EX75839

Ship?	Crate#	Container	H x W x L	Weight	Location	ReceDate
	1	CRT	52 x 42 x 64	201	Floor1	03/12/92
✓	2	PLT	30 x 42 x 64	198	Floor2	03/14/92
	3	CRT	31 x 42 x 64	245	Floor1	03/14/92
✓	4	CTN	52 x 42 x 64	1600	Floor3	03/14/92
	5	BDL	30 x 42 x 64	489	Floor1	03/15/92

Accumulated Wgt: 1798

<SPACE> Tag to select/unselect crates <F1> Help

Figure 2.7 Ship Crate Screen



```
                 EXPORT SHIPPING SYSTEM
                 Ship - View Crate Contents
            Order Number: 2612806          Crate#:  4
   Item    Catalog Number  Description         Crate Qty

   TA       PKA36125GN01    -5                    15
   TB       PKA36125GN02    -5                     8
   TC       PKA36125GN03    -5                     5
   TD       PKA36125GN04    -5                    12

 Hit <ESC> to leave viewing screen                  <F1> Help
```

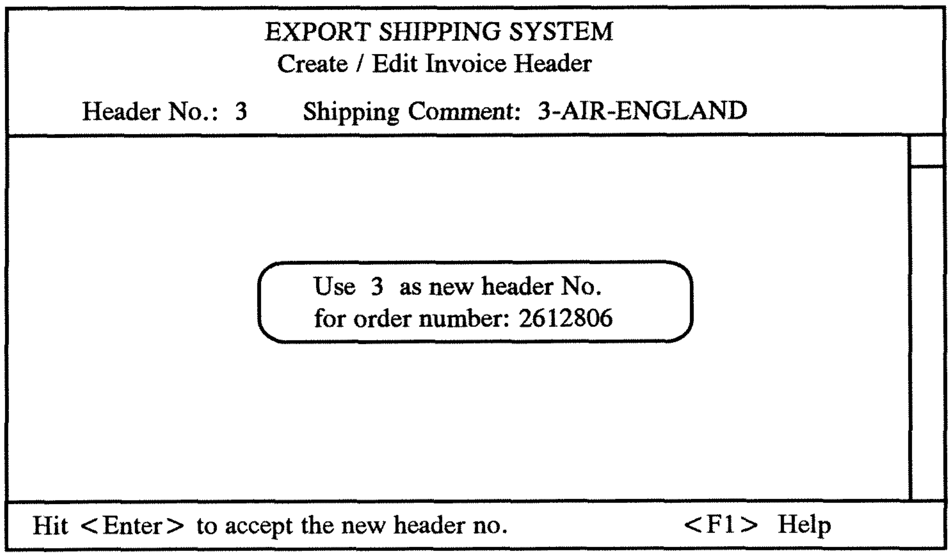
Figure 2.8 View Crate Contents Screen

- **Packing List**: this option allows the user to create a packing list from a list of crates for a specific order when the user puts these crates into a load and ships them out of door, as shown in Figure 2.11. It also provides the user with a chance to modify the shipping item information such as item catalog number, billing charges, and to modify the order template as well (this part of screen is same as Figure 2.10).

- **Invoice**: this option allows user to create, edit, and view shipping order invoices. It includes the three major screens, as shown in Figure 2.12.

  1). View/Edit Invoice Header (same as Figure 2.10), at this stage, all fields should be filled;
  2). View/Edit Invoice Items (similar to the Figure 2.11, but rearranged to reflect invoice layout.);
  3). View/Edit Total Billing, shown in Figure 2.13. It used especially for the freight, installation, or other specified charges. After the user confirms the invoice and saves it, he can not change it again, but he can view the existing invoice and print out a duplicate copy.

5). **Maintenance Functions**: rudimentary screens will be available for editing the raw data which are downloaded from the mainframe to reflect the local changes. The system provides four fundamental database table editors: orders, items, billing information and sold to address table.



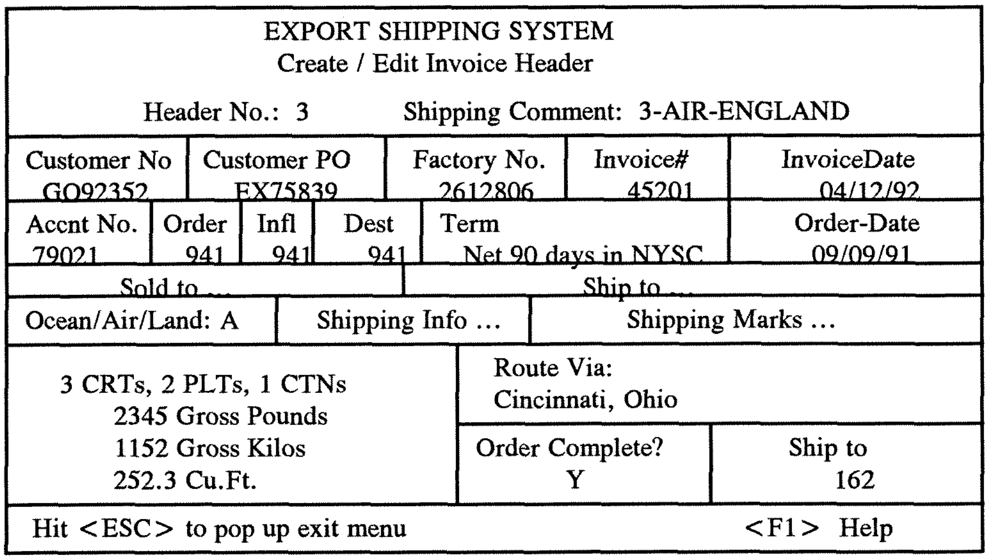
EXPORT SHIPPING SYSTEM
Create / Edit Invoice Header

Header No.: 3 Shipping Comment: 3-AIR-ENGLAND

Use 3 as new header No.
for order number: 2612806

Hit <Enter> to accept the new header no. <F1> Help

Figure 2.9 Template Create Screen (I)



EXPORT SHIPPING SYSTEM
Create / Edit Invoice Header

Header No.: 3 Shipping Comment: 3-AIR-ENGLAND

Customer No	Customer PO	Factory No.	Invoice#	InvoiceDate
GO92352	EX75839	2612806	45201	04/12/92

Accnt No.	Order	Infl	Dest	Term	Order-Date
79021	941	941	941	Net 90 days in NYSC	09/09/91

Sold to ... Ship to ...

Ocean/Air/Land: A Shipping Info ... Shipping Marks ...

3 CRTs, 2 PLTs, 1 CTNs
2345 Gross Pounds
1152 Gross Kilos
252.3 Cu.Ft.

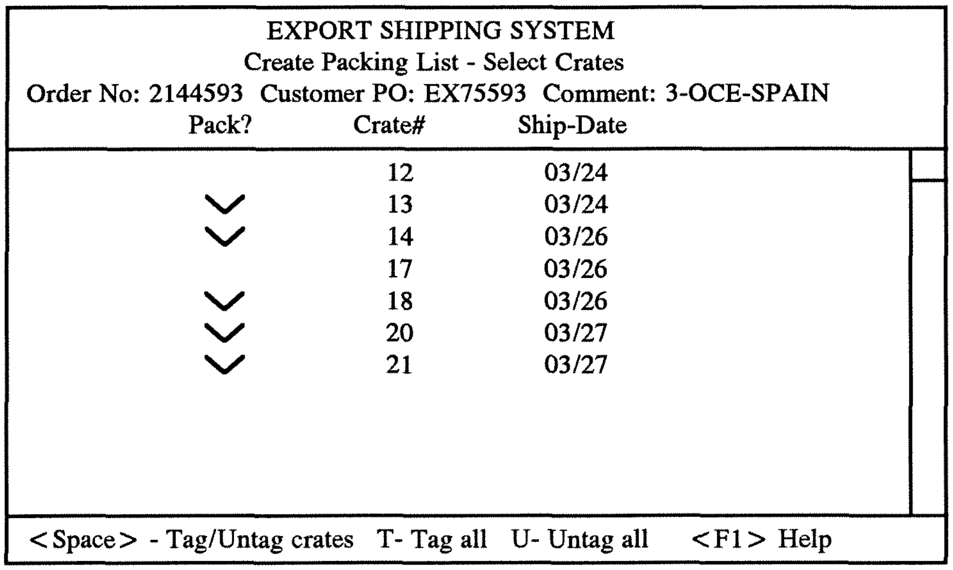
Route Via:
Cincinnati, Ohio

Order Complete?
Y

Ship to
162

Hit <ESC> to pop up exit menu <F1> Help

Figure 2.10 Template Create Screen (II)

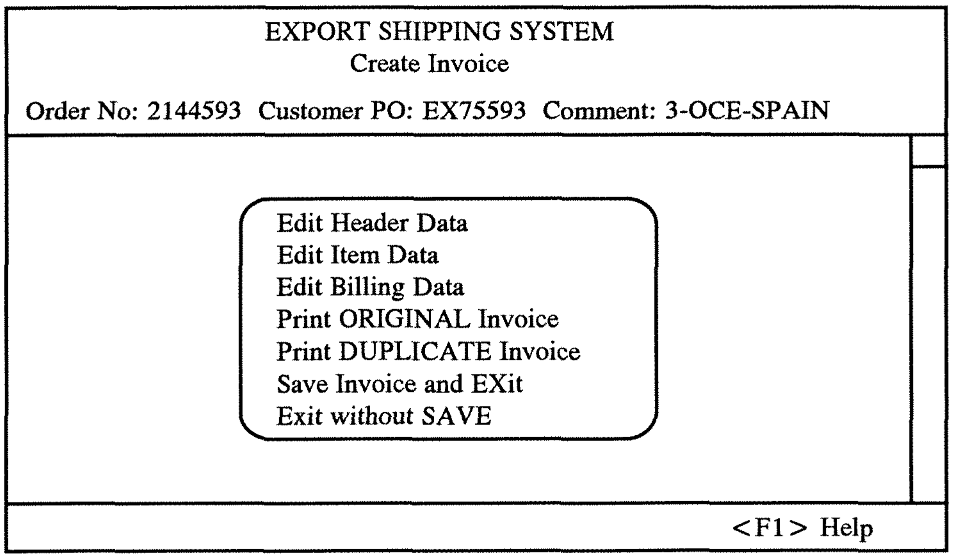


EXPORT SHIPPING SYSTEM
Create Packing List - Select Crates
Order No: 2144593 Customer PO: EX75593 Comment: 3-OCE-SPAIN

Pack?	Crate#	Ship-Date
	12	03/24
✓	13	03/24
✓	14	03/26
	17	03/26
✓	18	03/26
✓	20	03/27
✓	21	03/27

<Space> - Tag/Untag crates T- Tag all U- Untag all <F1> Help

Figure 2.11 Pack Crates Screen

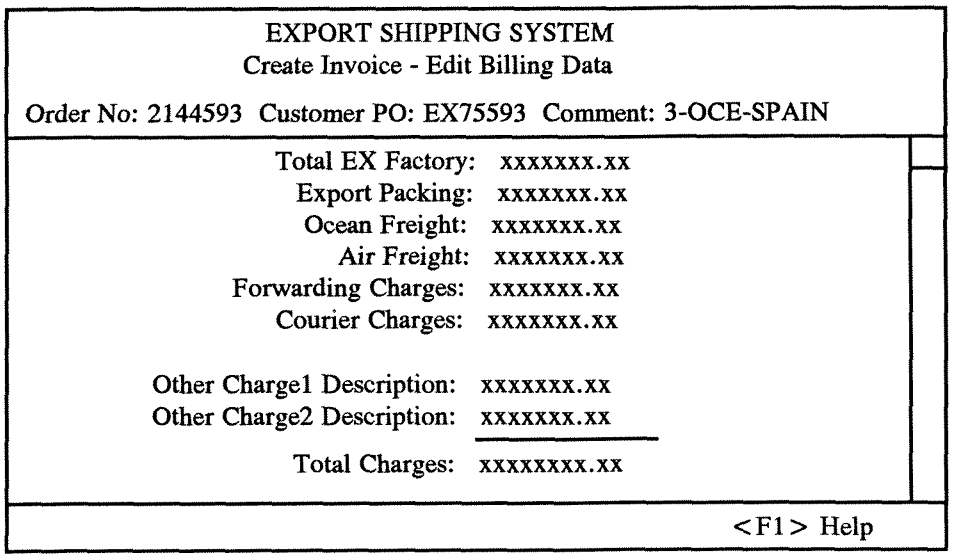


EXPORT SHIPPING SYSTEM
Create Invoice
Order No: 2144593 Customer PO: EX75593 Comment: 3-OCE-SPAIN

Edit Header Data
Edit Item Data
Edit Billing Data
Print ORIGINAL Invoice
Print DUPLICATE Invoice
Save Invoice and EXit
Exit without SAVE

<F1> Help

Figure 2.12 Invoice Screen


EXPORT SHIPPING SYSTEM
Create Invoice - Edit Billing Data

Order No: 2144593 Customer PO: EX75593 Comment: 3-OCE-SPAIN

Total EX Factory:	xxxxxxx.xx
Export Packing:	xxxxxxx.xx
Ocean Freight:	xxxxxxx.xx
Air Freight:	xxxxxxx.xx
Forwarding Charges:	xxxxxxx.xx
Courier Charges:	xxxxxxx.xx
Other Charge1 Description:	xxxxxxx.xx
Other Charge2 Description:	xxxxxxx.xx
Total Charges:	xxxxxxxx.xx

<F1> Help

Figure 2.13 Invoice - Edit Billing Screen

The above explanation roughly introduces the Export Shipping System functional requirements and the user interface prototyping. Next, The database design of the Export System will be presented.

3. Data Model of Export Shipping System

Data modeling consists of organizing the required data structure into a form which can be represented by computer software [10]. The end product of data analysis is a data structure diagram which represents the structure of the data which is to be stored in the database.

By comparing the current manual system with what the user was asking for, an overview of the system and its data model has been worked out. This process was a continuous and repetitive process, but the principal part definition of the data tables (the primary key of each table and the basic mapping relationship among the tables) was determined at system design stage. After brainstorming and studying the forms and raw data that were collected during the user requirements phase, a basic data flow was found in the Export Shipping System, that could be represented as follows: order information downloaded from mainframe, which was passed to the ship department to generate crates, to pack a load, and then was passed to the accounting department to generate invoices and subsequent accounting reports. The physical data flow currently crosses the system as shown in Figure 3.1.

For the order part, two database tables were created: Table006 - Order Item and Table020 - Export Order. Table006 contains all information about an item. Table 020

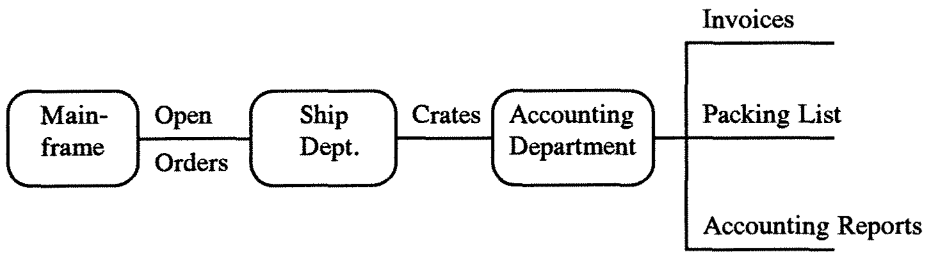


Figure 3.1 Physical Dataflow of Export System

contains all information about an order, so that the relationship is a one to many mapping from Table020 to Table006. For crate parts, two tables were created, Table027 - Crate Quantity Table, and Table028 - Crate Physical Description Table to describe length, width, height, location of each crate. For invoice part, another two database tables were employed, Table021 - Invoice Item Information Table which contains the item catalog number, invoice quantity, item billing, and Table026 - Invoice Header Table, which contains all information about an invoice besides the item information such as invoice number, invoice date, sold to and ship to address, and statistical information.

Some support tables were provided, for example, Table023, containing customer accounting number and correspond shipping to country name; Table037, letting the user preload sold to address according to the customer account number. We number these tables according to the general information system plan in the Oxford plant.

For each table, the key0 is the primary key to identify the table; other keys, called index in Btrieve database, are based on the program implementation and the efficiency of running the system. The database table structure can be represented in Figure 3.2. A detailed definition of each table, please see Appendix A.

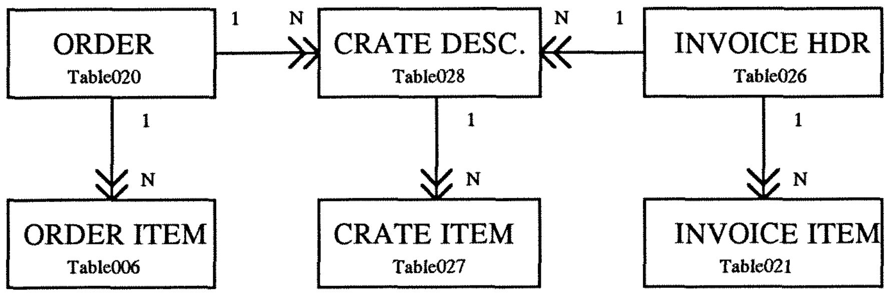


Figure 3.2 Database Structure of Export System

4. System Design of Export Shipping System

The normal design method involved considering the design as a number of functional components. The system started with a high level viewpoint and then was progressively refined into a more detailed design. This methodology is exemplified by the top-down technique which is based on the notion that the structure of the problem should determine the structure of the software [14]. By employing the top-down method, the Export Shipping System was divided into the following seven components, as shown in Figure 4.1, which represents the main menu choice of the Export Shipping System.

Another major function, Import, was implemented as a stand-alone function because it was used primarily by the shipping coordinator as a batch file running nightly. The Import function downloaded the order and its item information into Table006 and Table020.

Each function further refined into its own fundamental parts which were introduced briefly below.

Receive	The Receive function allows the user to define a new crate and to assign items to that crate
Ship	The Ship function allows the user to select crates to be shipped. A picking list and crate label are available as well
EditCrate	The Receive function allows the user to edit the specification or contents of an existing crate as well as to define a new crate
UnShip	The UnShip function allows the user to "unship" crate which were previous "shipped" under the Ship function
Documents	This choice presents a submenu from which the user may create / edit the invoice and packing lists
Maintenance	This choice presents a submenu from which the user may do maintenance on the data tables directly
Help	This choice presents a full screen help file viewer

Figure 4.1 Export Shipping System Funtions

Receive function

This function's primary purpose is to allow the user in the shipping department to log a new crate into the system. The user will go through the following steps:

1). Select the order number of the items in the crate from a list of open orders displayed on the screen;
2). Select a crate number for the crate. The default will be the next unused crate number for this order;
3). Enter the crate physical parameters such as container type, dimensions, weight and the location where the crate will be stored;
4). Enter quantity for each item in the crate. A list of all items due for the current order number and the total quantity still due are displayed;
5). Hit <ESC> to bring up Exit menu;
6). Select "Review Crate Contents" to see a list of the items and quantities selected for review. This step is optional;
7). Select "Log Crate to system" to save the crate into the system with the items and quantities specified.

This function may also be used to edit a crate already logged into the system by selecting an existing crate number. The user will have the option of editing any of the descriptions or the quantity of each item.

Ship Function

This function's purpose is to allow the user to select crates to be shipped. The user will have the option of shipping all crates for a given order number (Shown in Figure 4.3), or selecting individual crates to be shipped for a specific order number. A number of aids are provided to help the user determine what to ship:

1). Orders which are complete are noted;
2). The total accumulated weight and volume for all crates for each order number is displayed;
3). The accumulated weight in pounds and kilos is displayed as individual crates are selected for shipment;
4). A "zoom" function is provided to review the contents of an individual crate;
5). A picking list can be printed at any time for the selected order number or crates.

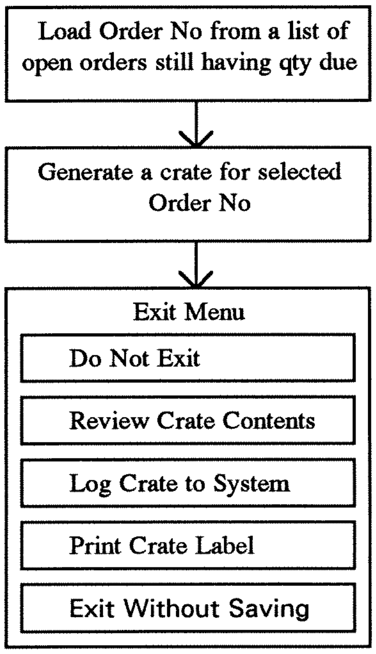

Load Order No from a list of open orders still having qty due
Generate a crate for selected Order No
Exit Menu
Do Not Exit
Review Crate Contents
Log Crate to System
Print Crate Label
Exit Without Saving

Figure 4.2 Control Flow of Receive

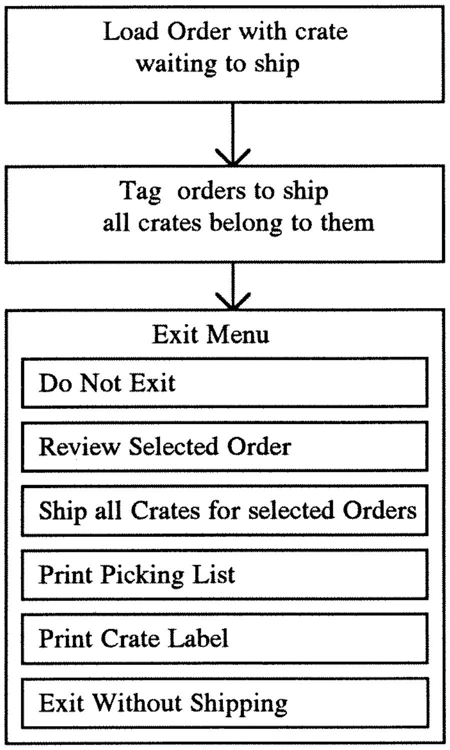

Load Order with crate waiting to ship
Tag orders to ship all crates belong to them
Exit Menu
Do Not Exit
Review Selected Order
Ship all Crates for selected Orders
Print Picking List
Print Crate Label
Exit Without Shipping

Figure 4.3 Ship by Order Control Flow

Edit Crate Function

This function's primary purpose is to allow the user in the shipping department to edit the physical parameters and contents of a crate already logged to the system.

This function is identical to the Receive Function except that the order list displays only the ones which have been received at least one crate so far to be shipped. This function may also be used to receive new crates from an order. With the limitation that you cannot receive a new crate for an order number which does not have any crates so far. It may, in fact, be fast because of the shorter order number listing. Also see the explanation for the Receive Function

UnShip Function

This function's purpose is to allow the user to "unship" crates which have already been logged as shipped under the Ship Function. The user will have the option of "unshipping" all crates for a given order number or selecting individual crates to be "unshipped".

Document Function

This function has three major purposes: to add shipment control comments; to pack crates and generate packing list; and to generate invoice and subsequent accounting report, see Figure 4.4.

It is important to note that after the user saves the invoice, the program will automatically purge the database tables (Table027 and Table028) in order to keep the system continuously running otherwise it will run out of storage space soon.

Templates	This option presents a sub-menu of choices which allow the user to create, edit and assign header templates
Packing List	This option allows the user to create a packing list
Invoice	This option presents a sub-menu of choices which allow the user to create, edit, and view shipping invoices

Figure 4.4 Document Sub-functions

Maintenance Function

The basic purpose of the Maintenance function is to provide a tool to let the shipment coordinator modify the raw data downloaded from the mainframe to reflect the local changes. Four editing functions were provided, i.e. editors of Table006, Table020, Table021 and Table037.

On-Line Help

Provides a structured text help file to support on-line help service when the user hits the <F1> key.

5. System and Interface Implementation

Programming is a craft. It is dependent on individual skill, attention to detail, and knowledge of how to use available tools in the best way [14]. The Export Shipping System was written in C language using the commercial tool such as C-Scape for user interface, and Btrieve and Xtrieve for database management.

C-scape is a powerful and flexible tool for controlling the user interface of C programs with which the user can create and modify virtually any type of text or data entry screen. The user can add a number of sophisticated features to the program such as windowing, graphics support, context-sensitive help, text editing with word wrap, scrolling lists and validation [23]. C-scape is even more powerful when used with the Look & Feel Screen Designer which lets the user experiment with various interfaces for the application before selecting a final one. All screens are saved in "filename.lnf" files, and can be called from a program at run time. This helps shrink executable code size and allows the user to change an application's screens without recompiling the source codes.

The Btrieve is a server-based record management system for workstation applications. All Btrieve requests from network stations are processed at a network server. A concise interface call to all the Btrieve request was implemented during the development of the Export Shipping System.

It was started with Look & Feel tool to create a screen file called Export.lnf which included all of the screens required by the system. For each individual screen, it was designed the almost the same layout which was confirmed by the user at the system design stage. There were three types of screens, SED - the basic screen, SLED - a screen which can be scrolled up or down, and TED - text editor screen. The field movement, screen refreshing and special key handling were handled by calling the C-scape library function.

The database tables were generated by using the Xtrieve, see Appendix A for the detail definition of each table. In C language program, the corresponding data structure was set up to hold the data read from the database in order to process them in the program. Btrieve is a very low level record management system, the control of retrieving the database is implemented by the programmer. For example, if the user wanted to retrieve all the open orders which still had quantity due, the following structure would be used:

```
Btrieve (B_GETFIRST, T006, Key0)
while (!EOF)
    {
    if (still having quantity due)
        display the order on the screen;

    Btrieve(B_GETNEXT, T006, Key0);
    }
```

The Export Shipping System directory structure was set up as shown in Figure 5.1. For each major function, a sub-directory was set up, and the object files were put into next level sub-directory called OBJ. It is usually wise to partition larger projects into functional subsystems with each subsystem having its own directory. The directories for the project form a hierarchical relationship identical to the partitioning of the project itself. This kind of directory structure has several advantages over just putting all the source and object files under one directory:

1). Easier to control the complexity of the system as it becomes large and large;
2). Easier for a team to develop the project, because each programmer generally works only on a part of the project at any one time;
3). Have a clear and short list of source files under each directory because all OBJ and MAP files have been put into their own next level sub-directories called OBJ.

For the implementation details of this kind directory structure, please see the NMAKE and LINK.LRF file in Appendix B.

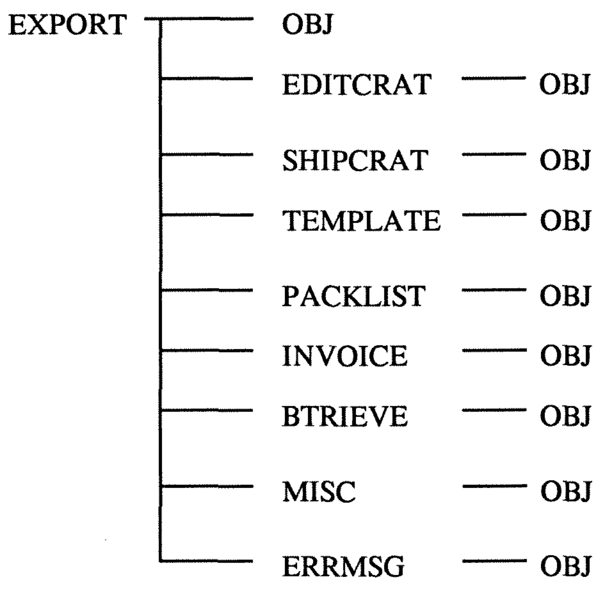


Figure 5.1 The Export Directory Structure

6. Survey of Software Life-Cycles in the Literature

Software development life-cycle is a project management tool, used to plan, execute, and control systems development projects. It breaks down the phases and stages of the projects into tasks that are essential to systems development, no matter what type or size of system you may try to build. Here four general software life-cycle methods are presented.

6.1 Waterfall Model

The waterfall model was popularized in the 1970s and was used in most of the software development activities. The waterfall model is illustrated in Figure 6.1 [9]. As it shows, the process is structured as a cascade of phases, where the output of one phase constitutes the input to the next one. Each phase, in turn, is structured as a set of activities that might be executed by different people concurrently. Each phase of activity is stated below [9].

- **Feasibility Study:** this evaluates the costs and benefits of the proposed application. It is necessary to analyze the problem, at least at a global level, and then try to anticipate future scenarios of software development. The result is a document called a *feasibility study document* that should contain at least the following items:

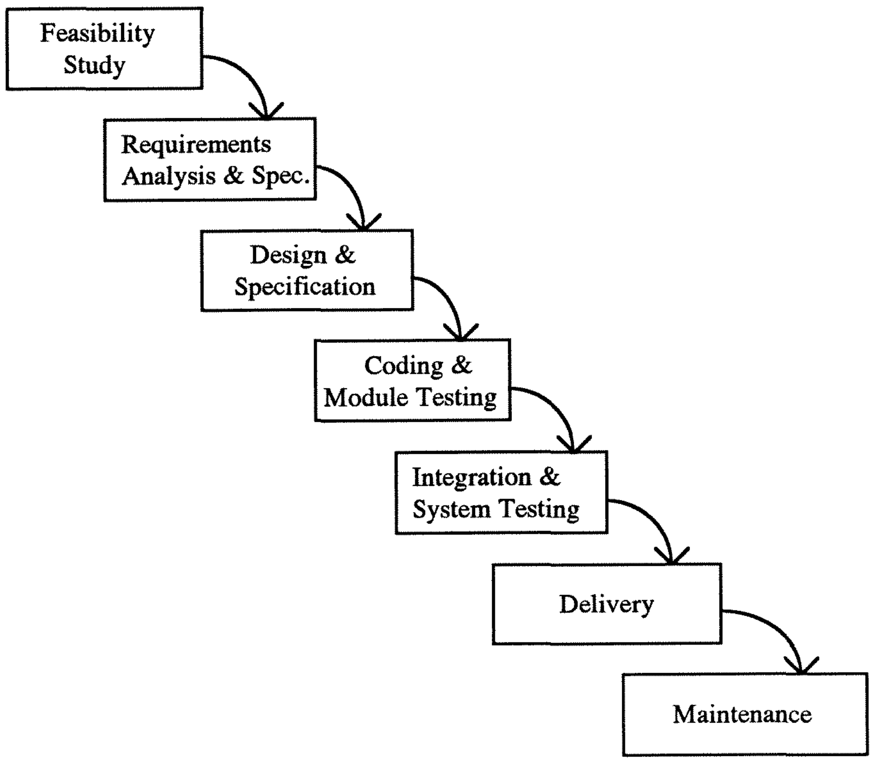


Figure 6.1 The Waterfall Software Life-Cycle [9]

1). A definition of the problem;
2). Alternative solutions and their expected benefits;
3). Require resources, costs, and delivery for each proposed alternative solution.

- **Requirements Analysis and Specification:** this states *what* qualities the application must exhibit, *not how* these qualities are achieved by design and implementation. For example, it should include definitions of what functions the software must be provided, without stating how a certain module structure or an algorithm may help. The requirements should not constrain the software engineer in the design and implementation activities. The requirements specification should include the following.

  1). Functional requirements, which describe what the product does by using informal, semiformal, formal notations, or a suitable mixture;
  2). Non-functional requirements, which include the following categories: reliability (availability, integrity, security, safety, etc.), accuracy of results, performance, human-computer interface issues, operating constraints, physical constraints, portability issues, and others;

3). Requirements on the development and maintenance process, these include quality control procedures - in particular, a system test procedure - priorities of the required functions, likely changes to the system maintenance procedures, and other requirements

- **Design and Specification:** the design involves decomposing the system into modules. The result is a *design specification document*, which contains a description of the software architecture or what each module is intended to do and the relationships among modules. The exact format of the design specification document is usually defined as a part of company wide standards. The standards may also indicate suggested design methods and practice, along with notations that should be used to document the design.

- **Coding and Module Testing:** this actually writes programs using a programming language. The output of this phase is an implemented and tested collection of modules. It was the only recognized development phase in early developmental processes. Module testing is the main quality control activity that is carried out in this phase.

- **Integration and System Testing:** Integration amounts to assembling the application from the set of components that were developed and tested separately. This phase is not always recognized as being separate from coding the program. In fact, the use of incremental development may provide progressive integration and testing of components as they are developed. The difference between the two is that the coding phase deals with programming on a small scale, while integration deals with programming on a large scale. The system testing is to test the system under realistic conditions.

- **Delivery and Maintenance:** maintenance is a set of activities that are performed after the system is delivered to the customer. Basically, it consists of correcting any remaining errors in the system, adapting the application to changes in the environment, and improving, changing, or adding features and qualities to the application.

A Critical Evaluation of the Waterfall Model

The waterfall model has played an important role because it has imposed much-needed discipline on the software development process, thus overcoming unstructured code-and-fix processes. The model has made two fundamental contributions to the understanding of software processes [9].

1). The software development process should be subject to discipline, planning, and management;

2). Implementing the product should be postponed until after the objects of doing so are well understood.

The waterfall model provides a phased view of the software life-cycle. It is based on the assumption that software development may proceed in a linear fashion from analysis down to coding. The waterfall model is monolithic in the sense that all planning is oriented to a single delivery date [2]. All analysis is performed before any designing and implementation is done.

In practice, this is not realistic, because the development of a software system requires constant feedback, and disciplined forms of feedback loops should be accounted for. If mistakes are made during the analysis, and these mistakes are not caught during analysis, then these errors be identified only after delivery of the system to the user. Moreover, since the development process may be long for complex applications - perhaps years - the application may be delivered when the user's needs have changed. Thus, all these will require immediate rework on the application.

Another underlying assumption of the waterfall model is phase rigidity, that the result of each phase are frozen before proceeding to the next phase [9]. This assumption does not recognize the need for customer-developer interaction regarding the requirements throughout the life cycle.

Some activities, however, are ongoing and span the entire life-cycle. Among these activities are documentation, verification, and management. Documentation is

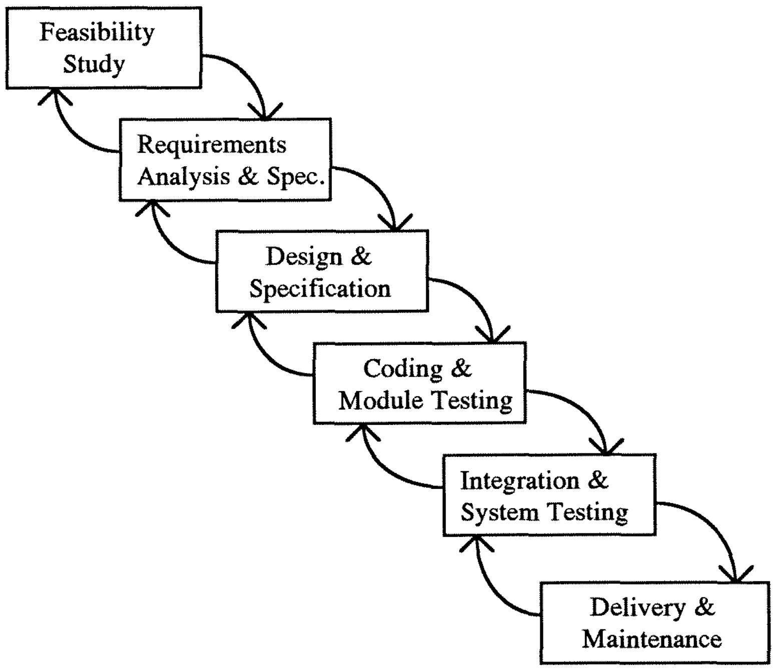


Figure 6.2 The Waterfall with Feedback [9]

intrinsic to the waterfall model, since most deliverables of the various phases are in fact documents. Based on the output, a transition to the next phase may be permitted or denied. Indeed, the waterfall process may be called a *document driven* method [3].

Verification is performed as a process of quality control, and is done at every stage on various kinds of activities even though we only singled out two specific phases where verification is performed (module testing and system testing). Management is a fundamental activity that shapes and monitors the entire development and maintenance process.

Software evolution is vital, but it is rarely anticipated nor planned [1]. Thus, it is usually done under pressure and within limited budget. Moreover, since the system that is eventually delivered may not match the user's expectations, maintenance must start immediately. This brings the question of who is responsible for the additional costs incurred during maintenance?

The reasons for the high maintenance costs of today's software systems can be traced to the characteristics of the waterfall model. In particular, because it is difficult to produce complete and correct requirement specification, this results in greater maintenance later. In fact, much of maintenance amounts to eliminate requirements errors, such as, introducing into the system exactly those functions that the user wants, but that were disregarded or misunderstood in the first place during requirement analysis and specifications [9].

To allow explicit and disciplined feedback, a revised waterfall life-cycle model is introduced as shown in Figure 6.2 [9]. It confines the feedback loops to the immediately preceding stages, in order to minimize the amount of rework involved in unconstrained repetition of previous phases.

In conclusion, the waterfall model has introduced much discipline into the software development process, but this discipline is accomplished through too much rigidity. This rigidity, in turn, introduces new problems into the process, especially when the software is being developed for poorly understood requirements [9].

6.2 Spiral Model

The spiral model creates a risk-driven approach to the software process rather than a primarily document-driven or code-driven process. The spiral model may be viewed as a *metamodel*, as shown in Figure 6.3 [5], because it can be accommodated with any process development life-cycle model (e.g. waterfall).

Risks are potentially adverse circumstances that may impair the development process and the quality of products. Risk management is a discipline whose objectives are to identify, address, and eliminate software risk items before they become either threats to successful software operation or a major source of expensive software rework [5].

The main characteristic of the spiral model is that it is cyclic and not linear like waterfall model. Each cycle of the spiral consists of four stages, and each stage is represented by one quadrant of the Cartesian diagram. The radius of the spiral represents the cost accumulated so far in the process; the angular dimension represents the progress in the process [9]. The spiral model focuses on identifying and eliminating high-risk problems through a carefully processed design, rather than treating both trivial and severe problems uniformly. Here a four stage frame was presented to develop a system under the spiral model [5].

- **Stage 1**: identifies the objectives of the portion of the product under consideration, in terms of qualities to achieve. Furthermore, it identifies alternatives - such as whether to buy, design, or reuse any of the software - and the constraints on the application of the alternatives. The alternatives are then evaluated in stage 2.

- **Stage 2**: The alternatives in stage 1 are evaluated and the potential risk areas are identified and dealt with. Risk assessment may require different kinds of activities to be planned, such as prototyping or simulation.

- **Stage 3**: consists of developing and verifying the next level product, the strategy followed by the process is dictated by risk analysis also.

- **Stage 4**: consists of reviewing the results of the stages traversed so far and planning for the next iteration of the spiral, if any.

A Critical Evaluation of the Spiral Model

An important feature of the spiral model is that each cycle is completed by a review involving the primary people or organizations concerned with the project [5]. The spiral model accommodates the favorable features of the existing software process model, and its risk driven approach avoids many difficulties.

It focuses on options for reusing of existing software early in the process, and provides a mechanism for incorporating software quality objectives into the software product development.

This model provides a viable framework for integrated software system development and eliminates errors and unattractive alternatives early [9]. It accommodates preparation for life-cycle evolution, growth, and changes of the software product.

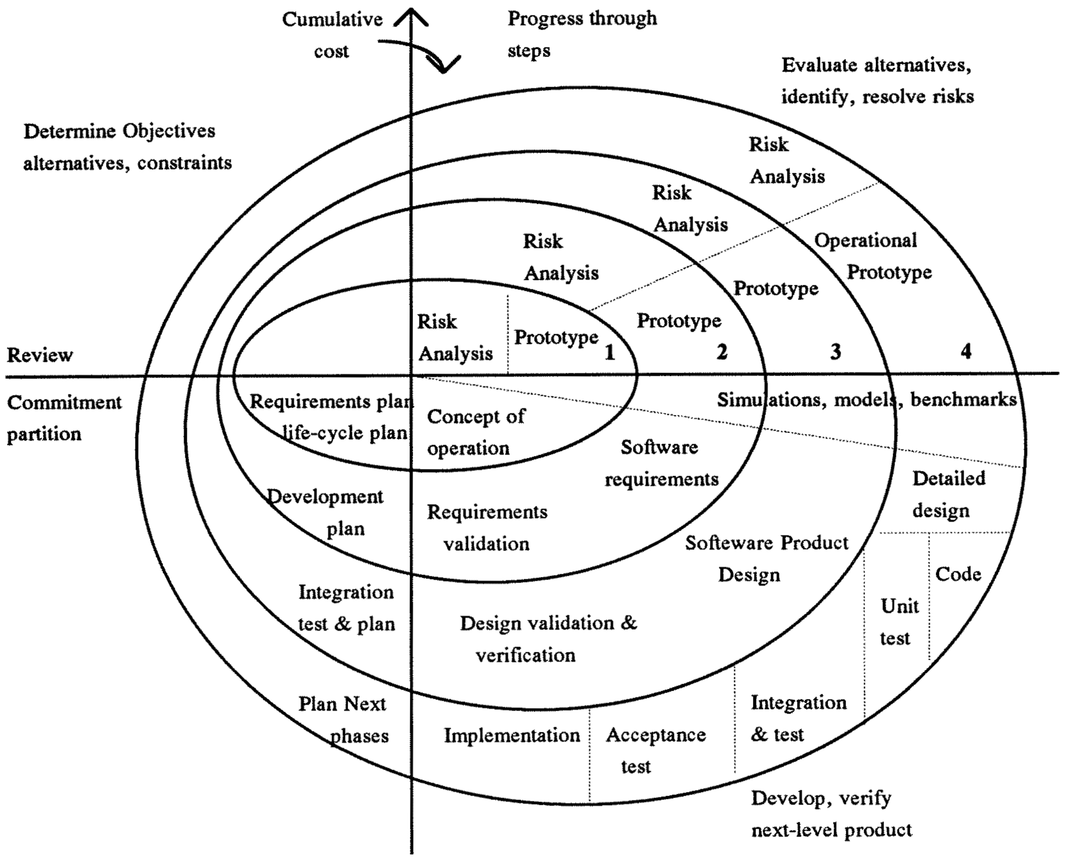


Figure 6.3 The Spiral Model [5]

Challenges still exist regarding the spiral model. It relies on risk-assessment expertise to identify and manage sources of project risk. However, it is sometimes difficult to estimate the risk-assessment. The spiral model needs further elaboration of its steps to ensure that all software development participants operate in a consistent context [5].

If the requirements for the application are understood reasonably well, a conventional waterfall model may be chosen, which leads to a simple one-turn spiral. In less understood end-user applications, however, the next step may be evolutionary in nature, several spiral turns may be required in order to achieve the desired results [9].

6.3 Fountain Model

Through the discussion above, some of the general concepts of the software development life-cycle can be clear. The overall life-cycle should take into account implicitly a high degree of overlap and iteration, although the basic activities in the software development are almost the same, i.e. analysis of user requirements, design of the system, and implementation and maintenance of the system. The fountain model, as shown in Figure 6.4 [8], provides a vehicle to implement the system in an iteration viewpoint and meanwhile emphases on the early phases of system development.

Fountain model represents the activities of different software development phases by the overlap circle. The phases in the model can be related to those of the waterfall model.

A Critical Evaluation of the Fountain Model

The fountain model provides a diagrammatic version of the development phases present in the software life-cycle and a clearer representation of the iteration and overlap in the system development [8].

The foundation of a successful software project is its requirements analysis and specification; this phase has been placed at the base in the fountain model. The life-cycle thus grows upward to a pinnacle of software use, falling only in terms of necessary maintenance [8]. The fountain model effectively emphasizes the early phases

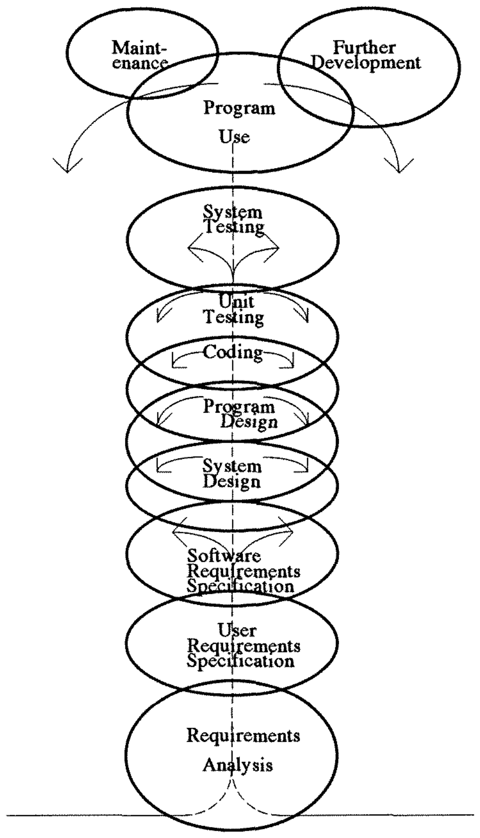


Figure 6.4 The Fountain Model [8]

of the software development life-cycle.

The modification of a software system can be more easily made interactively between requirements analysis and system design since designs are not based upon the first decisions made. Contrasting to the top-down functional design, it means there is no longer a need to freeze the overall systems requirement specification at an early phase of the system life-cycle.

6.4 Object-Oriented Model

As mentioned above, traditional software development life-cycle consists of three major activities: analysis, design and implementation. However, the Object-Oriented software life-cycle blurs the boundary of these activities because the items of interest in Object-Oriented design are objects and information hiding. Objects are independent entities, which may readily be changed because all states and representative information is held within the object itself. The information hiding is a design strategy in which as much information as possible in hidden with design components [9]. The following is a specification of a seven-step methodological framework for Object-Oriented system development [6].

- **Step 1:** Undertake system requirements specification. This step is a high-level analysis of the system in terms of objects and their services, as opposed to the system functions. An Object-Oriented requirements specification includes timing details, hardware usage, cost estimates and other documentation.

- **Step 2:** Identify the objects (entities) and the services each can provide (interface). At both the analysis and the high-level design stage, it is necessary to identify the objects or entities, their attributes and the services they provide. Objects can often be identified in terms of the real-world objects, like abstract nouns, which provide excellent objects. In this step, the functional features are defined, including defining the visible interface, although no indication of implementation is required.

- **Step 3:** Establish interactions between objects. For this step, the services required and services rendered will be defined.

- **Step 4:** Analysis stage merges into design stage to illustrate more internal details of the objects. From this step onward, bottom-up concerns should be taken into consideration. The identification of reusable design components, or classes, from previous designs is an important part of the Object-Oriented strategy.

- **Step 5:** Bottom-up concerns, using of library classes. The libraries contain object classes created as one of the successful outcomes of a previous application of this (or other) proposed development methodology. Initially, implementation (coding plus testing) of low-level classes may begin at this step.

- **Step 6:** Introduce hierarchical inheritance relationships as required. As more objects are identified within the detailed design, reevaluation of the total set of classes will require an iterative analysis of whether new super classes or new subclasses will be useful. Thus, creating a need for inheritance diagrams.

- **Step 7:** Aggregation and/or generalization of classes. As undertaken in the previous step, it may require iteration back to consider the relationship between objects.

In conclusion, Object-Oriented model follows this sequence:

1). Identifies the classes;
2). Assigns attributes and behavior;
3). Finds relationships between the classes;
4). Arranges the classes into hierarchies.

While these steps are being performed in the order shown, remember that Object-Oriented design is an iterative process. Each step in the process may alter the assumptions used in a previous step, which requires to go back and repeat that step with new information.

A Critical Evaluation of the Object-Oriented Model

Object-Oriented design differs dramatically from the tradition waterfall model. In an Object-Oriented design, it is unnecessary to analyze a problem in terms of tasks or processes. It is also unnecessary to describe the problem in terms of data. Instead of asking "What data does the program act upon?", it is necessary to ask firstly "What are the objects?" or "What are the active entities in this system?". The problem is analyzed as a system of objects interacting.

Problems with traditional development using the classical life-cycle include no iteration, no emphasis on reuse, and no unifying model to integrate the phases. Each system is built from scratch and maintenance costs account for a notoriously large share of total system costs [8]. The Object-Oriented paradigm addresses each of these issues. Inheritance facilitates extendibility and reuse within a given system, but also supports reuse across systems. Information-hiding guidelines dictate that all data within a class be private [20]. It differs from the more familiar functional approach to design in that it views a software as set of interacting objects, with their own private state, rather than as a set of functions. That is to guarantee that the interface of the class is in fact an abstraction.

The Object-Oriented model gives more attention to data specifications than the procedural approach but still utilizes functional decomposition to develop the architecture of a system [19]. Objects and their relationships between other objects are identified in both the analysis and design phases. Analysts, designers and programmers are working with a common set of items upon which to build [14].

Procedural decomposition is a top-down approach, starting with an abstract view of the system and ending with a detailed view. However, the Object-Oriented design is not a top-down technique. It does not require identifying large classes and breaking them down into smaller ones. It is also not necessarily a bottom-up process, where the system starts with small classes and builds up from them. Object-Oriented design involves working at both high and low levels of abstraction at all stages.

An Object-Oriented view of system design, however, is not always the most natural. At some levels of abstraction, a functional view is easier to derive from system requirements than an Object-Oriented view. In particular, where the system retains only minimal state information, a functional rather than an Object-Oriented design may be used [14].

7. Comparison of Life-Cycle used in Export Shipping System to Literature with Suggestions for Improvement

Normally for a large-scale information system, the waterfall model with feedback is commonly used to build the system. This is the model employed in the Export Shipping System described in this paper. The waterfall model provides a framework for people to think about the problem and to control the progress of each developmental phase. It is unusual for a software designer to arrive at a final design graph immediately. The design process appears to be a process of adding formality as the design progresses with constant backtracking to correct earlier and less formal designs [9].

We first analyzed the functional requirements of the Export Shipping System to get a rough idea of the system requirements. Then we used Look & Feel screen design tool to prototype the basic functions and screens for the user evaluation. After the user approved the initial design, we worked out the details of the program.

Most initial considerations were regarding implementation of the functional requirements, when the Export Shipping System was started. As these requirements were implemented, however, we were often forced to rethink the reasonable response-time and good program structure. Thus, the re-programming part of the source codes was often required to improve these considerations. This iteration also provided a significant impact on improving individual programming skills.

Response-time requirements are derived from the tasks required of the system users or emerge simply from human factors considerations [17]. If the response-time is too long, the user will neither like the system nor use it. During our second reviewing of the source codes and enhancement of the system, much of the effects were to improve the response-time, for example, creating a new index in the database tables or changing the retrieve control structure.

We also found that in most of the modules in our system the internal control structure can be represented as: Read data from database, process these data, then save the data back to the database. In each of these three parts there exists much redundant source code. It seems if we can consider reusing part of the source code through the Object-Oriented methodology before the design is implemented, it will reduce much of the repetitive work.

As the last feature of our system, we provided on-line help for when the user is confused. In an in-house system, it may not be necessary that the help function be available at every conceivable interaction. Because the system is required to respond to a help request at any instant and to provide assistance with request to the immediate context of the request, it takes up too much memory and slows down the executive. Perhaps help needs to be available only at major turning points, for example, when the user has the ability to switch from one screen to another [17]. This would save significant memory space in the system.

The system design or database model design is not a straight top-down or bottom-up approach. At the design level the designer can not foresee all of the technical difficulties and can not take into account all detailed aspects of the project, this requires repeating several cycle during the system development. For example, we combined the creating crate function and editing crate function into one module in programming implementation, but at the main menu choice stage we provided two separate function choices. The reason is that the basic control structure of these two functions was almost the same. However, it is reasonable to think of them as individual functions if the top-down method is used. The bottom-up approach is much more drawn out and tends to hide the overall picture behind a purely mechanical technique. Hence, perhaps the most successful approach is to combine the these two techniques, and use both approaches when most appropriate.

A significant learning curve is involved in writing a large-scale application system. Programming skills are gained through the application development. From the Export Shipping System, the following points could be made:

- We were not concerned about reusing parts of the source codes when we first started the system. Similar code was used several places in the system. It caused considerable redundant work, since each sub-system had to be built from scratch.

- We did not use risk analysis at each step before moving to the next step. Incorporating a spiral model would be a good practice to catch problems as the development progressed. If any major problems had occurred, such as incurring too much expense, this would not have been caught until the very end of the project.

- It is most realistic to recognize that no one life-cycle can necessarily solve all of the problems. It is prudent to be aware of several alternatives and to have a thorough understanding of perhaps two or more options which can be applied as circumstances direct. Often, the experience gained through out the project development will help to overcome the shortcomings of only applying one software life-cycle to the system development.

8. Conclusion

This paper has discussed the development procedure of the Export Shipping System. The main objective of this program is to develop a reliable, effective and easy to use system. The waterfall with feedback loop methodology was employed. By following the data flow through the system, we were able to systematically develop the system from the initial receiving order function to the finally generating invoices and accounting report functions.

The waterfall model with feedback loop is probably best suited to the development of information systems, especially for individual systems built from scratch. However, it does not have the risk assessment analysis, such as economic considerations, and reusability of the source code. The spiral model does not provide an adequate framework to systematically work out problems. Thus, a more feasible approach would be to combine the spiral model concepts with the waterfall model. The fountain model is a revised waterfall model and puts the requirement analysis at the basis in order to emphasize the early stages of the software development life-cycle. The Object-Oriented model builds the program on the objects of a system and places emphasis on the reuse and object hierarchies. An Object-Oriented view of an information system, however, is not always the most suitable because sometimes a functional view point is more appropriate.

In summary, none of the four software development life-cycles discussed in this paper are all inclusive and thus no one approach is used solely. In practical circumstances, a hybrid or combination of life-cycles is used to enable the programmer to design, develop and implement the information system most efficiently.

Acknowledgments

This paper benefited from comments and suggestions by Prof. Douglas Troy, Dr. James Kiper and Dr. Alton Sanders. Special thanks to Mr. Dave Spencer, who is a system analyst at Square D Co. and is my primary source for the project, for providing me the opportunity to pursue the development of a large-scale project. Without his help, this paper would not have been possible, so I dedicate this paper to his memory.

9. Bibliography

1. Alan M. Davis, *Software Requirements - Analysis & Specification*, Prentice-Hall Inc., 1990.

2. Alan M. Davis, "A Taxonomy for the Early Stages of the Software Development Life Cycle", Journary of System Software, Aug., 1988, pp. 297 - 311.

3. Alan M. Davis, "A Strategy for Comparing Alternative Software Development Life Cycle Models", IEEE Transaction of Software Engineering, 14 (1988), pp. 1453 - 1460.

4. Andrew Harbert, "A Graphical Specification System for User-Interface Design", IEEE Software, July 1990, pp. 12 - 20.

5. Barry W. Boehm, "A Spiral Model of Software Development and Enhancement", Computer, May 1988, pp. 61 - 72.

6. Bjarne Stroustrup, *The C++ Programming Language*, Addison-Wesley Co., 1987.

7. Bjarne Stroustrup, "What is Object-Oriented Programming?", IEEE Software, May 1988, pp. 10 - 20.

8. Brian Henderson, "Objected-Oriented Systems Life Cycle", Communication of the ACM, Vol. 33, No. 9, Sept. 1990, pp. 143 - 159.

9. Carlo Ghezzi, *Fundamentals of Software Engineering*, Prentice-Hall Inc., 1991.

10. D.S.Bowers, *From Data to Database*, Van Nostrand Reinhold (UK), 1988.

11. Ed Lee, "User-Interface Development Tools", IEEE Software, May 1990, pp. 31 - 36.

12. Edward H. Bersoff, "Impacts of Life Cycle Models on Software", Communication of ACM, Aug. 1991, pp. 104 - 118.

13. Gerhard Fischer, "Human-Computer Interaction Software: Lessons Learned, Challenges Ahead", IEEE Software, Jan. 1989, pp. 44 - 52.

14. Ian Sommerville, *Software Engineering*, Third Edition, Addison-Wesley Publishing Co., 1989.

15. Jeannette M. Wing, "A Specifier's Introduction to Formal Methods", Computer, Sept. 1990, pp. 8 - 24.

16. Nabajyoti Barkakati, *Microsoft C Bible*, The Waite Group's, SAMS Publish, 1990.

17. Michael F. Rothstein and Burt Rosner, *The Professional's Guide to Database Systems Project Management*, John Wiley & Sons, Inc., 1990.

18. Russell J. Abbott, *Software Development*, John Wiley & Sons, Inc., 1986.

19. Stephen Fickas, "Critiquing Software Specifications", IEEE Software, Nov. 1988, pp. 37 - 47.

20. Tim Korson, "Understanding Object-Oriented: A Unifying Paradigm", Communication of the ACM, Vol. 33, No. 9, Sept. 1990, pp. 40 - 60

21. Steve Schustack, *Variations in C*, Microsoft Press, 1989

22. Victor R. Basili, "Viewing Maintenance as Reuse-Oriented Software Development", IEEE Software, Jan. 1990, pp. 19 - 25.

23. *C-scape Interface Management System Manul*, Oakland Group, Inc., 1989

24. *Btrieve Programmer's Manul*, Novel Incorporate, 1990

25. *Xtrieve Plus*, Novel Incorporate., 1990

10. Appendix

A. Btrieve Database Table Definitions

B. NMAKE & LINK.LRF of Export Shipping System

C. Picking List, Packing List, Invoice and Accounting Reports

Appendix A: Btrieve Database Table Definitions

```
Xtrieve 4.10                                              Dictionary Print Utility
File : T06 CustOrdr Items                                                  Page   1

Location          : TABLE006.BTR
System File       : No

Page Size         : 2560
Variable Records  : No
Total Records     : 1143
Unused Pages      : 0


FIELD DEFINITIONS

Position Key Name                    Type     Size Dec Delimiter Case
-------- --- -------------------- -------- ----- --- --------- ----
       1  *  Sqd Order No         String       7                 Yes
       8  *  Order Item Seq No    Integer      2
      10  *  Item No              String      17                 Yes
      27     Phys Desc            String      12                 Yes
      39     Phys Desc w/Notes    String      12                 Yes
      51     Order Qty            Integer      4
      55  *  Qty Due              Integer      4
      59     Qty in Shppng        Integer      4
      63     Qty Invoiced         Integer      4
      67     Qty Shipped          Integer      4
      71     Distr Net Mult       Float        4
      75  *  Shpmt Ctrl No        Integer      2
      77  *  Item Letter          String       4                 Yes
      81     Last Update          Date         4
      85     Line Code            String       4                 Yes
      89     List Price           Float        8
      97     Net Mult             Float        4
     101     Order Split          Integer      2
     103  *  Prom Date            Date         4
     107     Status Bits          String       1                 Yes
     108     Item Marks           String      20                 Yes


INDEX DEFINITIONS

Key Name                 Field                Type     Dups Mod Case Asc  Total
--- -------------------- -------------------- -------- ---- --- ---- ---  ------
  0                      Sqd Order No         String     No Yes   No Yes   1143
                         Order Item Seq No    Integer    No Yes   No Yes
  1                      Sqd Order No         String    Yes Yes   No Yes    919
                         Item No              String    Yes Yes   No Yes
  2                      Sqd Order No         String     No Yes   No Yes   1143
                         Item Letter          String     No Yes   No Yes
  3                      Sqd Order No         String    Yes Yes   No Yes    255
                         Qty Due              Integer   Yes Yes   No  No
  4                      Sqd Order No         String    Yes Yes   No Yes    261
                         Shpmt Ctrl No        Integer   Yes Yes   No Yes
                         Qty Due              Integer   Yes Yes   No  No
  5                      Shpmt Ctrl No        Integer   Yes Yes   No Yes    104
```

```
"DEX DEFINITIONS (Continued)

.ey  Name                  Field                  Type     Dups Mod Case Asc  Total
---  --------------------  --------------------  --------  ---- --- ---- ---  ------
                           Sqd Order No          String     Yes Yes   No Yes
                           Prom Date             Date       Yes Yes   No Yes


'IELD ATTRIBUTES

'ield Name           Attr. Type Attribute Value
-------------------- ---------- ----------------------------------------------------------------
```

```
Xtrieve 4.10                                                    Dictionary Print Utility
File : T20 Cstmr Orders                                                         Page   1

  cation           : TABLE020.BTR
  stem File        : No

Page Size          : 512
Variable Records   : No
Total Records      : 434
Unused Pages       : 0


FIELD DEFINITIONS

Position Key Name                    Type     Size Dec Delimiter Case
-------- --- -------------------- -------- ----- --- --------- ----
       1  *  Sqd Order No         String       7                 Yes
       8     Cstmr Accnt No       String       5                 Yes
      13     Cstmr Purch Order No String      20                 Yes
      33     Date Recvd           Date         4
      37     Dest Terr            String       3                 Yes
      40     Infl Terr            String       3                 Yes
      43     Info Needed          String      20                 Yes
      63     Job Name             String      39                 Yes
     102     MD509 No             String       8                 Yes
     110     Order Terr           String       3                 Yes
     113     Quoted Billg         Float        8
     121     Trbl File No         String       8                 Yes
     129     Ship to Cntry Code   String       3                 Yes
     132     Sold to Cntry Code   String       3                 Yes
     135     Status               Integer      2
     137     Terms                String      30                 Yes
     167     Complete/Close Date  Date         4
     171     Shppng Marks         String      30                 Yes
     201     Cmnts                String      40                 Yes


INDEX DEFINITIONS

Key Name                 Field                   Type    Dups Mod Case Asc  Total
--- -------------------- -------------------- -------- ---- --- ---- --- ------
  0                      Sqd Order No         String     No Yes   No Yes    434


FIELD ATTRIBUTES

Field Name           Attr. Type Attribute Value
-------------------- ---------- -------------------------------------------------
```

```
Xtrieve 4.10                                               Dictionary Print Utility
File : T21 Exprt BillgItm                                                 Page    1

  ation          : TABLE021.BTR
  stem File      : No

Page Size        : 2560
Variable Records : No
Total Records    : 4635
Unused Pages     : 0


FIELD DEFINITIONS

Position Key Name                 Type     Size Dec Delimiter Case
-------- --- -------------------- -------- ----- --- --------- ----
       1  *  Sqd Order No         String       7                Yes
       8  *  Shpmt Ctrl No        Integer      2
      10  *  Invoice Seq No       Integer      2
      12     Back Order Qty       Integer      4
      16  *  Crate No             Integer      2
      18     Item No              String      17                Yes
      35     Distr Net Mult       Float        4
      39  *  Item Letter          String       4                Yes
      43     Line Code            String       4                Yes
      47     List Price           Float        8
      55     Net Mult             Float        4
      59     Norm List Price      Float        8
      67     Order Qty            Integer      4
      71     Phys Desc            String      12                 No
      83     Invoice Qty          Integer      4
      87     Unit Matl Cost       Float        8
      95     Unit Lab Cost        Float        8
     103     Unit Fixed Bur Cost  Float        8
     111     Unit Var Bur Cost    Float        8


INDEX DEFINITIONS

Key  Name                 Field                Type     Dups Mod Case Asc  Total
---  -------------------- -------------------- -------- ---- --- ---- --- ------
  0                       Sqd Order No         String     No Yes   No Yes   4635
                          Shpmt Ctrl No        Integer    No Yes   No Yes
                          Invoice Seq No       Integer    No Yes   No Yes
  1                       Sqd Order No         String    Yes Yes   No Yes   2099
                          Crate No             Integer   Yes Yes   No Yes
  2                       Sqd Order No         String    Yes Yes   No Yes   4220
                          Item Letter          String    Yes Yes   No Yes


FIELD ATTRIBUTES

Field Name           Attr. Type Attribute Value
-------------------- ---------- --------------------------------------------------
  str Net Mult       Mask       Z.ZZZZ
```

```
Xtrieve 4.10                                                      Dictionary Print Utility
File : T21 Exprt BillgItm                                                          Page   2

FIELD ATTRIBUTES (Continued)

Field Name            Attr. Type Attribute Value
--------------------- ---------- -----------------------------------------------------------
List Price            Mask       ZZZZZZZ.ZZ
Net Mult              Mask       Z.ZZZZ
Norm List Price       Mask       ZZZZZZZ.ZZ
Unit Matl Cost        Mask       ZZZZZZ.ZZZZ
Unit Lab Cost         Mask       ZZZZZZ.ZZZZ
Unit Fixed Bur Cost   Mask       ZZZZZZ.ZZZZ
Unit Var Bur Cost     Mask       ZZZZZZ.ZZZZ
```

```
Xtrieve 4.10                                                       Dictionary Print Utility
File : T26 Exprt ShpmtHdr                                                        Page    1

  cation          : TABLE026.BTR
  stem File       : No

Page Size         : 1024
Variable Records  : No
Total Records     : 684
Unused Pages      : 0


FIELD DEFINITIONS

Position Key Name                   Type     Size Dec Delimiter Case
-------- --- -------------------- -------- ----- --- --------- ----
       1  *  Sqd Order No         String       7                Yes
       8  *  Shpmt Ctrl No        Integer      2
      10     Air Frt Charge       Float        8
      18     B/L /Bookng/Rcpt No  String      20                Yes
      38     Cntnr No             String      20                Yes
      58     Courier Charge       Float        8
      66     Cubic Ft             Float        4
      70     Dest Terr            String       3                 No
      73     Estm Time Arrival    String      10                Yes
      83     Export Invoice No    String       5                Yes
      88     Frt Foreward Charge  Float        8
      96     Gross Kilos          Integer      4
     100     Gross Lbs            Integer      4
     104     Infl Terr            String       3                 No
     107     Intl Accnt No        String       5                Yes
     112     Intl Purch Order No  String      20                Yes
     132  *  Invoice Date         Date         4
     136     No of Bundles        Integer      2
     138     No of Cartons        Integer      2
     140     No of Crates         Integer      2
     142     No of Pallets        Integer      2
     144     Ocean Frt Charge     Float        8
     152     Order Terr           String       3                 No
     155     Other Charge-1       Float        8
     163     Other Charge-2       Float        8
     171     Other Charge-3       Float        8
     179     Other Charge Desc-1  String      20                Yes
     199     Other Charge Desc-2  String      20                Yes
     219     Other Charge Desc-3  String      20                Yes
     239     Other Total          Float        8
     247     Other Total Desc     String     120                 No
     367     Pack Charge          Float        8
     375     Pack Date            Date         4
     379     Partial/Cmplt        String       1                Yes
     380     Prepaid/Collect      String       1                Yes
     381     Route Via            String      35                Yes
     416     Sailing Date         Date         4
     420     Shpmt Header Name    String       5                Yes
     425     Ship From Loc        String       3                Yes
```

ELD DEFINITIONS (Continued)

```
osition Key Name                 Type     Size Dec Delimiter Case
-------- --- -------------------- -------- ----- --- --------- ----
     428     Shipped Via          String     25                Yes
     453     Shppng Cmnts         String     20                Yes
     473     Shppng Method        String      1                 No
     474     Total Desc           String     40                 No
     514     USA Port             String     25                Yes
     539     Vessel               String     20                Yes
     559     Ship to Name/Addr    String    200                 No
     759     Shppng Marks         String    180                Yes
```

INDEX DEFINITIONS

```
Key Name                 Field                Type     Dups Mod Case Asc  Total
--- -------------------- -------------------- -------- ---- --- ---- --- ------
  0                      Sqd Order No         String     No Yes   No Yes    684
                         Shpmt Ctrl No        Integer    No Yes   No Yes
  1                      Invoice Date         Date      Yes Yes   No Yes    684
                         Shpmt Ctrl No        Integer   Yes Yes   No Yes
                         Sqd Order No         String    Yes Yes  Yes Yes
```

FIELD ATTRIBUTES

```
ield Name           Attr. Type Attribute Value
-------------------- ---------- ----------------------------------------------------------------
```

```
:trieve 4.10                                                      Dictionary Print Utility
'ile : T27 Exprt CrateItm                                                        Page    1

 cation          : TABLE027.BTR
 stem File       : No

?age Size        : 3584
/ariable Records : No
Total Records    : 4508
Jnused Pages     : 0


FIELD DEFINITIONS

Position Key Name                    Type     Size Dec Delimiter Case
-------- --- -------------------- -------- ----- --- --------- ----
       1  *  Sqd Order No         String       7                 Yes
       8  *  Crate No             Integer      2
      10  *  Order Item Seq No    Integer      2
      12     Crate Qty            Integer      4
      16     Last Update          Date         4


INDEX DEFINITIONS

Key  Name                 Field                Type     Dups Mod Case Asc  Total
---  -------------------- -------------------- -------- ---- --- ---- --- ------
 0                        Sqd Order No         String     No Yes   No Yes   4508
                          Crate No             Integer    No Yes   No Yes
                          Order Item Seq No    Integer    No Yes   No Yes


FIELD ATTRIBUTES

Field Name           Attr. Type Attribute Value
-------------------- ---------- --------------------------------------------------------------
```

```
trieve 4.10                                                    Dictionary Print Utility
ile : T28 Exprt Crates                                                      Page    1

 ation           : TABLE028.BTR
 stem File       : No

age Size         : 2560
ariable Records  : No
otal Records     : 1715
nused Pages      : 0


IELD DEFINITIONS

osition Key Name                 Type     Size Dec Delimiter Case
-------- --- -------------------- -------- ----- --- --------- ----
       1  *  Sqd Order No         String       7               Yes
       8  *  Crate No             Integer      2
      10     Contnr Type          String       3               Yes
      13     Date Recvd in Shppng Date         4
      17     Height               Integer      2
      19     Lgth                 Integer      2
      21     Loc                  String       7               Yes
      28  *  Status               String       1               Yes
      29     Wgt                  Integer      2
      31     Width                Integer      2
      33     Last Update          Date         4
      37  *  Shpmt Ctrl No        Integer      2


NDEX DEFINITIONS

Key Name                 Field                Type     Dups Mod Case Asc  Total
--- -------------------- -------------------- -------- ---- --- ---- --- ------
 0                       Sqd Order No         String     No Yes   No Yes   1715
                         Crate No             Integer    No Yes   No Yes
 1                       Sqd Order No         String    Yes Yes   No Yes    436
                         Shpmt Ctrl No        Integer   Yes Yes   No Yes
 2                       Status               String    Yes Yes   No  No   1715
                         Sqd Order No         String    Yes Yes   No Yes
                         Crate No             Integer   Yes Yes   No Yes
 3                       Status               String    Yes Yes   No Yes    436
                         Sqd Order No         String    Yes Yes   No Yes
                         Shpmt Ctrl No        Integer   Yes Yes   No Yes


FIELD ATTRIBUTES

Field Name           Attr. Type Attribute Value
-------------------- ---------- ----------------------------------------------------------------
```

Appendix B: NMAKE and LINK.LRF of the Export Shipping System

```
# NMAKE Descriptor file for EXPORT.EXE
# Written 10/24/91

compiler_options = /Fo$@ -AL -c -DM5 -Zp1 -W3 -Os -I..\BTRIEVE -IMISC
compiler_options2 = /Fo$@ -AL -c -DM5 -Zp1 -W3 -I..\BTRIEVE -IMISC
link_options     = /se:512
DIR              = ..\BTRIEVE^\
TABLES           = $(DIR)TABLE006.H \
                   $(DIR)TABLE020.H \
                   $(DIR)TABLE021.H \
                   $(DIR)TABLE023.H \
                   $(DIR)TABLE026.H \
                   $(DIR)TABLE027.H \
                   $(DIR)TABLE028.H \
                   $(DIR)TABLE037.H \
                   $(DIR)TABLE038.H

EXPORT.EXE: OBJ\EXPORT.OBJ   \
            EDITCRAT\OBJ\EDITCRAT.OBJ \
            EDITCRAT\OBJ\SAVECRAT.OBJ \
            EDITCRAT\OBJ\LOADRECV.OBJ \
            EDITCRAT\OBJ\LOADEDIT.OBJ \
            EDITCRAT\OBJ\EDIT_SPC.OBJ \
            SHIPCRAT\OBJ\SHIPCRAT.OBJ \
            SHIPCRAT\OBJ\LOGSHPMT.OBJ \
            SHIPCRAT\OBJ\LOADORDR.OBJ \
            SHIPCRAT\OBJ\LOADSHIP.OBJ \
            SHIPCRAT\OBJ\PICKLIST.OBJ \
            SHIPCRAT\OBJ\SHIP_SPC.OBJ \
            TEMPLATE\OBJ\TEMPLATE.OBJ \
            TEMPLATE\OBJ\LOADTMPL.OBJ \
            TEMPLATE\OBJ\SAVETMPL.OBJ \
            TEMPLATE\OBJ\TMPL_SPC.OBJ \
            PACKLIST\OBJ\PACKLIST.OBJ \
            PACKLIST\OBJ\LOADPACK.OBJ \
            PACKLIST\OBJ\SAVEPACK.OBJ \
            PACKLIST\OBJ\PRNTPACK.OBJ \
            PACKLIST\OBJ\BLUE_TAG.OBJ \
            PACKLIST\OBJ\PACK_SPC.OBJ \
            INVOICE\OBJ\INVOICE.OBJ   \
            INVOICE\OBJ\LOADORDR.OBJ  \
            INVOICE\OBJ\EX_DECL.OBJ   \
            INVOICE\OBJ\PRINTINV.OBJ  \
            INVOICE\OBJ\PRINTCAN.OBJ  \
            INVOICE\OBJ\INV_SPC.OBJ   \
            DOCUMENT\OBJ\LOADHDR.OBJ  \
            DOCUMENT\OBJ\DOCU_SPC.OBJ \
            VEDIT006\OBJ\VEDIT006.OBJ \
            VEDIT020\OBJ\VEDIT020.OBJ \
            VEDIT021\OBJ\VEDIT021.OBJ \
            VEDIT037\OBJ\VEDIT037.OBJ \
            ERRMSG\OBJ\B_ERRMSG.OBJ   \
            ERRMSG\OBJ\O_ERRMSG.OBJ   \
            ERRMSG\OBJ\D_ERRMSG.OBJ   \
            MISC\OBJ\LABEL.OBJ    \
            MISC\OBJ\HP_GL2.OBJ   \
              MISC\OBJ\MISC.OBJ     \
              MISC\OBJ\TABLE.OBJ    \
              MISC\OBJ\PRINT.OBJ    \
              MISC\OBJ\MSCXBTRV.OBJ \
              MISC\OBJ\GETORDNO.OBJ
     LINK @EXPORT.LRF
     echo done!!

  OBJ\EXPORT.OBJ:              EXPORT.C \
                               EXPORT.H \
                               SYMBOL.C;
     CL /Fo$@ $(compiler_options) EXPORT.C

  EDITCRAT\OBJ\EDITCRAT.OBJ: EDITCRAT\EDITCRAT.C \
                             EDITCRAT\EDITCRAT.H \
                             MISC\MISC.H \
                             MISC\TABLE.H \
                             MISC\PRINT.H \
                             MISC\MSCXBTRV.H \
                             $(TABLES);
     CL /Fo$@ $(compiler_options) EDITCRAT\EDITCRAT.C

  EDITCRAT\OBJ\LOADEDIT.OBJ: EDITCRAT\LOADEDIT.C
     CL /Fo$@ $(compiler_options) EDITCRAT\LOADEDIT.C

  EDITCRAT\OBJ\LOADRECV.OBJ: EDITCRAT\LOADRECV.C
     CL /Fo$@ $(compiler_options) EDITCRAT\LOADRECV.C

  EDITCRAT\OBJ\SAVECRAT.OBJ: EDITCRAT\SAVECRAT.C \
                             EDITCRAT\SAVECRAT.H \
                             MISC\MISC.H  \
                             MISC\TABLE.H \
                             $(TABLES);
     CL /Fo$@ $(compiler_options) EDITCRAT\SAVECRAT.C

  EDITCRAT\OBJ\EDIT_SPC.OBJ: EDITCRAT\EDIT_SPC.C \
                             EDITCRAT\EDIT_SPC.H \
                             MISC\MISC.H \
                             MISC\TABLE.H \
                             $(TABLES);
     CL /Fo$@ $(compiler_options) EDITCRAT\EDIT_SPC.C

  SHIPCRAT\OBJ\SHIPCRAT.OBJ: SHIPCRAT\SHIPCRAT.C \
                             SHIPCRAT\SHIPCRAT.H \
                             SHIPCRAT\LOGSHPMT.H \
                             SHIPCRAT\PICKLIST.H \
                             SHIPCRAT\LOADORDR.H \
                             MISC\PRINT.H \
                             MISC\MISC.H \
                             MISC\TABLE.H \
                             $(TABLES);
     CL /Fo$@ $(compiler_options) SHIPCRAT\SHIPCRAT.C

  SHIPCRAT\OBJ\PICKLIST.OBJ: SHIPCRAT\PICKLIST.C \
                             SHIPCRAT\PICKLIST.H \
                             $(TABLES);
```

```
   CL /Fo$@ $(compiler_options) SHIPCRAT\PICKLIST.C

SHIPCRAT\OBJ\LOADORDR.OBJ: SHIPCRAT\LOADORDR.C \
                           SHIPCRAT\LOADORDR.H \
                           MISC\MISC.H \
                           MISC\TABLE.H \
                           $(TABLES);
   CL /Fo$@ $(compiler_options) SHIPCRAT\LOADORDR.C

SHIPCRAT\OBJ\LOADSHIP.OBJ: SHIPCRAT\LOADSHIP.C \
                           SHIPCRAT\LOADSHIP.H \
                           MISC\MISC.H \
                           MISC\TABLE.H \
                           $(TABLES);
   CL /Fo$@ $(compiler_options) SHIPCRAT\LOADSHIP.C

SHIPCRAT\OBJ\LOGSHPMT.OBJ: SHIPCRAT\LOGSHPMT.C \
                           SHIPCRAT\LOGSHPMT.H \
                           MISC\MISC.H \
                           MISC\TABLE.H \
                           $(TABLES);
   CL /Fo$@ $(compiler_options) SHIPCRAT\LOGSHPMT.C

SHIPCRAT\OBJ\SHIP_SPC.OBJ: SHIPCRAT\SHIP_SPC.C \
                           SHIPCRAT\SHIP_SPC.H \
                           MISC\MISC.H \
                           MISC\TABLE.H \
                           $(TABLES);
   CL /Fo$@ $(compiler_options) SHIPCRAT\SHIP_SPC.C

TEMPLATE\OBJ\TEMPLATE.OBJ: TEMPLATE\TEMPLATE.C \
                           TEMPLATE\TEMPLATE.H \
                           MISC\PRINT.H \
                           MISC\MISC.H  \
                           MISC\TABLE.H \
                           $(TABLES);
   CL /Fo$@ $(compiler_options) TEMPLATE\TEMPLATE.C

TEMPLATE\OBJ\LOADTMPL.OBJ: TEMPLATE\LOADTMPL.C \
                           TEMPLATE\LOADTMPL.H \
                           MISC\MISC.H \
                           MISC\TABLE.H \
                           $(TABLES);
   CL /Fo$@ $(compiler_options) TEMPLATE\LOADTMPL.C

TEMPLATE\OBJ\SAVETMPL.OBJ:  TEMPLATE\SAVETMPL.C \
                           TEMPLATE\SAVETMPL.H \
                           MISC\MISC.H \
                           MISC\TABLE.H \
                           $(TABLES);
   CL /Fo$@ $(compiler_options) TEMPLATE\SAVETMPL.C

TEMPLATE\OBJ\TMPL_SPC.OBJ: TEMPLATE\TMPL_SPC.C \
                           TEMPLATE\TMPL_SPC.H \
                           MISC\MISC.H \
                           MISC\TABLE.H \
                                          $(TABLES);
    CL /Fo$@ $(compiler_options) TEMPLATE\TMPL_SPC.C

PACKLIST\OBJ\PACKLIST.OBJ:    PACKLIST\PACKLIST.C \
                              PACKLIST\PACKLIST.H \
                              MISC\MISC.H \
                              MISC\TABLE.H \
                              $(TABLES);
    CL /Fo$@ $(compiler_options) PACKLIST\PACKLIST.C

PACKLIST\OBJ\LOADPACK.OBJ:    PACKLIST\LOADPACK.C \
                              PACKLIST\LOADPACK.H \
                              MISC\MISC.H \
                              MISC\TABLE.H \
                              $(TABLES);
    CL /Fo$@ $(compiler_options) PACKLIST\LOADPACK.C

PACKLIST\OBJ\SAVEPACK.OBJ:    PACKLIST\SAVEPACK.C \
                              PACKLIST\SAVEPACK.H \
                              MISC\MISC.H \
                              MISC\TABLE.H \
                              $(TABLES);
    CL /Fo$@ $(compiler_options) PACKLIST\SAVEPACK.C

PACKLIST\OBJ\PRNTPACK.OBJ:    PACKLIST\PRNTPACK.C \
                              PACKLIST\PRNTPACK.H \
                              MISC\MISC.H \
                              MISC\TABLE.H \
                              $(TABLES);
    CL /Fo$@ $(compiler_options) PACKLIST\PRNTPACK.C

PACKLIST\OBJ\BLUE_TAG.OBJ:    PACKLIST\BLUE_TAG.C \
                              PACKLIST\BLUE_TAG.H \
                              MISC\MISC.H \
                              MISC\TABLE.H \
                              $(TABLES);
    CL /Fo$@ $(compiler_options) PACKLIST\BLUE_TAG.C

PACKLIST\OBJ\PACK_SPC.OBJ:    PACKLIST\PACK_SPC.C \
                              PACKLIST\PACK_SPC.H \
                              MISC\MISC.H \
                              MISC\TABLE.H \
                              $(TABLES);
    CL /Fo$@ $(compiler_options) PACKLIST\PACK_SPC.C

INVOICE\OBJ\INVOICE.OBJ:      INVOICE\INVOICE.C \
                              INVOICE\INVOICE.H \
                              MISC\MISC.H \
                              MISC\TABLE.H \
                              $(TABLES);
    CL /Fo$@ $(compiler_options) INVOICE\INVOICE.C

INVOICE\OBJ\LOADORDR.OBJ:     INVOICE\LOADORDR.C \
                              MISC\MISC.H \
                              MISC\TABLE.H \
                              $(TABLES);
```

```
   CL /Fo$@ $(compiler_options) INVOICE\LOADORDR.C

INVOICE\OBJ\INV_SPC.OBJ:      INVOICE\INV_SPC.C \
                              INVOICE\INV_SPC.H \
                              MISC\MISC.H;
   CL /Fo$@ $(compiler_options) INVOICE\INV_SPC.C

INVOICE\OBJ\PRINTINV.OBJ:     INVOICE\PRINTINV.C \
                              INVOICE\PRINTINV.H \
                              MISC\MISC.H \
                              MISC\TABLE.H \
                              MISC\HP_GL2.H \
                              $(TABLES);
   CL /Fo$@ $(compiler_options) INVOICE\PRINTINV.C

INVOICE\OBJ\PRINTCAN.OBJ:     INVOICE\PRINTCAN.C \
                              INVOICE\PRINTCAN.H \
                              MISC\MISC.H \
                              MISC\TABLE.H \
                              MISC\HP_GL2.H;
   CL /Fo$@ $(compiler_options) INVOICE\PRINTCAN.C

INVOICE\OBJ\EX_DECL.OBJ:      INVOICE\EX_DECL.C \
                              INVOICE\EX_DECL.H \
                              MISC\MISC.H \
                              MISC\TABLE.H \
                              $(TABLES);
   CL /Fo$@ $(compiler_options) INVOICE\EX_DECL.C

DOCUMENT\OBJ\LOADHDR.OBJ:     DOCUMENT\LOADHDR.C \
                              DOCUMENT\LOADHDR.H \
                              MISC\MISC.H;
   CL /Fo$@ $(compiler_options) DOCUMENT\LOADHDR.C

DOCUMENT\OBJ\DOCU_SPC.OBJ:    DOCUMENT\DOCU_SPC.C \
                              DOCUMENT\DOCU_SPC.H \
                              MISC\MISC.H;
   CL /Fo$@ $(compiler_options) DOCUMENT\DOCU_SPC.C

VEDIT006\OBJ\VEDIT006.OBJ: VEDIT006\VEDIT006.C \
                           VEDIT006\VEDIT006.H \
                           MISC\MISC.H \
                           $(TABLES);
   CL /Fo$@ $(compiler_options) VEDIT006\VEDIT006.C

VEDIT020\OBJ\VEDIT020.OBJ: VEDIT020\VEDIT020.C \
                           VEDIT020\VEDIT020.H \
                           MISC\MISC.H \
                           $(TABLES);
   CL /Fo$@ $(compiler_options) VEDIT020\VEDIT020.C

VEDIT021\OBJ\VEDIT021.OBJ: VEDIT021\VEDIT021.C \
                           VEDIT021\VEDIT021.H \
                           MISC\MISC.H \
                           $(TABLES);
   CL /Fo$@ $(compiler_options) VEDIT021\VEDIT021.C
```

```
VEDIT037\OBJ\VEDIT037.OBJ: VEDIT037\VEDIT037.C \
                           VEDIT037\VEDIT037.H \
                           MISC\MISC.H \
                           $(TABLES);
   CL /Fo$@ $(compiler_options) VEDIT037\VEDIT037.C

ERRMSG\OBJ\B_ERRMSG.OBJ:   ERRMSG\B_ERRMSG.C;
   CL /Fo$@ $(compiler_options) ERRMSG\B_ERRMSG.C

ERRMSG\OBJ\D_ERRMSG.OBJ:   ERRMSG\D_ERRMSG.C;
   CL /Fo$@ $(compiler_options) ERRMSG\D_ERRMSG.C

ERRMSG\OBJ\O_ERRMSG.OBJ:   ERRMSG\O_ERRMSG.C;
   CL /Fo$@ $(compiler_options) ERRMSG\O_ERRMSG.C

MISC\OBJ\MSCXBTRV.OBJ:     MISC\MSCXBTRV.C \
                           MISC\MSCXBTRV.H;
   CL /Fo$@ $(compiler_options) MISC\MSCXBTRV.C


#   Problems with the roundoff function require that we turn off -Os

MISC\OBJ\MISC.OBJ:         MISC\MISC.C \
                           MISC\MISC.H \
                           MISC\TABLE.H \
                           $(TABLES);
   CL /Fo$@ $(compiler_options2) MISC\MISC.C

MISC\OBJ\TABLE.OBJ:        MISC\TABLE.C \
                           MISC\TABLE.H \
                           MISC\MISC.H \
                           $(TABLES);
   CL /Fo$@ $(compiler_options) MISC\TABLE.C

MISC\OBJ\PRINT.OBJ:       MISC\PRINT.C \
                          MISC\PRINT.H;
   CL /Fo$@ $(compiler_options) MISC\PRINT.C

MISC\OBJ\LABEL.OBJ:        MISC\LABEL.C \
                           MISC\LABEL.H \
                           $(TABLES);
   CL /Fo$@ $(compiler_options) MISC\LABEL.C

MISC\OBJ\HP_GL2.OBJ:       MISC\HP_GL2.C \
                           MISC\HP_GL2.H;
   CL /Fo$@ $(compiler_options) MISC\HP_GL2.C

MISC\OBJ\GETORDNO.OBJ:    MISC\GETORDNO.C \
                          MISC\MISC.H \
                          MISC\TABLE.H;
   CL /Fo$@ $(compiler_options) MISC\GETORDNO.C
```

```
OBJ\EXPORT   +
MISC\OBJ\LABEL    +
MISC\OBJ\HP_GL2   +
MISC\OBJ\MISC     +
MISC\OBJ\TABLE    +
MISC\OBJ\PRINT    +
MISC\OBJ\MSCXBTRV +
MISC\OBJ\GETORDNO +
EDITCRAT\OBJ\EDITCRAT +
EDITCRAT\OBJ\SAVECRAT +
EDITCRAT\OBJ\LOADRECV +
EDITCRAT\OBJ\LOADEDIT +
EDITCRAT\OBJ\EDIT_SPC +
SHIPCRAT\OBJ\SHIPCRAT +
SHIPCRAT\OBJ\PICKLIST +
SHIPCRAT\OBJ\LOGSHPMT +
SHIPCRAT\OBJ\LOADORDR +
SHIPCRAT\OBJ\LOADSHIP +
SHIPCRAT\OBJ\SHIP_SPC +
TEMPLATE\OBJ\TEMPLATE +
TEMPLATE\OBJ\LOADTMPL +
TEMPLATE\OBJ\SAVETMPL +
TEMPLATE\OBJ\TMPL_SPC +
PACKLIST\OBJ\PACKLIST +
PACKLIST\OBJ\LOADPACK +
PACKLIST\OBJ\SAVEPACK +
PACKLIST\OBJ\BLUE_TAG +
PACKLIST\OBJ\PRNTPACK +
PACKLIST\OBJ\PACK_SPC +
INVOICE\OBJ\INVOICE   +
INVOICE\OBJ\LOADORDR  +
INVOICE\OBJ\EX_DECL   +
INVOICE\OBJ\PRINTINV  +
INVOICE\OBJ\PRINTCAN  +
INVOICE\OBJ\INV_SPC  +
DOCUMENT\OBJ\LOADHDR  +
DOCUMENT\OBJ\DOCU_SPC +
(VEDIT006\OBJ\VEDIT006) +
(VEDIT020\OBJ\VEDIT020) +
(VEDIT021\OBJ\VEDIT021) +
(VEDIT037\OBJ\VEDIT037) +
ERRMSG\OBJ\B_ERRMSG +
ERRMSG\OBJ\D_ERRMSG +
ERRMSG\OBJ\O_ERRMSG
,,,M6LCSCAP M6LOWL, /se:1024 /NOE /STACK:10000 /OL
```

Appendix C: Picking List, Packing List, Invoice and Accounting Reports

ATE: 06/27/92 Form: K-500-63(02/21/92)
IME: 16:41:35

EXPORT SHIPPING SYSTEM
SQUARE D COMPANY OXFORD, OHIO
Picking List

Factory Order No.: 3025387
Container No.: ________________ Customer Order No.: EX76040
Intl Purch Order No.: 2604067

rt#	Pack	H x W x L	Weight	Location	IT	Qty	Description	
3	CRT	31 x 46 x 62	353	D140	AU	1	CP504GLTI	-9
					AV	1	CP504GO21TI	-9
4	CRT	30 x 42 x 136	3123	F102125	AZ	20	CP504G10	-11
5	CRT	26 x 42 x 136	2276	F101146	AZ	1	CP504G10	-11
					BA	1	CP504G10FBA	-11
					BB	2	CP504G10FBB	-11
					BC	2	CP504G10FBC	-11
					BD	3	CP504G10FBD	-11
					BE	1	CP504G10FBE	-11
					BF	1	CP504G10FBF	-11
					BG	1	CP504G49	-11
					BH	4	CP504G6	-11
6	CRT	31 x 46 x 64	322	F106	BI	1	CP504G69LTIFB	-11
7	CRT	42 x 42 x 42	213	F102174	R	1	CP504GLFO	-3
8	CRT	36 x 46 x 44	444	F102174	AE	3	CP506GLTI	-6
					AF	2	CP506GLTO	-6
9	CRT	36 x 42 x 136	3252	F102	AJ	8	CP504G10	-8
					AK	1	CP504G10FBA	-8
					AL	2	CP504G10FBB	-8
					AM	1	CP504G10FBC	-8
					AN	4	CP504G10FBD	-8
					AO	1	CP504G10FBF	-8
					AP	1	CP504G49	-8
					AQ	5	CP504G6	-8
10	CRT	29 x 42 x 136	2514	F101150	DM	15	CP504G10	-23
					DW	1	CP504G66	-24
11	CRT	32 x 42 x 136	3302	E112	CT	11	CP504G10	-20
					CU	1	CP504G10FBA	-20
					CV	2	CP504G10FBB	-20
					CW	1	CP504G0FBC	-20
					CX	2	CP504G10FBD	-20
					CY	1	CP504G10FBE	-20
					CZ	1	CP504G10FBF	-20
					DB	4	CP504G6	-20
		694.9 CuFt	15799 lbs 7166 Kilos			107		

SQUARE D COMPANY

PACKING LIST

SHIPPED FROM: 5735 COLLEGE CO[illegible]ER ROAD
OXFORD, OHIO 45056 U.S.A.

Form: K-500-61 (02/21/92)

SOLD TO:	SHIPPED TO:	SHIPPING MARKS:
SQUARE D COMPANY (THAILAND) LTD. 10TH FLOOR MBK TOWER 444 PHAYATHAI ROAD BANGKOK 10330 THAILAND	SQUARE D COMPANY (THAILAND) LTD. 10TH FLOOR MBK TOWER 444 PHAYATHAI ROAD BANGKOK 10330 THAILAND	EX-76136 (THAILAND) 3225786 ORDER NO. 270615 (IP615-EM) SQUARE D THAILAND - BANGKOK PUNTIP PARK PROJECT

ROUTE VIA: OCEAN - COLLECT

CASE#	PACK	POUNDS			KILOS			DIM. IN INCHES			CU. FT.	QTY	CATALOG NUMBER		RUN NUMBER
		GROSS	LEGAL	NET	GROSS	LEGAL	NET	HEIGHT	WIDTH	LENGTH					
2	CRT	595	90	505	270	41	229	16	42	100	38.9	3	AF2516G76		A
3	CRT	800	58	742	363	26	337	40	46	64	68.1	3	AF2516G10FEB		A
												3	AF2516G10FES		A
												6	AF2516GLEM11		A
4	CRT	308	40	268	140	18	122	31	46	44	36.3	1	AF2508G22FEB		B
												1	AF2508G30OF	S11O8B11	B
												1	AF2508G37LE	S11B26	B
												1	AF2508G38DL	S11C16B11	B
5	CRT	1529	122	1407	694	55	639	24	42	136	79.3	9	AF2508G10		B
												1	AF2508G23		B
												1	AF2508G30		B
												1	AF2508G34		B
												1	AF2508G55		B
												1	AF2508G59		B
6	CRT	424	38	386	192	17	175	28	42	42	28.6	1	PIF34080GN		N/A
												4	PIK34150GN		N/A
												1	PIK34175GN		N/A
												3	PIK34250GN		N/A
7	CRT	596	58	538	270	26	244	58	42	64	90.2	2	PBIL36300GN		N/A
8	PLT	218	38	180	99	17	82	15	42	42	15.3	1	AT2		A
												8	HF88F		A
												1	ACF43EC		B
												1	AT2		B
												14	HF43E		B
												24	HFV		B
9	CRT	298	58	240	135	26	109	33	42	64	51.3	2	AF2508GLEM11		B
												1	AF2508GLFM11		B
7	Crts	4550 Lbs			2064 Kilos						392.7	47			
1	Plts	218 Lbs			99 Kilos						15.3	49			
8		4768 Lbs			2163 Kilos						408.0	96			

PRINTED BY: GZHOU DATE: 06/27/92 CONTAINER NO: SHIP VIA:

SQUARE D COMPANY

5735 COLLEGE CORNER ROAD
OXFORD, OHIO 45056 U.S.A.

ORIGINAL INVOICE

REMIT TO:
SQUARE D COMPANY
P.O. BOX 85248
CINCINNATI, OHIO 45201-0999, U.S.A.

CUST. ORDER NO.	EXPORT ORDER NO.	FACTORY ORDER NO.	INVOICE NO.	INVOICE DATE	PAGE	OF
XU-0242	EX76060	3081586	40448	06-26-92	1	1

ACCOUNT NO.	ORDER	INFL.	DEST.	TERMS - PAYABLE IN NEW YORK EXCHANGE	ORDER DATE
00930	943	943	943	NET 90 DAYS	02-14-92

SOLD TO
SQUARE D COMPANY (UK) LTD.
CHENEY MANOR TRADING ESTATE, SWINDON
WILTSHIRE, ENGLAND SN2 2QG

SHIPPED TO
ELECTRICAL CENTRE
P. O. BOX NO. 4037
ABU DHABI
U.A.E.

SHIPPING INFORMATION
HAWB: 90088832
DATE SHIPPED: 06-26-92
SHIPPED VIA: BURLINGTON AIR EXPRESS
U.S.A. PORT: CINCINNATI, OH U.S.
EST. TIME OF ARRIVAL:

SHIPPING MARKS
EX-76060/3081586/XU-0242
MARKS:
ELECTRICAL CENTRE
P. O. BOX 4037
ABU DHABI, U.A.E.

SEE ATTACHED PACKING LIST
6 CRT
4864 LBS 2207 KGS 322.9 CU. FT.

ROUTE VIA AIR - PREPAID

PARTIAL [✓ X] COMPLETE ☐ COUNTRY CODE 155

ITEM	LINE CODE - ITEM LETTER	BUS-BAR TRUNKING CATALOG NUMBER		QUANTITY ORDERED	QUANTITY BACK/ORDERED	QUANTITY SHIPPED	UNIT PRICE	EXTENDED PRICE
1	9005 - BS	CF2510G45LF	S34B11	1	0	1	160.60	160.60
2	9005 - BT	CF2510G73LF	S41B32	1	0	1	227.79	227.79
3	9005 - BW	CF2510GLEM11		1	0	1	105.33	105.33
4	9005 - BK	CF2508G45LF	S34B11	1	0	1	143.92	143.92
5	9005 - BL	CF2508G52LF	S41B11	1	0	1	158.22	158.22
6	9005 - BJ	CF2508G106		1	0	1	215.54	215.54
7	9005 - BM	CF2508G6		1	0	1	146.41	146.41
8	9005 - BN	CF2508G80		1	0	1	162.65	162.65
9	9005 - BO	CF2508G88		1	0	1	178.99	178.99
10	9005 - BP	CF2510G107		1	0	1	257.08	257.08
11	9005 - BR	CF2510G4		1	0	1	115.32	115.32
12	9005 - BU	CF2510G9		1	0	1	259.46	259.46
13	9005 - BV	CF2510G93		1	0	1	223.48	223.48
14	9005 - BB	CF2516G100		1	0	1	388.74	388.74
15	9005 - BG	CF2516G6		3	0	1	279.89	279.89
16	9005 - BH	CF2516G8		1	0	1	373.19	373.19
17	9005 - BI	CF2516G94		1	0	1	365.47	365.47
18	9005 - BG	CF2516G6		3	0	2	279.89	559.78
19	9005 - BC	CF2516G34LE	S12B22	1	0	1	204.75	204.75
20	9005 - BD	CF2516G36FEB		1	0	1	212.59	212.59
21	9005 - BE	CF2516G46LF	S34B12	1	0	1	251.40	251.40
22	9005 - BF	CF2516G53LF	S41B12	1	0	1	278.53	278.53
23	9005 - BQ	CF2510G2		1	0	1	57.66	57.66

TOTAL EX FACTORY:	5326.79
EXPORT PACKING:	160.00
AIR FREIGHT:	5404.70
: US $	10891.49

PRINTED BY: GZHOU

Form: K-600-62 (03/01/92)

THESE COMMODITIES LICENSED BY U.S. FOR ULTIMATE DESTINATION ABU DHABI DIVERSION CONTRAY TO U.S. LAW PROHIBITED - COUNTRY OF ORIGIN - UNITED STATES OF AMERICA - CERTIFIED TRUE AND CORRECT.

SQUARE D COMPANY

5735 COLLEGE CORNER ROAD
OXFORD, OHIO 45056 U.S.A.

SHIPPER

SQD ORDER: **3225786**

Form: K-500-66 (05/25/92)

EX76136

SHIPPING INFORMATION	
CUSTOMER ORDER:	EX76136
CUSTOMER ACCNT:	00103
PACK DATE:	06-27-92
CONTAINER NO:	
ROUTING:	OCEAN
PREPAID/COLLECT:	COL
SHIPPED VIA:	

SHIPPED TO

SQUARE D COMPANY (THAILAND) LTD.
10TH FLOOR MBK TOWER
444 PHAYATHAI ROAD
BANGKOK 10330
THAILAND

ITEM LETTER	QUANTITY SHIPPED	CATALOG NUMBER		LIST PRICE	NET MULT	LINE CODE
E	1	AT2	ODC-1	0.00	0.0000	9255
F	8	HF88F	ODC-1	0.00	0.6500	9255
G	3	AF2516G10FEB	-2	855.00	0.3187	9055
H	3	AF2516G10FES	-2	855.00	0.3187	9055
J	6	AF2516GLEM11	-2	1112.00	0.3187	9055
K	1	ACF43EC	ODC-3	115.00	0.3187	9255
L	1	AT2	ODC-3	0.00	0.0000	9255
M	14	HF43E	ODC-3	0.00	0.0000	9255
N	24	HFV	ODC-3	0.00	0.0000	9255
O	9	AF2508G10	-4	1190.00	0.3187	9055
P	1	AF2508G22FEB	-4	630.00	0.3187	9055
Q	1	AF2508G23	-4	228.00	0.3187	9055
R	1	AF2508G30	-4	298.00	0.3187	9055
S	1	AF2508G30OF	S11O8B11/-4	1222.00	0.3187	9055
T	1	AF2508G34	-4	337.00	0.3187	9055
U	1	AF2508G37LE	S11B26/-4	829.00	0.3187	9055
V	1	AF2508G38DL	S11C16B11/-4	1301.00	0.3187	9005
W	1	AF2508G55	-4	545.00	0.3187	9055
X	1	AF2508G59	-4	585.00	0.3187	9055
AC	2	PBIL36300GN	-7	4982.00	0.1966	9270
AD	1	PIF34080GN	-7	1842.00	0.1966	9270
AE	4	PIK34150GN	-7	4177.00	0.1966	9270
AF	1	PIK34175GN	-7	4177.00	0.1966	9270
AG	3	PIK34250GN	-7	4736.00	0.1966	9270

PRINTED BY: GZHOU   CONTAINER NO:

STATEMENT OF PASS-THRU INTERNATIONAL SHIPMENTS
FROM OXFORD TO UNITED KINGDOM
FOR THE MONTH OF APRIL

SOLD TO COUNTRY	SHIP TO COUNTRY	INVOICE	FACTORY BILLING	FREIGHT &OTHER	TOTAL
UNITED KINGDOM	EGYPT	40200	12936.43	6435.37	19371.80
			12936.43	6435.37	19371.80
	UNITED KINGDOM	40198	227.06	214.94	442.00
		40199	135.94	100.06	236.00
		40201	12918.97	4908.00	17826.97
		40202	1333.88	555.00	1888.88
		40203	19364.83	5166.00	24530.83
		40204	765.90	84.00	849.90
		40206	4251.03	1542.00	5793.03
		40207	5436.62	625.48	6062.10
		40208	4188.59	2642.53	6831.12
		40209	149.02	94.98	244.00
			48771.84	15932.99	64704.83
			61708.27	22368.36	84076.63