This is to certify that the

dissertation entitled

An Investigation of Abstraction
in Events-Based Accounting Systems

presented by

Cheryl Lynn Dunn

has been accepted towards fulfillment
of the requirements for

PhD degree in Accounting

William E McCarthy
Major professor

Date March 25, 1994

AN INVESTIGATION OF ABSTRACTION IN EVENTS-BASED ACCOUNTING SYSTEMS

By

Cheryl Lynn Dunn

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Accounting

1994

ABSTRACT

AN INVESTIGATION OF ABSTRACTION IN EVENTS-BASED ACCOUNTING SYSTEMS

By

Cheryl Lynn Dunn

Researchers have recently expounded the virtues of events-based accounting systems. The most prominent criticism of events-based accounting systems is that they provide more information than human users can handle, resulting in information overload. The REA accounting model proposed by McCarthy (1982) suggests that the inclusion of an abstraction hierarchy in the user interface of an events-based accounting system will make the information load manageable. Abstraction is the suppression of irrelevant details and the emphasis of details appropriate to a given decision. The principle of abstraction has been highly touted in the computer science literature as a means of controlling complexity.

Abstraction hierarchies decompose system models into multiple views with varying levels of detail and are thus advocated for systems which have multiple users with varying needs. The concepts of abstraction and abstraction hierarchies have been subject to little empirical testing in the computer science literature. The only studies found examined abstraction by comparing user performance with data models said to be more or less abstract than one another. These studies found conflicting results. The current study compares performance between two groups of users of an events-based accounting system: one with an abstraction hierarchy built into its user-interface and one without the abstraction hierarchy in its user-interface.

ACKNOWLEDGEMENTS

I would like to thank Matthew Anderson, William Punch III, and Jon Sticklen for serving as committee members for this dissertation. Their comments and assistance were invaluable to me.

Special thanks are due to William McCarthy for being the chairperson of this committee. Bill's expertise, guidance, and friendship were essential in enabling me to see this project through to fruition.

I am also grateful to Deloitte & Touche and to the Department of Accounting at Michigan State University for providing financial support for this dissertation.

The most heartfelt appreciation I feel is for my family: Jim and Jimmy Dunn. Besides providing moral support, Jim took over almost all household chores in order for me to devote maximum attention to this project. Jimmy was as patient and understanding as any 2-3 year old could be, considering the number of times he wanted Mommy to play and Mommy had to work. My parents, Bob and Issie Scott, and my parents-in-law, James and Janet Dunn, also provided a great deal of moral support throughout my doctoral program.

TABLE OF CONTENTS

CHAPTER 3 - CONSTRUCT AND HYPOTHESIS DEVELOPMENT (Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

CHAPTER 1 - INTRODUCTION

The REA (Resources-Events-Agents) accounting model was proposed by McCarthy (1982) as a new method for implementing accounting systems with "events" orientations (Sorter 1969). The most prominent criticism of events accounting systems is that information overload causes the benefit of disaggregated information to be outweighed by the cost of additional cognitive processing which is necessary (Davidson and Trueblood 1961, Revsine 1970). Several empirical studies have compared user performance with aggregated versus disaggregated data. Each study yielded results which support this criticism (e.g. Chervany and Dickson 1974; Benbasat and Dexter 1979; Casey 1980; Otley and Dias 1982). No empirical refutation of this criticism has been published in the accounting literature.

The REA solution for information overload in an events-based accounting systems implementation is abstraction. Abstraction is the suppression of detail that is irrelevant for a given decision. Abstraction has been proposed as a means for controlling complexity in the fields of computer science and cognitive psychology. McCarthy (1982, 1987) has proposed applications of abstraction to various elements of the REA model. These applications promote a user orientation because they can be used to select various levels of detail of a company's financial database for further examination. By allowing selective suppression of detail, the use of abstraction is consistent with FASB Concept Statements 1 and 6 (FASB 1989). These concept statements attempt to provide

guidelines which will result in provision of information useful to present and potential investors and creditors and to other users at a reasonable cost. The Board considers it a dilemma that "the optimal information for one user will not be optimal for another" (FASB 1989, 5199). It has sought means for allowing users to select only the information that is relevant for given decisions. The REA model's recommended use of abstraction provides a possible solution because it permits users to look at an overall schema of the available pieces of information and to then select only the pieces deemed useful for further examination.

The research question addressed in this study is whether the inclusion of abstraction in an interface to a financial database enhances user performance by mitigating data overload. This question is addressed by first building two interfaces to a database of financial information, one with abstraction and one without. User performance with the two systems is then compared. Results have implications for systems researchers and designers who are interested in determining how best to implement events-based accounting systems and also for systems instructors who are interested in determining how best to teach events-based accounting systems. This project also contributes to the investigation of the feasibility of database financial reporting, a goal which is consistent with a proposition by the University of Southern California Financial Accounting Study Group (1991, 11):

> Research and field tests should be undertaken to determine the feasibility of developing databases and systems to permit users to obtain the information they perceive necessary to meet their decision needs in whatever format they may desire, with the

expectation that such information would be subjected to the attest process.

The remainder of this dissertation is organized as follows: Chapter Two discusses the theoretical foundations for the experimental interfaces: the principle of abstraction for the abstraction interface and the theory of normalization for the non-abstraction interface. Chapter Three discusses construct development and presents hypotheses arising from these discussions. Chapter Four describes the methodology used to test the hypotheses. Chapter Five discusses statistical tests and presents the results of the experiment. Concluding comments and suggestions for future research are offered in the final chapter.[1]

End Notes:

1. This paper assumes some knowledge of technical terminology as it is used in the events-based accounting and in the database literature. The reader who is unfamiliar with such terminology is directed to McCarthy (1987) or to Date (1986).

CHAPTER 2 - THEORETICAL FOUNDATIONS: PRINCIPLE OF ABSTRACTION AND NORMALIZATION THEORY

2.1 THE PRINCIPLE OF ABSTRACTION

Brodie (1981) claims "The principle of abstraction is to suppress irrelevant details of an object under consideration and to emphasize details appropriate to the current context" (p. 102).[1] This is a commonly accepted definition of abstraction as it is used in computer science. Three types of abstraction are most widely used in computer science. These include (types along with their inverses): (1) aggregation (decomposition), (2) generalization (specialization), and (3) classification (instantiation). These were proposed by Smith and Smith (1977, 1978) and further developed by Brodie (1981, 1984) and by Brodie and Ridjanovic (1984). As of the present, these three types of abstraction seem to form the standard in the database literature (Batini, Ceri and Navathe 1992).

Aggregation is a concept which implies a relationship between objects (e.g. bride, groom, priest) connoting a higher level object or concept (e.g. marriage) (Smith and Smith 1977). This allows suppression of detail in that many component object details can be ignored, and focus may be placed on details which apply to the overall relationship. **Generalization** is a process in which a set of similar objects (e.g. cow, horse) is considered to be a generic object (e.g. animal) (Smith and Smith 1977). Any individual differences between objects (e.g. gives milk, has mane) may be ignored, and emphasis is placed on those attributes common to all (e.g. breathes). **Classification** is an abstraction mechanism in which instances of an object class (e.g. Clint Eastwood, Charles Bronson) are

categorized (e.g. movie star). As with generalization, any individual differences between the instances may be ignored in such an abstraction. These three types of abstraction will be discussed in more detail shortly.

2.1.1 Abstraction Hierarchies

The current study will investigate the use of an "abstraction hierarchy" as advocated by Smith and Smith (1977, 1978). An abstraction of a system is a model of that system that deliberately omits certain details. A single abstraction consists of one view of data resulting from the application of one or more of the three abstraction methods. An abstraction hierarchy decomposes the model into multiple views at varying levels of detail. Smith and Smith suggest two disadvantages of single abstractions which can be alleviated by the use of abstraction hierarchies. First, decomposition of a single abstraction into a hierarchy may be necessary to make a system with many relevant details intellectually manageable.

For example, a telephone directory for a company consisting of employee name, position, department and telephone number (in alphabetical order by employee name) contains many relevant details. Depending on the information needed, it may not be intellectually manageable. If a user wanted to contact the Personnel Director, but did not know that person by name, it would be quite a task to find it. If abstraction methods are used to group employee names and phone numbers according to their departments and positions, the information would become more intellectually manageable.

Second, an abstraction hierarchy permits sharing of a single model by several diverse users without compromising their access requirements. A single abstraction omits details as dictated by the expected users and by the intended application of the abstraction. That particular view may not be useful for other applications or for other users. For example, consider a person with complex cash flow management responsibilities. This user would need very detailed cash receipt information, including trend analysis of which day(s) of the week cash is most likely to come in, etc.

Contrast such information needs with those of a user who simply needs to know overall accounts receivable information. If a single abstraction is provided which contains the detailed information needed by the cash flow manager, the accounts receivable user is likely to be lost in the detail. If a single abstraction which contains total cash receipts categorized by type (e.g. sales, stock issuances, loan proceeds, etc.), the accounts receivable user will be satisfied, but the cash flow manager will not have enough detail. An abstraction hierarchy would allow the user to select a presentation of the cash receipt information congruent with the level of detail needed.

The use of abstraction hierarchies suggested by Smith and Smith is consistent with the concept of *recursive decomposition* which Palmer and Kimchi (1986) claim is central to information processing theories. They discuss this concept, as it applies to aggregation, as follows:

> Any complex (non-primitive) informational event at one level of description can be specified more fully at a lower level by *decomposing* it into (1) a number of components, each of which is

> itself an informational event, and (2) the temporal ordering relations among them that specify how the information "flows" through the system of components (p. 47).

Palmer and Kimchi claim the rationale for using recursive decomposition is to reduce complexity. They note that (at least in principle) the decomposition should "factor out" some portion of the complexity implicit in a unitary information event by making it explicit in the flow relations among a number of simpler events. They note that as a user moves down a decomposition tree, the internal complexity of the component operation should decrease, warning that this reduction comes at the cost of more components and more complex flow relations among them.

2.1.2 REA in an Abstraction Hierarchy

The abstraction hierarchy which is included in the interface tested in this study is based on the REA applications of abstraction proposed by McCarthy (1982, 1987) and further developed by Gal and McCarthy (1992). When used in an interface to an events-based database, an abstraction hierarchy allows a user to first examine an overall picture of a company and then to select an area of the firm on which to "zoom in." From an overall picture of that area, the user can zoom in further to examine sub-areas, then specific data types, and finally data instances. The concept of progressive "zooming" is presented clearly in Figure 2.1, which is taken from Davis and Olson (1985, 550) and in turn from Herot, Carling, Friedell, Kramlich, and Rosenberg (1981).

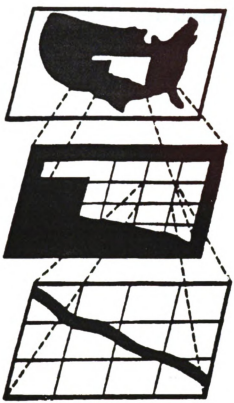


Illustration of progressive "zooming" through a spatial data management system. [Figure 17-10 from Davis and Olson (1985) and Figure 2.2 from Herot et al. (1981)]

FIGURE 2.1: PROGRESSIVE ZOOMING

As this example illustrates, the user is first presented with an overall map of the United States, on which low level details are omitted (i.e. with only major interstate highways and names of major cities included). The user can select a portion of the United States (e.g. a state) to see more detail. The user would be presented with a map of that state, with not only major interstate highways and large cities, but also intrastate highways, main roads, and names of all cities. The

user can then focus on a part of that state, for example a city. The user would zoom in to bring up a city map, with all streets identified, and any other low-level details that expected users of the map may need.

2.1.3 An REA Abstraction Hierarchy Example

To make the three abstraction techniques as applied to REA more clearly understood, a series of conceptual models is provided as an example.[2] The first model presented (Figure 2.2a) is a natural language description of the economic philosophy of business enterprises. The second (2.2b) depicts an accounting cycle template (pattern or guide) which is an overall picture of the inherent duality of economic transactions of business enterprises (Dunn and McCarthy 1992). A company will transfer economic resources to outside agents with the optimistic expectation that these transfers-out will eventually result in transfers-in of more valuable economic resources.

Figure 2.2(c) expands the overall accounting template into a more detailed diagram. Some of the various cycles which are typically found in manufacturing companies are presented. The user can get a quick overall picture of the firm from this diagram without needing to see detailed information about every transaction. The user can choose a cycle to focus on (for example the conversion cycle). Figure 2.2(d) presents an entity-relationship[3] (E-R) diagram which partially represents the conversion cycle. With E-R modeling, entities are portrayed as boxes, and the relationships between entities are represented by diamonds. Typically only two boxes are present in an overall diagram of a given

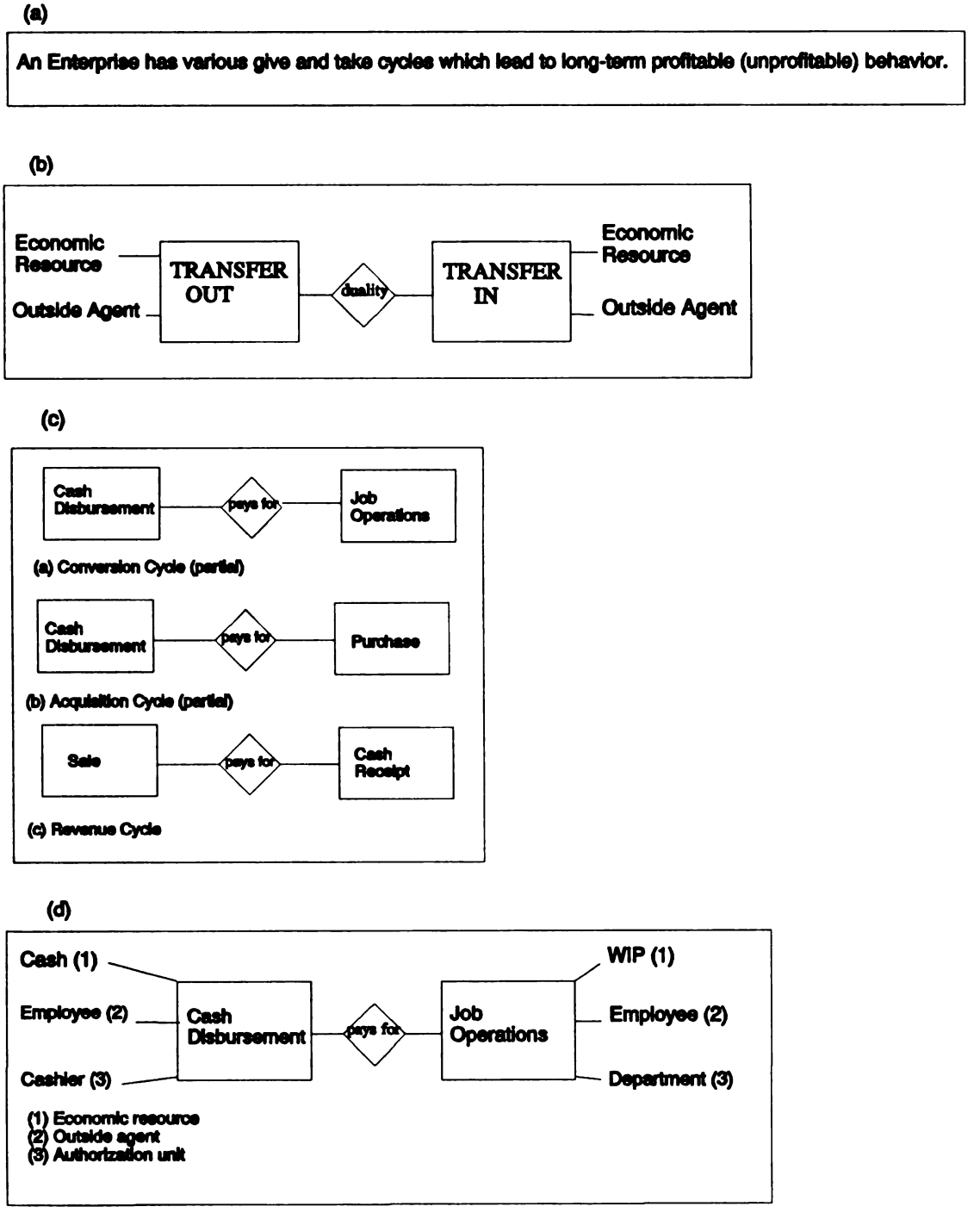


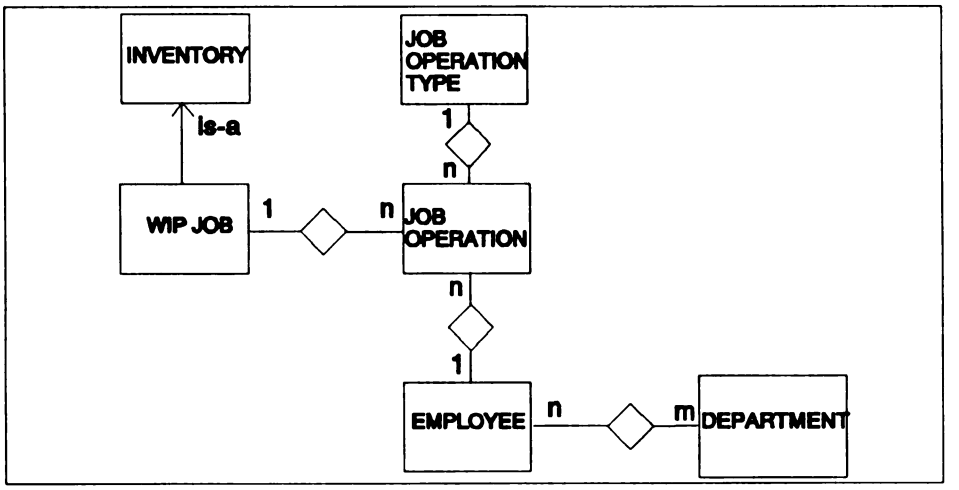
FIGURE 2.2: AN ABSTRACTION HIERARCHY
[adapted from Gal and McCarthy (1992)]

cycle. One represents the transferring in of a resource, the other a transferring out, consistent with the overall diagram in Figure 2.2(b).

The model in Figure 2.2(d) consists of two entities: **Job Operations** (the transferring in of a resource) and **Cash Disbursements** (the transferring out of a resource). These are two parts of a company's conversion cycle (**Raw Material Issues** and **Transfers of WIP to Finished Goods** would be two additional parts which could be modeled separately). The diagram in 2.2(d) does not provide much detail; it portrays the company's job operations portion of the conversion cycle at a very high level of abstraction (low level detail is suppressed). A user can, however, "zoom in" on this cycle and get a more detailed picture.

Figure 2.3(a) shows a more detailed, partial E-R diagram of the job operations portion of the firm's conversion cycle. The notations of "1" or "n" on the lines connecting entities indicate the cardinality of the relationship between those entities. The cardinality of a relationship refers to the maximum number of objects of one entity set that can be related to another set, and vice versa. For each mapping of a relation we can specify cardinalities for both directions. For example, the "1" by **WIP Job** and the "n" by **Job Operation** indicate that for this company, one WIP Job can have more than one job operation, but any job operation can belong to only one WIP Job. It is also possible to model "many-to-many" or "one-to-one" relationships in an E-R diagram. For example, many-to-many relationships are portrayed with an "m" on one side of the diamond and an "n" on the other. Such a relationship is depicted in Figure 2.3(a) between **Department** and **Employee**. This relationship indicates that for this company, a

(a)



(b)

INVENTORY

Inventory Number	Description	Quantity on Hand

WIP JOB

Job Number	Total Cost

EMPLOYEE

Employee SSN	Employee Name	Employee Wage Rate

EMPLOYEE - DEPARTMENT

Employee SSN	Department Code

DEPARTMENT

Department Code	Department Name	Employee SSN

JOB OPERATION TYPE

Job Oper Type	Standard Time

JOB OPERATION

Job Timecard	Actual Job Time	Job Oper Type	Employee SSN	Job Number

(c)

EMPLOYEE

Employee SSN	Employee Name	Employee Wage Rate
001-01-0001	Joe	$8.72
002-02-0002	Dan	$6.80
003-03-0003	Bob	$6.13

FIGURE 2.3: AN ABSTRACTION HIERARCHY (continued)

department can have more than one employee, and that an employee may be assigned to more than one department. The "is-a" relationship included in this diagram represents a generalization and will be discussed directly in the Generalization/Specialization subsection.

Although the picture in Figure 2.3(a) portrays the major objects in this cycle (entities and the relationships between them) it does not include lower level details about the attributes of these entities and relationships. The user can progress down yet another level to examine the relational tables for any of the objects in this diagram. Figure 2.3(b) illustrates example table headings. Such a view informs users of which attributes of the entities and relationships are included in the database. A user can then select specific tables for detailed examination. For example, if the user is interested in the employees of the firm, looking at Figure 2.3(b) may reveal that all of the relevant attributes are represented in the **Employee** table. The user can then zoom down as in Figure 2.3(c) and look at the specific **Employee** instances to obtain the desired information.

The reader should now have a general idea of what the individual levels within an REA model abstraction hierarchy might look like. The levels used in Figures 2.2, 2.3 and 2.4 are theoretically based on the ordering of the abstractions used by McCarthy (1982, 1987) and by Gal and McCarthy (1992). Figure 2.4 presents a picture of the overall hierarchy, and is discussed further in the Abstraction Hierarchies subsection. Following is a more detailed discussion of the various types of abstraction, and of how they are represented in this hierarchy.

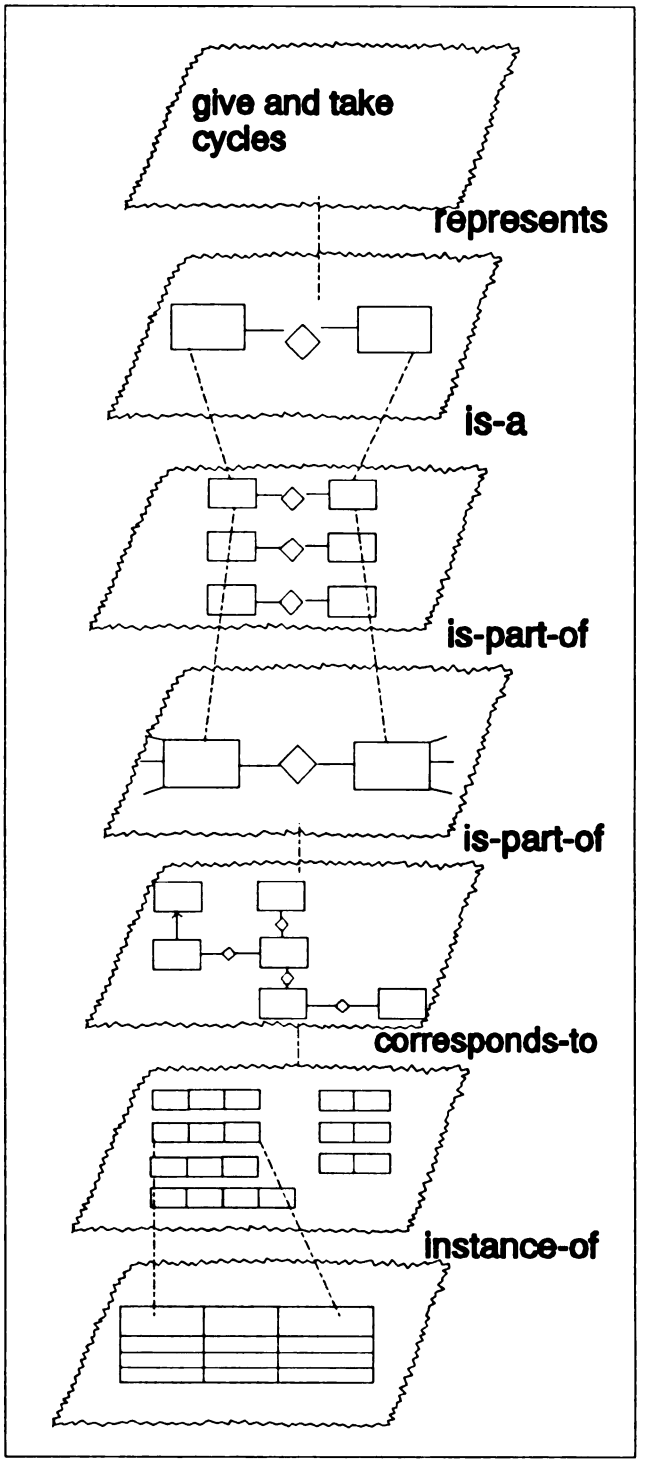


FIGURE 2.4: AN ABSTRACTION HIERARCHY SUMMARIZED

2.1.4 Aggregation/Decomposition:

Aggregation is the clustering of different items to form an aggregate entity. This is a type-to-type relationship in that it deals only with the object types (table intension in a relational model), and not with the instances (table extension). Aggregation is used as an abstraction technique in this conceptual model in two ways. First, the entities are aggregations of different attributes. For example, the two attributes ***Job #*** and ***Total-Cost*** are aggregated to form the entity **WIP Job** (see Figure 3b). Or inversely, the entity **WIP Job** can be decomposed into the attributes ***Job#*** and ***Total-Cost***. The aggregation relationship is often described as "is-part-of." For example, ***Job #*** is part of **WIP Job**, and ***Total-Cost*** is part of **WIP Job**. A higher level of aggregation is also included in this model, in that entities may be aggregated to form relationships. For instance, the two different entities **WIP Job** and **Job Operation** are aggregated to form a relationship containing the ***attributes Job #*** and ***Job-Time-Card***. Or inversely, the relationship of **WIP Job** to **Job Operation** can be decomposed into the two separate entities.

2.1.5 Generalization/Specialization:

Another type-to-type relationship that can be specified in a conceptual model is that of generalization. Generalization clusters related items to form a generic entity. This type of relationship is often discussed in terms of subtypes and supertypes. For example, "hammer," "screwdriver," and "wrench" could be considered subtypes of the supertype "tools." "Tools" is a generalization of those three subtypes. The presence of generalization as an abstraction technique is

usually indicated by an "Is-A" relationship in the model (as in Figure 2.3a). In this example, **WIP Job** is a type of **Inventory**. Other types of inventory would be **Raw Materials** (RM) and **Finished Goods** (FG). These are related items. Thus **Inventory** is a generalization of **RM, FG** and **WIP Job**, and **RM, FG** and **WIP Job** are specializations of **Inventory.**

2.1.6 Classification/Instantiation:

Classification is a simple form of data abstraction in which a set of instances is grouped into a single class. It represents a type-instance relationship (a relationship between an object type and the specific instances of that object), thus it represents a lower level of abstraction than do aggregation and generalization. That is, lower level details are highlighted, and higher level parts of the model are suppressed. Figure 2.3(c) zooms in further on the conversion cycle example, to reveal the actual instances of the entity **Employee**. *Joe*, *Dan* and *Bob* are all instances of the **Employee** entity. Inversely, *Joe*, *Dan* and *Bob* all could be classified as employees.

2.1.7 Abstraction Hierarchies Revisited:

Abstraction hierarchies consist of multiple single abstractions which are connected to each other by one of the abstraction techniques. Figure 2.4 shows the entire abstraction hierarchy which represents Figures 2.2(a-d) and 2.3(a-c). Figure 2.2(b) is simply a data model representation of the natural language found in 2.2(a). Figures 2.2(b) and 2(c) are related by generalization (the conversion cycle **is a** general economic cycle, the acquisition cycle **is a** general economic

cycle, etc). Figures 2.2(c) and 2.2(d), and Figures 2.2(d) and 2.3(a) are related by aggregation/decomposition. **Cash**, **Employee**, **Inventory**, etc are all components of the job operations portion of the conversion cycle. The relationships between those various entities are also components of the partial conversion cycle. Figure 2.3(b) is a relational representation (a decomposition) of the E-R diagram found in 2.3(a). Figures 2.3(b) and 2.3(c) are related by classification/instantiation. *Joe*, *Dan*, and *Bob* are instances of the entity **Employee**.

The order of the abstractions used in this abstraction hierarchy is the same as those used in the REA literature (McCarthy 1982, 1987; Gal and McCarthy 1992). Application of the abstraction mechanisms in various orders is certainly possible. However, there is no theoretical foundation for alternative orderings, as there is for the order suggested by the REA model. The abstraction hierarchy interface used in this experiment thus represents the REA model.

2.2 NORMALIZATION THEORY AND THE UNIVERSAL RELATION MODEL

Normalization theory was introduced to database design as a means of controlling integrity maintenance problems such as insertion, deletion, and update anomalies[4]. These problems cause inconsistency (e.g. requiring the same changes to be applied to multiple data instances) or accidental loss of data. Normalization is driven by functional dependencies (Date 1986; Loomis 1987; Batini et al. 1992). A functional dependency exists between two single-valued attributes if each value of attribute 1 (a_1) corresponds to exactly one value of attribute 2 (a_2). In a well-normalized database, the only theoretically desirable functional dependencies are

between the key(s) of an entity and each of the other attributes of that entity. The process of normalization is a set of rules for systematic detection and elimination of undesired functional dependencies. Each rule which is successfully applied results in a particular "form" of normalization, e.g. First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), etc.

The universal relation is often discussed in conjunction with normalization. This model suggests that it is possible to define an initial universal relation, involving all attributes relevant to a database under consideration. A decomposition algorithm can then be applied to create a well-normalized structure for that database (Kent 1981, 1983; Ullman 1983). A database created by such an algorithm would be identical to one created by conceptual modeling, with the exception that the tables would not be given semantic names. Thus the tables would be called something like R1, R2, R3, etc.

An example of a dependency diagram for a universal relation is presented in Figure 2.5. This figure contains the attributes associated with an example sale order file.[5] The functional dependencies between the attributes are represented by the arrows in the diagram. The attribute at the flat end of each arrow determines the attribute toward which it is pointing. For example, ***Customer Number*** determines ***Customer Credit Rating***. Alternatively this can be read as ***Customer Credit Rating*** is functionally dependent on ***Customer Number***.

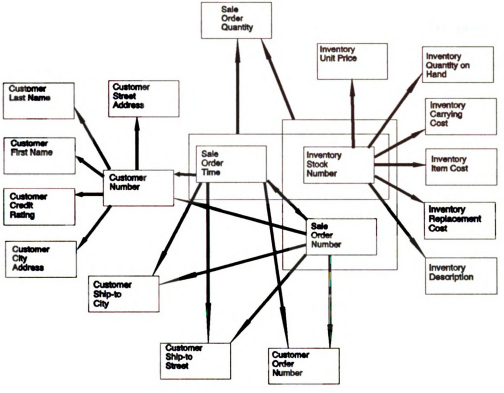


FIGURE 2.5: EXAMPLE UNIVERSAL RELATION WITH FUNCTIONAL DEPENDENCIES FOR SALE ORDER EVENT

A database table representing this universal relation (with the primary key double underlined) would be structured as follows:

R1: [***Sale_order_time***, ***Sale_order_number, Customer_order_number, Customer_ship-to_street, Customer_ship-to_city, Customer_number, Customer_last_name, Customer_first_name, Customer_credit_rating, Customer_street_address, Customer_city_address,*** {***Sale_order_quantity, Inventory_stock_number, Inventory_description, Inventory_unit_price, Inventory_quantity_on_hand, Inventory_carrying_cost, Inventory_item_cost, Inventory_replacement_cost***}]

The attributes from ***Sale_order_quantity*** through ***Inventory_replacement_cost*** are enclosed in {}'s to indicate that there can be more than one inventory item included on a sale order, with different quantities for each item This notation is consistent with the definition conventions of DeMarco (1979, 133) who defines {} as "ITERATIONS OF the component enclosed."

Examination of this database table reveals several maintenance anomalies that would exist. An insertion anomaly would be present, because one would not be able to add a customer unless a sale order had been received from that customer. Thus it would be impossible to add prospective customers. A deletion anomaly would occur in that if the last sale order for a customer[6] was deleted all information about that customer would also be removed. An update anomaly would also exist. For example, if the inventory carrying cost of an item changed, the change would have to be made to multiple instances in the database (i.e. to every instance of sale order that included the associated inventory item).

2.2.1 Rules of Normalization -- First Normal Form:

Several rules of normalization can be applied to this relation in order to convert it into a well-normalized structure which eliminates these anomalies. A relation is in first normal form if, and only if, every attribute in every row can contain only a single value (Date 1986; Loomis 1987). Thus, repeating groups must be removed from the relation. The attributes in the sale order relation which were included inside the {}'s make up a repeating group. The key attribute ***Sale_order_time*** could correspond to multiple values of ***Inventory_stock_number***

since a sale order can be for more than one kind of inventory. To convert this relation to first normal form, the table must be flattened. One way to achieve this is to decompose the universal relation into multiple relations, propagating the original key down to form a composite key for the repeating group (Howe 1983). In the example discussed earlier, the repeating group could be broken out as a separate relation, with the original key ***Sale_order_time*** propagated down to form a composite key with ***Inventory_stock_number*** for R2. The resulting 1NF relations are as follows:

R1: [***Sale_order_time, Sale_order_number, Customer_order_number, Customer_ship-to_street, Customer_ship-to_city, Customer_number, Customer_last_name, Customer_first_name, Customer_credit_rating, Customer_street_address, Customer_city_address***]

R2: [***Sale_order_time, Inventory_stock_number, Sale_order_quantity, Inventory_description, Inventory_unit_price, Inventory_quantity_on_hand, Inventory_carrying_cost, Inventory_item_cost, Inventory_replacement_cost***]

2.2.2 Rules of Normalization -- Second Normal Form:

Second normal form calls for the elimination of non-prime (partial) functional dependencies (that is, if an attribute depends on only part of the primary key of the relation). A non-prime functional dependency can only exist if the primary key (identifier) of a relation is composite. R1 is in 2NF since it has only a single key. R2 has partial functional dependencies. The attributes ***Inventory_description, Inventory_unit_price, Inventory_quantity_on_hand, Inventory_carrying_cost, Inventory_item_cost,*** and ***Inventory_replacement_cost*** all are dependent only on the ***Inventory_stock_number*** portion of the composite key. This

problem can be remedied by replacing the composite key in R2 with ***Inventory_stock_number*** and breaking out an additional relation (R3) with the composite key and any attributes that are dependent on both parts. In this example, only ***Sale_order_quantity*** depends on both ***Sale_order_time*** and on ***Inventory_stock_number***. The revised relations in 2NF are as follows:

R1: [***Sale_order_time, Sale_order_number, Customer_order_number, Customer_ship-to_street, Customer_ship-to_city, Customer_number, Customer_last_name, Customer_first_name, Customer_credit_rating, Customer_street_address, Customer_city_address***]

R2: [***Inventory_stock_number, Inventory_description, Inventory_unit_price, Inventory_quantity_on_hand, Inventory_carrying_cost, Inventory_item_cost, Inventory_replacement_cost***]

R3: [***Sale_order_time, Inventory_stock_number, Sale_order_quantity***]

2.2.3 Rules of Normalization -- Third Normal Form:

Third normal form calls for the elimination of transitive functional dependencies. That is, no non-key attribute can be dependent on another non-key attribute. In the sale order event example, there is a transitive functional dependence. The non-key attributes ***Customer_last_name, Customer_first_name, Customer_credit_rating, Customer_city_address,*** and ***Customer_street_address***, are dependent on the non-key attribute ***Customer_number***[7]. Since ***Customer_number*** is dependent on ***Sale_order_time***, there is a transitive dependency. This can be removed by further decomposing R1. The relations in third normal form are presented as follows:

R1: [***Sale_order_time, Sale_order_number, Customer_order_number, Customer_ship-to_street, Customer_ship-to_city, Customer_number***]

R2: [***Inventory_stock_number, Inventory_description, Inventory_unit_price, Inventory_quantity_on_hand, Inventory_carrying_cost, Inventory_item_cost, Inventory_replacement_cost***]

R3: [***Sale_order_time, Inventory_stock_number, Sale_order_quantity***]

R4: [***Customer_number, Customer_last_name, Customer_first_name, Customer_credit_rating, Customer_street_address, Customer_city_address***]

2.2.4 Rules of Normalization -- Boyce-Codd Normal Form:

A stronger version of 3NF is Boyce-Codd Normal Form (BCNF), which accomplishes the same objectives as 3NF with only one rule. A relation is BCNF if, and only if, every determinant is a candidate key (Date 1986; Loomis 1987). A candidate key is an attribute which has a unique value for each relation instance, but has not been identified as the primary key. Thus it is one which *could* be used as the primary key, but hasn't been declared as such. The attribute ***Sale_order_number*** is a candidate key. Relations R1 through R4 meet the criteria of BCNF.

2.2.5 Rules of Normalization -- Higher Normal Forms:

Higher forms of normalization have been discussed in the database design literature, such as Fourth and Fifth Normal Forms (4NF and 5NF), both of which were created to handle very rare occurrences in database tables. 4NF eliminates all multi-valued dependencies that are not also functional dependencies. A relation is 5NF if it cannot be split into smaller relations and then rejoined

without changing its facts and meaning. In practice, 4NF and 5NF rules are rarely applied; therefore they are not discussed in detail in this project.

The tables created by starting with a universal relation and decomposing it according to normalization theory coincide with the tables created by the use of conceptual modeling with the exception that the decomposition tables would not be assigned semantic names. The decomposition algorithm may be applied to the universal relation by a database designer. Alternatively, the algorithm may be employed by a computer which is furnished with the attributes and functional dependencies which make up the universal relation. Either way, the tables that result from the decomposition approach are often used in practice. The interface to such a database typically will either allow users to enter a table name in order to view it, or it will allow them to scroll through the tables one at a time. The non-abstraction interface used in this experiment represents such a system.

End Notes:

1. The determination of which details are irrelevant or appropriate to a given context is made by the individual who applies the abstraction.

2. For simplicity sake, the REA abstraction hierarchies used in this explanation and in the actual study have been modified from the theoretical norms originally specified in McCarthy (1982, 564). On pages 570-575, he suggests alterations to full REA specification to accommodate current GAAP practice. These include items such as combination of entities and treatment of claims as base objects that have been incorporated here. For further detail on the use of such procedures, see Geerts and McCarthy (1992).

3. Entities are defined as classes of objects or events (either real or conceptual). Relationships represent associations between two or more entities.

4. Such anomalies are problems which result when adding data to, removing data from, or updating existing data in a database. Specific examples of each type of anomaly are discussed later in this section.

5. The database designer would obtain the set of attributes which are to be included in the database, along with their functional dependencies, during the requirements analysis phase of database design. There is no "correct" set of attributes -- the decision of which attributes to include depends on the information needs of the intended users.

6. Such a deletion may occur as a result of a practice companies may use to reduce the storage requirements of their database. For example, a company may decide to keep sale order detail for only six months. After that time, the amounts are rolled into a summary table, and the detail erased.

7. Note that ***Customer_ship-to_street***, ***Customer_ship-to_city***, and ***Customer_order_number*** are related to the entity "Customer", but they can vary from one sale order to another. That is, a customer can order something and have it sent to an address other than his or her own address. Thus, those three attributes are not functionally dependent on ***Customer_number***, but instead are determined by ***Sale_order_time***.

CHAPTER 3 - CONSTRUCT AND HYPOTHESIS DEVELOPMENT

The question researched in this study is whether the use of an abstraction hierarchy in an interface to an events-based accounting system will enhance user performance in preparing financial statements from an events-based accounting database. Use of such a system is compared to that of an events-based accounting system without an abstraction hierarchy in its interface. Comparisons of both user performance and user perceptions are made.

3.1 Financial Statement Preparation and Information Overload

Financial statement preparation was chosen as the task of interest in this study because this task represents a recurring theme in events accounting research. The events-accounting literature began with Sorter's (1969) discussion of how financial statements may be appropriately prepared and presented using an events approach. Financial reporting has continued to be the focus of many events accounting and database accounting papers (e.g. Beaver and Rappaport 1984; Abramson 1986; Cushing 1989). The feasibility of providing an interface to a financial reporting database that is manageable for external users is among the issues discussed in these studies. Revsine's (1970) criticism of data expansion approaches (such as the REA model) is made with regard to the provision of data to external users. He agrees that expanding the range of data provided could help to overcome the many limitations of traditional financial statements without requiring detailed knowledge of user decision models. However, he contends that such an approach ignores the user processing constraint of finite channel capacity

and warns that information overload (implicitly defined as information provided in excess of that which users can manage) would likely result.

3.2 Information Overload and Manageability

Information overload is defined by Casey (1980) as: "a decline in user performance due to the assimilation of additional information" (pp. 36-37). Information overload occurs because of the limited ability of humans to absorb and process information. Simon (1990) claims that human attention, not information, is the scarce resource which puts a tight constraint on how much information can be input. The construct of manageability appears to be very closely related to overload. For instance, Snowball and Brown (1979) note that aggregation or condensation of data is performed "to reduce decision inputs to *manageable* proportions" (p. 527, italics added). Schick, Gordon and Haka (1990) define information overload in terms of time rather than user performance. Still, they note that "the occurrence of information overload would signify problems in organizing (i.e. problems in the *management* of time)" (p. 33, italics added). Like information overload, manageability is indicated by user performance. The American Heritage Dictionary definition of manageability includes the ability "to direct or control the use of; handle, wield, or use", and also the ability "to succeed in doing or accomplishing something, especially with difficulty."

3.3 Proposed Means of Mitigating Information Overload

Ogden (1991) claims that there's no such thing as information overload; the user just has to have ways of organizing the data so that it is manageable. Some

researchers claim that filtration and condensation mechanisms enable users to access a larger base of information while limiting the amount the user must actually assimilate (Ackoff 1967; Morris, Kasper and Adams 1992). An abstraction hierarchy filters and condenses data via the abstraction mechanisms. Filtration occurs in that the user can choose from a high level of the hierarchy (an overall picture in which low level detail is suppressed) specific items for which more detail is required. The low level detail for irrelevant items can be ignored entirely (thus accomplishing filtration). Condensation is incorporated in the abstraction hierarchy since each level is a condensation of the level below it. These filtering and condensing features may provide the organization suggested by Ogden to make large data loads manageable.

Conversely, over-filtration or over-condensation can pose problems in accessing the needed information (Rappaport 1968; Chervany and Dickson 1974). A well-normalized database created through decomposition may not need any further filtration or condensation. As mentioned, the end-result of decomposition via normalization rules resembles commercial entity-oriented databases. If such a system provides adequate filtering and condensing, then addition of an abstraction hierarchy may result in over-filtration or over-condensation, thus leading to reduced manageability. The following section discusses benefits expected from the abstraction hierarchy which suggest that manageability (and performance) should be enhanced rather than reduced.

3.4 Information Overload and User Performance

This study examines information overload from a user performance perspective. Libby and Luft (1993) identify determinants of decision performance using a relation from Einhorn and Hogarth (1980) and Libby (1983):

Performance = f(Ability, Knowledge, Environment, Motivation) (1)

Ability is defined[1] as the capacity to complete information encoding, retrieval, and analysis tasks. Knowledge is described as highly task-specific. Libby and Luft (1993) note that both the content and the organization of knowledge can be changed by learning opportunities and that both can independently affect judgment performance. Environment includes such features as judgment guidance, technological aid, and substantial monetary incentives for good performance. Motivation is determined jointly by environmental features such as monetary incentives and by individual characteristics such as utility functions and abilities.

3.4.1 Ability and Knowledge as Determinants of Performance

The preparation of financial statements from a relational database of a particular company involves several cognitive abilities and types of knowledge. The overall activity can be broken down into several subtasks.[2] One subtask is recall[3] of potential elements of financial statements and of the types of events comprising those elements. The user must also recall the format of the various financial statements. This subtask requires accounting domain knowledge. Another subtask is navigation of the database. This requires data modeling

domain knowledge. Recognition[4] of potential elements present in a particular company in searching its database is a subtask which requires accounting domain knowledge. Extraction of database information needed to compute a financial statement line item is a subtask which requires both accounting domain knowledge and data modeling domain knowledge. The user must have some knowledge of how relational database tables are structured in order to understand how to navigate through them and extract information from them. Accounting domain knowledge is required for understanding which information to extract, and for understanding which arithmetic or other operations must be performed to derive the financial statement elements.

The information extraction process will differ for various financial statement line items. Some will involve retrieval of numeric values from a single table and application of summation. Other elements require retrieval of data from multiple tables, matching on a common factor, and application of various arithmetic operations. For example, to compute Accounts Payable for a given company, a user must determine acquisition amounts for all items acquired on credit (such as Inventory, General and Administrative Services, and Fixed Assets.) Cash Disbursement amounts corresponding to those acquisitions must then be retrieved and subtracted from the acquisition amounts.

3.4.2 Expected Effect of the Abstraction Hierarchy on Ability and Knowledge

The abstraction hierarchy interface is believed to aid users in performing several of the identified subtasks. The views of a company presented at varying

levels of detail are intended to aid the user in understanding the company's operations and in determining which financial statement elements are likely to be present. The hierarchy is also expected to facilitate search for particular tables which are needed to compute a financial statement line item. By identifying which transaction cycle would contain the needed table(s), the search space is reduced by as much as 80%. Examination of the cycle diagrams and table intensions should lead the user directly to the table(s) of interest and assist the user in matching related tables.

The abstraction hierarchy interface is expected to substitute for some of the task-specific domain knowledge. Navigation and extraction without the abstraction hierarchy require extensive use of database domain knowledge. Relational tables are structured so that relationships between entities are identified via matching keys of two or more tables. The user must apply knowledge of data modeling to determine what relationships are present. The abstraction hierarchy user need not recall data modeling knowledge from memory because the relationships are portrayed directly in the interface. The reduction in the amount of necessary recall for users of the abstraction hierarchy interface is expected to enhance performance both in terms of shorter completion time and in terms of greater accuracy.

The abstraction hierarchy also reduces the need for recall of financial statement elements from memory. Since users of the hierarchy interface are presented with adequate opportunities to recognize applicable financial statement elements, there would be little need to recall potential elements from memory.

Without the abstraction hierarchy this recall is essential in order to direct the search.

3.4.3 Environment as a Determinant of Performance

The interfaces in this study can be classified as technological aids and thus fall under the environment variable described by Libby and Luft (1993). The expected effects of the two interfaces (abstraction hierarchy versus normalization) on ability, knowledge, and motivation help to develop predictions about performance. As discussed in the previous subsection, the abstraction hierarchy interface is expected to interact with knowledge and abilities by reducing the amount of recall users must employ. Prior research has shown that recall requires more cognitive effort than does recognition (Libby and Lipe 1992). The normalization interface requires considerable recall. This is predicted to cause declines in performance, both in terms of time and in terms of accuracy. The time is expected to lengthen because of the increased effort. Accuracy is expected to decrease because people may recall items incorrectly or they may not recall some pieces of knowledge that are necessary.

No other environmental variables are present in this study with which the two technological interfaces are expected to interact. Possible environmental variables mentioned by Libby and Luft (1993) as being typical of accounting settings include hierarchical group and accountability relationships (as are often present in audit teams). Neither of these factors is present in this study, and there

appear to be no other environmental characteristics expected to vary between subjects in the two conditions.

3.4.4 Motivation as a Determinant of Performance

No motivation variables are expected to have a direct effect in this study. All subjects will be offered the same incentive to perform well: to achieve a desired grade. Although different students may aspire to different grades, there is no reason to expect the average desired grade to differ between groups. Some interaction may exist between motivation and the knowledge and ability variables. To the extent that the abstraction hierarchy interface reduces the cognitive effort required for recall, subjects in that condition may be willing to exert more effort in other aspects of the task, such as the computations needed for financial statement elements. This possible interaction lends additional support for the prediction that the abstraction hierarchy interface users will exhibit higher task accuracy. There is no clear prediction for task completion time; subjects may or may not increase their cognitive time spent on computations to the same level as their recall requirements were reduced.

3.5 Prior Studies of Database Query Performance

The behavioral accounting literature discussed in Section 3.4 provides valuable insights for studying user performance. Additional insights are gained from the database literature. Most of the studies in this literature focused on one or more query languages without varying the data model. Jih, Bradbard, Snyder and Thompson (1989) note that the data model portrays the logical organization

of a database and is therefore a critical part of the user-system interface. Those studies which did not vary the data model are thus not considered to be relevant for the current project and are not described herein.

Batra, Hoffer and Bostrom (1990) examined differences in user performance with different data models (relational versus extended-entity-relationship). However, their task involved the design of a database rather than the retrieval of information from a database. Their results can not necessarily be extended to a retrieval task such as is used in the current study. The remainder of this section thus describes studies which involved database querying. Several studies in the database literature have examined user performance with different data models and with different database query languages. Brief descriptions of these studies are followed by a discussion of how the current project compares to them.

Lochovsky and Tsichritzis (1977) attempted to examine the effect of the data model on user performance. They compared user performance using three different data models (hierarchical, network and relational), each with a different query language. They found that the relational model with its query language was superior to the others. Unfortunately, the confounding effect of having both different data models and different query languages makes it impossible to isolate the effect attributable to one or the other.

Jih et al. (1989) examined user performance using one query language (Structured Query Language or SQL) with both the entity-relationship data model and the relational model. They were interested in which model was an easier-to-

comprehend database interface for end-users and in whether the result was consistent for both simple and complex queries. They noted that the entity-relationship model had a higher power of abstraction than does the relational model. They claim:

> a data model at a higher level of abstraction not only shields users better from complexity of the system, hence is easier to work with, but also is capable of modeling more domain-specific semantics (p. 260).

Based on this claim, they expected that the users would perform better with the entity-relationship model than with the relational model. However, they found that fewer syntax errors were committed with the relational model. Users of the relational model also took more time. They found no significant difference in the number of semantic errors between the two models. Since SQL was originally created specifically for the relational model, that could account for the difference in syntax errors between the two groups. Jih et al. state that for one model to be declared as better than the other, there must be a difference in the number of semantic errors. A semantic error is a logical error resulting from a misunderstanding of the problem (a data model-related error) or from a mistake in problem-solving logic (possibly an intelligence-related error). They conclude that since no difference was found in semantic errors, neither can be considered easier to comprehend by end-users.

Chan, Wei, and Siau (1991) argue that it is not possible to compare separate data models using the same query language because no query language can suit two data models. They state that if two data models differ, it is because

they contain different constructs and their languages must necessarily differ. They contend that Jih et al.'s use of SQL to generate queries with the entity-relationship model confounded the results and made it impossible to draw any conclusions. Chan et al. constructed a language called KQL, tailored for the entity-relationship model. They compared performance between users of the entity-relationship model using KQL and users of the relational model using SQL. Like Jih et al., Chan et al. expected the entity-relationship model users to exhibit better performance in writing queries because of the higher power of abstraction offered in the entity-relationship model. They indeed found the entity-relationship user group performed significantly better, used significantly less time, and were more confident in their answers.

The current project follows Jih et al. (1989) in that two data models are used, with the same query techniques used for both. Although Chan et al. (1991) claim that KQL and SQL are so similar that their subjects' performance differences can be attributed only to the effect of the data models, we believe that the data model effect in their experiment can not truly be isolated. In this dissertation, the querying involved is not done through a formal query language, so it is not expected to be biased toward one interface or the other. All retrieval from the interfaces is done manually, with paper, pencil and calculator. While this makes the systems somewhat unrealistic, it is necessary to isolate the effect on users of the theoretical construct of interest -- the presence or absence of an abstraction hierarchy in an interface.

The current project differs from Jih et al. (1989) in that the task is much more complicated. Their data models consisted of only three entities and one relationship. Even their "complex" queries were quite simple. All of their subjects in both groups were nearly 100% accurate in their query writing, therefore it is possible that the simplicity of their task contributed to their finding of no difference. The use of a more complex task such as creating financial statements from a database of accounting information should produce a stronger treatment effect.

One important distinction of the current study from both Jih et al. (1989) and Chan et al. (1991) is that the data models used in those projects were single abstractions. There was no abstraction hierarchy present in any of their models. Thus while they argue that the entity-relationship model has more abstraction power, that power is not utilized in their systems. There is no provision of views with differing levels of detail to the users. Chan et al. are not really justified in making their claim that

> The results supported the basic hypothesis that users can perform better at higher abstraction levels. The reason is that the higher levels have semantics closer to the user's world (p. 34).

All they can legitimately claim is that users with the combination of KQL and the entity-relationship model exhibited better performance at their task. They cannot further isolate the reason for the increased accuracy and time. The current project thus goes beyond the testing of two data models to isolate the effect of an abstraction hierarchy.

3.6 Conceptual Framework for Hypotheses

The conceptual framework used in this project fits that used by Jih et al. (1989). Their framework is illustrated as Figure 3.1.

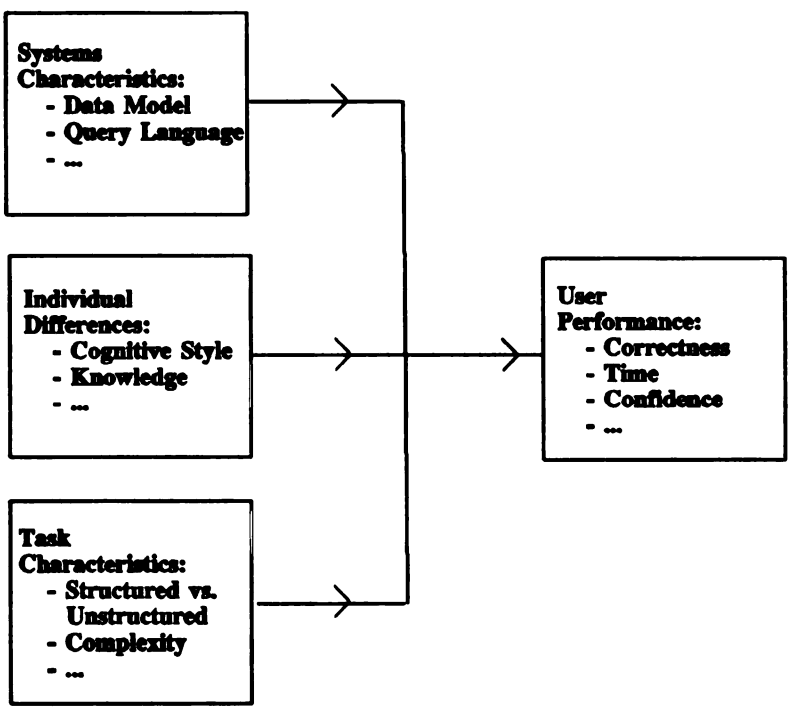


FIGURE 3.1: CONCEPTUAL FRAMEWORK OF JIH et al. (1989)

In this framework there are three types of variables that can influence user performance: system characteristics such as data model or data query language, individual differences such as cognitive style or knowledge, and task characteristics such as complexity or degree of structure. The measures of user performance in this framework include correctness, time, and confidence. Jih et al.'s framework is similar to Einhorn and Hogarth's (1980) determinants of user performance

described in Section 3.4. System characteristics fit into the environment category. Individual differences encompass ability, knowledge, and motivation. The current study examines the effect of the system characteristic (type of interface) on user performance while controlling for individual differences between subjects in accounting and data modeling domain knowledge.

3.7 User Performance Measures and Hypothesis

Prior research studies of information overload employing user performance measures have typically included two dependent variables: decision accuracy and decision time (e.g. Davidson and Trueblood 1961; Chervany and Dickson 1974, Benbasat and Dexter 1979; Casey 1980, Otley and Dias 1982). Studies in the database literature on database query performance have also used accuracy and time as dependent variables (Jih et al. 1989, Chan et al. 1991). Therefore task accuracy and task completion time are included as user performance measures in the current study. The discussion in section 3.4 illustrates how the abstraction hierarchy is expected to improve performance in terms of both accuracy and time. This expectation is supported by the results of Chan et al. (1991). The hypothesis regarding user performance in this study (stated in both null and alternative forms) is:

> H1: There will be no significant difference in financial statement preparation performance (measured as a combination of accuracy and speed) between users of an events-based accounting database with an abstraction hierarchy interface and users of the same database with a non-abstraction interface, controlling for domain knowledge in accounting and data modeling.

H1A: Financial statement preparation performance (measured as a combination of accuracy and speed) of users of an events-based accounting database with an abstraction hierarchy interface will be significantly better than that of users of the same database with a non-abstraction interface, controlling for domain knowledge in accounting and data modeling.

3.8 User Perception Measures and Hypothesis

Performance is not the only indicator of a manageable information system. User perceptions may also provide evidence as to whether a system is manageable. A person could conceivably perform well in a reasonable length of time and yet still feel overwhelmed. If a user feels overloaded, the user may reject the information system in spite of adequate performance. This is undesirable because acceptance of users has been identified as one of the critical determinants of system success (Shneiderman 1980). Prior research studies which examined information overload employing user perception measures have typically examined overall user satisfaction (e.g. Chervany and Dickson 1974; Casey 1980; Otley and Dias 1982). Prior studies in database query performance have typically included ease-of-use or user confidence as dependent variables.

In this project the user perception measured is perceived manageability[5]. This construct is believed to be quite similar to ease-of-use. Perceived manageability is expected to identify whether a subject felt overloaded, regardless of performance. Since the abstraction hierarchy interface is expected to ease cognitive efforts, it is expected to be perceived as more manageable than the

non-abstraction interface. The hypothesis regarding user perceptions in this study (in both null and alternative forms) is:

> H2: There will be no significant difference in the perceived manageability of financial statement preparation between users of an events-based accounting database with an abstraction hierarchy interface and users of the same database with a non-abstraction interface, controlling for domain knowledge in accounting and data modeling.
>
> H2A: Perceived manageability of financial statement preparation by users of an events-based accounting database with an abstraction hierarchy interface will be significantly greater than that of users of the same database with a non-abstraction interface, controlling for domain knowledge in accounting and data modeling.

Tests of the two hypotheses are described in the next chapter.

End Notes:

1. The definitions given here were adapted from Einhorn and Hogarth (1980) by Libby (1983) and by Libby and Luft (1993) to make them consistent with typical accounting settings.

2. Each of these subtasks could be broken down into finer detail. The level of detail presented seems adequate for understanding the overall benefits expected from the abstraction hierarchy.

3. Recall is generally defined as a retrieval process that may involve the generation of alternatives (Libby and Lipe 1992).

4. Recognition is defined by Libby and Lipe (1992) as the matching of a presented cue to prior knowledge. Alternatively the matching may be to a trace of some type.

5. There are multiple aspects of user satisfaction of which perceived manageability is only a part. Other constructs which are considered part of user satisfaction include perceived usefulness, the degree to which subjects' use is voluntary, compatibility of system with work, and perceived prestige (Moore and Benbasat 1991). While these constructs are interesting, they do not appear to be indicators of information overload.

CHAPTER 4: METHODOLOGY

4.1 Research Framework

In order to test the hypotheses put forth in this dissertation, two major projects were completed. March and Smith (1994) describe the difficulties inherent in information technology (IT) research and suggest a framework for use in planning and in evaluating IT research. They begin their framework development with a discussion of the differences and interactions between design science and natural science. March and Smith note that natural science typically consists of two stages -- theorize and justify. In natural science theories arise from naturally occurring phenomena. They can be observed and tested in controlled settings, resulting in justifcation. In design science the constructs, models, and methods are created or "built." These phenomena are artifactual rather than naturally occurring. March and Smith propose that design science consists of two stages -- build and evaluate -- which parallel the two stages of natural science. They point out that in the computer science literature it is widely recognized that constructs, models, and methods that work "on paper" do not necessarily work in real-world contexts. Thus, instantiations (physical implementations of the constructs, models, or methods) provide the real proof. Once adequate evaluations of the instantiations (and the underlying constructs, models, and methods) have been made to determine how well they work, theories may be developed and justified as to *why* they work. According to March and Smith (1994, 6)

> IT research builds and evaluates constructs, models, methods, and instantiations. It also theorizes about these artifacts and attempts to justify these theories. Building and evaluating IT artifacts have design science intent. Theorizing and justifying have natural science intent.

Based on this belief, March and Smith propose a 4 x 4 matrix of research possibilities for use in planning and evaluating IT research. This matrix is presented as Figure 4.1.

	BUILD	EVALUATE	THEORIZE	JUSTIFY
CONSTRUCTS				
MODELS				
METHODS				
INSTANTIATIONS				

FIGURE 4.1: IT RESEARCH FRAMEWORK per MARCH AND SMITH (1994)

Research in the *Build* column seeks to show that a construct, model, method or instantiation works, whereas work in the *Evaluate* activity attempts to determine how well something works. Research in the *Theorize* category tries to show how and why something works, and research in the *Justify* column tests those theories. March and Smith note that within their framework different cells have different objectives and different research methods are appropriate in different cells. Evaluation of research depends on the cell(s) in which the research lies.

4.1.1 Project 1: Building the Interfaces

The first major project in this dissertation was the creation of instantiations that operationalized the constructs, models and methods included in the REA model with abstraction hierarchies and those in normalization theory. Thus, this section of the dissertation fits the *Build-Instantiations* cell in March and Smith's matrix in Figure 4.1. According to March and Smith (1994) it is sometimes necessary for an instantiation to precede the complete articulation of its underlying constructs, models and methods, so that it can be studied and used in order to formalize the constructs, models, and methods on which it is based. They suggest criteria by which *Build* research should be evaluated, as follows:

> Building the first of virtually any set of constructs, model, method, or instantiation is deemed to be research, provided the artifact has utility for an important task. The research contribution lies in the novelty of the artifact and in the persuasiveness of the claims that it is effective. Actual performance is not required at this stage (1994, 13).

March and Smith go on to say that "'first' is usually interpreted to mean, 'never done within the discipline.'" (1994, 13) The abstraction hierarchy interface built in this dissertation is the first instantiation of the REA model including the abstraction hierarchies thus providing research contribution per March and Smith (1994).

4.1.2 Project 2: Evaluating the Interfaces

This dissertation continues beyond the *build* activity. The instantiations were also evaluated using a laboratory experiment. March and Smith (1994, 14) note that "research in the evaluate activity develops metrics and compares the

performance of constructs, models, methods, and instantiations for specific tasks." The remainder of this chapter describes the experimental procedures, including detailed illustrations of the two interfaces which were used.

4.2 Interface Software

An events-based accounting database for a sample company modeled with RE accounting is used in both experiments. The sample company used is the Wilson Company (McCarthy 1979). The interfaces were built using an object-oriented, message-passing software package called VisualWorks (Version 1.0, Parc Place Systems 1992). VisualWorks is a graphical user interface builder that uses Smalltalk as its engine. This environment allows the creation of user interfaces with features such as windows, buttons, and graphics that are purported to be user-friendly (Shneiderman 1992).

4.3 Experimental Treatments

Both experiments consisted of two treatments: abstraction and non-abstraction.

4.3.1 Abstraction condition

Subjects in the abstraction condition used a computerized abstraction hierarchy interface to Wilson Company's database. Figures 4.2 through 4.9 represent example screens in the computerized abstraction hierarchy interface. Figure 4.2 shows a representation of the initial screen. This gives a natural language description of Wilson Company.

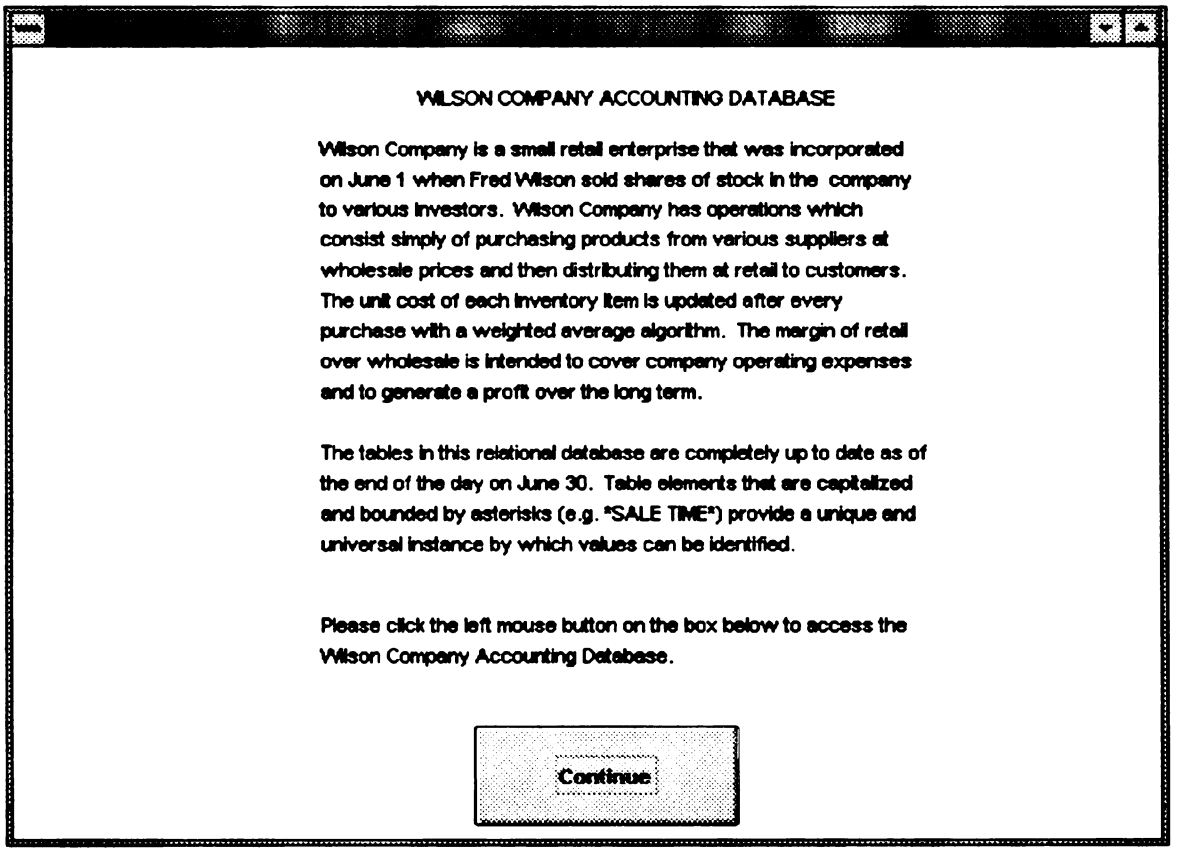
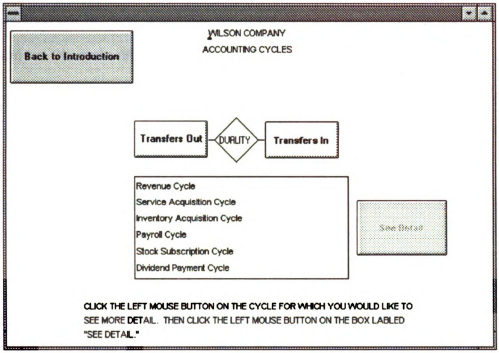


FIGURE 4.2: ABSTRACTION INTERFACE INITIAL SCREEN

When the user clicks the left mouse button on the *Continue* button, Figure 4.3a appears.

(a) Cycle List Screen as it appears before the user highlights a cycle:



(b) Cycle List Screen as it appears after the user highlights *Revenue Cycle*:

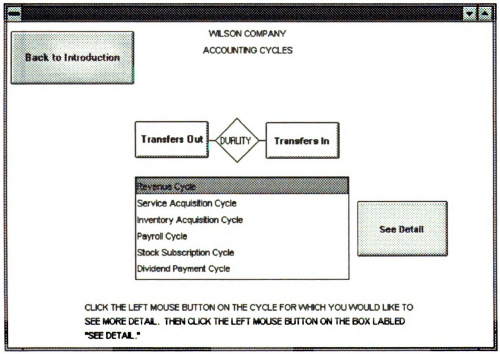


FIGURE 4.3: ABSTRACTION INTERFACE CYCLE LIST SCREEN

The six cycles which encompass Wilson's financial activities are portrayed. On this screen the user highlights the cycle of interest by clicking the left mouse button on the cycle name. This makes the *See Detail* box active, allowing the user to click the left mouse button on the box, as in Figure 4.3b. An overall REA template of that cycle appears, as portrayed in Figure 4.4. This gives the user an overall picture of the transfers in and out (events) in the cycle, and of the resources, outside agents, and authorization units that are involved.

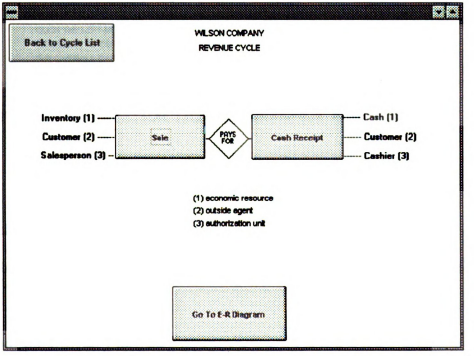


FIGURE 4.4: ABSTRACTION INTERFACE CYCLE TEMPLATE SCREEN

Users then click the left mouse button on the box labeled *Go To E-R Diagram* which reveals a detailed E-R diagram, as illustrated in Figure 4.5.

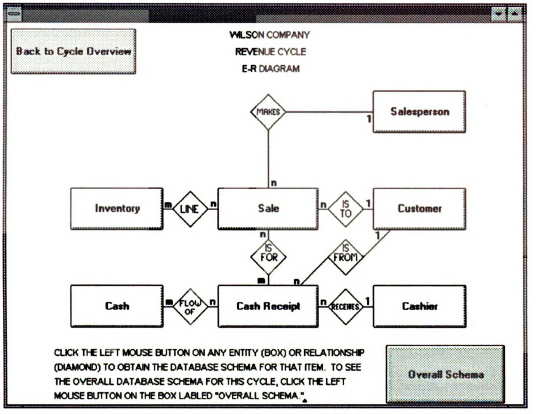


FIGURE 4.5: ABSTRACTION INTERFACE E-R DIAGRAM SCREEN

The user can utilize the E-R diagram to better understand what entities and relationships exist in the cycle. From this screen, the user can choose to see the table intension(s)[1] for a particular entity or relationship. For example, the user may choose to click the left mouse button on the relationship labeled *LINE* between the *Inventory* and *Sale* entities, bringing up the screen portrayed in Figure 4.6.

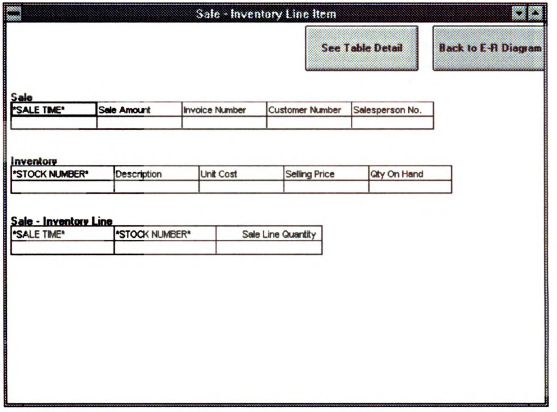


FIGURE 4.6: ABSTRACTION INTERFACE RELATIONSHIP SCHEMA SCREEN

The user can look at the table intension to see if the attributes of interest are included. If not, the user may return to the E-R Diagram by clicking the left mouse button on the box labeled *Back to E-R Diagram*. If the attributes of interest are included, the user can click the left mouse button on the box labeled *See Table Detail* to bring up the screen shown in Figure 4.7.

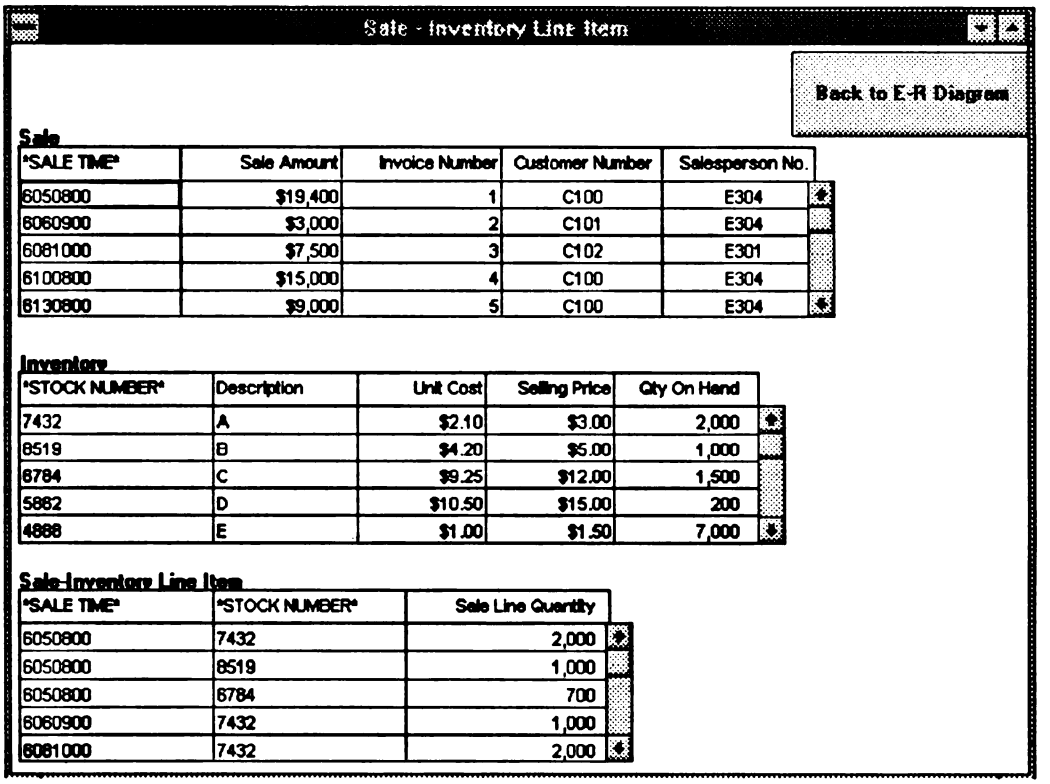


FIGURE 4.7: ABSTRACTION INTERFACE RELATIONSHIP DETAIL SCREEN

Each table contains a scroll bar, so the user may scroll to see every instance in a table. From the table detail screen, the user returns directly to the E-R Diagram, rather than having to view the table schema again. In pilot testing, the system required users to go back up the hierarchy through exactly the same path as they had come down. Several subjects complained that this was unnecessary, time-consuming, and frustrating.[2]

From the E-R Diagram screen, the user may opt to view the overall database intension for the cycle, as in Figure 4.8, rather than focusing on a particular entity or relationship. This is accomplished by clicking the left mouse button on the box labeled *Overall Schema*.

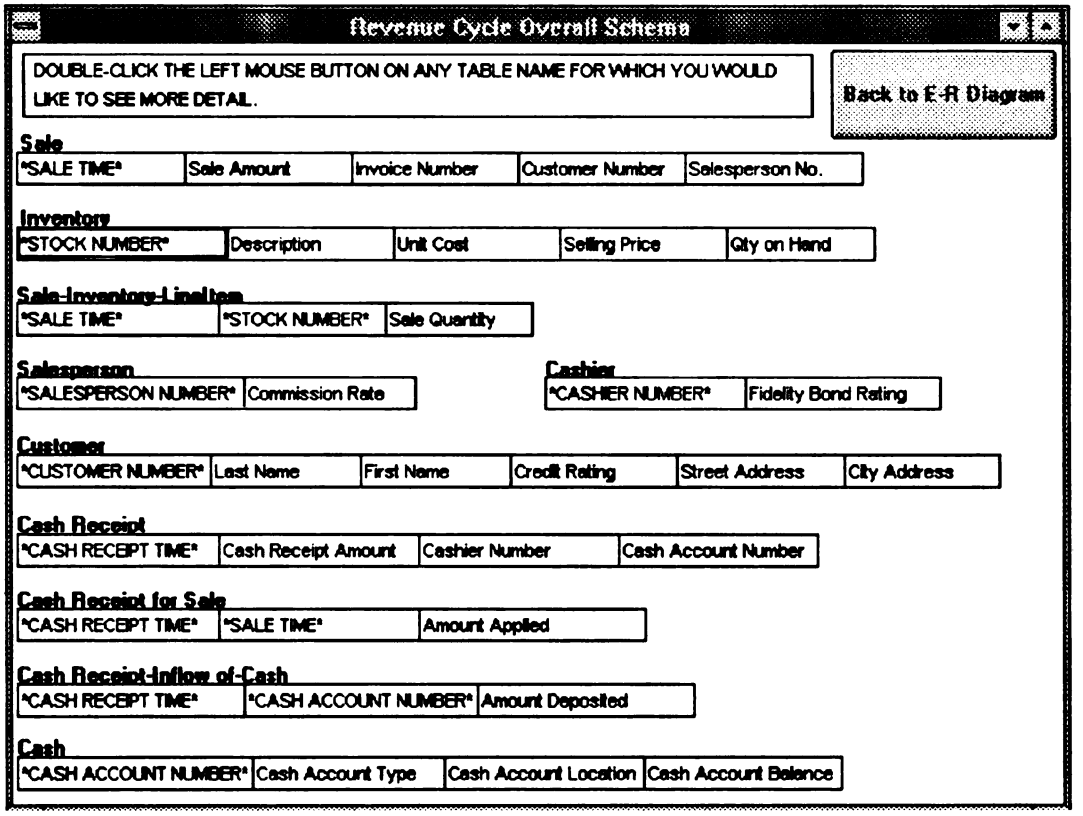


FIGURE 4.8: ABSTRACTION INTERFACE OVERALL SCHEMA SCREEN

From the overall schema, the user may select a table of interest by double-clicking the left mouse button on the table name. This brings up a screen such as that shown in Figure 4.9.

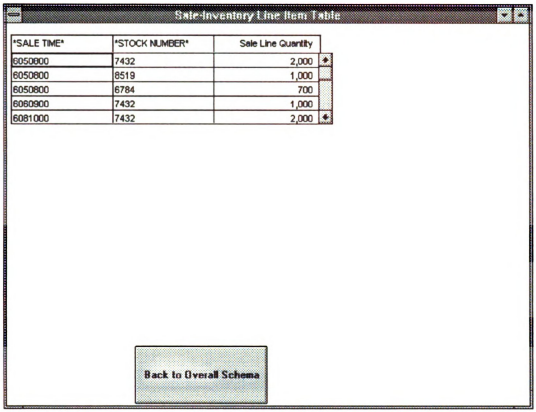


FIGURE 4.9: ABSTRACTION INTERFACE SCHEMA DETAIL SCREEN

Any table with more than five instances has a scroll bar to allow users to see every instance. All tables in both interfaces were limited to having at most five instances visible at a time. The purpose for this was twofold -- (1) to ensure that any differences in performance were due to the abstraction hierarchy as opposed to more table detail being visible at one time, and (2) to make the task as realistic as possible. The sample company's database was fairly small with a limited number of transactions. Although it would be possible to view entire tables at one time for Wilson Company, this would be impossible for most real-world companies.

4.3.2 Non-abstraction condition

Subjects in the non-abstraction condition used a computerized non-abstraction interface to Wilson's database. Consistent with some commercial database interfaces, this interface allowed users to simply scroll through the database tables to obtain the needed information. This system provided access to all of the same data as the abstraction system (i.e. the detailed Boyce-Codd Normal Form relational tables). The difference was in the exclusion of the abstraction hierarchy from the interface. The initial screen for the non-abstraction system is identical to that of the abstraction system, as illustrated in Figure 4.10.

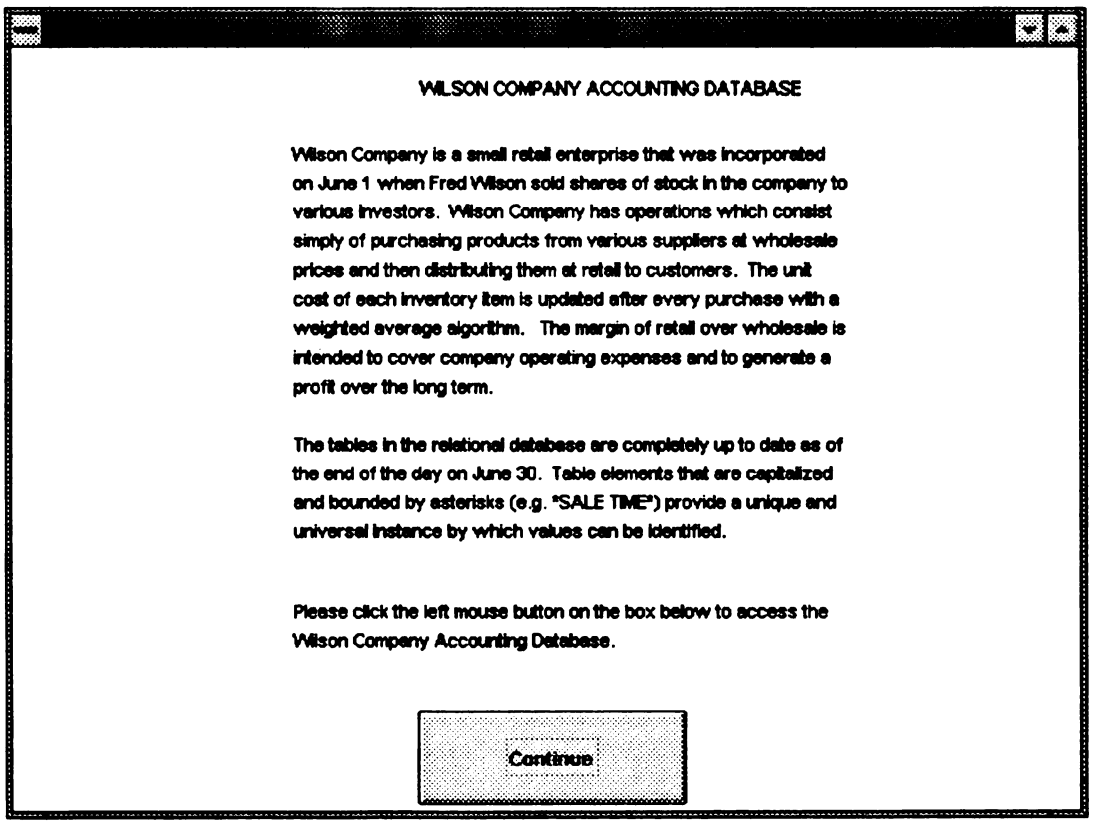


FIGURE 4.10: NON-ABSTRACTION INTERFACE INITIAL SCREEN

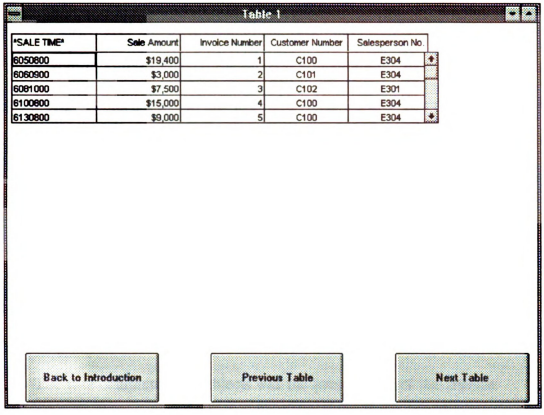


FIGURE 4.11: NON-ABSTRACTION INTERFACE TABLE 1 SCREEN

Figure 4.11 portrays the screen for Table 1 in the non-abstraction interface. Subsequent table screens have a similar appearance (identical except for the table content). Users may scroll both forward and backward through the tables, by clicking the left mouse button on either the *Next Table* box or the *Previous Table* box. Alternatively, the user may return to the initial screen at any time by clicking the left mouse button on the box labeled *Back to Introduction*. Consistent with the decomposition approach to database design, the table names in the non-abstraction interface are non-semantic. Thus they are labeled Table 1, Table 2, Table 3, etc.

4.3.3 Querying the Database

No automated querying capabilities were built into either interface. Subjects were required to locate the table(s) containing the data they needed, and to make any necessary calculations with only the assistance of a calculator. This reduces the realism of the systems (i.e. any commercial database interface would have automated query capabilities). However, the inclusion of automated querying would likely have introduced other confounds into the task and obscured the effects of the abstraction hierarchy on the dependent variables.

4.4 Experimental Environment

4.4.1 Task

The task completed by subjects was the preparation of the Income Statement, Statement of Changes in Retained Earnings, and Balance Sheet for Wilson Company's first month of operations. Preparation of simple[3] retail company financial statements is a task with which subjects were expected to be very familiar and is one that does not require subspecialty knowledge (general accounting domain knowledge should be sufficient). If the task required subspecialty knowledge, per Bonner and Lewis (1990), it would be advisable to use subjects experienced in that subspecialty.

The experiment was conducted in a computer laboratory in six separate sessions. Subjects were given a set of written instructions which were read aloud by the experimenter. These instructions are included as Appendix 1. The instruction period lasted 10 minutes for each session. All subjects claimed they

had used a mouse and felt comfortable using a mouse to point and click. No extensive mouse training was given, since only pointing and clicking were required. There was no need for window movement or resizing (which have been reported to cause problems with lack of training in other studies). Pilot testing indicated that, with only one exception, students felt comfortable using the mouse for pointing and clicking. During the instruction period subjects were told that they were to write their financial statements on paper that was provided to them. Along with the financial statement line items and numbers, subjects were instructed to write a brief explanation as to how they computed each number (e.g. which attribute(s) of which table(s) they used and what mathematical operation(s) they performed). An example explanation was included in the instructions.

Subjects had been instructed to bring their calculators with them, and the experimenter had extras available so that every subject had a calculator to use. Subjects were not allowed to use any notes or accounting textbooks; rather, they were expected to rely on their own accounting and data modeling knowledge in combination with their interface to the database. This represented a change from pilot testing, in which subjects were given a chapter from an introductory accounting textbook which included example financial statements. This change was made because it was believed that the textbook chapter in effect provided an abstraction mechanism for the subjects. This belief was formulated through experimenter observation. For example, although Wilson Company has no fixed assets (it rents them), several subjects included "Fixed Assets - $0" as a line item on their balance sheets. The example financial statements in the textbook chapter

were for a company which had fixed assets. The students thus seemed to be using the example financial statements as a template or abstraction to which they tried to match Wilson Company. Another reason the use of example financial statements is a potential problem is that the abstraction interface is expected to aid users in recall of accounting domain knowledge. Any such effect would be obscured because the example financial statements would make recall largely unnecessary (replacing it with recognition).

Subjects were given scrap paper to use if needed, but were instructed to use it as little as possible. Minimal use was expected to be necessary. At least one financial statement line item (Cost of Goods Sold) required use of scrap paper (or use of the memory function on a calculator, which some people feel uncomfortable using). Therefore, the scrap paper could not be taken away altogether. However, in pilot testing, many subjects used the scrap paper not only to make computations, but also to create abstractions for themselves. For example, several subjects in the non-abstraction condition wrote notes on their paper such as "Table 1 = Sales, Table 2 = Inventory," etc. Others went so far as to write out the attributes for each table (i.e. the table intension).

To curb this creation of abstraction in the experiment, subjects in both conditions were only allowed to keep a given piece of scrap paper for 10 minutes. The task time period for the experiment was 60 minutes, thus there were six 10 minute cycles. Subjects were given six different colored pieces of scrap paper. Subjects worked uninterrupted for the first 9 minutes of each cycle. After 9 minutes, a bell was rung as a one-minute warning. At that point, students could

complete whatever calculation they were working on. If they were starting a new calculation, they could turn their scrap paper over and start with the next color. After the one-minute transition period, the designated color scrap paper was collected. Subjects did not appear to be bothered by the interruptions, although no conclusion can be drawn as to their effect.

Figures 4.12 - 4.14 illustrate the financial statements for Wilson Company. For each line item, an explanation is provided as to which database tables contained the necessary information and what computations were necessary.

Income Statement:

Sales	$125,500	Add up the "Sale Amount" column of the Sale table
COGS	(93,250)	Get the "Line Qty" column of the Sale-Inv-Line table Get the "Unit Cost" column of the Inventory table Match the two columns by "Agleclap Number" For each item number, multiply the qty by the cost Add all of them together.
Gross Margin	$ 32,250	
G&A Expense	(10,495)	Add up the "Amount" column of the G&A Service Acquisition table
Wages Expense	(7,216)	Add up the "Gross Pay" column of the Personnel Service Acquisition table
Net Income	$ 14,539	

FIGURE 4.12: WILSON COMPANY INCOME STATEMENT

Statement of Changes in Retained Earnings:

Beginning Balance	0	from Introduction screen information
+ Net Income	14,539	
- Dividends	(6,000)	Add up "Amount" column of Dividend Declaration table
= Ending Balance	$ 8,539	

FIGURE 4.13: WILSON COMPANY STATEMENT OF CHANGES IN RETAINED EARNINGS

Balance Sheet:

Cash	$ 60,095	Add up "Balance" column of Cash table
Accts Receivable	76,850	Add up "Sale Amount" column of Sale table Add up "Amount Applied" column of Cash Receipt for Sales table Subtract the latter from the former to get A/R
Inventory	31,375	Get "Unit Cost" column of Inventory table Get "QOH" column of Inventory table Multiply unit cost by qoh for each item Add all item subtotals to get total Inventory
Total Assets	$ 168,320	
Accounts Payable for Purchases	4,625	Add up "Amount" column from Purchase table Add up "Amount Applied" column from Cash Disbursements for Purchases table Subtract the latter from the former
for G&A Services	4,500	Add up "Amount" column from G&A Service Acquisition table Add up "Amount" column from Cash Disbursements for G&A Services table Subtract the latter from the former
Wages Payable	656	Add up "Amount" column from Personnel Service Acquisition table Add up "Amount" column from Cash Disbursements for Personnel Services table Subtract the latter from the former
Capital Stock	150,000	Add up "Amount" column of Stock Subscriptions table
Retained Earnings	8,539	From Stmt of Changes in Retained Earnings
Total Liab & Eq	$ 168,320	

FIGURE 4.14: WILSON COMPANY BALANCE SHEET

4.4.2 Variables

Three independent variables are measured in this study. The first independent variable is the type of interface to which subjects were assigned. There are two levels of this variable -- abstraction and non-abstraction -- as described in Section 4.3. This independent variable fits into the "System Characteristics" box of the conceptual framework of Jih et al. (1989). The second and third independent variables are the subjects' accounting domain knowledge and data modeling domain knowledge. These variables fit into the "Individual Differences" box of the conceptual framework of Jih et al. (1989) and were expected to have an influence on user performance based on Einhorn and Hogarth's (1980) equation in Section 3.4.

The subjects' grade in the first intermediate accounting course was used as a surrogate for accounting domain knowledge. This was determined to be the most adequate surrogate for two reasons. First, several subjects had taken their two introductory accounting courses at other schools (typically community colleges). This makes it difficult to compare these grades across subjects. Second, several subjects had not yet taken any accounting courses higher than the first intermediate accounting course. To include higher accounting course grades for some subjects and not for others would diminish the comparability. Comparability of the first intermediate accounting course grade is believed to be quite strong because all sections are taught by one of two instructors who work together closely to ensure consistency of material and presentation. Data modeling domain knowledge was measured as the numerical score each subject earned on a data

modeling question on the first exam administered in the accounting information systems course for which they were concurrently enrolled.

The dependent variables measured in this study include task accuracy, task completion time, and perceived manageability. These all fit into the "User Performance" box of Jih et al.'s framework. Because there is a correct answer for the individual line items and amounts which should be included in Wilson's financial statements, it was possible to derive an accuracy measure. Subjects received points for each necessary account name (e.g. Cash, Accounts Receivable), points for each correct numerical value, and points for having the correct explanation of how to derive the figure. Subtotals and totals were not awarded points in order to avoid double-counting errors as much as possible.

Accuracy scores consisted of total points earned. More points were awarded for those line items that required more mathematical operations. For example, on the Income Statement a correct answer for Sales was worth four points -- one for including it as a line item, two for getting the right number (since it involved two operations: [1] isolating the "Sale Amount" column of the Sale table and [2] adding up the numbers in the column), and one for giving the correct explanation. Cost of Goods Sold was worth a total of fifteen points -- one for including it as a line item, thirteen for getting the right number (since it involved thirteen operations as illustrated in Figure 4.12) and one for providing the correct explanation. This grading scheme was followed with one exception -- if a subject gave the correct explanation for deriving a number, and the scrap paper revealed that the error was clerical in nature, the subject was only penalized

one point. This situation only occurred three times. The grading scheme was determined before the experiment was administered and was based on relational algebra operators typically used in database query languages.

Task completion time was measured as the number of minutes a subject worked on the task. Subjects were allowed up to 60 minutes to complete the task. Completion time was recorded before the questionnaire was administered in order to isolate time spent on the task itself. The 60 minute time period turned out to be inadequate for most subjects to complete the entire task. This time period was chosen based on pilot testing. In pilot testing, subjects did not have incentive to work quickly. They were allowed up to 120 minutes to complete both the task and the questionnaire. As in an examination situation, many subjects probably kept their financial statements for the entire period, even though they could not improve their performance by doing so. The fastest completion time was 65 minutes, with 100% accuracy (again, this included time to fill out the questionnaire).

In conversations after the pilot test sessions, several subjects said they had computed figures for every line item within approximately 40-45 minutes, and spent the rest of their time "spinning their wheels" because their financial statements did not balance. The resulting situation was like starting a race between a Corvette and a bicycle and allowing them both an hour to go 10 miles. Arriving at the finish line after an hour, it is seen that both have indeed crossed the finish line; however, the discrimination between the two performances is lost. It was expected that the 60 minute time period would encourage subjects to work

as quickly as possible, and provide a clearer measure of how long it took subjects to develop a set of financial statements (correct or not). To further discourage "wheel spinning," subjects were thus told their grades would be based on both accuracy and completion time, with statements that are 90% accurate completed in 40 minutes being worth more than 100% accurate statements completed in 60 minutes.

Perceived manageability was measured using a seven point Likert scale questionnaire. The five questions used to measure this variable are reproduced in Appendix 2. These five questions are identical to those used in Batra, Hoffer, and Bostrom (1990), except their words "data modeling technique" are replaced by "database interface". Batra et al. (1990) had adapted their instrument from Davis (1989). The reported reliability for their instrument was .83. The Batra et al. (1990) and Davis (1989) instruments purport to measure "perceived ease of use," which Davis defines as the degree to which an individual believes that using a particular system would be free of physical and mental effort. That definition is consistent with the definition of perceived manageability used in this study.

Several concomitant and prior influence variables were identified and measured via subject's completion of background questions which were mostly closed-ended (see Appendix 2). These data were run as covariates to ensure that they did not account for variation in the dependent variable.

4.4.3 Subjects

Subjects were students enrolled in an intermediate level accounting information systems course. Students at this level were expected to have the necessary domain knowledge, in both accounting and in data modeling. All subjects had either completed or were concurrently taking the first intermediate level financial accounting course. All had been taught and tested on data modeling techniques and on REA modeling of accounting phenomena in the systems course prior to administration of the experiment. Subjects received class credit for participation, with varying grades based on performance. To ensure fairness, the grading was done separately for each condition. For example, the top performers in each condition received top grades, the lowest performers in each condition received low grades, and similarly in-between.

All subjects completed the task as an in-class computer assignment. Students enrolled in the course were offered a choice between completing this task as 5% of their grade or assigning an extra 5% to their second examination. All but one student chose to complete this task. Subjects were randomly assigned to the two experimental conditions. Sixty-two subjects participated in the experiment; however, fifteen were dropped from the main data analysis. Three were dropped because they arrived at their sessions late (just as the experimenter had finished reading the instructions aloud). Two were dropped because it was discovered that they were using their accounting textbooks as an aid (in spite of being told to keep all personal books and materials other than their calculators put away). Ten were eliminated because they did not grant permission for the

experimenters to obtain their intermediate accounting grade, which was the measure of accounting domain knowledge.[4] Thus, the final sample used for the experiment consisted of forty-seven subjects (20 Abstraction, 27 Non-abstraction).

Endnotes:

1. The terms *intension* and *extension* refer to a database's structure and its detail. Table headings make up the database intension. Instances or rows in the database tables constitute the extension.

2. Other minor adjustments were made to the database tables in both systems due to feedback of pilot subjects. For example, in using the original database, students were confused between "Amount" fields and "Quantity" fields. Since amounts were monetary and quantities were numeric, dollar signs were inserted into all of the amount fields. Students were also confused by the terms "Replacement Cost," "Carrying Cost," and "Volume" in the Inventory table, and by the terms "Interest Cost," and "Withdraw Cost" in the Cash table. Since these fields were not required for calculation of any financial statement items, they were deleted from the database.

3. Classification of these statements as simple is based on the fact that they only involve line items which are common to most companies and which are typically taught to students in introductory accounting courses.

4. Data were also run with the full sample of 57 (27 abstraction, 30 non-abstraction) substituting a grosser measure of accounting domain knowledge. This measure is a category variable indicating the range in which each subject's self-reported average grade point in accounting classes falls, with 1 for < 2.5, 2 for 2.5-2.99, 3 for 3.0-3.49, and 4 for 3.5-4.0. Results of these tests did not differ substantially from the tests of the 47 subjects.

CHAPTER 5 - STATISTICAL TESTS AND EXPERIMENTAL RESULTS

This chapter presents the statistical tests used to test the experimental hypotheses and provides a discussion of the results of each of the tests.

5.1 Analysis of User Performance Hypothesis

Hypothesis One in null form suggests no difference in task performance (in terms of accuracy and of completion time) between groups of subjects using the abstraction interface versus the non-abstraction interface, controlling for accounting and data modeling domain knowledge. This hypothesis is tested using multivariate analysis of variance (MANOVA). The test, which tests the effect of the interface on the combination of task accuracy and task completion time, is necessary because task accuracy and task completion time are believed to be inextricably linked. Subjects were told their class grade for the project would depend both on their accuracy and on their speed, thus their performance is expected to reflect an accuracy-speed tradeoff.

Hypothesis One calls for the measurement of the effect of the interface (abstraction versus non-abstraction) to be done while holding the accounting and data modeling domain knowledge levels constant. Accounting domain knowledge and data modeling domain knowledge measures were thus included as covariates in the MANOVA model. The model included *Score* and *Time* as a combined dependent variable, *Group* as the factor, and *IntAcc* and *DataMod* as covariates. The variables in the model are defined as follows. *Score* represents the task accuracy score earned by a subject. *Time* is measured as the weighted number of

financial statement line items a subject attempted to complete[1], divided by the number of minutes the subject took to complete the project. *Group* represents the interface to which the subject was assigned (0 = non-abstraction, 1 = abstraction). *IntAcc* is the grade each subject earned in the first intermediate accounting course (on a scale from 0.0 to 4.0). *DataMod* is the numerical score each subject earned on a data modeling question on the first exam administered in the accounting information systems course for which they were concurrently enrolled.

The MANOVA model was run using SPSSX. Descriptive statistics for the model are presented in Table 5.1. The direction of effect results indicate that overall, subjects in the non-abstraction group demonstrated higher accuracy and used less time to complete the task. They had a higher grade point average in intermediate accounting but exhibited a lower performance on the data modeling test problem than did subjects in the abstraction group. However, no conclusion can be drawn from these trends without performing significance tests. Results of significance tests indicate no statistical differences, as discussed later in this chapter.

Table 5.1: Descriptive Statistics for User Performance Model

SCORE	Mean	Std Deviation	N
Non-abstraction	47.7	16.5	27
Abstraction	39.3	11.7	20
Total	44.1	15.1	47
TIME	**Mean**	**Std Deviation**	**N**
Non-abstraction	1.09	.22	27
Abstraction	.97	.21	20
Total	1.04	.22	47
INTACC	**Mean**	**Std Deviation**	**N**
Non-abstraction	3.09	.76	27
Abstraction	2.85	.86	20
Total	2.99	.80	47
DATAMOD	**Mean**	**Std Deviation**	**N**
Non-abstraction	91.6	26.9	27
Abstraction	99.4	22.7	20
Total	94.9	25.3	47

Results of testing for homogeneity of variance are presented in Table 5.2. The Cochran's C test and Bartlett-Box F test reveal no significant violations of the assumption of homogeneous variances.

Table 5.2: Tests of Homogeneity of Variance for User Performance Model

	Value	Significance
SCORE		
Cochran's C	.67	P=.101 (approx)
Bartlett-Box F	2.47	P=.116
TIME		
Cochran's C	.51	P=.936 (approx)
Bartlett-Box F	.01	P=.937
INTACC		
Cochran's C	.56	P=.558 (approx)
Bartlett-Box F	.33	P=.565
DATAMOD		
Cochran's C	.58	P=.421 (approx)
Bartlett-Box F	.61	P=.436

The MANOVA procedure of SPSSX generates within-cells regression tests of significance, which measure the strength of the relationship between the covariates and the dependent variables. The within-cells results are summarized in Table 5.3. Part (a) of the table contains the multivariate test, which indicates whether the combined covariates are contributing to the overall model (using a linear, equally weighted combination of the dependent variables). Three test statistics are generated with significance levels: Pillai's criterion, Hotelling's trace, and Wilks' lambda. Part (b) of the table contains univariate F tests which indicate the effect the combined covariates have on each of the dependent variables separately. Part (c) of the table contains individual univariate tests indicating the strength of the relationships between each covariate and each dependent variable.

Table 5.3: Within-Cell (Covariate) Analysis for User Performance Model

(a) Multivariate tests of significance

Test Name	Value	Approx F	df	Error df	Significance
Pillai's	.225	2.77	4	86	.035
Hotelling's	.286	2.93	4	82	.025
Wilks'	.777	2.83	4	84	.030

(b) Univariate test of significance

Variable	F	Significance
SCORE	5.78	.006
TIME	4.59	.016

(c) Individual Univariate tests of significance

Covariate	B	Beta	Std Error	t-value	Significance
SCORE					
INTACC	6.46	.354	2.501	2.58	.013
DATAMOD	.14	.245	0.080	1.79	.081
TIME					
INTACC	.094	.350	0.037	2.50	.016
DATAMOD	-.002	-.159	0.001	1.31	.197

The data in Table 5.3 indicate that the combined covariates *IntAcc* and *DataMod* contribute significantly to the combined dependent variables *Score* and *Time*. This data also suggests that the combined covariates contribute significantly to each dependent variable separately. The individual effect tests indicate that

IntAcc is the stronger of the two covariates, affecting both *Score* and *Time* significantly. *DataMod* approaches significance in relation to *Score*, but does not contribute significantly to *Time*. Overall, the tests in Table 5.3 indicate that *IntAcc* is a valid covariate to include in the model, but that *DataMod* is somewhat questionable. To be consistent with H1, which calls for the controlling of both *IntAcc* and *DataMod*, both covariates are included in the analysis.

The next step in the MANOVA procedure is to compute multivariate and univariate tests of the effect of *Group*. These tests indicate the effect strength of the interface type that remains after the effect of the covariates has been removed. This data is presented in Table 5.4. Results of these tests suggest that once the effects of *IntAcc* and *DataMod* are removed, there is no statistically significant difference in performance between users of the abstraction interface and users of the non-abstraction interface.

Table 5.4: Main Effect (*Group*) Analysis of User Performance Model

(a) Multivariate tests of significance

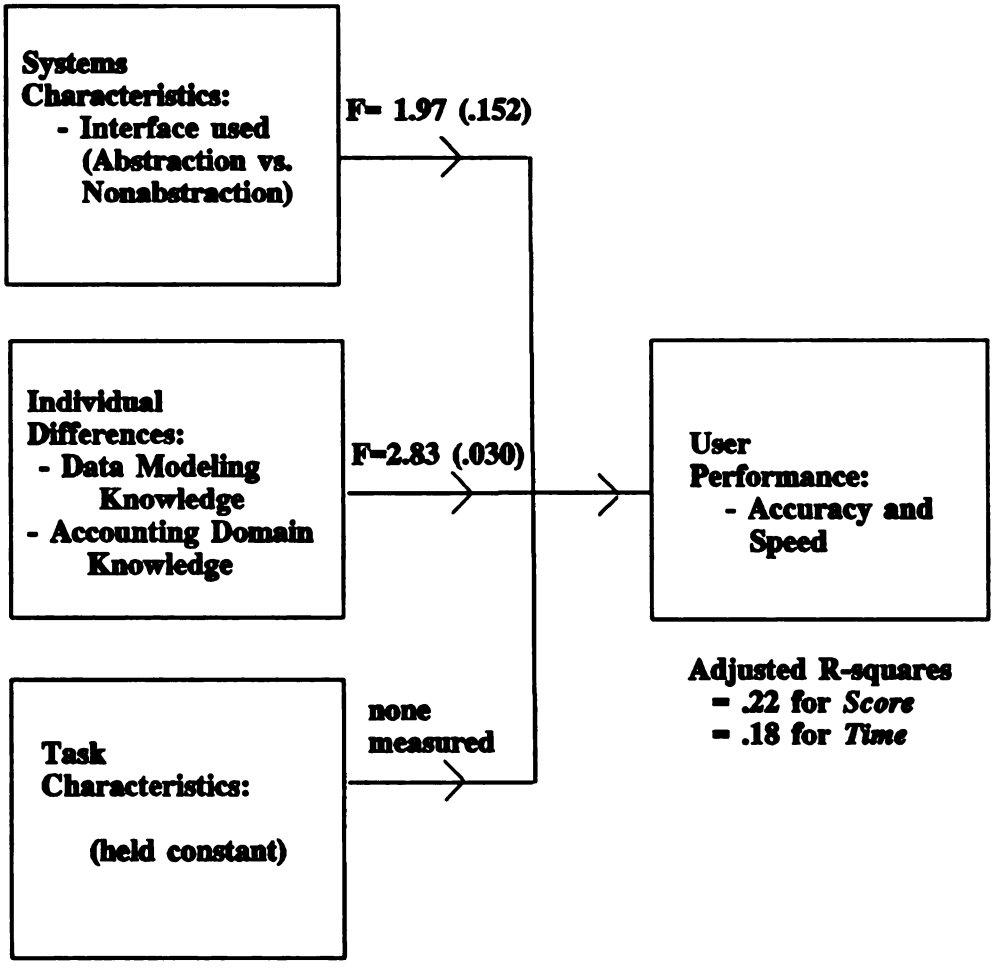
Test Name	Value	Exact F	df	Error df	Significance
Pillai's	.086	1.97	2	42	.152
Hotelling's	.094	1.97	2	42	.152
Wilks'	.914	1.97	2	42	.152

(b) Univariate test of significance

Variable	SS_h	SS_e	MS_h	MS_e	F	Significance
SCORE	687.95	7636.25	687.95	177.59	3.87	.056
TIME	.12	1.72	.12	.04	3.03	.089

The overall adjusted R-squares for *Score* and *Time*, respectively, are .22 and .18. This suggests that approximately 22 percent of the variance in subjects' accuracy scores and 18 percent of the variance in their scaled task completion time is explained by the combination of their accounting domain knowledge, their data modeling knowledge, and the type of interface they used.

Figure 5.1 summarizes these results in terms of the conceptual framework adapted from Jih et al. (1989).



Note: effect strength given is the F-statistic for each variable followed by the significance level in parentheses. The F-statistic for the effect of Individual Differences on User Performance is for the combined effect of accounting knowledge and data modeling knowledge.

FIGURE 5.1: SUMMARY OF USER PERFORMANCE RESULTS

Analysis of the results reveals that subjects' grades in intermediate accounting are a stronger determinant of their task performance (in terms of accuracy and speed) than was either the type of interface they used or their data

modeling problem score. The accounting grade was significant with higher accounting grades associated with higher task performance. Data modeling problem scores approached significance with higher data modeling scores associated with higher task performance. The type of interface was not statistically significant. Instead of abstraction group subjects exhibiting superior performance, they demonstrated similar performance. This suggests the abstraction hierarchy does not aid the user as we believed it would. Perhaps, as suggested in Chapter 3, the abstraction hierarchy over-filters or over-condenses the information presented to the users. This possibility is discussed further in Chapter 6.

5.2 Analysis of User Perception Hypothesis

The user perception hypothesis is stated in terms of perceived manageability. Subjects answered Likert scale questions as to how manageable they perceived their database interface to be. Hypothesis Two in null form (H2) suggests no difference in perceived manageability between groups of subjects using the abstraction interface versus the non-abstraction interface, after controlling for accounting and data modeling domain knowledge. This hypothesis was tested using MANOVA in order to control for the covariates. A MANOVA model with *Satis* as the dependent variable, *Group* as the factor and *IntAcc* and *DataMod* as covariates was run. The *Satis* variable in the model is the average of a subject's responses to five 7 point Likert scale questions. *Group*, *IntAcc*, and *DataMod* are defined as described in Section 5.1.

This MANOVA model was also run using SPSSX. Descriptive statistics for the *Satis* variable are presented in Table 5.5. The descriptive statistics for *IntAcc* and *DataMod* remain as presented in Table 5.1. The direction of effects indicate that overall, subjects in the non-abstraction group perceived their interface as more manageable than did subjects in the abstraction group; however, the significance tests presented later in this section reveal the difference is not statistically significant.

Table 5.5: Descriptive Statistics for User Perception Model

SATIS	Mean	Std Deviation	N
Non-abstraction	3.67	1.52	27
Abstraction	3.54	1.30	20
Total	3.62	1.42	47

Results of testing for homogeneity of variance for the *Satis* variable are presented in Table 5.6. Results for *IntAcc* and *DataMod* remain as illustrated in Table 5.2. The Cochran's C test and Bartlett-Box F test reveal no significant violations of the assumption of homogeneous variances.

Table 5.6: Tests of Homogeneity of Variance for User Perception Model

	Value	Significance
SATIS		
Cochran's C	.58	P=.463 (approx)
Bartlett-Box F	.51	P=.477

The significance test results for the user perception model are summarized in Table 5.7. Part (a) of the table contains the test of significance for *Satis* using unique sums of squares. The line entitled "Within Cells" contains the error term data. The "Regression" row indicates the significance of the combined covariates (*IntAcc* and *DataMod*) in relation to *Satis*. The line called "Group" indicates the significance of the type of interface used in relation to *Satis*. Part (b) of the table breaks down the effect of the covariates, to indicate the individual strength of each covariate in relation to *Satis*.

Table 5.7: Significance Test Results for User Perception Model

(a) Test of significance for *Satis* using unique sums of squares

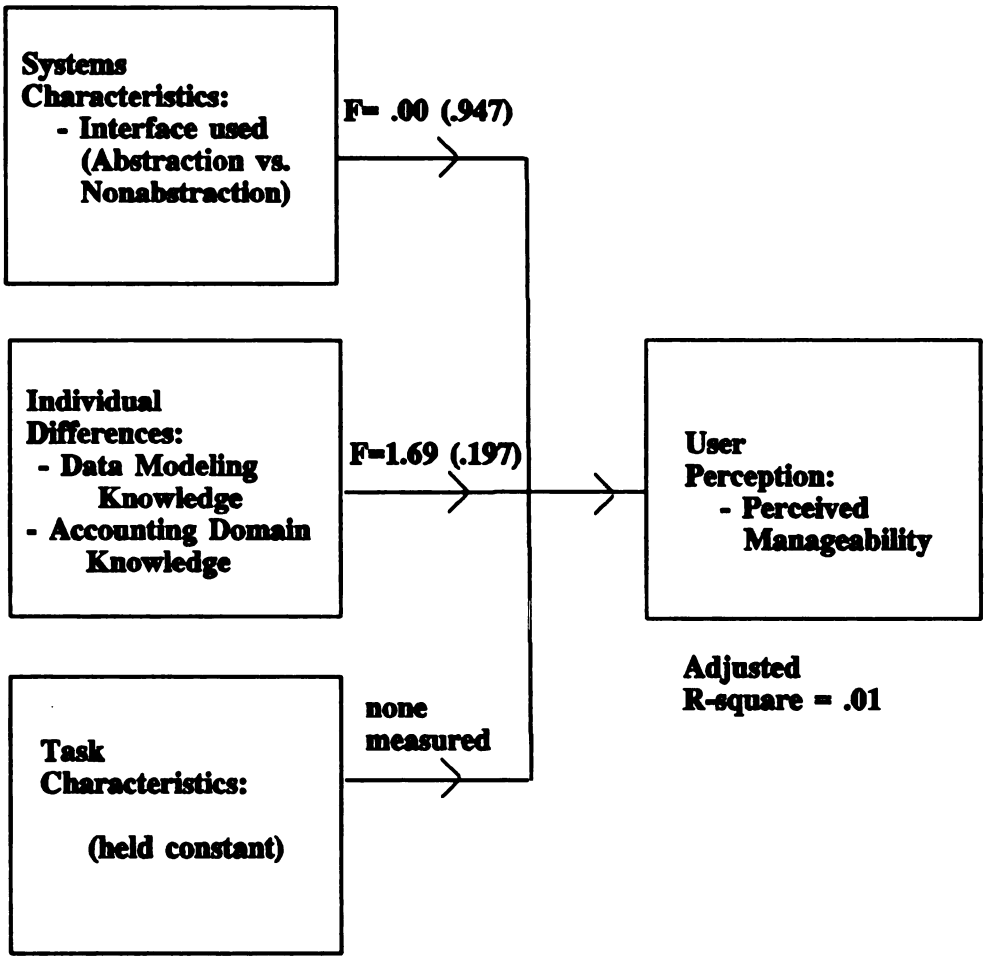
	SS	df	MS	F	Significance
Within Cells	85.91	43	2.00		
Regression	6.75	2	3.37	1.69	.197
Group	.01	1	.01	.00	.947

(b) Individual Univariate test of significance

Covariate	B	Beta	Std Error	t-value	Significance
INTACC	.184	.103	.265	0.693	.492
DATAMOD	-.015	-.266	.008	-1.788	.081

The overall adjusted R-square for *Satis* is .01. This suggests that only 1 percent of the variance in subjects' perceived manageability is explained by the combination of their accounting domain knowledge, their data modeling knowledge, and the type of interface they used.

Figure 5.2 summarizes these results in terms of the conceptual framework adapted from Jih et al. (1989).



Note: effect strength given is the F-statistic for each variable followed by the significance level in parentheses. The F-statistic for the effect of Individual Differences on User Perception is for the combined effect of accounting knowledge and data modeling knowledge.

FIGURE 5.2: SUMMARY OF USER PERCEPTION RESULTS

The data indicate that the combined covariates *IntAcc* and *DataMod* do not contribute significantly to the dependent variable *Satis*, although *DataMod* taken individually approaches significance in relation to *Satis*. The results also suggest that, after controlling for the covariates, there is no statistically significant difference in perceived manageability between users of the abstraction interface and users of the non-abstraction interface.

5.3 Summary of Findings

Based on the tests of the hypotheses described in this chapter, the following observations may be made. The provision of an abstraction hierarchy in the interface used by the subjects in this experiment for purposes of producing financial statements from an events-accounting database did not assist them as predicted, either in terms of task accuracy or in terms of task completion time. Therefore Hypothesis One in alternative form (H1A) is rejected. Hypotheses One in null form (H1), predicting no difference, may not be rejected at generally accepted levels of statistical significance as indicated by the MANOVA analysis in Table 5.4. Possible reasons for this unexpected result and future research directions resulting from it are discussed in Chapter 6.

Hypothesis Two in alternative form (H2A) predicted that users of the abstraction interface would perceive their interface as more manageable than would users of the non-abstraction interface. Thus, H2A is rejected. The null form of this hypothesis (H2), predicting no differences, cannot be rejected at

generally accepted levels of statistical significance. Subjects perceived both interfaces to be equally manageable.

End Notes:

1. The weighting of the line items was the same as used for computing task accuracy. Thus, the computation of Cost of Goods Sold (whether or not the result was correct) was assigned a higher weight than the computation of Sales. The line items assigned higher weights were expected to take more time to compute than those to which lower weights were assigned.

CHAPTER 6 - DISCUSSION AND SUGGESTIONS FOR FUTURE RESEARCH

6.1 Discussion of Results

The research question addressed by this study is whether the inclusion of an abstraction hierarchy in an interface to an events-based accounting database facilitates the preparation of financial statements. The expectation, based on previous studies in the behavioral accounting and database literatures, was that users of an interface including an abstraction hierarchy would exhibit higher accuracy, use less time, and perceive their system to be more manageable than would users of an interface without the abstraction hierarchy.

The computer science literature has taken as conventional wisdom the idea that abstraction and abstraction hierarchies are useful for controlling complexity. Brodie (1984) claims

> Abstraction is essential in database applications due to their inherent complexity which must be managed (p. 40).

Data flow diagrams are subdivided into progressively lower levels in order to provide greater amounts of detail, due to the belief that

> users have differing needs, and the differing levels can better meet users' needs for details about the system (Cushing and Romney 1993, 108).

The results of this dissertation suggest that this conventional wisdom should be subjected to further testing. In this study, user performance results suggest that there is no effect as predicted by the conventional wisdom, in terms of either accuracy or speed. No significant differences were observed for user perceptions, measured as perceived manageability.

These results do not appear to be an anomaly of the sample used in the study. Pilot test results revealed no significant differences for any of the three dependent variables, with the means showing the same directions as this experiment (i.e. non-abstraction subjects were more accurate than abstraction subjects, abstraction subjects took slightly more time, and abstraction subjects perceived their system as slightly less manageable).

6.2 Implications for Events-Based Accounting System Design

The results of this study suggest that the typical commercial database interfaces that do not contain any abstraction need not be replaced at this time by interfaces with abstraction hierarchies. This is only a preliminary suggestion as this study is the first to explore the user-computer interaction in events-based accounting systems for the task of financial statement preparation. No definitive statements may be made regarding events-based accounting system design until more research is done.

6.3 Implications for Events-Based Accounting System Instruction

Instruction in events-based accounting systems has been largely centered around teaching students how to **design** events-based databases. The results of this study suggest that more guidance needs to be given to students in how to **retrieve** information from an events-based database. One possibility that arose from the observations in this study is that perhaps emphasis needs to be placed on using REA models implemented in the form of E-R diagrams for information retrieval. Of the few schools teaching the REA accounting model in accounting

information systems courses, all seem to teach information retrieval using only the relational tables. The E-R diagrams are used for instruction on database design.

Students are taught to develop E-R diagrams from a textual description of a company's operations and information needs. Students then use the E-R diagrams to design the relational tables. One would expect them to understand that the E-R diagrams could be useful in helping them to retrieve information from the tables; however, this expectation requires analogical reasoning which students may not be using. Students may need explicit instruction as to the usefulness of E-R diagrams (and of the abstraction hierarchies present in the REA model) for information retrieval.

6.4 Future Research Directions

Overall, very little variance in user performance and in user perception was accounted for in this study by the type of interface used and by subjects' accounting and data modeling domain knowledge. This suggests the existence of variables which affect user performance and perceptions that were not measured in this study. The conceptual framework of Jih et al. (1989) may be examined to provide directions for future research. (See Figure 3.1).

6.4.1 Further Examination of System Characteristics

System characteristics are included as one category expected to affect user performance. In this study, both systems were created following the REA accounting conceptual model implemented as relational database tables. The abstraction system included in its interface an abstraction hierarchy intended to

aid users in navigating through the database. The result that users were not assisted by the interface requires consideration of two possibilities. First, users may not need any assistance in navigating the database tables. If that were the case, the accuracy scores should have been much higher than they were. The non-abstraction group scored an average of only 47 out of 84 possible points. These scores were depressed partly because of the time pressure. In pilot testing, where subjects had more than enough time to complete the task, the average score was 66 out of 84. These scores are still low enough to suggest a need for assisting users in navigating through the database tables.

The second possibility is that the abstraction hierarchy implemented may need to be modified. Perhaps it has too many layers, thus over-filtering the data. Perhaps the use of E-R diagrams in the hierarchy was confusing to subjects. Even though subjects had been thoroughly instructed in E-R implementation of the REA accounting model, many accounting information systems students claim they do not understand E-R diagramming. Thus, if the abstraction mechanisms were manifested in some format other than E-R diagrams, perhaps users would find them more useful. As discussed in Section 6.3, switching the focus of instruction from database design to information retrieval may reduce students' confusion regarding E-R diagrams and make the abstraction interface as implemented in this study more helpful.

Training on use of the system is another factor which could be handled differently in future research. Subjects were not given any advance training on the specific systems used in the experiment. The abstraction system may have a

steeper learning curve than does the non-abstraction system, thus leading to poorer performance for first-time use. The abstraction system had many screens which required the user to read one or two sentence instructions before proceeding, whereas the non-abstraction system's instructions fit entirely onto button labels (except for the initial screen which was identical to that of the abstraction system). Since subjects hadn't been specifically taught how to use the REA abstraction hierarchy for information retrieval, abstraction system users had to read and contemplate the instructions.

The non-abstraction subjects were able to concentrate immediately on the tables themselves, which was more consistent with the method of information retrieval they had been using in class. In future research, subjects should be provided with explicit instructions on the use of the REA abstraction hierarchy to retrieve data, and they should be allowed to practice using the system to which they are assigned (with a different database and a different task) before performing the experimental task. This may lead to very different results than were found in this study. A follow-up study is being planned to determine whether different learning curves for the two systems affected the results of this particular experiment.

Another direction for future research is the examination of this research question with a database that is made up of a greater number of tables and attributes than the one used in this experiment. The database used in this study was very simple. Wilson Company offered only one product line, consisting of six different items. There were no fixed assets and no manufacturing cycle. The

attributes in the tables did not include many items that would be considered useful for non-accounting decisions.

The database used in this experiment contained 33 tables with 132 attributes. Twenty of the 33 tables (61%) contained information needed for financial statement preparation. Of the 132 attributes included in the database, 64 were needed for identifying instances in the tables or for identifying relationships between tables in accordance with the relational model. Twenty-six of the attributes contained data needed for financial statement item computations. The remaining 42 attributes contained data useful for record-keeping or for other types of decisions (e.g. vendor address, customer credit rating). Thus, 90 out of 132 (68%) of the attributes were needed for the financial statement preparation task.

The expected benefit of abstraction is to assist users in filtering out data which is *irrelevant* for their task. Only 32% of the data in the Wilson Company database could be considered irrelevant for the preparation of Wilson's financial statements. A corporate-wide database of a realistically complex company would include many more attributes that would be irrelevant in generating financial statements but would be useful for making marketing, management, personnel, or logistics decisions. There does not appear to be any theory to guide us as to how high the percentage of irrelevant data would need to be in order to demonstrate the point at which abstraction becomes beneficial. Future research should manipulate the size and complexity of the database, to try to identify a point (if one exists) at which user performance and/or user perceptions are consistent with the predictions made in this study.

6.4.2 Further Examination of Individual Differences

Individual differences make up the next category of variables which may affect user performance. Only two types of domain knowledge were measured in this study. Much of the variance unaccounted for could have been due to other individual differences. Field dependence, cognitive style, and general problem-solving ability are examples of other human characteristics that could be incorporated into future research. Subjects used in this experiment were knowledgeable in financial statement preparation, but were novices in terms of real-world experience. Having not applied their knowledge on a regular basis, they may not have demonstrated their true potential performance because they were not confident in their knowledge.

Because the environment of the experiment was similar to an examination situation, some subjects could have been stricken by test anxiety. Alternatively, subjects may not have had adequate motivation to perform up to their potential. Students should be motivated to attain as high a course grade as possible, but many appear willing to settle for lower grades in exchange for less effort.

Data modeling domain knowledge was measured as an individual difference variable in this study. However, there are two potential problems with this variable's measurement. One problem is that the data modeling test problem on which they were scored was a database design problem. Subjects were given a textual description of a company and asked to draw an E-R diagram and a set of relational database tables to be used in that company. This may not adequately tap their knowledge of data modeling from an information retrieval standpoint.

The second problem is that it is possible that all subjects had enough data modeling domain knowledge such that those who did not have the abstraction hierarchy interface were able to mentally picture the necessary parts of the hierarchy. If this were the case, subjects without the abstraction interface could scroll through the tables to see what was there and determine a financial statement line item to compute. Seeing the tables may have triggered a mental image of the abstraction hierarchy for the relevant cycle, which directed the user as to the appropriate tables to use in computing a given financial statement line item. The user could then scroll through the tables to find the other necessary tables. Users with the abstraction interface who could already picture the necessary parts of the abstraction hierarchy may have been frustrated by the provision of useless overhead (particularly since it was accompanied by instructions they needed to assimilate, as discussed earlier). This would explain why the system may have actually hindered their performance and certainly did not help them.

An experiment is being planned as an extension of this dissertation to test this possibility. Subjects who have adequate accounting domain knowledge will be given a short training session on information retrieval from an events-based accounting database. These subjects will then prepare the same financial statements that the current study's subjects (with data modeling domain knowledge) completed. Such a study will help to isolate whether there is an interaction effect between the provision of an abstraction hierarchy interface and the existence of subjects' data modeling domain knowledge.

6.4.3 Further Examination of Task Characteristics

Task characteristics make up the third category which Jih et al. (1989) expect to affect user performance. The task was held constant in this study. Financial statement preparation is a task which is complex, but well-structured. Future research could examine whether the abstraction interface would have a different effect on user performance for tasks which are unstructured.

6.4.4 Refinement of Task Completion Time Measurement

Future research should refine the measurement of task completion time. The 60 minute time frame allowed in this study was not adequate to allow most subjects to complete the financial statement preparation task, thus causing a ceiling effect for the raw time measurement. Scaling of this variable results in a measure which is less precise than that which would have been obtained with a longer time frame. Protocol analysis via computerized process tracing would allow for an even finer measurement of time to be obtained.

6.5 Summary

The results of this dissertation can be summarized as follows. The hypothesis of no difference in task performance between users of the two interfaces could not be rejected. The user perception hypothesis put forth in this dissertation also could not be rejected. This study has shown that the conventional wisdom of abstraction as an aid to users requires empirical examination to determine the types of situations in which it will provide assistance and the occasions for which it may actually hinder users. Several future research

directions have been suggested as a result of the findings of this study. These constitute the beginning of a research program on the use of abstraction in user interfaces for information retrieval.

APPENDIX 1

APPENDIX 1: EXPERIMENTAL INSTRUCTIONS

To Computer Project Participants:
Thank you for coming today. The computer project in which you are participating is being conducted in order to enable a comparison between users of events-based accounting systems. The results should help in future design of such systems. This project will require approximately 80 minutes of your time, in one session.

General Instructions:
Your task is to prepare handwritten financial statements (Income Statement, Balance Sheet, and Statement of Changes in Retained Earnings) for Wilson Company as of June 30. All information needed to complete the statements is in Wilson's database on the computer you are using. IGNORE TAXES!

As a reminder, an Income Statement includes the company's revenues and expenses. A Balance Sheet includes the assets, liability, and stockholders' equity of the company. A Statement of Changes in Retained Earnings starts with the beginning balance of Retained Earnings, lists additions and reductions and concludes with the ending balance of Retained Earnings.

Beside each financial statement number, you must give a brief explanation of how you calculated it. For example, if you were asked to compute the total number of students enrolled at the Specialty Arts Academy (a student can only take 1 course) and you had the following table included in your database:

C-1

COURSE NUMBER	Course Name	Number of Students Enrolled
100	Art	20
101	Dance	20
102	Drama	30

Your answer would be presented as follows:

70 - Went to the C-1 table and summed the "Number of Students Enrolled" column.

Tools:
You may use a calculator. You should be able to complete your task by using the computer and your calculator. You have also been provided with a stack of scrap paper. Each piece of scrap paper is a different color. You may use this paper for temporary recording of information - if you need it. However, you may not use this paper to record information you need on a long-term basis (you are to refer back to the database for such things). To ensure that no-one uses the paper for such purposes, the paper will be collected every 10 minutes. For example, if your session starts at 8:10, the first page (e.g. yellow) will be collected at 8:20. The second page (e.g. green) will be collected at 8:30,

etc. You will be given a warning 1 minute before each collection so that you may begin new calculations on the next page. This will prevent collection of paper in the middle of a calculation you may be making. (Note: the ideal situation would be for you to not need to use the paper at all).

If you have any mechanical difficulties with your computer or your mouse, please raise your hand for assistance. No questions about Wilson Company, the database, or financial statements can be answered. You should be able to figure out everything you need from the database.

Maneuvering Through the Database:
On each screen you will see buttons with labels such as "Continue" to go from screen to screen. Only use these buttons to maneuver through the system. Because this system was created in a windowing environment, each screen is also a window. Like any windows, they may be closed or iconicized by clicking on the arrows in the upper left and right hand corners. DON'T DO THIS -- IT WILL THROW YOU OUT OF THE SYSTEM! If you accidentally do this, raise your hand, and I'll come get you back in. But this will be a waste of your time!

CAUTION!!: Each table ends with a blank row. Any table that has more than five rows has a scroll bar on the right side. There are more entries in these tables than you see on the screen! Click the left mouse button on the arrow-heads of the scrollbar to scroll down and up through the table. You will know you have reached the end of a table when you see the blank row.

Class Credit:
As you know, you are completing this project for class credit. Your grade will be assigned based on the following criteria:

1) Accuracy and completeness of the financial statements you turn in.
2) How quickly you complete the task.

For example, a paper that is 90% accurate (with complete and correct explanations) that is completed in 40 minutes would be graded higher than one which is 100% accurate and is completed in 60 minutes. Thus, although you are allowed an hour to finish the task, if you finish in less time, you should turn in your statements right away.

Questionnaire:
When you have turned in your financial statements, and your completion time has been recorded, you will be given a questionnaire to complete. You must fully complete this questionnaire in order to receive class credit for this project.

Confidentiality:
I will provide copies of your financial statements to your instructor. The financial statements will be seen only by your instructor and by me. Your questionnaire will be seen only by me (your instructor will not receive a copy). All responses will remain confidential, and are obtained only for statistical purposes. Any results will be reported in aggregate form. If you have any questions regarding this project, feel free to contact me through the Accounting Department at 355-7486.

APPENDIX 2

APPENDIX 2: EXPERIMENTAL QUESTIONNAIRE

For each question that follows, please circle the number which corresponds most closely to your experience in this project. (1 = Strongly Agree, 4 = Neutral, 7 = Strongly Disagree)

1. I found the database interface cumbersome to use.

1 2 3 4 5 6 7

Strongly Agree Neutral Strongly Disagree

2. Using the database interface was frustrating.

1 2 3 4 5 6 7

Strongly Agree Neutral Strongly Disagree

3. Using the database interface required a lot of mental effort.

1 2 3 4 5 6 7

Strongly Agree Neutral Strongly Disagree

4. The database interface is clear and understandable to me.

1 2 3 4 5 6 7

Strongly Agree Neutral Strongly Disagree

5. Overall, I found the database interface easy to use.

1 2 3 4 5 6 7

Strongly Agree Neutral Strongly Disagree

Demographic information (strictly confidential - for statistical purposes only)

Circle the appropriate category:

Your age:	0-19	20-21	22-24	25+	
Your grade level:	Fr. So.	Jr. Sr.	PPA	Grad student (non-PPA)	
Your GPA in accounting classes:	<2.5	2.5-2.99	3.0-3.49	3.5-4.0	
Your Overall GPA:	<2.5	2.5-2.99	3.0-3.49	3.5-4.0	

Place a check mark next to the accounting courses you have taken (besides 321). If you are currently taking one of the courses, add a "c" next to the check mark. If you took an equivalent of the MSU course at another school, add a "*" next to the check mark.

_____ ACC 201 (Financial Accounting Principles)

_____ ACC 202 (Managerial Accounting Principles)

_____ ACC 251H (Honors Accounting Principles)

_____ ACC 300 (Intermediate Financial Accounting I)

_____ ACC 301 (Intermediate Financial Accounting II)

_____ ACC 308 (Governmental/Not for Profit Accounting)

_____ ACC 341 (Cost and Managerial Accounting - used to be 303)

_____ ACC 411 (Auditing)

_____ ACC 419 (Auditing Theory)

_____ ACC 431 (Federal Income Tax)

_____ ACC 439 (Federal Taxes)

_____ ACC 490 (Independent Study)

Please list the computer science courses you have taken:

How many months of work experience (paid or unpaid) involving accounting responsibilities have you had (do not include being an accounting TA):

_____

Are you (or have you been) an Accounting Teaching Assistant? _____

If yes, for which course (circle one): Acc 201 Acc 202 Acc 230

If yes, for how many months have you been an Accounting TA? _____

How many months of work experience (paid or unpaid) involving regular use of computers or computer systems have you had:

_____

LIST OF REFERENCES

LIST OF REFERENCES

Abramson, D.H. 1986. The future of accounting: Scenarios for 1996. *Journal of Accountancy*. October. 120-124.

Ackoff, R.L. 1967. Management misinformation systems. *Management Science*. December. B147-B156.

Batini, C., S. Ceri, and S.B. Navathe. 1992. *Conceptual Database Design: An Entity-Relationship Approach*. Benjamin/Cummings, California.

Batra, D., J.A. Hoffer, and R.P. Bostrom. 1990. Comparing representations with relational and EER models. *Communications of the ACM*. February. 126-139.

Beaver, W.H. and A. Rappaport. 1984. Financial reporting needs more than the computer. *Business Week*. August 13. 16.

Benbasat, I. and A.S. Dexter. 1979. Value and events approaches to accounting: An experimental evaluation. *The Accounting Review*. October. 735-748.

Bonner, S.E. and B.L. Lewis. 1990. Determinants of auditor expertise. *Journal of Accounting Research*. Supplement. 1-20.

Brodie, M.L. 1981. Data abstraction for designing database-intensive applications. *Proceedings of the Workshop on Data Abstraction, Databases and Conceptual Modeling (1980)*. Pingree Park, Colorado. SigArt, SigMod, SigPlan, 101-103.

_____. 1984. On the development of data-models. *On Conceptual Modelling*. eds. Brodie, M.L., Mylopolous, J. and Schmidt, J.W. Springer-Verlag, New York. 19-47.

_____. and D. Ridjanovic. 1984. On the design and specification of database transactions. *On Conceptual Modelling*. eds. Brodie, M.L., J. Mylopolous, and J.W. Schmidt. Springer-Verlag, New York. 277-306.

Casey, C., Jr. 1980. Variation in accounting information load: The effect on loan officers' predictions of bankruptcy. *The Accounting Review*. January. 36-49.

Chan, H.C., K.K. Wei, and K.L. Siau. 1991. Conceptual level versus logical level user-database interaction. *Proceedings of the 12th International Conference on Information Systems*. New York. December. 29-40.

Chervany, N.L. and G.W. Dickson. 1974. An experimental evaluation of information overload in a production environment. *Management Science*. June. 1335-1344.

Cushing, B. 1989. On the feasibility and consequences of a database approach to corporate financial reporting. *The Journal of Information Systems*. Spring. 29-51.

_____ and M.B. Romney. 1993. *Accounting Information Systems: A Comprehensive Approach*. Addison-Wesley.

Date, C.J. 1986. *An Introduction to Database Systems*. Fourth edition. Addison-Wesley.

Davidson, H.J. and R.M. Trueblood. 1961. Accounting for decision making. *The Accounting Review*. October. 577-582.

Davis, F.D. 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*. September. 319-340.

Davis, G.B. and M.H. Olson. 1985. *Management Information Systems: Conceptual Foundations, Structure and Development*. Second edition. McGraw-Hill.

DeMarco, T. 1979. *Structured Analysis and System Specification*. Prentice-Hall.

Dunn, C.L. and W.E. McCarthy. 1992. Conceptual models of economic exchange phenomena: History's third wave of accounting systems. *Collected Papers of the Sixth World Congress of Accounting Historians*. Volume I. August 20. 133-164.

Einhorn, H.J. and R.M. Hogarth. 1980. Rationality and the sanctity of competence. Working paper.

Financial Accounting Standards Board. 1989. *Accounting Standards, Original Pronouncements July 1, 1973 - June 1, 1989*. Irwin.

Gal, G. and W.E. McCarthy. 1992. Semantic specification and automated enforcement of internal control procedures within accounting systems. Working paper.

Geerts, G. and W.E. McCarthy. 1992. The extended use of intensional reasoning and epistemologically adequate representations in knowledge-based accounting systems. *Proceedings of the Twelfth International Workshop on Expert Systems and Their Applications*. Avignon, France. June 1992. 321-332.

Herot, C., R. Carling, M. Friedell, D. Kramlich, and R. Rosenberg. 1981. Overview of the spatial data management system. Computer Corporation of America, Technical Report CCA-81-08. November.

Howe, D.R. 1983. *Data Analysis for Data Base Design*. Edward Arnold.

Jih, W.J.K., D.A. Bradbard, C.A. Snyder, N.G.A. Thompson. 1989. The effects of relational and entity-relationship data models on query performance of end users. *International Journal of Man-Machine Studies*. 257-267.

Kent, W. 1981. Consequences of assuming a universal relation. *ACM Transactions on Database Systems*. December. 539-557.

_____. 1983. The universal relation revisited. *ACM Transactions on Database Systems*. December.

Libby, R. 1983. Determinants of performance in accounting decisions. *Accounting Research Convocation*. University of Alabama.

_____. and Lipe. M. 1992. Incentives, effort, and the cognitive processes involved in accounting-related judgments. *Journal of Accounting Research*. Autumn. 249-273.

_____. and Luft, J. 1993. Determinants of judgment performance in accounting settings: Ability, knowledge, motivation, and environment. *Accounting, Organizations, and Society*. July. 425-450.

Lochovsky, F.H. and D.C. Tsichritzis. 1977. User performance considerations in DBMS selection. *ACM-SigMod International Conference on Management of Data*. August. 128-134.

Loomis, M.E.S. 1987. *The Database Book*. Macmillan. New York.

March, S.T. and G.F. Smith. 1994. Design and natural science research on information technology. *Decision Support Systems*. forthcoming.

McCarthy, W.E. 1979. An entity-relationship view of accounting models. *The Accounting Review*. October. 667-685.

_____. 1982. The REA accounting model: A generalized framework for accounting systems in a shared data environment. *The Accounting Review*. July. 554-578.

_____. 1987. On the future of knowledge-based accounting systems. *The D.R. Scott Memorial Lecture Series*. The University of Missouri. 19-42.

Moore, G.C. and I. Benbasat. 1991. Development of an instrument to measure the perceptions of adopting an information technology innovation. *Information Systems Research*. September. 192-222.

Morris, A.H., G.M. Kasper. and D.A. Adams. 1992. The effects and limitations of automated text condensing on reading comprehension performance. *Information Systems Research*. 3:1. 17-35.

Ogden, F. 1991. Dr. Tomorrow searches for electronic future. *Computerworld*. September 9. 19.

Otley, D.T. and F.J.B. Dias. 1982. Accounting aggregation and decision-making performance: an empirical investigation. *Journal of Accounting Research*. Spring. 171-188.

Palmer, S.E. and R. Kimchi. 1986. The information processing approach to cognition. In T. Knapp & L. Robertson (Ed.) *Approaches to Cognition*. Erlbaum. 37-77.

ParcPlace Systems. 1992. VisualWorks version 1.0.

Rappaport, A. 1968. Management misinformation systems - another perspective. *Management Science*. December. B133-B136.

Revsine, L. 1970. Data expansion and conceptual structures. *The Accounting Review*. October. 704-711.

Schick, A., L. Gordon, and S. Haka. 1990. Information overload: A temporal approach. *Accounting, Organizations, and Society*. 15:3. 199-220.

Shneiderman, B. 1980. *Software Psychology: Human factors in computer and information systems*. Winthrop Publishers, Massachusetts.

_____. 1992. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley.

Simon, H.A. 1990. Information technologies and organizations. *The Accounting Review*. July. 658-667.

Smith J.M. and D.C. Smith. 1977. Database abstractions: aggregation and generalization. *ACM Transactions on Database Systems*. June. 105-133.

_____. and _____. 1978. Principles of database conceptual design. *Data Base Design Techniques I: NYU Symposium*. New York. May. 114-146.

Snowball, D. and C. Brown. 1979. Decision making involving sequential events: some effects of disaggregated data and dispositions toward risk. *Decision Sciences*. Vol. 10. 527-546.

Sorter, G.H. 1969. An 'events' approach to basic accounting theory. *The Accounting Review*. January. 12-19.

The American Heritage Dictionary. 1980. W. Morris, ed. Houghton Mifflin.

Ullman, J.D. 1983. On Kent's "Consequences of assuming a universal relation." *ACM Transactions on Database Systems*. December 1983.

University of Southern California Financial Accounting Study Group. 1991. *Setting Financial Accounting Standards for the Twenty-first Century*. Topical Issue Study No. 4. School of Accounting, USC.