Modeling Accounting Processes Using UML: The Drifters Example

David H. Olsen, (E-mail: dolsen@b202.usu.edu), Utah State University
Robert J. Mills, (E-mail: mills@cc.usu.edu), Utah State University

Abstract

This paper includes a comprehensive case that details a simplified business system for the Drifters Snowboard Company. We include a description of the business processes of Drifters relative to the data needs which includes an entity-relationship (ER) diagram and a unified modeling language (UML) diagram. Both diagrams are constructed to compare and contrast each method and to give those that perform modeling tasks a sense of some of the real world issues that are involved.

In the past ten years, object-orientation has become viable and important in both the modeling and implementation arenas. Programming languages such as C++ and Java incorporate object-oriented (O-O) features and database management systems (DBMS) such as O₂, ObjectStore, Poet, etc. and are advertised as "pure" O-O DBMS. Conversely, some systems such as Oracle, DB2, Sybase, SQL Server etc. have incorporated some O-O features into essentially relational database technology in order to remain competitive.

On the modeling dimension, the entity-relationship (E/R) diagramming method with its notable variants has been dominant for more than 20 years. This is probably due to its simplicity and its power to accurately represent the data needs of nearly all applications. In the last 10 years however, O-O modeling has become increasingly popular because it correlates nicely to O-O implementations and because its modeling power is requisite for the greater complexity inherent in contemporary information systems. Recently, UML has become the de facto O-O modeling variant, so hence, we illustrate UML in the same case we use ER diagramming. The comparison and contrast is useful to those who are migrating to UML data modeling.

I. Introduction

Data modeling is the single most important task in building an accounting information system. A poorly developed accounting data model will never lead to an implementation that yields accu-

rate, timely information no matter the talent level of the implementation specialists. It seems that though there are relatively numerous database implementation specialists, there are few that are accomplished data modelers and even fewer that understand the needs of the accounting community.

Readers with comments or questions are encouraged to contact the authors via email.

The team that designs a new accounting information system must have a thorough understanding of areas as diverse as accounting principles, database design, database implementation, business processes, and strategic initiatives. An ideal candidate might be a seasoned accountant with considerable database design and implementation knowledge.

In this paper we concentrate on the data modeling aspects of designing a basic accounting information system as we introduce the Drifters business case. We compare and contrast the entity/relationship (E/R) data modeling method (Chen, 1976) with the Unified Modeling Language (UML) method in the context of the Drifters case.

In the end, the purpose of a data model is to yield a robust database that is properly constructed, eliminates anomalies, and contains business logic. We demonstrate how the ER and UML methods accomplish these objectives. This paper continues with a review of the accounting database literature in Section II while Section III is an introduction to a standard, simplified database design that supports basic business functions in the context of the Drifters case. The entities that are derived from the Drifters design in Section III are the basis for some normalized tables. In Section IV, a set of UML diagrams germane to the Drifters case is presented and explained. Specifically, we show a sample Use Case diagram, a Class diagram, and a Sequence diagram. Section V concludes with a discussion of the future of accounting data modeling and specifically, the role of UML in designing accounting systems.

II. Literature Review

A principal understanding of database concepts has historically been important in the accounting profession. For example, the AAA Committee on Contemporary Approaches to Teaching AIS (AAA, 1986) recommended that instructors include substantial coverage of data-

base topics in their AIS courses. The extent of database coverage recommended by the Committee includes 1) data coding, 2) file/record design, 3) batch/on-line processing, 4) data structures and file organization, 5) database organization, 6) conceptual data modeling, 7) defining database requirements, 8) model databases, 9) extracting data from databases, and 10) maintenance procedures.

A particularly important part of building a database involves conceptual data modeling which is listed above in item number six. Some alternative data modeling approaches include the E/R model, the REA model and UML. The E/R model is presented in (Chen, 1976) and the REA model, which is a specialized accounting data modeling paradigm is presented in (McCarthy, 1982). UML was a collaborative effort that combined three different approaches and is described more fully in section IV.

The rationale for proficiency using database management systems (DBMS) is evident from the changing role of accountants as information providers in business and industry. While accountants are still very important providers of financial information to managers and parties external to a business enterprise, their role is rapidly changing (Elliott, 1994). Managers, investors, and others now have significantly greater access to sophisticated databases and can query those databases directly to obtain timely custom reports. The accountant's role then seems to include more systems analysis and design as well as supporting querying the information system.

Research seems to support the increasing role that database systems play in the accounting field. Olsen and Kimmell (1998) reported on advanced database technologies that are either beginning to be used or that may be used in the accounting profession. These technologies include 1) object-oriented databases, 2) Internet databases, 3) data warehousing, 4) data mining, 5) active databases, and 6) business rules. These advanced technologies have important implica-

tions for accountants because they will be the basis for future accounting systems. Additionally, research such as (Olsen, 1998) and (Olsen, 2000) report on SQL usage in accounting information systems.

III. Case Study Description And ER Diagram

In this section we introduce a sample case that is used to illustrate some basic UML diagrams and some differences between the UML method and E/R methods. Drifters Snowboard Company is an Internet-based snowboard reseller that markets snowboards for three major manufacturers which include Burton, Simms, and Marrow. Drifters sells three different styles of boards including the Freestyle, Freerider, and Downhill styles. The Freestyle and Freerider snowboards cost between \$800 and \$1200 with quality and size being the determining factors. Downhill boards cost between \$2500 and \$3000 and are mostly used by professional riders.

Credit cards are the only acceptable form of payment as customers enter their orders online. American Express, Discover, Visa and MasterCard are the four types of credit cards that Drifters currently accepts. Drifters advertises extensively at races and resorts during the ski season and also e-mails existing customers during special promotions. Drifters currently processes 1400 orders a day during the fall and winter seasons and 800 orders per day during spring and summer.

Orders received from customers are passed electronically from Drifters' database to the manufacturer's database. Orders are then filled and the product is shipped directly to the customer. Since Drifters does not manufacture snowboards and works directly with the manufacturer inventory, shipping facilities are not needed. Replacement parts are also not needed since the manufacturer warrants all the purchased products.

Customers enter their personal information

directly into the Internet site. This personal information includes the customer's name and complete address along with phone and fax numbers. Likewise, the e-mail address of the customer is also included. Once the customer has confirmed the personal information is correct, they are given a unique customer ID number that can be used to order future additional products.

Products are ordered by selecting the model and quantity and providing credit card information (credit card number, expiration date) into the Internet site. The customer receives a confirmation of the order (order #), the total amount charged to the credit card, and the estimated delivery date. Ordering products via the Internet is not only efficient, but it gives the customer autonomy over the ordering process.

All orders are processed in real-time and are immediately checked for input errors. This allows for immediate handling of orders. If an order does contain incorrect data, such as U8tah or 18\$4 South State Street, the record is flagged, stored, and sent back to the user via e-mail. An order is also sorted by the product manufacturer and sent to an electronic mailbox where it is transmitted in real-time. Once the manufacturer has confirmed an order, the order is returned with a product barcode and an estimated date of delivery. If the product is not in stock, a copy of the order is returned and is placed in the backorder pile. The customer is notified through e-mail about any backordered items. The manufacturer notifies Drifters once the product becomes available.

At the end of each day the status of each order is determined. This information is made available to the customer, who is given an order number so they can track their order from the time the order is received and processed until the package is shipped from the manufacturer(s) to the customer.

An E/R diagram that reflects this case is given in Figure 1 whereas relational tables con-

sistent with the Drifters case are given in Appendix A.

IV. Unified Modeling Language

This section is included to give an overview of the Unified Modeling Language (UML) so that the UML diagrams can be understood in a

proper context. The section also describes the use of UML diagrams used in the Drifter's case. Comprehensive descriptions of UML can be found in (Eriksson & Penker, 1998, 2000) and (Rumbaugh, Jacobson & Booch, 1999).

UML is an object-oriented modeling language that is the result of combining the ap-

Figure 1 - Entity/Relationship Diagram

proaches of Booch, Rumbaugh, and Jacobson (Muller, 1999, p. 127). A major benefit of UML is that it is independent of programming language and development techniques (Evitts, 2000). UML can also be used to describe virtually any business process which widens the utility and usability of the modeling language well beyond traditional object-oriented development.

There are actually many diagram types in the UML that are described in a visual format expressed by graphical symbols and connections (Eriksson & Penker, 1998). As such, UML is a natural choice for the development of systems because they are easily understood and are easy to communicate to others.

UML provides modeling support and solutions for all phases of the systems development process. UML offers a number of views, diagrams, and elements available for modeling systems. Views are useful to represent a "projection of the complete system description, showing a particular aspect of the system" (Eriksson & Penker, 1998, p. 14). Diagrams show model elements symbols arranged to illustrate a particular part or aspect of the system" (Erikson & Penker, 1998, p. 17). Model elements are a significant feature of UML and include the concepts used in the diagrams (Erikson & Penker, 1998). A list of common UML views, diagrams, and

model elements are provided in Table 1. These views, diagrams, and model provide a blueprint for the design and development of complex systems.

In this manuscript, we include a Use-Case Diagram, a Class Diagram, and a Sequence Diagram to demonstrate a few of the design features available in the Unified Modeling Language. The software program Rational Rose Enterprise Edition 2000 was used to develop the diagrams for this case.

Use-Case Diagram

The Use-Case is a basic modeling technique that describes the interactions between outside users and the system being developed. The Use-Case model is used by system developers and outside users to help lead to a requirement specification of the system which describes what the system should do. The goals of the external actors are described in the system. The Use Case Diagram is ideal for proper communication of ideas between the software developer and the customer. It is simple enough for domain experts who specialize in their own field to understand and also serves as an unambiguous method of communication between them and the software developer unlike other types of software specification or natural language.

Table 1. UML Components

<u>Views</u>	<u>Diagrams</u>	<u>Model Elements</u>
Use-case view	Use-case Diagram	Adornments
Logical view	Class Diagram	Notes
Component view	Object Diagram	Specifications
Concurrency view	State Diagram	Stereotypes
Deployment view	Sequence Diagram	Tagged Values
	Collaboration Diagram	Constraints
	Activity Diagram	
	Component Diagram	
	Deployment Diagram	

*Items in **bold** are included in this manuscript

Figure 2 – Use Case Diagram

For instance, a customer using the Drifter's system is an example of an external actor. The customer has three primary goals from the Drifter's system point of view. The first goal is to provide the Drifter's database with personal information (i.e, name, address, e-mail address). The second goal of the customer is to place an order. Finally, the customer enters payment information into the Drifter system. In part, the Use-Case Diagram is designed to describe how this snowboarder uses and interacts with the system using a very general, macro-level approach.

Two additional external actors described in Figure 2 include the snowboard supplier and the credit card company. In the context of this system, the goals of the supplier include receiving and confirming orders, providing delivery information (i.e., estimated date of delivery), and shipping the product to the customer. In addition, the credit card company includes the goal of processing the credit card of the customer.

The Use-Case model does not describe the internal structures of the system that are hidden

in this model. Designing an overly complex Use-Case model is a common mistake made by software designers. Detailed interactions among external actors and the system being analyzed should be saved for other models available in the UML modeling language.

Class Diagram

The Class diagram “is a structural or static diagram with which you model the structure of a system of classes” (Muller, 1999, p. 128). Various types of relationships like dependency, aggregation, association, composition, generalization etc. are used to link these classes. In a class diagram these links need to be persistent i.e. they have to stick to static structuring and should not model any dynamic behavior. A class is a description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. The Class diagram in UML corresponds closely to the structure in ER diagramming since the OO notations on which UML was developed were based upon ER notations.

In UML the class diagrams are drawn in a phased manner. Firstly, the diagrams are drawn from a conceptual perspective, next from a specification perspective and finally from the implementation perspective. With each phase the model becomes more specific and low level and the detail increases, until the class diagrams take us very near to the implementation. At the end of the three phases all the necessary foundation is laid to go in directly for implementation.

Figure 3 is a Class diagram that represents the Drifter’s system. It is interesting to note some similarities between the Class diagram in Figure 3 and the ER diagram in Figure 1. This is not surprising as ER and Class diagrams attempt to accomplish the same goal; namely to capture the structure of a given system which are often ultimately used to design relational tables.

There are some notational differences between the two methods that are illustrated with

the Customers and Orders classes. A customer may place 0, 1, or many orders which is illustrated with a “0..*” on the connecting line closest to the Orders class. An order is assigned to exactly one customer which is illustrated with a “1” on the connecting line closest to the Customers class.

Another interesting part of a Class diagram is constraints. We include a constraint on the credit card name to only allow “Visa”, “American Express”, “Discover”, or “Mastercard” as valid entries.

A useful Rational Rose UML implementation feature is the code generation which includes facilities for C++, Visual Basic, and SQL database DDL (data definition language). Appendix B includes a sample of the DDL that was generated by Rational Rose from the Class Diagram. Note that the data types are generic but they could be changed when the classes are designed or modified.

Sequence Diagram

The Sequence Diagram is very similar to the traditional flowchart used by systems analysts. Unlike in the Class Diagrams the relationships and links in the Sequence Diagrams exist between specific objects and not between classes. The Sequence Diagram reflects “two of the basic control structures: sequence and condition. The sequence is the directed connection between the elements; a condition is the equivalent of an IF statement” (Muller, 1999, p. 94). A Sequence Diagram models the dynamic behavior of the objects and is not intended to represent a static structure. The Sequence Diagram is very effective to describe procedural information pertaining to the system being described. The Sequence Diagrams are generally used to show a few important sequences of actions involving the various objects in the system. In the Drifter’s case, the Sequence Diagram is used to describe the procedural interactions among actors. For

Figure 3 – UML Class Diagram

Figure 4 – UML Sequence Diagram

instance, the Sequence Diagram in Figure 4 indicates that the credit check takes place before determining product availability with the manufacturer database on product availability. The Drifter's Sequence Diagram also describes the responsibilities of the Drifter's system as well as the manufacturer's system with regards to shipping and the notification of shipping to the customer.

UML also includes a host of other diagrams and elements such as the State Diagram, Collaboration Diagram, Activity Diagram, Component Diagram, and the Deployment Diagram to assist in the modeling of business and software systems that are not described in this manuscript.

V. Conclusion

We presented a comprehensive case that compared some UML diagrams with the well-known E/R method in order to illustrate some of the differences between the two approaches and also to underscore the importance of the data modeling process.

The UML data modeling method has two distinct advantages. First, because UML is object-based, the data structures that are the results of a UML CASE tool diagram are similar to the diagram itself whereas the data structures that are the results of an ER CASE tool sometimes bear less resemblance to the diagram. This seems to make it easier to proceed from the logical modeling phase to the relational table design phase.

The second advantage of UML over the ER method is that there are many more UML constructs and diagrams that are available. Though the ER method is simple and elegant, it is restricted in its expressive power. This implies that the UML method has a much longer learning curve but it has greater capability in expressing business rules. UML is moving our field toward a "more mature vision of a collaborative nature of software development – one that is typical of genuine professional practice" (Evitts, 2000, p. 3). As a result, there has been a valuable shift of focus from an exclusive programming perspective toward an equally valuable focus on the design and modeling processes of software development.

Future research in this area includes focusing on UML as a diagramming tool to describe the audit process, financial accounting processes, and web-based human/computer interaction processes relative to accounting and auditing instruction. Additionally, as the use of UML in modeling accounting process enters the mainstream of the accounting profession, future research is necessary to facilitate this transition. Additional research is necessary to discover

"best practices" that will improve the integration of UML and accounting processes and maximize the impact of UML on the design of accounting information systems. Future research is also necessary to examine more carefully the effects of using components of UML at various phases of the data modeling process, particularly the class and activity diagrams.

Finally, future research in this area should also examine the expansion on UML as a diagramming tool to describe the audit process, financial accounting processes, and web-based human/computer interaction processes relative to accounting.

References

1. American Accounting Association, "Report of the AAA Committee on Contemporary Approaches to Teaching Accounting Information Systems." May 1986.
2. Chen, P.P., "The Entity-Relationship Model: Toward a Unified View of Data." *ACM Transactions on Database Systems* (January 1976): 9-36.
3. Elliott, R. K., "The Future of Audits." *Journal of Accountancy*, (September 1994), pp. 74-82.
4. Eriksson, H. and M. Penker, *Business Modeling with UML: Business Patterns at Work*, New York, John Wiley & Sons, Inc., 2000.
5. Eriksson, H. and M. Penker, *UML Toolkit*, New York, John Wiley & Sons, Inc., 1998.
6. Evitts, P., *A UML Pattern Language*, Indianapolis, IN, Macmillan Technical Publishing, 2000.
7. McCarthy, William E., "The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment," *The Accounting Review* (July 1982) pp. 554-78.
8. Muller, R. J., *Database Design for Smarties: Using UML for Data Modeling*, San Francisco, CA, Morgan Kaufmann Publishers, 1999.
9. Olsen, D., "Accounting Database Design

- and SQL Implementation” Review of Accounting Information Systems (Summer, 1999).
10. Olsen, D., “Accounting Database Design and SQL Implementation Revisited” Review of Accounting Information Systems Spring 2000).
 11. Olsen, D., and S. Kimmell, “Towards Integrating Advanced Database Concepts into Accounting” Review of Accounting Information Systems (Summer, 1998).
 12. Rumbaugh, J., Jacobson, I, and G. Booch, *The Unified Modeling Language Reference Manual*, Reading, MA, Addison-Wesley Longman, Inc., 1999

Appendix A Proposed Tables

Customer Table (10 tuples):

{CustomerID, FirstName, LastName, Address, City, State, ZipCode, Country, Phone, Fax, Email, ConfirmedCorrectness}

Supplier Table (10 tuples) :

{SupplierID, Name, Address, City, State, ZipCode, Country, Phone, Fax, WebAddress, ContactsEmail}

Order Table (20 tuples) :

{OrderID, *CustomerID*, DateOrdered, EstimatedDeliveryDate, TotalOrder, PaidInFull}

Payment Table (20 tuples) :

{PaymentID, *OrderID*, *CreditCardID*, CCNumber, ExpirationDate, PaymentAmount}

CreditCardID Table (5 tuples):

{CreditCardID, CreditCardName}

Style Table (3 tuples from the case description):

{StyleID, StyleName}

Model Table (10 tuples):

{ModelID, *StyleID*, *SupplierID*, ModelName, Description, Cost, Price, Size}

Order Items Table (40 tuples):

{*OrderID*, *ModelID*, Quantity, Extended_Price, OnBackOrder, ActualDeliveryDate}

Appendix B Sample SQL CREATE Generated From UML Class Definitions

```
CREATE TABLE T_Customers(
    CustomerID VARCHAR UNIQUE,
    FirstName VARCHAR,
    LastName VARCHAR,
    Address VARCHAR,
    City VARCHAR,
    State VARCHAR,
    ZipCode VARCHAR,
    Country VARCHAR,
    Phone VARCHAR,
```

Fax VARCHAR,
Email VARCHAR,
ConfirmedCorrectness VARCHAR,
OrderID VARCHAR,
FOREIGN KEY (OrderID) REFERENCES T_Orders,
PRIMARY KEY(CustomerID))

CREATE TABLE T_Orders(
OrderID VARCHAR,
CustomerID VARCHAR,
DateOrdered VARCHAR,
EstimatedDeliveryDate VARCHAR,
TotalOrder VARCHAR,
PaidInFull VARCHAR,
CustomerID VARCHAR UNIQUE,
FOREIGN KEY (CustomerID) REFERENCES T_Customers,
PRIMARY KEY(OrderID))

Notes