A Systems Development Life Cycle Project for the AIS Class

Ting J. Wang, (E-mail: t-wang@gsu.edu), Governors State University
Georgia Saemann, (E-mail: gsaemann@uwm.edu), University of Wisconsin – Milwaukee
Hui Du, (E-mail: huidu@utpa.edu), University of Texas – Pan American

ABSTRACT

The Systems Development Life Cycle (SDLC) project was designed for use by an accounting information systems (AIS) class. Along the tasks in the SDLC, this project integrates students' knowledge of transaction and business processes, systems documentation techniques, relational database concepts, and hands-on skills in relational database use. After completing this project, students will comprehend major steps in the process of designing, developing, and implementing a database application. Instructors can add additional and/or modify any requirements and elements based on their emphases in the process, as well as align the tasks according to the systems design approach used in the textbook. This project is constructed on the Systems Understanding Aid (SUA) manual case of Arens and Ward (2001). However, instructors can adopt any case by using the project as a template. This project can be used as a semester final project, split into parts, worked along with the systems development topics covered in class, or assigned to a second AIS course.

INTRODUCTION

The topic of systems development and its importance in forming a general understanding of detailed information systems design, development, and implementation have been covered in accounting information systems (AIS) textbooks for more than a decade. Systems development related topics, such as transaction processing cycles, systems analysis and design, systems documentation techniques, relational database concepts, and data modeling techniques, have received significant coverage in AIS textbooks and in the classroom. The average coverage for these topics is more than 50% in most AIS textbooks and close to 50% of the class time spent by the instructors (Bain, Blankley, and Smith, 2002). There are cases and projects developed for some individual topics in the systems development process, such as REA data modeling and cardinalities (Geerts and Waddington, 2000; Geerts, Waddington, and White, 2002, respectively). However, a comprehensive instructional project has not been developed for systems development as a whole for the AIS class. The purpose of this project is to provide students an opportunity to integrate their knowledge of systems development topics acquired from the textbook and class lectures with hands-on skills in relational database use in order to enhance their understanding of the process of creating a relational database application.

Many different approaches are available for systems development, such as the Systems Analysis and Design Life Cycle (SDLC) by Hoffer, George, and Valacich (1999) (adopted in this project) and a System Development Process by Whitten, Bentley, and Barlow (1994). In addition, there are a variety of other approaches that may be adopted with respect to specific tools or technologies, including object-oriented analysis and design, systems engineering, joint application design, participatory design, essential systems design, and automating the SDLC using CASE tools (Hollander, Denna, and Cherrington, 2000). Table 1 provides a comparison of the SDLC phases covered in this case with those followed by six popular AIS textbooks. The depicted lineups are not as clear as shown here because the textbook authors include certain tasks in different phases and/or add different tasks due to applications of unique computer technologies, e.g., Jones and Rama (2006) use CASE tools.

There are cases available for us to achieve similar purpose of this project, such as the Belgian Chocolate Company case (Geerts & Waddington, 2000). Instructors can use whatever cases they are using for systems

development topics such as transaction/business processing cycles, flowcharts, etc. with appropriate modifications to the project. However, one primary criteria of using a case is to assure that students are able to acquire knowledge of and experience with detailed business activities and processes, and are familiar with traditional accounting cycles before engaging in this project. Understanding of business processes, including rules and policies, is a pre-requisite to systems design.

This project utilizes the materials from the Systems Understanding Aid (SUA) for Waren Distributing, Inc., a company that purchases and then re-sells small household appliances, such as toasters, can openers, etc. (Arens and Ward, 2001). The document flowcharts can serve as the background for the project with any supplementary information provided by the instructors. This project is flexible and can be assigned by any instructor regardless of the textbook used. The objectives of the SUA case are to help students become more knowledgeable about aspects of business processes and accounting systems. The case provides students opportunities to actually see and experience the paths of information flow, and help them understand relationships among the devices used in accounting systems (i.e., documents, chart of accounts, double-entry bookkeeping, journals, subsidiary ledgers, general ledgers, and financial statements).

We adopted the SUA manual case because of its popularity and because it helps students gain business and accounting experience which reinforce their knowledge acquired from business and accounting courses that they have taken before. Since the SUA case requires students to go through a manual process of preparing and processing raw documents, record transactions, understand and follow document flowcharts, go through an accounting cycle, and prepare financial statements, we believe it will be more effective in terms of the students' level of understanding of the business processes and accounting cycles, compared to other cases. As a result, students will have a broader and more complete knowledge about the system in order to handle the process of designing and implementing it in the SDLC. It will be an invaluable opportunity for students to experience and compare both manual and database systems with the same case.

This project, along with the SDLC, helps students integrate their knowledge of business processes acquired from the SUA case, knowledge of systems documentation techniques (i.e., document flowcharts and data flow diagrams) and relational database concepts (i.e., entity, normalization, & cardinalities) learned from the textbook and lectures, and hands-on skills obtained from relational database use. It provides students opportunities to understand and experience the process of creating a relational database application from scratch in a small scale system. The SDLC processes that this project follows include phases such as Systems Analysis, Logical Design, Physical Design, and Implementation, which are illustrated with the tasks involved in each phase in Figure 1.

In this project, we followed the SDLC and provided documentation of the process of systems development for the current "charge sale" of the sales/collection subsystem (or revenue cycle) in the SUA case and required students to do the same for the current "purchase/payment" subsystem (purchases of fixed assets and office supplies are excluded). We completed certain processes of the SDLC, except Project ID & Selection, Project Initiation & Planning, and Maintenance phases. We only covered the Coding and Testing tasks in the Implementation phase. Figure 2 presents the work flow of the project. It shows the flows and the interrelationships of the tasks between the phases. The arrows represent influences of one task over other tasks and/or phases.

We adopted this project in both undergraduate and graduate AIS classes as an individual final project for 2 years by one instructor (as described in this paper) and split it into 6 assignments as a group project by another instructor. However, there are many other alternatives in terms of how this project can be carried out. For example, we can require students to design and implement the current "Payroll" subsystem, go through a business process re-engineering for any or all subsystems (e.g., combine the "Charge sale" and "Cash sale"), use REA (or E-R) in the "Conceptual Data Modeling" phase, use Microsoft Access (or FoxPro) in the "Implementation" phase, as an individual or a group project, and/or a combination of the above. Any of these will serve the purpose of this project, that is, integration of students' knowledge of systems development topics acquired over the AIS course and hands-on skills in relational database use by providing them opportunities to understand and experience the process of creating a relational database application.

Manual completion of the SUA (or any other case) will provide the necessary foundation for this SDLC project. In addition, certain coverage is required before each phase in the SDLC. For example, coverage of systems documentation techniques is a prerequisite to the Systems Analysis Phase, coverage of relational database concepts is required before the Logical and Physical Phases, and coverage of MS Access are essential before the Implementation Phase.

This paper describes the process of systems development for the current "charge" sales/collection subsystem in section I and provides some teaching notes in section II.

PROJECT DESCRIPTION

You have been asked to design and implement a relational database application for Waren Distributing, Inc. (i.e., SUA manual case). Below we documented the work involved in each phase of a database design following the traditional SDLC: 1) Systems analysis, 2) Logical design, 3) Physical design, and 4) Implementation for Waren's sales/collection subsystem, specifically their current "charge" sales/collection subsystem.

For this project, you are required to follow the same processes and document your work for their "Purchase/Payment" subsystem (excluding purchase of fixed assets and office supplies).

Phase 1: Systems Analysis

The systems analysis phase requires the following tasks: 1) definition of a company's systems requirements; 2) development of process; 3) logical, and 4) conceptual data modeling. A company considers costs, benefits, and feasibility in making its decision to implement a systems project. It must also establish responsibilities and a project timeline. We will discuss each of these below.

Task I: Defining Company's Systems Requirements

The objectives of most of the systems projects are for efficiency and/or effectiveness of the information system process. From a company's point of view, they are looking at the bottom line, such as cost savings, possible benefits from more and better information provided to the decision maker, rate of return, etc. From the employees' point of view, they are looking for more and/or better information, ease of operations, ease of use and ease to learn, user-friendliness, etc. As a result, we must answer several questions about the system: what are users' expectations of the system? What decisions will the system support and what objectives will the system help the company to achieve?

Suppose that Waren wants to computerize their manual system to speed up the accounting process and cut down on human processing costs by capturing and processing most of its data on-line. Let's also suppose that Waren is not willing to get involved with a major business process re-engineering because of the costs involved and possible resistance from employees. As a result, we are going to design and implement a computerized system primarily based on their current manual system. We will educate the employees about the benefits of re-engineering business processes in the future.

Once we define the scope of the systems requirements, we will work on our understanding of the current systems involved by going through process, logical, and conceptual data modeling in this "Systems Analysis" phase.

Task II: Process Modeling

The objective of this step is to formalize our knowledge about the processes involved in the current system using graphical representation, i.e., Data Flow Diagrams (DFD). Process modeling involves preparation of hierarchical DFD to describe data flows and processes in the targeted system at different levels, such as a Context Diagram, Level 0 and Level 1 DFD. In sophisticated cases, more levels of descriptions may be required. We will follow the processes in the SUA case very closely based on its "current" system depicted in the document flowcharts,

which is concluded from our previous task, Defining Systems Requirements. We will not change any process without justifications.

According to Waren's "current" sales/collection subsystem, the DFD are depicted in Figure 3. The Context Diagram and Level 0 DFD are very similar to those depicted in some of the textbooks because they tend to follow the same processes of general information. There are three major processes in the Level 0 DFD: 1) Process order/return approvals, 2) Process shipments/receipts, and 3) Process payments/credits; two in the Level 1.0 DFD (in Figure 3-A): 1) Approve order requests and 2) Approve return requests; seven in the Level 2.0 DFD (in Figure 3-B): 1) Fill orders, 2) Prepare bill of ladings, 3) Generate bill of lading information, 4) Prepare sales invoices, 5) Generate sales invoice information, 6) Prepare receiving reports, and 7) Generate receiving report information; and six in the Level 3.0 DFD (in Figure 3-C): 1) Endorse checks, 2) Record cash receipts, 3) Prepare bank deposits, 4) Generate cash receipts information, 5) Prepare credit memos, and 6) Generate credit memo information.

Task III: Logical Modeling

The objective of logical modeling is to produce structured descriptions and diagrams that enumerate the logic contained in each process denoted by the most detailed level of DFD. Again, we will follow the exact procedures from the SUA case closely, based on the conclusion from our Defining Systems Requirements task mentioned above. As noted in Figure 3-B, the Level 1 DFD of the Shipments/Receipts Process includes seven processes of which three of them (2.3, 2.5, and 2.7) relate to generating reports and do not require logical models (assuming there is only one simple type of reports provided to users in each process in this case). Table 3 provides an example of logical descriptions for processes 2.1 and 2.4 in the Level 1 DFD, which are **not** described in the DFD; they explain more specifically about how data are used and processed. As an example of this, Table 3 shows that the process of "Fill Orders" begins with picking each ordered item from inventory. Pick the quantity if there are sufficient amount of items in stock, or prepare for partial or back-order shipment if there are insufficient amounts or out of stock, respectively. Table 3 also shows the logical process of the "Prepare Sales Invoices". In addition, we can also use program flowcharts to describe these logical processes (not included in the project).

Task IV: Conceptual Data Modeling

In this phase, a conceptual data model will be developed to depict 1) the entities for which we want to store data about business activities and 2) the business rules and policies governing interrelationships between these data entities. This data model provides the blueprint for the design of the relational database structure. There are many ways to come up with the data model, such as using REA or E-R data modeling. However, we will adopt the E-R data modeling using the data stores identified from the Process Modeling task.

To obtain the entities required for the sales/collection subsystem, we use the DFD from the most detailed level depicted in Figure 3-A, 3-B, and 3-C. All of the data stores depicted in the DFD will serve as a basis for figuring out the entities required. Table 4 lists all of the processes and data stores identified from the "current" sales/collection subsystem at Waren Distributing, Inc. In a few cases, some of these data stores and processes may be retained as manual files/processes that will not be included in the data model, or may be accessed via an Electronic Data Interchange device linking to suppliers/customers systems directly. As a result, not all of the data stores will be included depending on the proposed design of the system. In this project, we will include everything except the "approval" and "fill orders" processes which will remain unchanged as manual processes in the system. As a result, we will not include data stores such as "Customer Order" and "Return Request" in the data model. We assume that all of the information generation processes involve only one type of report.

Figure 4 shows the conceptual data model based on the data stores summarized in Table 4 and business rules and policies adopted by Waren Distributing, Inc. from the SUA case (Note: Customer, Employee, and Zip Code tables are included during the normalization process in Task I, Phase 2 below).

Phase 2: Logical Design

The logical design phase of a database involves a detailed draft of tables, forms, reports, interfaces, and dialogues, which will eventually be used in the Implementation phase later (see Figure 2). The work that we have come up with from the previous phase, Systems Analysis, is used in the logic design phase to further develop the system. Specifically, the data model that we obtained from the "Conceptual Data Modeling" task is used in the "Define Data Structure" and the work from the "Logical Modeling" and "Define Systems Requirements" in "Design Program Interfaces", "Design Input Interfaces", and "Design Output Format" tasks (See Figure 2).

Task I: Data Structures

The context dictionaries created previously in the "Process Modeling" task (in Figure 3-A, 3-B, and 3-C) provide a list of the data elements to be included in the data stores. However, the data stores obtained from this list are not normalized. They are simply recognized through the process modeling of its current sales/collection subsystem. To achieve database normalization, certain forms (also called properties or constraints) must be imposed on the data stores (entities/tables) in the database. The least restrictive form is called the first normal form, followed by the second, third, Boyce-Codd, fourth, fifth, and Domain/Key forms. Most accounting systems require the use of the first three normal forms (Perry and Schneider, 2005). The first normal form restricts repeating attributes (or fields) and divisible data in an attribute. The second normal form requires dependency of the non-key attribute(s) on the key attribute(s), and the third normal form eliminates transitive dependency which means that non-key attributes cannot be dependent on any other non-key attributes (for more detailed information, see Gelinas, Sutton, and Fedorowicz, 2004).

To apply the normalization process, please follow the instructions from the textbooks for all entities to the third normal form. All data fields that flow into and out from a data store will be used as attributes to populate the specific entity (see the context dictionaries in the DFD in Figure 3-A, 3-B, and 3-C). Apply the relationships based on the business policies and rules adopted in the SUA case. Figure 4 shows the simplified E-R data model of the sales/collection subsystem with normalized entities for the SUA case.

To obtain the detailed attributes (fields) for tables, we will need to use the normalized entities shown in Figure 4. Table 5 lists the normalized table structures. We included two additional tables, Carrier and Location, in order to reduce the degree of redundancy and eliminate inconsistency. Foreign keys, bridge tables, and detailed table structures are included later in Phase 3, Physical Design.

Task II: Program Interfaces

A switchboard (or main menu) will be created, which contains buttons linking to each data entry screen and each report display designed for the current sales/collection system application. There will be six buttons for recording and reviewing each account in Carrier, Customer, Employee, Location, Product, and Zip Code tables, five buttons for recording and reviewing records in the processes such as Prepare Sales Invoice, Prepare Bill of Lading, Prepare Receiving Report, Record Cash Receipt, and Prepare Credit Memo, and eleven buttons for previewing and printing reports for all of the six accounts and five processes (see Table 5). A sketch of the switchboard needs to be created as a guide for the implementation. However, Figure 5 shows the actual switchboard created later in the "Implementation" phase in the process.

Tasks III & IV: Input Interfaces & Output Format

For the input and output screens/formats in which data will be captured, displayed, and/or printed, we will follow those existing forms/reports used in the manual system from the SUA case closely as concluded from the Define Systems Requirements task (i.e., possible resistance from employees for new looks). To ease data entry and prevent typos in some data fields, we will create user-friendly interfaces and dialogues, such as pull-downs, record navigation and operation buttons, etc. We will create these input interfaces on displays/screens for all six accounts and five processes mentioned previously in the Program Interfaces task which is originated from Table 5. Figure 6 shows

the input screen for the sales invoice created later in the Implementation phase, which appears as the actual sales invoice document used in the case.

We will follow the formats of the actual documents and reports from the SUA case in the design of the output for all eleven reports mentioned in the Program Interfaces task.

Phase 3: Physical Design

In this phase, the specific structure of the tables and the interrelationships between the tables are set forth. Structure details include field name, data type, field size, format, and other field properties. The field name should be concise but descriptive of the element. Define the data type, field size, format, and other field properties if there are any. The specifications can be found in MS Access software (e.g., field size and format).

Task I. File Design

The basic elements in the tables and the interrelationships between the tables were determined in the Data Structures task in the Logical Design phase (see Figure 4 and Table 5). Since these are normalized tables (i.e., data redundancy is reduced and data inconsistency can be eliminated when referential integrity is imposed), we need to create foreign keys in the related entities so that these entities can be physically linked to each other (this is the reason that data redundancy is minimized rather than eliminated). Primary and foreign keys must be identified in the physical design phase to establish the actual links between tables. Since we are using relational database management software in the implementation for the project, we need to create bridge (relationship) tables for many-to-many relationships as well.

The primary key is listed first with underlines and followed by any foreign keys in the brackets. Table 5 presents a partial list of these table structures.

Task II: Database Design

Based on the table structures listed in Table 5 and their interrelationships, we will be able to come up with a database structure shown in Figure 7 which is taken from the Relationships in Access created later in the Implementation phase.

Phase 4: Implementation

Implementation includes coding, testing, and installing the new database system based on the logical and physical designs already developed in previous phases. MS Access is used to implement the database for this project. Table 6 lists tables, forms, and reports needed for this project.

Task I: Coding

1. Create the tables based on the field names, data types, field size, and format specified in the File Design task in the Physical Design phase in Table 5. It is not necessary to specify foreign keys in MS Access because they are recognized by the system once the relationships are established.

2. Establish links between the tables created based on the relationships specified in the Database Design task in the Physical Design phase in Figure 7.

3. Use Access Forms object to create the switchboard and data entry forms according to the specifications determined in the Logical Design phase (see Figure 5 and 6). The layout of the data entry forms should correspond as closely as possible to the paper documents in the manual SUA in order to minimize possible resistance from employees (discussed in the Define System Requirements task in the System Analysis phase). Use Macros to make record navigation and operation as user-friendly as possible. On each data entry form,

include a form navigation button which can close the data entry form and return to the switchboard (or main menu). Use Combo Box devices for the data fields to ease the data entry, which link to an account table, such as Carrier, Customer, Employee, Location, Product, and Zip Code.

4. Use Access Reports object to create reports according to the specifications discussed in the Logical Design phase. Use Queries as needed for retrieving data. The layout of the reports should correspond as closely as possible to actual paper documents in the manual SUA.

Task II: Testing

Test the data entry forms by entering some fictitious data in Table 7 (see the actual case for more data). Test data entries to all data fields, combo boxes, and buttons on data entry forms for six accounts and five processes.

TEACHING NOTES

Overview

This project was targeted on the SDLC topic and designed for use in an AIS class at either the graduate or undergraduate level. It requires coverage of transaction and business processes, systems documentation techniques (i.e., both document flowcharts and data flow diagrams) and relational database concepts (i.e., E-R or REA, normalization, and cardinalities), and familiarity with MS Access. Perry and Schneider's MS Access 2003 (2005) provides a good resource aid, reducing the amount of time required in class to cover the technical aspects of working with the program. Pillsbury and Wang (2002) also provided a series of assignments targeting most of the objects in Access (e.g., Tables, Forms, Queries, and Macros). The use of MS Access, as opposed to some other database program, is beneficial because many students already have it on home computers, which eliminates the need for them to work exclusively in University computer labs. However, the authors are willing to provide additional instructions about MS Access if requested.

We strongly recommend assigning work (i.e., tasks in the phases) throughout the semester to give students, especially undergraduate, immediate hands-on examples of systems design topics. We also recommend presenting to students the completed processes and their interrelationship and connections of tasks between phases (see Figure 2) before and after they begin to learn about the next task and/or phase in the SDLC.

Systems Analysis Phase

Systems Requirements

In this project, we required students to design and development a relational database system based on the "current" manual system of the company in the SUA case. However, as we mentioned in the project, instructors can modify the requirements based on their emphasis in the class. For example, instructors can ask students to propose a new design by going through a business process re-engineering for a subsystem (e.g., use the interview method in Geerts & Waddington, 2000) and/or use REA data modeling instead. The solution is going to be varied depending on the conclusions from the Define System Requirements task that students come up with. We are willing to discuss solutions for different scenarios.

Process Modeling

Information about DFD can be found in Supplement Chapter A and throughout the text in Hollander, Denna and Cherrington (2000), in Chapter 2 of Gelinas, Sutton, and Fedorowicz (2004), and in Chapter 3 in Romney and Steinbart (2006). Although context dictionaries are only found in Hollander, et al. (2000), they are very easy to comprehend. They provide a list of data items required in a data flow, which helps to form the table structure in the later development.

For this project:

- The context diagram provides a very basic overview with only one process (sales/collection).
- The Level 0 DFD includes the three processes included in the sales/collection subsystem: customer order/return, shipment/receipt, and payment/credit.
- The context dictionaries define the specific elements or attributes in each of the data flows. Each element represents a data field that is needed to process a transaction and/or provide information for decision-making purposes. For this project, we closely followed the data provided in the current system in the SUA case. It is important, however, to discuss with students the need to evaluate data included in existing forms and reports with information users and providers before creating a context dictionary for the proposed system. This evaluation is essential to assure more efficient and effective modeling.

Logical Modeling

In this section, students document the logical procedures required for some of the processes in the Level 1 DFD in this case. These detailed procedures vary company by company and change with advancing technologies. Logical modeling is the focus of a business process re-engineering. More information can be found in Chapter 4 of Hollander et al. and in Chapter 10 and 11 of Gelinas, Sutton, and Fedorowicz. Students should recall how purchases and payments were processed in the manual SUA and modify, as needed, for a computerized system.

Conceptual Data Modeling

This project utilizes an Entity-Relationship (E-R) conceptual method (Chen 1976; Batini et al. 1992) to come up with a data model. However, the REA data modeling can be used as well (see Geerts & Waddington, 2000).

Logical Design Phase

The logical design phase focuses on the layout of user interfaces (i.e., the display of the input and output screens), organization of the interfaces, and output reports. For this project, students will simply follow the design of manual documents and reports from the SUA. It is worthwhile, however, to describe ways in which user interfaces, windows and dialogues, and navigation aids can be designed to ease data entry and ensure data are entered correctly. This will help students to complete the project. Further, although the logical design for this project is relatively simple, Chapter 6 of Post (1999) provides more extensive material.

Good discussions of cardinalities are provided in Chapter 4 of Hollander et al. (2000), Chapter 16 of Romney and Steinbart (2006), and Chapter 3 of Gelinas, Sutton, and Fedorowicz (2004). The cardinality indicates the number of records of a table that are related to a record of another data file. For example, a zero-to-one minimum cardinality between customer and invoice would indicate that the Customer data file might include entries that have never purchased merchandise from Waren (i.e., new customer). Further, at least one customer must be identified for every invoice. Note that a database can be designed without defining minimum cardinalities but maximum cardinalities are required. The maximum cardinality must be identified because it determines how related data files are physically linked. Further, there are many ways to present the cardinalities and relationships, such as Batini, Elmasri, and maximums-only conventions (see Romney and Steinbart, 2006).

Physical Design Phase

Chapter 6 of Perry and Schneider (2005) provides an excellent discussion of table layout specifications including field names, data type, field size, and other properties for the revenue cycle.

Foreign keys must also be included in main tables to establish many-to-one links with other tables. As already noted, separate relationship tables are required for many-to-many relationships. For main tables (with a many-to-one relationship), the foreign key is entered in the table with the *many* cardinality. For example, a *many* cardinality is recorded for the Invoice process in its relationship with Customer. Therefore, the Invoice table must include a

foreign key to link it to the Customer table, which has a cardinality of *one*. The foreign (or linking) key in the Invoice table is the key from the Customer table.

Students may find it more efficient to complete the physical design of their data files (tables) directly in their MS Access database, but it may not be effective because there may be too many things involved at the same time. However, in the reality, these tasks are performed by different people (e.g., system analysts and programmers).

Implementation Phase

Implementation is the most difficult and time-consuming element of the project. If possible, it is helpful to provide computer lab time with assistants who understand MS Access. When competent lab assistants are not available, we found it worthwhile to set aside an hour of class time to discuss problems encountered and demonstrate solutions. In either case, the instructor will also need to be available to answer questions. Incorrectly designed tables are very difficult to fix down the road and, based on our experience, the process is more frustrating than worthwhile from a teaching perspective.

Refer students to Chapters 4 and 5 of Perry and Schneider (2005) for help in creating forms and reports. Assignment #2 & #3 in Pillsbury and Wang (2002) also targeted at the Access objects such as Forms and Macros. We are also willing to provide instructions if requested.

Grading & Schedule

Figure 1 provides a summary of the SDLC tasks along with work involved and work presented in the project. Instructors can use it as a guide to allocate the grades and schedule for the coverage in class and deadlines.

REFERENCES

1. Arens, Alvin A. and D. Dewey Ward. 2001. *Systems Understanding Aid*. 5th edition. Okemos, MI: Armond Dalton Publishers, Inc.
2. Bain, Craig E., Alan I. Blankley, and L. Murphy Smith. 2002. An Examination of Topical Coverage for the First Accounting Information Systems Course. *Journal of Information Systems* (Fall): 143-164.
3. Batini, Carlo, Stefano Ceri, and Sham Navathe. 1992. *Conceptual Database Design: An Entity-Relationship Approach*. Redwood City, CA: Benjamin/Cummings Publishing Co.
4. Chan, P. 1976. The Entity Relationship Model – Toward a Unified View of Data. *Transactions on Database Systems* 1: 9-36.
5. Geerts, Guido L. and Barbara A. Waddington. 2000. The Belgian Chocolate Company: A Dynamic Data Modeling and Database Design Case for the Accounting Information Systems Class. *Journal of Information Systems* (Spring): 53-74.
6. Geerts, Guido L., Barbara A. Waddington, and Clinton E. White. 2002. Stevie: A Dynamic, Between-Instructor Collaborative Internet Tool for Learning Cardinalities. *Journal of Information Systems* (Spring): 75-90.
7. Gelinas, Ulric J., Jr., Steve G. Sutton, and J. Fedorowicz. 2004. *Business Processes and Information Technology*. Mason, OH: South-Western College Publishing.
8. Hall, James A. 2004. *Accounting Information Systems*. 4th Edition. Cincinnati, OH: South-Western College Publishing.
9. Hoffer, J. A., J. F. George, and J. S. Valacich. 1999. *Modern Systems Analysis and Design*. Reading, MA: Addison-Wesley.
10. Hollander, Anita S., Eric L. Denna, and J. Owen Cherrington. 2000. *Accounting, Information Technology, and Business Solutions*. 2nd Edition. Burr Ridge, IL: Irwin.
11. Jones, Frederick and Dasaratha Rama. 2006. *Accounting Information Systems: A Business Process Approach*. 2nd Edition. Mason, OH: South-Western College Publishing.
12. Perry, James T. and Gary P. Schneider. 2005. *Building Accounting Systems Using Access 2003*. Mason, OH: South-Western College Publishing.

13. Pillsbury, Ceil M. and Ting J. Wang. 2002 . Supplemental Materials for Database Management System Knowledge and Skills in the Accounting Information System Course. *The Review of Business Information Systems* (Winter): 1-6.
14. Post, Gerald V. 1999. *Database Management Systems: Designing and Building Business Applications*. Boston: Irwin/McGraw-Hill.
15. Romney, Marshall B. and Paul J. Steinbart. 2006. *Accounting Information Systems*. 10th Edition. Upper Saddle River, NJ: Prentice Hall.
16. Whitten, Jeffrey L., Bentley, Lonnie D., and Barlow, Victor M. 1994. *Instructor's Guide to Accompany Systems Analysis and Design Methods*. 3rd Edition. Burr Ridge, IL: Richard D. Irwin.
17. Wilkinson, Joseph W., Michael J. Cerullo, Vasant Raval, and Bernard Wong-On-Wing. 2000. *Accounting Information Systems: Essential Concepts and Applications*. 4th Edition. New York: John Wiley and Sons.

Availability: Upon request, the authors will provide a sample MS Access database file for this project.

TABLE 1
Comparison of Systems Development Approaches

SUA Project			Systems Analysis	Logical Design	Physical Design	Implementation	
Hollander, Denna, and Cherrington (2000)	Project ID and Selection	Project Initiation and Planning	Systems Analysis	Logical Design	Physical Design	Implementation	Maintenance
Romney and Steinbart (2006)	Systems Analysis			Conceptual Design	Physical Design	Implement. and Conversion	Operation and Maintenance
Gelinas, Sutton, & Fedorowicz (2004)	Analysis			Design		Implementation	Operation
Jones and Rama (2006)	Systems Investigation		Systems Analysis	Systems Design		Systems Implementation	
Hall (2004)	Systems Strategy		Project Initiation	In-House Development or Commercial Packages			Maintenance
Wilkinson, Cerullo, Raval, and Wong-On-Wing (2000)	Systems Planning		Systems Analysis	Systems Design	Systems Selection	Systems Implement.	Systems Operation

FIGURE 1
Systems Development Life Cycle Phases and Tasks

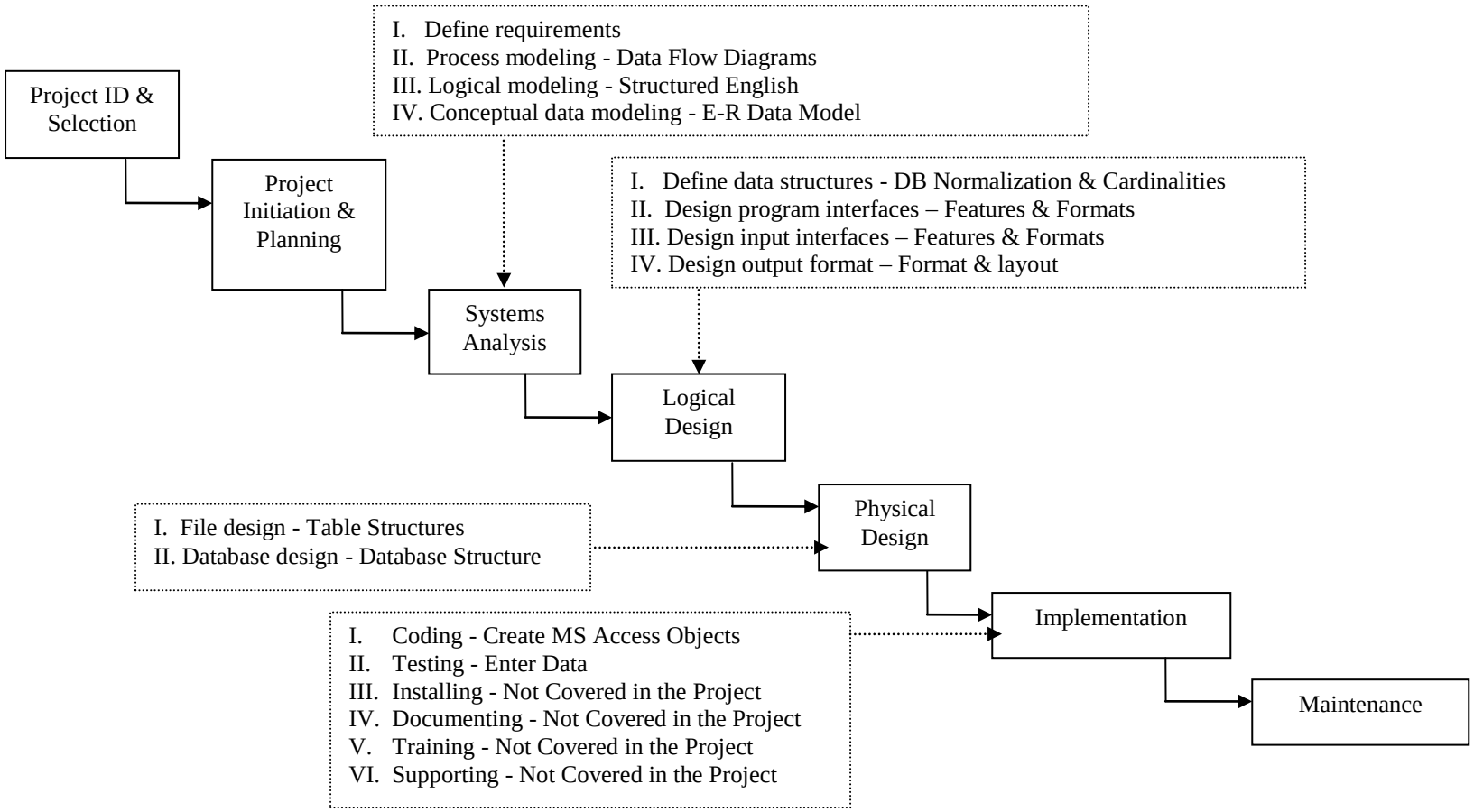


FIGURE 2
Task Flow of the Project

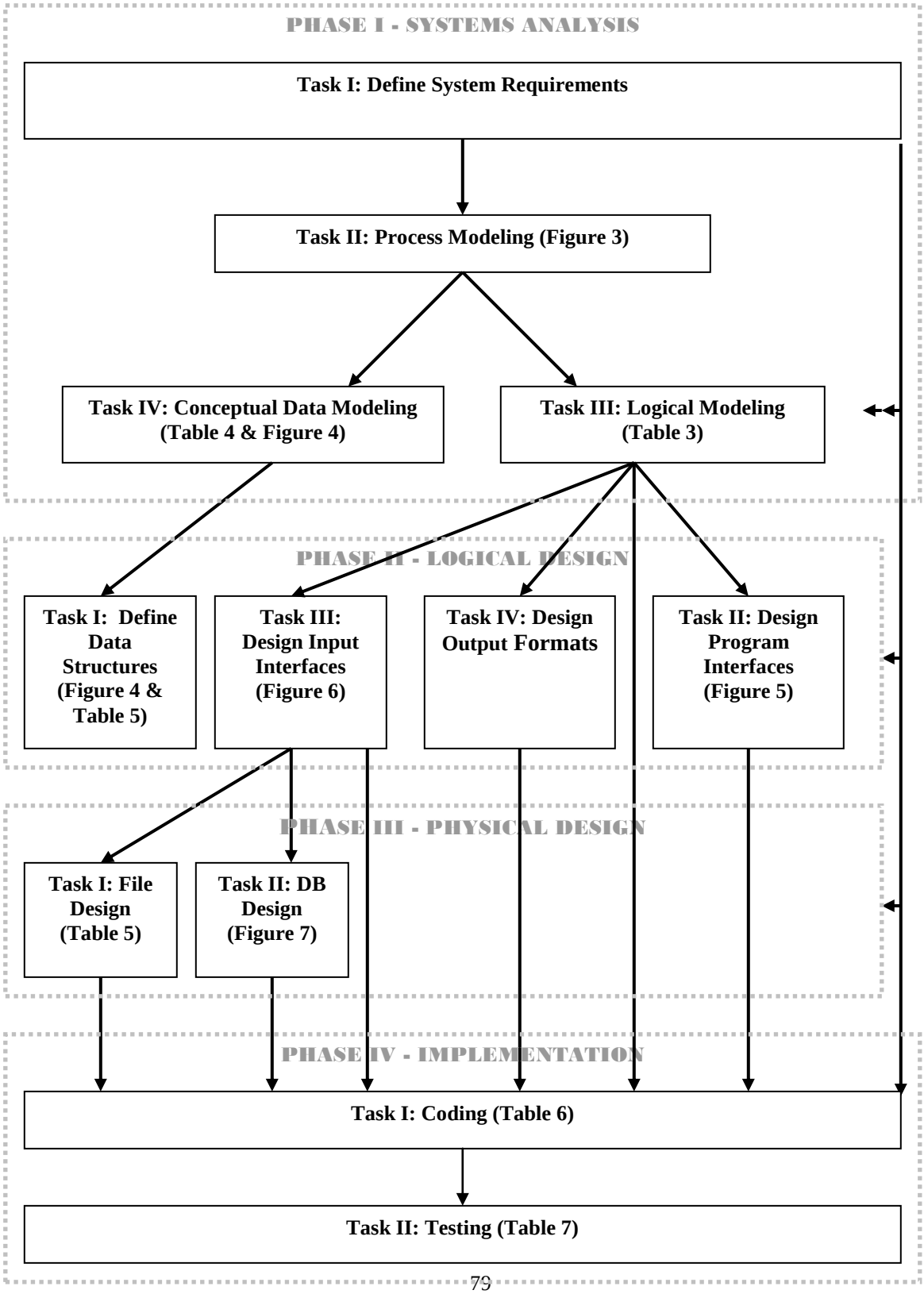
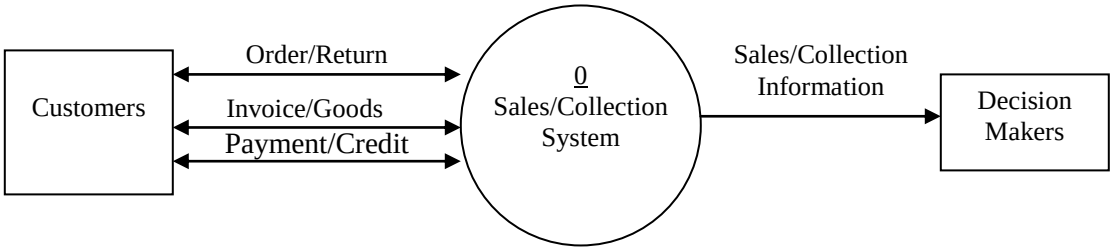
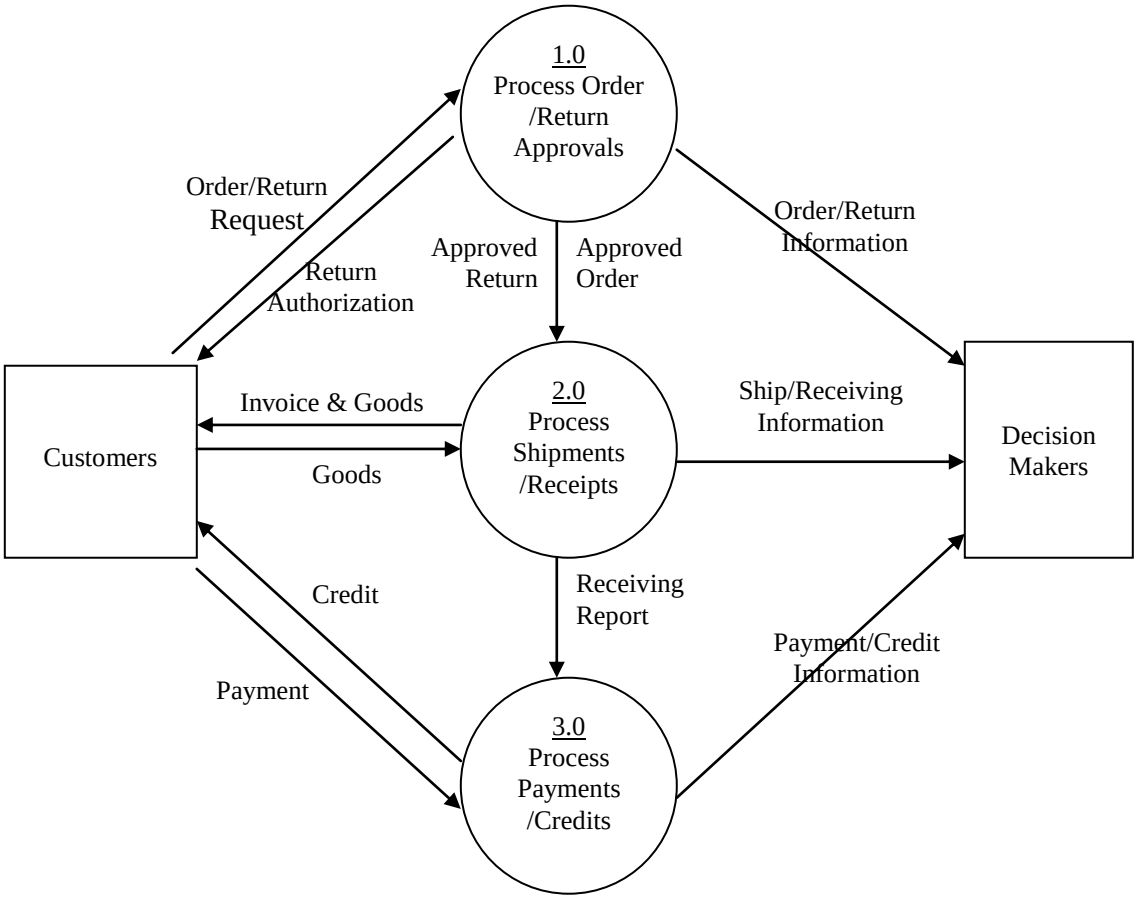


FIGURE 3
DFD of Current Sales/Collection Cycle for Waren Distributing, Inc.

Context Diagram



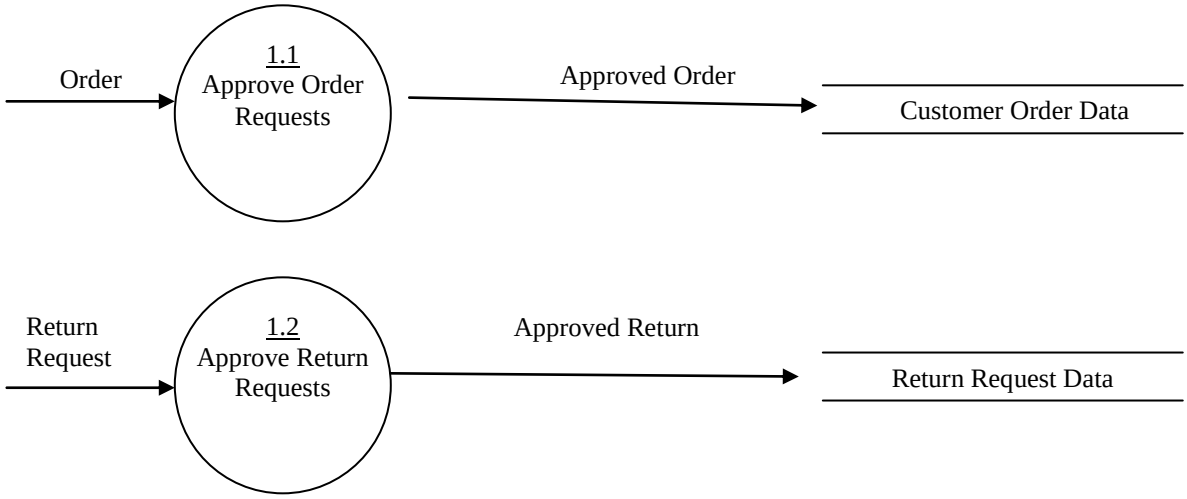
Level 0 Data Flow Diagram



Note: These diagrams (i.e., context and Level 0) may be similar to those depicted in some of the textbooks because they all follow the same processes of general information.

FIGURE 3-A
DFD of Current Sales/Collection Subsystem for Waren Distributing, Inc.
Level 1.0: Customer Order/Return Process

Level 1 Data Flow Diagrams



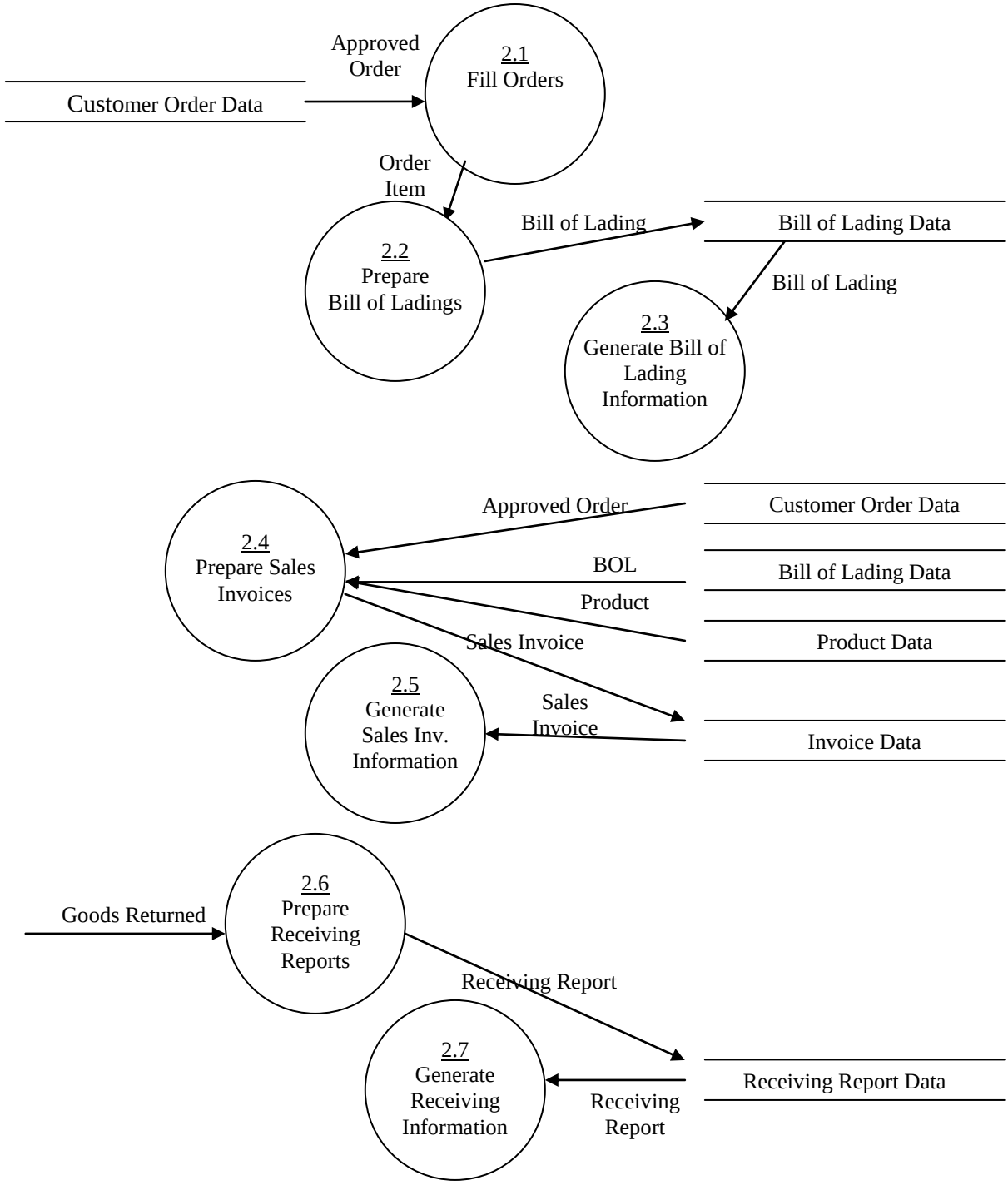
Context Dictionary (see note below)

Order =	Customer Name + Customer Address + Order # + Order Date + Bill To + Ship To + Ship Via + {Product# + Qty Order + Description} + Sign By + Pay Method + CK# + CK$
Ret. Req. =	Customer Name + Customer Address + Ret Req# + Ret Date + Refund Method + Customer# + Req By + Contact + Phone + {Qty Req + Product# + Description + Ret Code + Inv# + Inv Date + Qty Ret + Unit Price + Disc + Extension} + Total
Approved Order =	Same as Order + Approved By* [for credit sales only].
Approved Ret. =	Same as Ret. Req. + Ret Auth By + Ret Auth Date

Note: The context dictionary provides contents for the data flows. This technique is only seen in Hollander, et al. (2000) among all six AIS textbooks. It is an excellent method to completely present the detailed information involved in the process. It shows us what data are flown into and out from (thus available in) a process and a data store (required to be stored).

FIGURE 3-B
DFD of Current Sales/Collection Subsystem for Waren Distributing, Inc.
Level 2.0: Shipment/Receiving Process

Level 1 Data Flow Diagrams

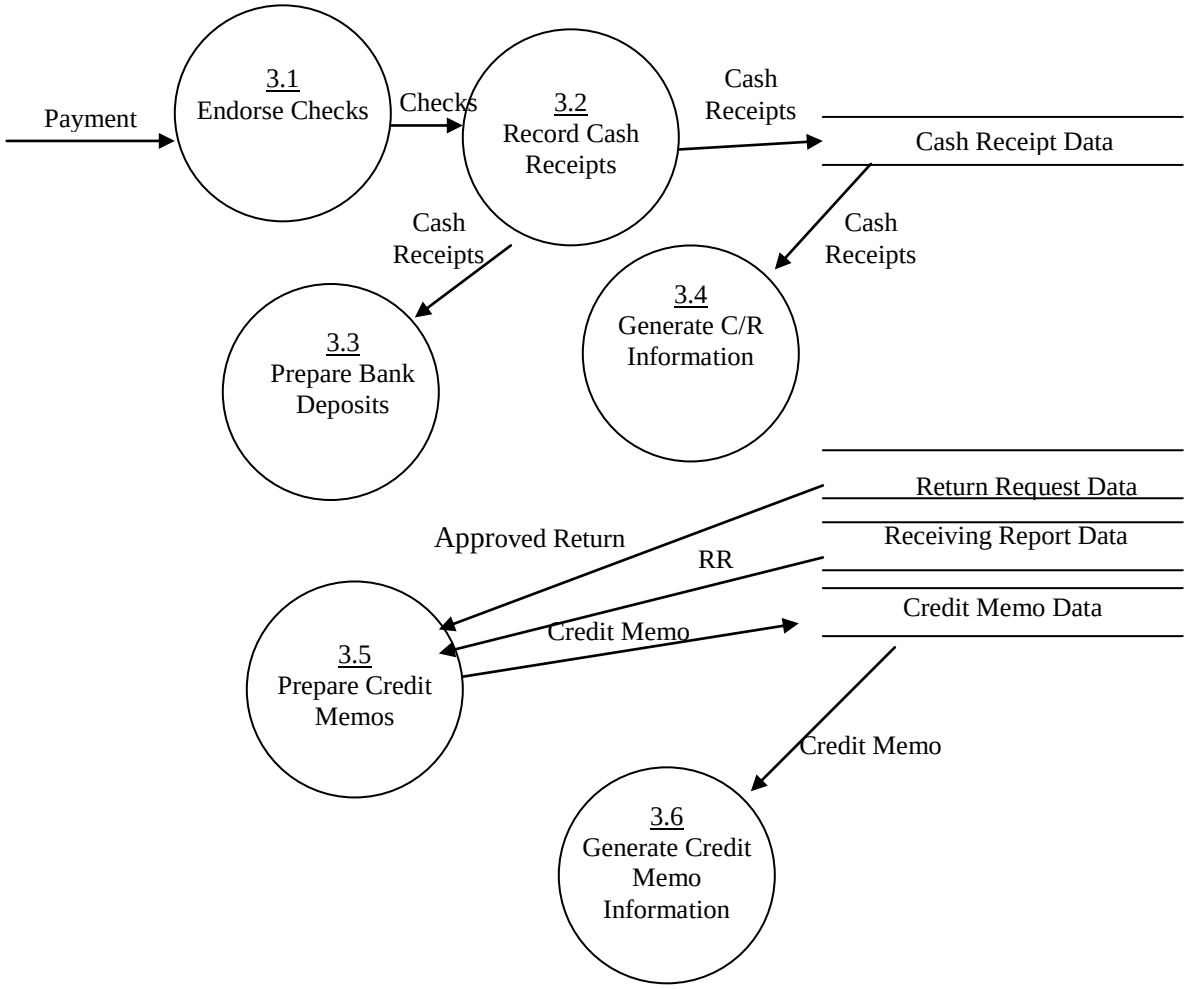


Context Dictionary

Approved Order =	Customer Name + Customer Address + Order# + Order Date + Bill To + Ship To + Ship Via + {Product# + Qty Order + Description} + Sign By + Pay Method + CK# + CK$ + Customer# + Auth By + Approved By* [for credit sales only].
Sales Invoice =	Invoice# + Customer Name + Customer Address + City + State + Zip + terms + Contact + Inv Date + Prep By + PO# + PO Date + Sign By + Ship Date + Ship Via + Bill of Lading# + {Qty order + Product# + Description + Qty Ship + Unit Price + Extension + Code} + Total + Customer# + Verify By
Bill of Lading =	Shipper # + Carrier# + Date + Consignee + Street + City + State + Zip + Carrier Name + Route + Veh# + {#Unit Ship + Description + Weight + Rate + Charges} + COD Address + COD$ + COD Fee + Total Charges + Freight Charges Method + Value + Value Per + Shipper Name + Ship Per + Carrier Name + Carrier Per + Date
Receiving Report =	RR# + Date + Received From + Address + City + State + Zip + Carrier Name + PO#/Ret# + Prepaid + Collect + Freight Bill # + {Qty + Item # + Description} + Remarks + Received By + Delivered To
Goods Returned =	Ret. Req.# + Date + Vendor + Acct.# + Req. By + {Qty + Product # + Desc. + Ret. Code + Inv. # + Inv. Date + List Price + Ext}
Approved Order =	Customer Name + Customer Address + Order # + Order Date + Bill To + Ship To + Ship Via + {Product# + Qty Order + Description} + Sign By + Pay Method + CK# + CK$+ Approved By* [for credit sales only].
Product =	Product# + Description + Unit Price + Unit Cost
Order Item =	{Product # + Description + Quantity}

FIGURE 3-C
DFD of Current Sales/Collection Subsystem for Waren Distributing, Inc.
Level 3.0: Payment/Credit Process

Level 1 Data Flow Diagrams



Context Dictionary

Payment =	Customer Name + Address + City + State + Zip + CK Date + CK# + CK$ + {Inv. #} + Memo + Bank Name + Bank Address + Auth By
Checks =	Customer Name + Address + City + State + Zip + CK Date + CK# + CK$ + {Inv. #} + Memo + Bank Name + Bank Address + Auth By
Cash Rec. =	Date + Customer Name + Disc + Cash$ + CK# + {Inv#}
Approved Return =	Ret. Req.# + Date + Vendor + Acct.# + Req. By + {Qty + Product # + Desc. + Ret. Code + Inv. # + Inv. Date + List Price + Ext} + Ret. Auth. By + Date
RR =	RR# + Date + Received From + Address + City + State + Zip + Carrier Name + PO#/Ret# + Prepaid + Collect + Freight Bill # + {Qty + Item # + Description} + Remarks + Received By + Delivered To
Credit Memo =	CM# + Date + Credit To + Acct. # + Ret. Req. # + Inv. # + Inv. Date + RR# + {Item # + Desc. + Qty + Unit Price + Amt} + Total + Ret. Method + Prep. By

TABLE 3
Logical Models for Processing Customer Order/Return in the Level 1 DFD

2.1 Fill Orders

For each customer order, do the following:

1. Pick each ordered item from inventory
   - If found with ordered quantity — Do nothing
   - If found without enough quantity — Prepare partial shipment and Backorder
   - If out of stock — Place an order and backorder
2. Check for inventory level (added; not in the current system)
   - If satisfied — Do nothing
   - If not satisfied — Place an order
3. Proceed with the next process

2.4 Prepare Sales Invoices

For each Bill of Lading, do the following:

1. Issue a new pre-numbered invoice
2. Find customer information from customer file
   - If found — Write information on the invoice
   - If not found — Establish an account & write on the invoice (added)
3. Find sale price for each product from current price list & write on the invoice
4. Fill out all related information using customer's order form
5. Review the invoice and support documents (performed by Adams)

TABLE 4
Summary of Processes and Data Stores from Level 1 DFD

All Processes from Level 1 DFD:

Order/return
1.1 Approve order request (performed manually on the paper documents)
1.2 Approve return request (performed manually on the paper documents)

Shipment/Return
2.1. Fill order (performed manually)
2.2. Prepare bill of lading
2.3. Generate bill of lading information (producing a single report)
2.4. Prepare sales invoice
2.5. Generate sales invoice (producing a single report)
2.6. Prepare receiving receipt
2.7. Generate receiving report information (producing a single report)

Payment/Credit
3.1. Endorse check (performed manually on the paper documents)
3.2. Record cash receipts
3.3. Prepare bank deposit (Assume a manual process)
3.4. Generate cash receipts information (producing a single report)
3.5. Prepare credit memo
3.6. Generate credit memo information (producing a single report)

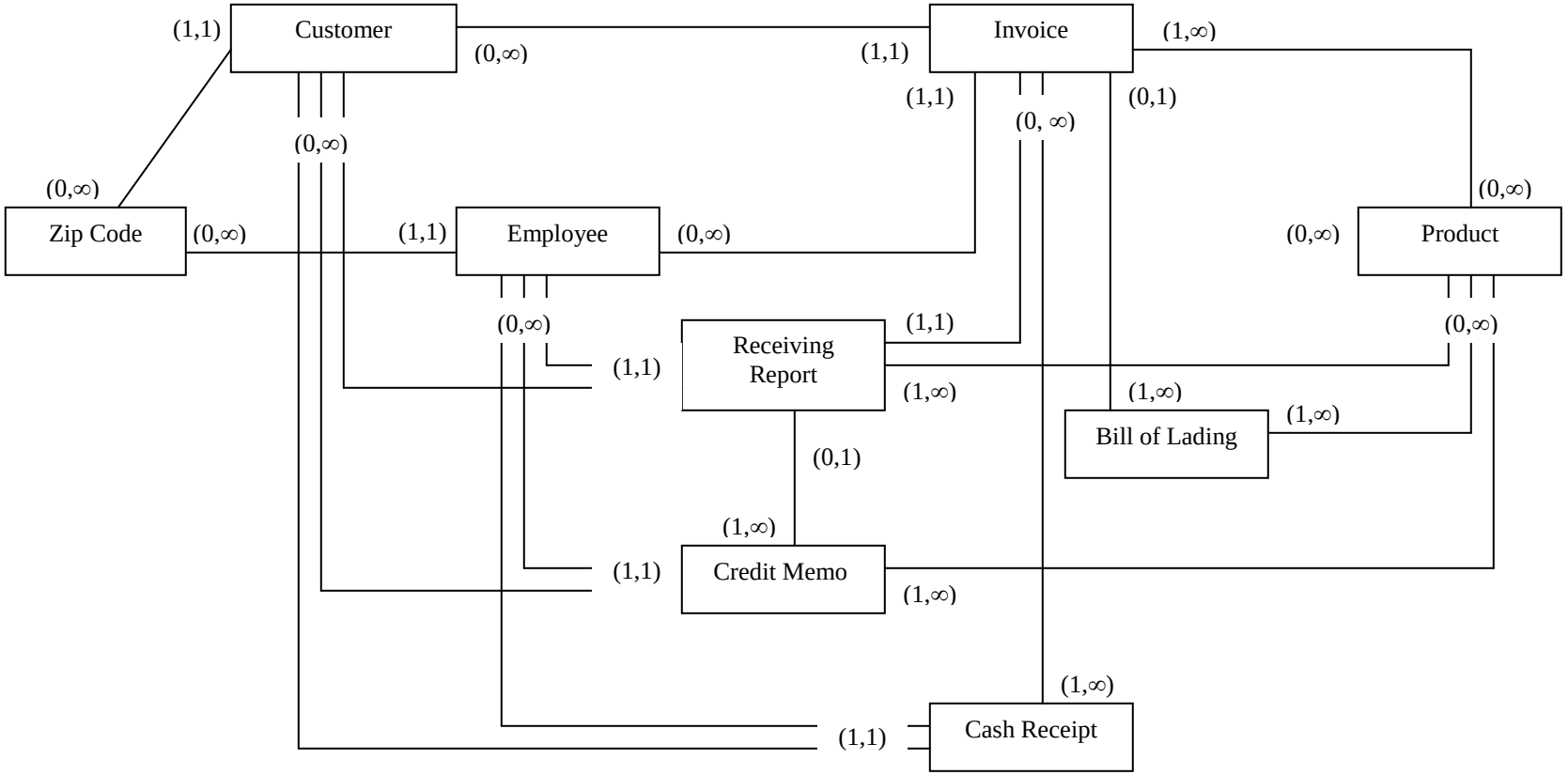
All Data Stores from Level 1 DFD:

Customer order/return
- Customer Order Data (remained manually)
- Return Request Data (remained manually)

Shipment/Return
- Customer Order Data (remained manually)
- Product Data
- Invoice Data
- Bill of Lading Data
- Receiving Report Data

Payment/Credit
- Cash Receipt Data
- Return Request Data (remained manually)
- Receiving Report Data
- Credit Memo Data

FIGURE 4
Simplified E-R Data Model of the Sales/Collection Subsystem for the SUA Case (Normalized Entities)



- Batini notation is adopted here for representing the cardinality information based on the business rules and policies adopted in the SUA case. Some of the relationships are assumed so that they can be more flexible in handling the operations.
- Assuming that a 9-digit zip code determines the city in addition to the state.

TABLE 5
Table Structures (Normalized)

Processes:

Invoice = Inv#, Inv Date, [PSS#], POID, PO Date, Sign, [BOL#], Total, [Customer#], [VSS#])
Bill of Lading = (BOL #, Date, Consignee, Street, [Zip], [CarrierID], Route, Veh#, COD Address, COD$, COD Fee, Total Charges, Freight Charges Method, Value, Value Per, Shipper Name, Ship Per, Sign Date)
Receiving Report = (RR#, Date, [Customer#], [CarrierID], PO#/Ret#, Prepaid, Collect, Freight Bill #, Remarks, [SS#], [LocID], [Inv#])
Cash Rec. = (CRID, Date, [Customer#], Disc, Cash$, CK#)
Credit Memo = (CM#, Date, [Customer#], [Ret. Req#], [Inv#], [RR#], Total, Ret. Method, [SS#])

Bridge Tables:

InvProd = ([Inv#], [Product#], Qty order, Qty Ship, Extension, Code)
BOLProd = ([BOL#], [Product#], #Unit, Weight, Rate, Charges)
RRProd = ([RR#], [Product#], [RetReqID], Qty)
CRInv = ([CRID], [Inv#], Amt)
CMProd = ([CM#], [Product#], Qty, Amt)

Accounts:

Customer = (Customer#, Customer Name, Address, [Zip], Contact, Phone)
Employee = (SS#, First, Mid, Last, Initials, Address, [Zip], Phone, DOB, DateEmp, Occup, DateTerm)
Product = (Product#, Description, Unit Price, Unit Cost)
ZipCode = (Zip, City, State)
Carrier = (CarrierID, Name)
Location = (LocID, Place)

Note:
1. Credit Term is a constant (i.e., 2%/10 day & net/30 days) and thus is excluded.
2. To reduce the degree of redundancy, we also created tables, such as Carrier and Location.
3. The **underline** indicates the table's **primary key** and the brackets, **[ ],** designate a **foreign key**.

Customer

Field Name	Data Type	Field Size	Format
Customer#	Text	5	None
CustName	Text	50	None
CustAddress	Text	50	None
Zip	Text	9	None
Contact	Text	50	None
Phone	Text	10	None

FIGURE 5
Program Interface -Switchboard for the Sales/Collection Subsystem

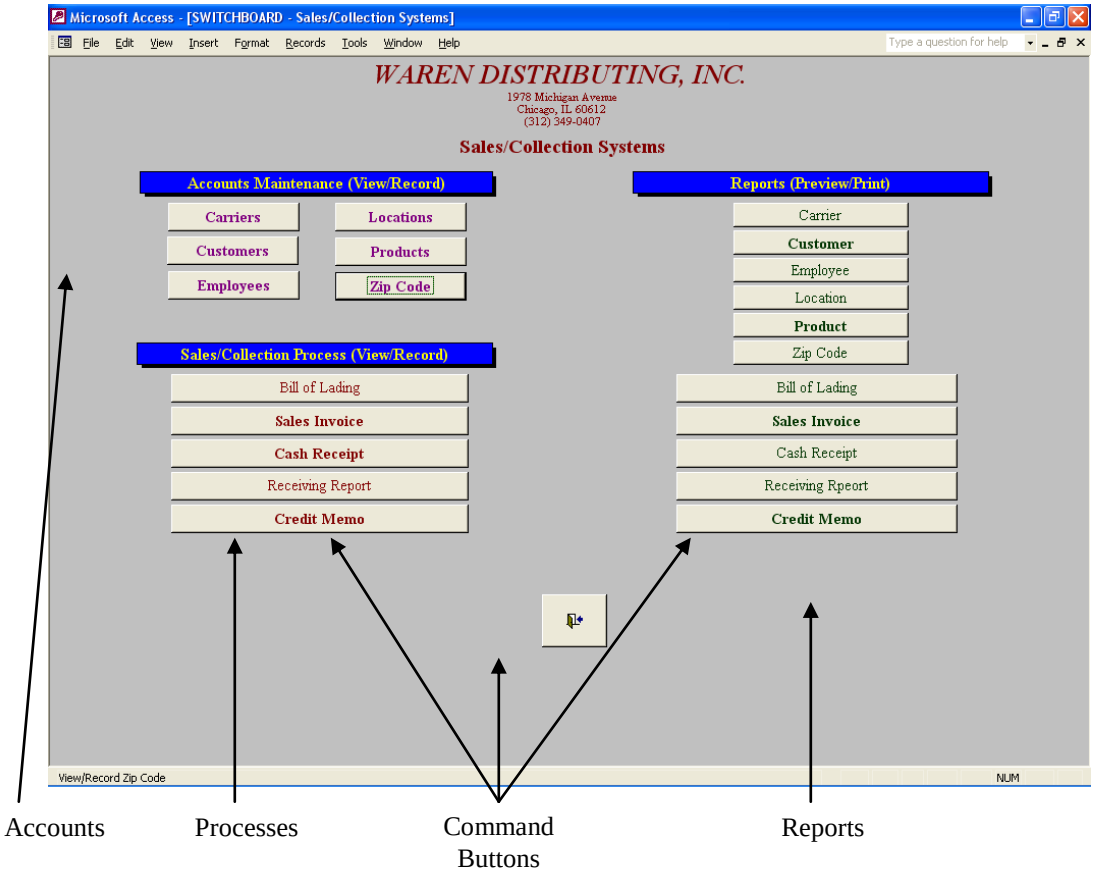


FIGURE 6
Input Interfaces -- Sales Invoice for the Sales/Collection Subsystem

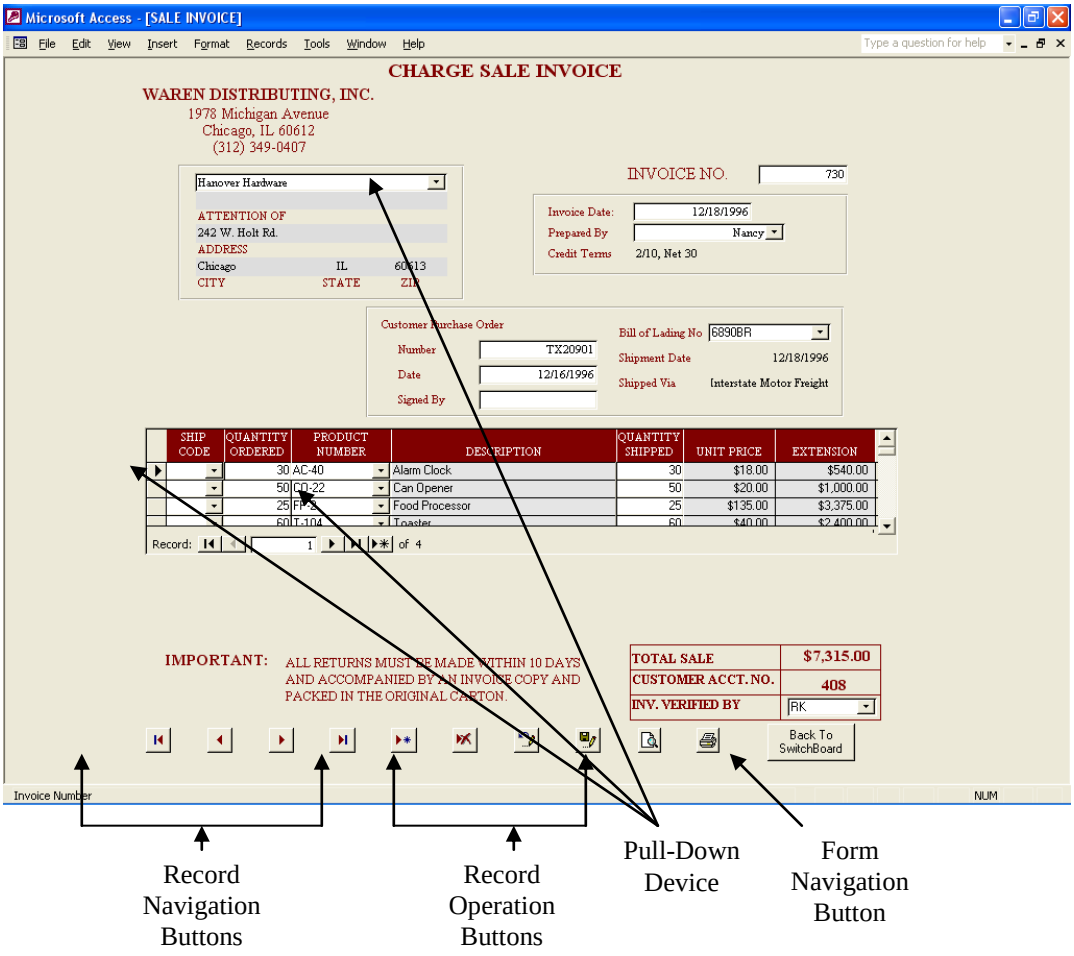


FIGURE 7
Database Structure for the Sales/Collection Subsystem

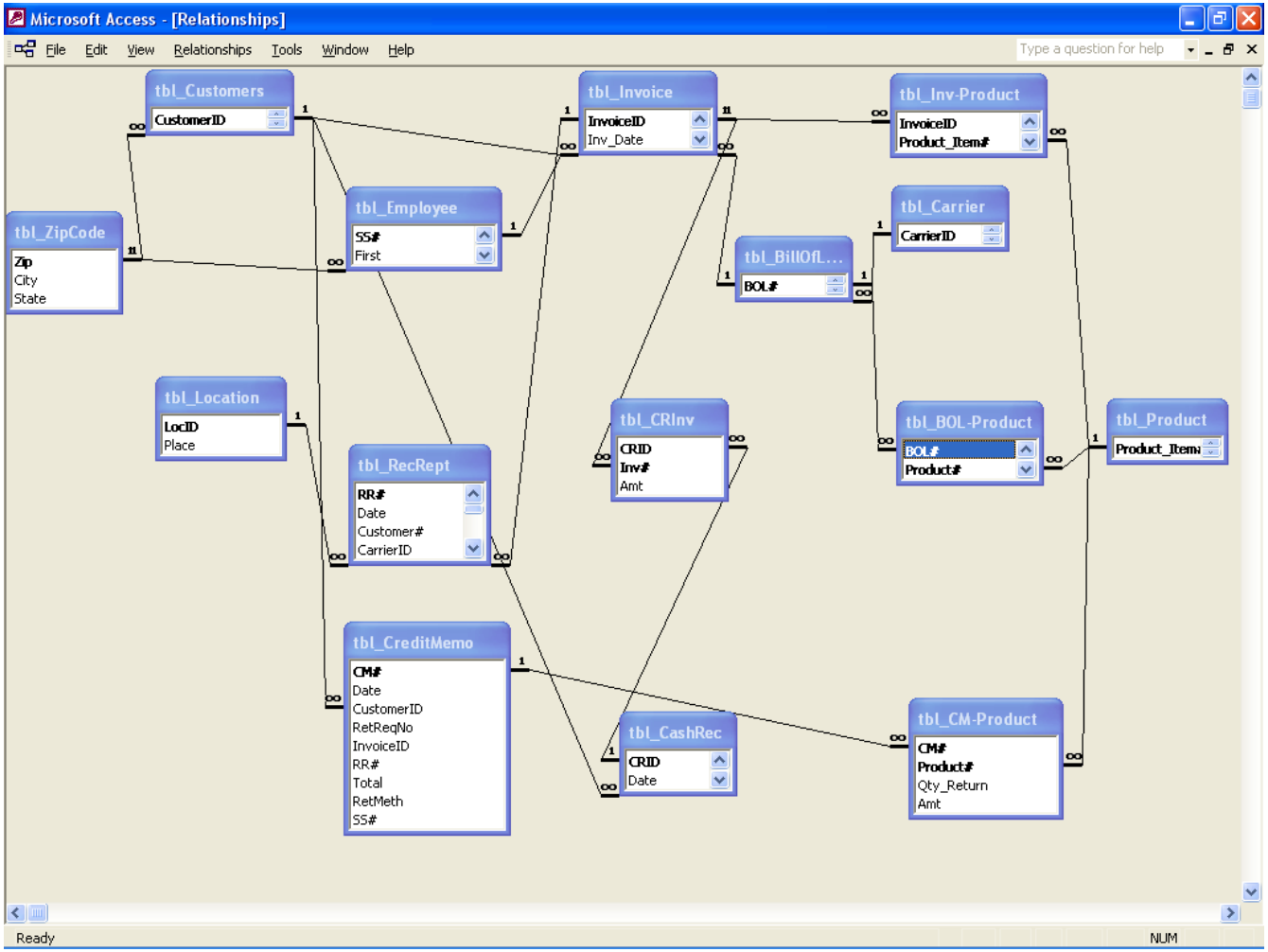


Table 6
List of Tables, Forms, and Reports for the Sales/Collection Subsystem

Tables (Based On The Files Identified In The Conceptual Data Model And Logical Design):

1. Customer
2. Employee
3. Product
4. Invoice
5. Invoice-Product (for bridging many-to-many relationship)
6. BillofLading
7. Carrier
8. BOL-Product (for bridging many-to-many relationship)
9. Receiving Report (or RR)
10. Location
11. RR-Product (for bridging many-to-many relationship)
12. Cash Receipt (or CR)
13. CR-Invoice (for bridging many-to-many relationship)
14. Credit Memo (or CM)
15. CM-Product (for bridging many-to-many relationship)
16. Zip Code

Forms (Based On Some Of The Processes And Files Identified In The DFD Summary):

Forms are created to provide user interfaces for viewing, adding, deleting, and updating data in the database. Forms can also be used to communicate between the system and the users (e.g., users' choices). As a result, the following forms are created:

1. Customer (based on Customer table)
2. Employee (based on Employee table)
3. Product (based on Product table)
4. Invoice with a sub-form (based on Invoice & Invoice-Product tables)
5. BillofLading with a sub-form (Billof Lading & BOL-Product tables)
6. Carrier (based on Carrier table)
7. RecReport with a sub-form (Receiving Report & RR-Product tables)
8. Location (based on Location table)
9. CashRecpt with a sub-form (Cash Receipt & CR-Invoice tables)
10. CreditMemo with a sub-form (Credit Memo & CM-Product tables)
11. Zip Code (based on Zip Code table)
12. Switchboard (navigate between forms and reports)

Reports (Based On Some Of The Processes Identified In The DFD Summary):

1. Customer list in alphabetical order based on the customer's name
2. Employee list (alphabetical)
3. Product list (sorted by product number)
4. Invoice (based on a specific Invoice number)
5. Bill of Lading (based on a specific Bill of Lading number)
6. Carrier list (alphabetical)
7. Receiving Report (based on a specific Receiving Report number)
8. Location list (alphabetical)
9. Cash Receipt (based on a specific Cash Receipt number)
10. Credit Memo (based on a specific Credit Memo number)
11. Zip Code list (alphabetical based on State & City, ascending order)

TABLE 7
Testing

Zip Code

Zip	City	State
60613	Chicago	IL
60614	Chicago	IL
60615	Chicago	IL

Customer

Cust#	Name	Address	Zip	Contact	Phone
406	Bertram Appliance	121 E. Front	60613	Mr. Bertram	555-5512
407	Fritter Appliance	13210 S. Logan	60614	Ms. Fritter	555-1359
408	Hanover Hardware	242 W. Holt Road	60613		555-8957
409	Reliance Electric	1231 S. Cedar	60615	Amy	555-6872

Employee

SS#	First	Middle	Last	Initials	Zip	…
365-73-2376	Ray		Kramer	RK	60613	
379-54-9833	Jim		Adams	JA	60614	
367-55-4832	Nancy		Ford	NF	60615	

Product

Product #	Description	Unit Cost	Unit Price
AC-40	Alarm Clock	$15.00	$18.00
B-28	Blender	$38.00	$53.00
CM-15	Coffee Maker	$39.00	$60.00
CO-22	Can Opener	$16.00	$20.00
EFP-510	Electric Frying Pan	$29.00	$40.00

Invoice

Inv#	Cust#	InvDate	PrepBy	PO#	PODate	Prod#	Qty
730	408	12/18/1996	Nancy Ford	TX20901	12/16/1996	AC-40	30
						CO-22	50
						FP-2	25
						T-104	60
731	406	12/22/1996	Nancy Ford	GR253	12/20/1996	B-28	15
						CM-15	35

NOTES