1990

ABSTRACTION BASED MODELING: AN EMPIRICAL STUDY OF THE PROCESS

Ananth Srinivasan
Case Western Reserve University

Dov Te'eni
Case Western Reserve University

ABSTRACTION BASED MODELING: AN EMPIRICAL STUDY OF THE PROCESS

Ananth Srinivasan
Dov Te'eni
The Weatherhead School of Management
Case Western Reserve University

ABSTRACT

This paper takes a cognitive view of data modeling to address the usability of semantic modeling techniques and their potential impact on end user productivity. Specifically, we use a process tracing methodology to study the manner by which end users are able to use abstraction based data modeling to represent a reasonably complex problem and construct queries to address represented objects. The paper views the process of modeling as a constrained problem solving process. Drawing from research that has studied the systems design process, a cognitive model is developed and used to examine the semantic modeling process.

We define a semantic modeling environment for the subjects of the experiment. Two specific abstractions are used: generalizations and composite objects. The representation constructs and the valid operations that are permitted on them are described.

The results show that abstraction based modeling is a viable end user development methodology. Tentative recommendations for the design of platforms and training programs emphasize the need to encourage users to work at high levels of abstraction and utilize particular modeling heuristics that lead to better task performance.

1. INTRODUCTION

User productivity has been central in the arguments proposed in favor of end user computing (e.g., Rockart and Flannery 1983; Munro, Huff, and Moore 1987). Several factors have been cited as being conducive to its nurturing and development. The main factors are increased technical awareness and literacy of a large cadre of users, more powerful technologies, and affordability. Linking end users with appropriate development tools is now seen as a desirable response to the productivity issue. One such development technology is data management. There is considerable evidence to suggest that a large number of systems applications are data management applications (Batini and Ceri 1985). It is therefore meaningful to examine the role of this technology in the context of end user computing. In this research, we empirically examine the link between end users and a viable data management approach, namely, semantic modeling. Specifically, our focus is to understand the process by which end users utilize semantic modeling, in order to translate a problem in an application domain to its representation in the modeling environment, and the subsequent use of the model to address specific tasks.

The abilities of users with varying degrees of data modeling expertise have been empirically studied by a number of researchers. Presumably, this interest has been a reflection of the fact that this particular technology may be a suitable vehicle for wider user participation in application development. The earliest set of relevant studies focussed on the database querying capabilities of users with the motivation of designing appropriate query interfaces (Zloof 1975; Reisner 1981; Vassiliou et al. 1983). More recently, with conceptual developments in more powerful modeling approaches (such as semantic modeling in its various forms), there have been studies on the abilities of users to develop adequate representations of a problem (Mantha 1987; Jarvenpaa and Machesky 1989; Batra, Hoffer, and Bostrom 1990). Many of these studies attempted to test hypotheses related to the usability of semantic modeling as an effective representation vehicle. The verdict is mixed; it is not clear whether semantic modeling, in the final analysis, produces better user performance in terms of addressing specific tasks (constructing and executing queries against a particular representation). The research presented in this paper is an attempt at addressing this process oriented perspective of problem representation and its subsequent use within a semantic modeling framework.

The perspective on modeling, as described above, is fundamentally a cognitive process. There is a representation of a problem in an application domain that the user has to convert into a representation as per the dictates of the modeling environment. Further, tasks then have to be translated to operations that have to be carried out on represented objects. We examine this process as one of cognitive processing in order to enable us in understanding

the conditions that make for different types of usage behaviors. Systems design has been similarly viewed and insights have been gained in our understanding of that process (Rouse and Boff 1987). There are many similarities between systems design in general and modeling as viewed in our research.

Since the *process* of representation is emphasized in this research, a process tracing methodology is utilized to study it (Ericsson and Simon 1980; Todd and Benbasat 1987). The methodology has been used quite successfully in a variety of different contexts (e.g., Bouwman 1982; Reitman 1976). In most cases, it has been used to identify the constituents of expertise in a particular area and to differentiate novice behavior from that of experts. In this research, we are more interested in studying how reasonably trained users may function in the modeling environment. It is not clear that understanding expert behavior with a view to emulating experts is feasible in the end user context. It appears to be more useful instead to identify successful usage patterns of users who are neither experts nor complete novices. Such an identification should have important implications for training end users and in the design of systems that support data modeling.

In Section 2, a cognitive model is developed that forms the basis of the research and describe how it guides our study of the modeling process. The historical development of semantic modeling as a field is briefly discussed. Section 3 describes the research method employed; specifically, the modeling environment that was utilized and the conduct of the experiments. The results and observations from the experiments, as well as the implications of the results obtained from this research, are discussed in Section 4. Section 5 presents comments on useful avenues for pursuing this line of research.

2. MODELING AS A COGNITIVE PROCESS

In this section, a cognitive model that describes data modeling as a constrained problem solving process is developed. Data modeling is generally viewed as a design activity in the information systems development life cycle; however, research on the data modeling process has not drawn from research about design in general. This may be due to the rather concrete connotation of design in contrast to the abstract nature of data modeling. We draw from this relationship of modeling to systems design and demonstrate its utility and briefly discuss the development of advanced data modeling concepts and highlight issues of interest.

2.1 The Systems Design Perspective

Rouse (1986) regards design as a four stage process: formulation of the design problem, generation of alternative design solutions, analysis of these alternatives, and selection of a preferred alternative. Furthermore, Rouse, building on principles of human cognition (Newell and Simon 1972), emphasizes the information processing nature of the problem solving process. This view of design is adopted to describe data modeling with two augmentations. First, we concentrate on the "understanding" phase in system design that is needed primarily for the generation and analysis of, and selection from, alternative design solutions. Second, the cognitive model of data modeling includes the use of the designed data model as opposed to being restricted to representation only.

Understanding in system design entails seeking information, translating it, and representing knowledge. These three activities (and the control over these activities) are the primary constituents of data modeling. In this context, knowledge representation includes both internal representation (in memory) and external representation (the conceptual data model). Knowledge representation is therefore central to our discussion. The definition of knowledge representation must incorporate additional elements, such as operators showing the potential use of the data model. Indeed, in their discussion of problem solving, Greeno and Simon (1984, p. 4) describe the product of understanding as a problem representation that incudes

> an individual's representation of the *objects* in the problem situation, the *goal* of the problem, and the *actions* that can be performed and *strategies* that can be used in working on the problem. It also includes knowledge of *constraints* in the problem situation: restrictions on what can be done, as well as limits on the way in which objects or features of objects can be combined.

In the context of data modeling, it is important to explain how this (internal) representation interacts with the development of the (external) data model. Two points are made. First, the interaction works both ways: internal representation affects the data model and vice versa. Second, constraints play a crucial part in this interaction. Ballay (1987) proposed a transaction model of the design process, which accounts for the information held partly in the designer's memory and partly in the external environment. According to this model, designers begin sketching an initial design with only partial information. By retrieving (seeks and translates, in our terminology) additional information, the designer is able to progressively refine initial representations. Figure 1 is an adaptation of Ballay's model. It depicts data modeling, procedurally, in the simplified setting of modeling from a narrative description of the real world. The internal representation and the external data model interact and evolve over time; the narrative remains static. Figure 1 emphasizes the dynamic nature of data modeling, rendering obsolete the notion that data modeling proceeds from a general understanding to

a more detailed understanding to some external representation (e.g., relational), to testing and model refinement. The implication of semantic modeling being a natural representation vehicle implies a smooth transition from the internal to the external representation in our model.

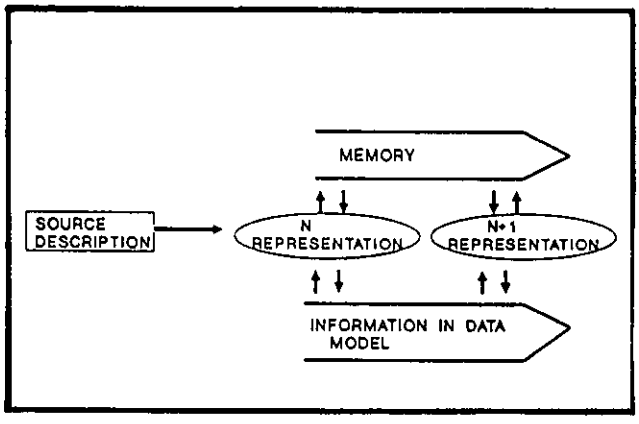


Figure 1. Transaction Model of the Design Process

The second theme of our model is the central role of constraints on the design process. Guindon (1987) presents a model of cognitive processes in software design that shows how the design process is affected by constraints. Constraints may be classified into several types: design-process, cognitive, problem domain, functional, transitional, and artifact-performance constraints. Interestingly, Guindon identifies several heuristics that help the software designer control the design process in the face of these constraints. Here, we use these ideas, but concentrate on a limited set of relevant constraints and heuristics.

Figure 2 shows a functional model of data modeling that includes the modeling activities, the constraints placed on these activities, and the heuristics used to control the activities.

2.2 Components of the Model

Data-modeling activities are seeking information, translating information, representing objects, relationships and operators internally, using tools to represent the objects, relationships and operators externally, and testing and, as a result, refining the representations (Jeffries et al. 1980; Borgida, Greenspan, and Mylopoulos 1985). Figure 2 shows these activities within the double-lined rectangle, emphasizing the interactive, rather than sequential, nature of data modeling. Moreover, these activities are performed at different levels of abstraction. Duncker (1945) demonstrated how problem solving progresses through shifts from one level of abstraction to another, and Rasmussen (1983) demonstrated the use of varying levels of abstraction to represent the functional properties of a system. In a study of design activities, Ballay (1987) found that routine design activities were accompanied by short "bursts" of planning at a higher level, which appeared to be necessary for effective design. The conditions that induce these shifts are discussed below.

Constraints are limitations on the effectiveness of data modeling activities due to modeling tools, cognitive ability, problem requirements, functions, and transitions between states in the design process. **Modeling tools** provide a limited set of symbols to describe objects, relationships between objects, and operators to use the model. **Cognitive ability** is limited in knowing the tools and the problem domain and structuring and restructuring of representations. **Problem domain requirements** can be characterized by the number of objects and the complexity of their interrelationships. **Functions** are the user's information requirements that have to be produced from the data model. **Transitions** are the restrictions on the progression from one state to another in the design process.

Heuristics help the user control data modeling activities when faced with constraints. The focus here is on heuristics that could be observed in the interaction between the internal and external representation. We expect to find explicit use of the following heuristics:

(a) decompose a problem into sub-problems,

(b) temporarily leave a sub-problem with the intention of resolving it later,

(c) test a solution by simulating some operations,

(d) shift to a higher level of abstraction when you get lost in detail,

(e) pause to plan ahead (a special case of the previous heuristic), and

(f) shift to a lower level of abstraction when the higher level can no longer guide you.

Figure 2 is a tentative model of data modeling and our exploratory analysis is aimed at refining it. The first objective was to study verbal protocols describing the modeling process to detect elements and interactions described in the proposed cognitive model. Our observational focus was on the effect of constraints, particularly the effects of existence dependency chains (a problem-domain requirement) and tool constraints, the appearance of the above mentioned heuristics, and the dynamic nature of data modeling exhibited by shifts between data-modeling activities and between levels of abstraction within activities, as governed by the heuristics. The second objective was to examine the relationship between process, effort, and performance.

2.3 Semantic Modeling

The remainder of this paper uses the model presented in Figure 2 to investigate semantic modeling. The primary argument proposed in favor of semantic modeling is that the objects that are represented bear close semantic

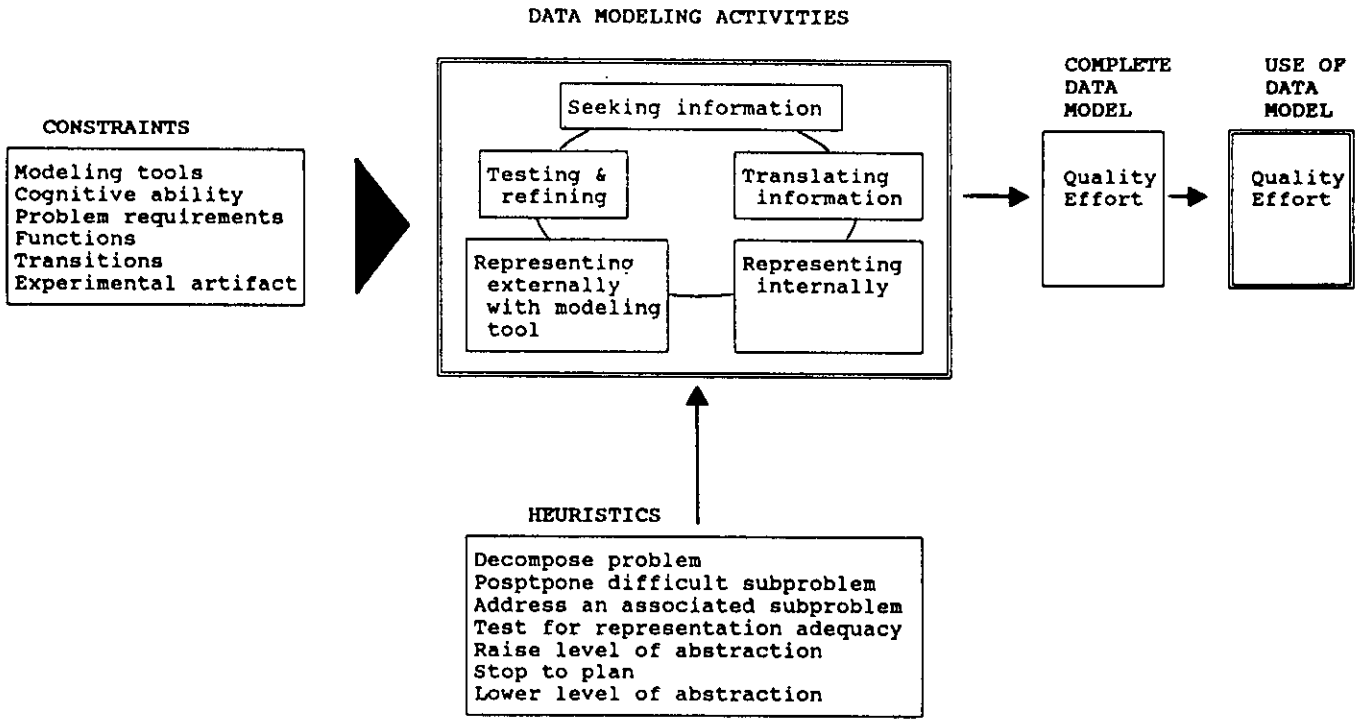


Figure 2. Cognitive Model of Data Modeling

resemblance to real world entities (Brodie 1984). The benefit of this is that the behavior of these objects during user manipulation is more easily related to the real world activities that are reflected in these manipulations. If users are expected to be proficient in the use of data management systems, then it is incumbent upon designers to provide environments that allow the user to capture a problem in a manner that is as "natural" as possible. It is this motivation that has led to the development of a multitude of advanced modeling ideas over the past few years. In this section, the highlights of this development are briefly reviewed.

Schmid and Swenson (1975) were of the opinion that it is important for the user to clearly comprehend the manner in which the world is to be represented. Smith and Smith (1977) argued that issues of understandability, efficiency, and consistency must be concurrently addressed by a representation vehicle. Convincing arguments against record based structures are provided by Kent (1979), where he demonstrates that the relational approach is quite awkward in handling commonly encountered complexities. Hammer and McLeod (1981) stated that the issue of "semantic relativism" becomes increasingly important in situations of reasonable complexity. The link between reasonably complex problems and the absence of mechanisms that help the user to interpret the data is an issue identified by Tsichritzis and Lochovsky (1982).

Responses to these concerns have taken the form of proposing vehicles of representation and subsequent manipulation that seek to specifically address the identified shortcomings. The clear delineation between entities, and relationships between them, was highlighted initially by Chen (1976) and subsequently expounded upon by Elmasri, Hevner, and Weeldreyer (1985) and Teorey, Yang, and Fry (1986). Smith and Smith (1977) specifically addressed the issue of task relevance and proposed how certain commonly used abstractions may be incorporated in the design of relational models. Hammer and McLeod (1981) presented an approach to representation that showed how elaborate grouping and classification schemes and structural inter-connections could be implemented. The object oriented approach has taken the view that it is important to encapsulate the structure and the behavior of represented objects thereby emphasizing the dynamic nature of represented entities (Stefik and Bobrow, 1986). It should be noted however, that there are critics of the semantic approach. Stonebraker (1988) argued that a query language supported by a semantic model may not naturally support ad-hoc ways of retrieving data.

3. RESEARCH METHOD

3.1 Modeling with Abstractions

In the experimental task, our aim was to provide an environment that allowed users to represent a problem using commonly discussed semantic abstractions. Specifically, we focused on two abstraction mechanisms: generalizations and composite objects. The fundamental represen-

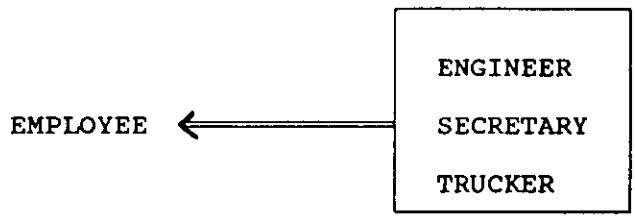
tation construct is an entity with associated properties. Collections of entities could be viewed at an abstract level as a single object; abstractions are the vehicles that allow this particular perspective on represented objects.

Generalizations allow for the definition of an entity type along with its associated subtypes. While all subtypes share a common set of properties, each one may have a unique property subset that is different from other subtypes of the same parent. In this experiment, the construct was confined to strict hierarchies and no more than one level of specialization. The generalization abstraction was constructed as a cluster that contains the supertype with all its associated subtypes. Each subtype of an entity has an associated property list that is exclusive to that subtype. Further, given that its supertype is identified, property inheritance from the supertype is implied.

The composite object abstraction allows for the treatment of a group of related, yet independent, entities as a "molecule." This allows us to capture the relationship concept from entity-relationship modeling (Chen 1976) as well as more complicated situations where the composite object in turn is associated with another entity.[1] A composite object is declared in terms of its component entities. A component entity may be another composite object; in such cases, it is meaningful to refer to a chain of related objects. Composite objects may or may not have properties.

The basic structure of a query followed typical SQL structure: a target list with set operators, if any, a source list, and a predicate list. Simple entities behave exactly like a relation. References to an object in a supertype or subtype allow the user to make target- or predicate-based references to properties in the entire cluster. References to a composite object allow the user to make target- or predicate-based references to properties in every associated component. If the component is itself a composite object, then the properties of all components in the chain are available for reference. Property references may be prefixed with a range variable or an entity name for clarity. Figures 3 and 4 provide examples of the represented objects and the associated operations that are valid.

This environment was defined purely from the perspective of an effective user interface; i.e., what are the objects that a user may represent and manipulate? Concerns regarding the implementation were not investigated by this research. While the shortcomings of SQL as a query language have been well documented, we used this approach for two compelling reasons. First, all subjects had more than passing knowledge about SQL. They had worked on a significant design project using SQL as an interface. Familiarity with the basic structure of query formulation was a necessary condition in this experiment. Second, it does appear that SQL is becoming the de facto standard based on industry trends. Almost without exception, all new commercially available systems offer SQL as a standard query interface. Proposals for object-based systems seem to be moving in the direction of modifying SQL to accommodate new constructs as opposed to designing new interfaces from scratch.



Representation

Property list for Employee (supertype):
EMP#, NAME, ADDRESS

Property list for Engineer (subtype of Employee:)
SPECIALIZATION

Property list for Secretary (subtype of Employee):
TYPING_SPEED, SKILL_LEVEL

Property list for Trucker (subtype of Employee:
ENDORSEMENTS

Sample Operations:

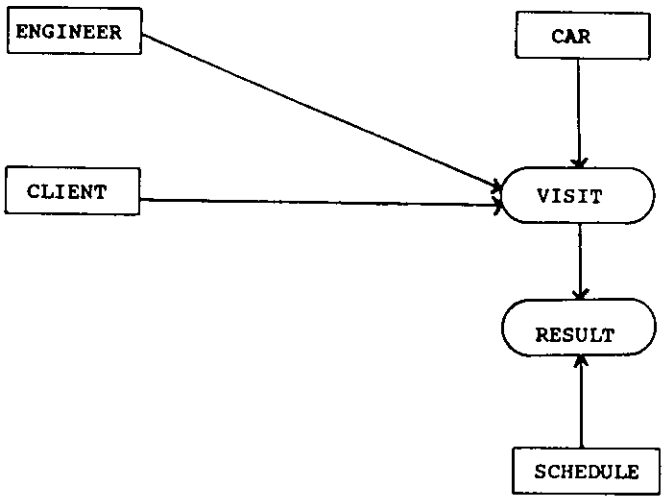
```
/*Retrieve all facts about all employees*/
SELECT *
FROM EMPLOYEE
/*Includes all properties of each of the subtypes*/

/*Retrieve all facts about engineers*/
SELECT *
FROM ENGINEER
/*Inherits relevant properties from EMPLOYEE*/

or

SELECT *
FROM EMPLOYEE
WHERE SUBTYPE='ENGINEER'
/*Note use of SUBTYPE as an implicit property*/
/*Includes all ENGINEER properties*/

/*What type of employee is Smith?*/
SELECT EMPLOYEE.SUBTYPE
FROM EMPLOYEE
WHERE NAME='Smith'
```

Figure 3. Modeling with Abstractions: Generalizations

3.2 Subjects

Given that the primary mode of data collection was by the use of process tracing, obtaining a large sample size was not a concern. The emphasis of this methodology, as suggested by its name, is on examining the process (of development and use, in this case) in detail; it is not on obtaining retrospective data for subsequent statistical analysis. Ten subjects were recruited to take part in the experiment. All ten had successfully completed a semester-long graduate level course in database manage-

ment. They were all familiar with the concepts of logical database design, normalization, relational systems, and the



Ellipses show composite objects.
Rectangles show entities.
Direction of the arcs is from the components to the composite object.

Representation:

VISIT: Ternary composite of ENGINEER, CAR, CLIENT
RESULT: Binary composite of VISIT, SCHEDULE
(Property lists are not shown for sake of brevity)

Operations:

```
/*Provide a list of cars*/
SELECT *
FROM CAR

/*Which cars were used on Engineer Jones' visit?*/
SELECT CAR.ID
FROM VISIT
WHERE ENGINEER.NAME='Jones'
/*Reference to the composite object VISIT in the source allows
references to attributes in all component objects*/

/*Provide a list of schedules drawn up for Ace Chemicals*/
SELECT SCHEDULE.*
FROM RESULT
WHERE CLIENT.NAME= 'Ace Chemicals'
/*Use of the composite chain. Reference to RESULT in the source
allows references to all components*/
/*in the chain of composites*/
```

Figure 4. Modeling with Abstractions: Composite Objects

structural and syntactic make up of SQL for the purposes of data representation and manipulation. All of them had implemented a fairly complex database system as part of the course requirement. This involved taking user requirements, developing a logical representation, implementing it using a commercially available (SQL based) database management system, and running queries against the database. None of the subjects could be considered experts in database design. Alternatively, none of them was a complete novice. Participation in the experiment was entirely voluntary and was not made part of any course requirement. They were informed that participation involved two training sessions of approximately three hours each followed by the experimental task. The whole process required a time commitment of about ten hours spread over a two week period. Three of the ten could not continue with the experiment due to time constraints. Seven of the ten subjects completed the whole sequence. The protocol obtained from one of the seven was unusable. It was later apparent that this individual's skill level was not comparable to that of the other six. Hence, this subject was dropped from further analysis. Our discussion, therefore, covers the output produced by the six remaining subjects.

3.3 Training

Two training sessions of approximately three hours each were conducted a week apart. In the first session, basic concepts of database design were reviewed. This was followed by introducing the notion of abstractions with specific examples. The examples helped illustrate what was permissible in terms of representation objects. Further, the benefits of using abstractions were illustrated by demonstrating the use of specific operators and their effects on the represented objects. Subjects were then given a sample problem to work with on their own time. The complexity of the sample problem indicated that it was not trivial. There were two supertype/subtype clusters, four binary composites, one ternary composite, and a composite chain involving five different entities at different levels of abstraction. Part of this composite chain exhibited an existence dependency, a condition in which the existence of one object depends on the existence of another. This condition usually complicates representation. The problem had to be modeled with abstractions and the set of tasks had to be handled by constructing queries that addressed the representation. The second training session was used to clarify and reiterate concepts using the sample problem as a context. Correct representation of the problem and the construction of SQL-like queries to handle the tasks were discussed to reinforce the concepts.

3.4 Experimental Task

The experimental task consisted of providing a narrative that described a problem. The complexity of this problem was very similar to that of the practice problem. Subjects were asked to read the narrative and represent the problem using the method of modeling with abstractions. Subjects were asked to verbalize their strategy as they proceeded to represent the problem. One of the authors played the role of the user to clarify the semantics of the problem for the subject. Subjects were often probed in a neutral manner during the course of this process. The

session was audio taped for later analysis. A set of trial problems were provided with the narrative so that the subjects could test the adequacy of their representations. Addressing the trial problems usually resulted in the revision of the initial representation. Subjects were required to transcribe their final representations on a sheet of paper so that the model at which they arrived at was available for further analysis along with the verbal transcripts. Finally, a set of eight tasks were given to the subject. They were asked to write down queries (using the SQL-like syntax described previously) in order to answer the problems contained in the tasks.

3.5 Coding of the Data

Based on the cognitive model that forms the basis for the study, we present the control strategies that subjects used to arrive at the final representation. By simple time stamping, we generated a transition graph for each subject that showed the following fundamental components of the process: the movement from one level of modeling abstraction to another, the points in time where non-abstraction based heuristics were employed to manage the complexity of the process, substantive transition points from one part of the problem to another, and the constraints that generated the need for employing a heuristic.

Five levels of abstraction were defined in order to categorize the verbal data from the audio tapes. This was done on the basis of the level at which the subject dealt with the problem at a particular point in time.

Level 0: Focus on individual properties of an object.

Level 1: Focus on basic entities. These were entities that had no subtypes; usually they were components of composite objects.

Level 2: Focus on a supertype/subtype cluster.

Level 3: Focus on basic composite objects.

Level 4: Focus on composite object chains, which may have included the entire model.

These levels were used to track the progression of the subjects (Figure 5). The use of non-abstraction based heuristics were coded as

A. Temporary transition to a different subproblem ("I'm not quite sure how to deal with this now....I'll come back to it later").

B. Testing for representation adequacy ("Let me see if this works").

C. Transition by association ("I see I've also got to take care of truckers here...so let me do that now while I'm thinking about it").

D. Planning ("How should I tackle this part?").

The two types of constraints that precipitated the need for employing the heuristics that were encoded were tool constraints and problem domain constraints. Specifically, dealing with the existence dependency in the problem was difficult for all the subjects and this is indicated on the graph.

4. RESULTS, OBSERVATIONS, AND DISCUSSION

4.1 Presentation of the Results

The first set of results that we present were obtained by analyzing the verbal transcripts obtained from each subject. Figure 5 shows the transition graphs for two of the subjects. Subject A is one whose performance is poor; Subject B's performance is good (relative to the performance of the six subjects).

The distribution of time spent at the various levels of abstraction by subject is presented in Table 1. The times shown are proportions of total time spent by the subject at each level on developing the representation.

Frequency counts of non-abstraction based heuristics by subject are shown in Table 2. The performance of the subjects was recorded in terms of the quality of the representations and the correctness of the queries that were constructed to address the tasks. With regard to the representation, we obtained counts of missing elements, mis-specified elements, and redundant elements. The queries were evaluated by counting the number of times incorrect statements were made about the target, the source object, and the predicate. Table 3 presents these results.

Table 1. Distribution of Time Across Abstraction Levels

Abstraction Level	Subject A	B	C	D	E	F
0	35.8	36.6	30.7	29.5	15.5	31.5
1	20.8	6.2	4.5	11.0	13.0	8.4
2	4.6	5.7	4.5	5.0	5.2	7.9
3	15.0	28.9	25.7	32.0	44.0	27.1
4	23.8	22.6	34.6	22.5	22.3	25.1

[Entries are percentages of total time spent by a subject.]

4.2 Observations

Our discussion essentially focuses on two important aspects of the results: what are the causes for transitions between abstraction levels and what can we learn from the use of heuristics.

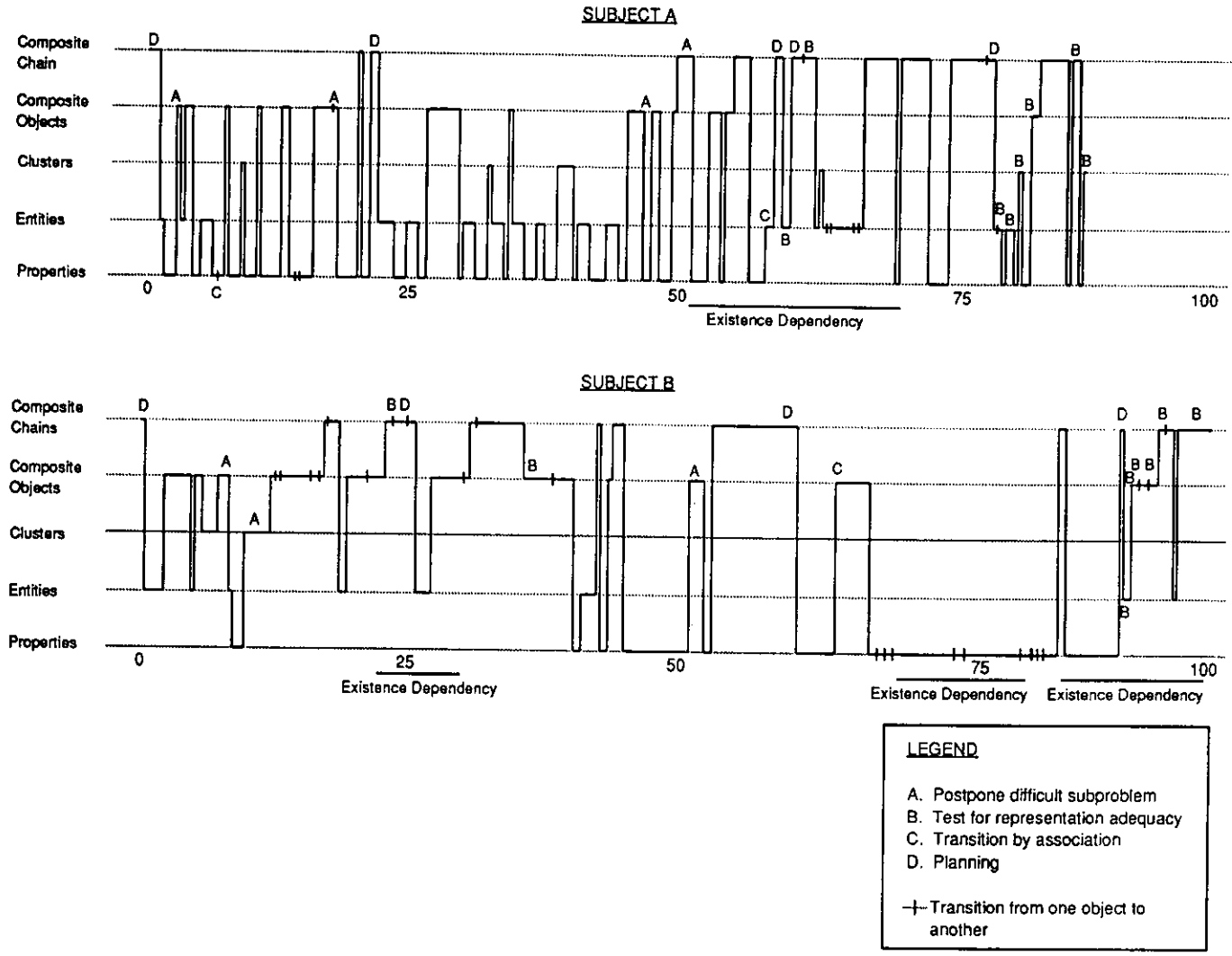


Figure 5. Transition Graphs for Subjects A and B

Table 2. Use of Non-Abstraction Based Heuristics

Heuristic	Subject A	B	C	D	E	F
Temporary transition to another subproblem	4	3	0	0	1	4
Final testing	8	8	8	8	8	8
Intermediate testing	0	2	0	0	3	1
Transition by association	2	1	3	1	1	2
Planning	5	4	1	4	4	4

[Entries in the table are frequency counts.]

Table 3. User Performance

	Subject A	B	C	D	E	F
Model Elements:						
Missing	2	0	3	2	1	0
Mis-specified	4	0	1	4	1	1
Redundant	0	1	0	0	0	1
Errors in Queries:						
Target	0	0	1	1	0	0
Source	5	3	4	2	2	0
Predicate	1	2	1	3	0	0

[Entries in the table are frequency counts.]

4.2.1 Transition Across Abstraction Levels

The transition graphs reveal some interesting patterns. Notice that the graphs primarily record the transition from one level of abstraction to another *in the order in which the transitions occur.* Several different patterns of transition are observable. Subject A's graph shows one pattern where there are rapid and huge transitions across several abstraction levels. This subject exhibits a strategy of frequently dropping down to the lowest level of abstraction to complement dealing with the problem at higher levels. Subject B exhibits a pattern whereby the problem is dealt with primarily at higher levels for almost half the total time before long bursts are used at the level of detail in the second half of the process. Similar graphs were generated for the other subjects. We show the two extremes in terms of performance here. Comparing these transitions to performance (see Table 3) reveals that working with the representation primarily at the lowest level of abstraction is unproductive (Subject A). The more successful performers appear to demonstrate that the abstractions are useful mechanisms. Subject F's strategy was to use the abstractions to guide the formulation of the representation in an orderly fashion. An alternative, yet beneficial, approach seems to be that exhibited by Subject E, where long bursts at higher abstraction levels served as a useful strategy to get a handle on the problem.

Although the patterns of using heuristics were different for each of the individual subjects, the reasoning offered by some of them appeared to be similar. Subjects left a subproblem because it was difficult to solve and wanted to first solve other subproblems. Subjects momentarily left a subproblem to augment a previously addressed subproblem because something in the first subproblem triggered an action on the second. The reasoning for testing was usually not mentioned explicitly, but recall that the experimental task included eight trial tasks that were positioned at the end of the narrative. It is therefore instructive to distinguish between the testing triggered by the trial tests and testing that was made independently of the trial tasks (see Table 2 for a breakdown of testing). Only three subjects took an initiative to test subproblems apart from the trial tests. Presumably testing is also a reaction to perceived complexity in the problem requirements.

The transition graphs can also be used to demonstrate the heuristics that trigger a shift in the level of abstraction. Although we did not use labels for these heuristics, the subjects explained on several occasions in the protocols why they shifted. Consider the following protocol of Subject F (at percentile 30):

- The subject was unclear about the composite object "site visit": "Cars are used by engineers on customer site visit. So there is hauling incident and site visits... those are not clear to me so I am just going to leave it alone until later."
- After a few moments the subject examines the properties of site visit and manages to represent the composite, building bottom up: "Properties of site visits, which are aggregates of something which I haven't worried about yet. Visit identifying number...date of visit are maintained with the site visits. An engineer will use any available car for the visit. Obviously a site visit is the coming together of an engineer, car and client....Site visit aggregate of: Engineer, Car, and Client."

4.2.2 Understanding the Heuristics

It is important to understand the nature of heuristics employed; specifically, how do they affect the data-modeling activities and when are they invoked. Several heuristics were identified that changed the order of representation and induced shifts in the level of abstraction. Problem-domain constraints (existence dependency in particular) and tool constraints that prompted several heuristics were also identified. Aside from requiring more time, these constraints appeared to prompt the subject to stop and plan a general course of action, leave difficult issues for later, and test specific subproblems. A fourth heuristic, jumping momentarily from one subproblem to another with the intention of immediately returning, was the result of an association (e.g., something similar should also have been done elsewhere) and not from an immediate difficulty. Why does the subject not finish the current subproblem and then attend the similar aspect in the other subproblem? This may be a way to eliminate the need to keep in mind a future activity – a reaction to constraints on cognitive ability. Problem-domain constraints also appeared to prompt a higher rate of shifting from one level of abstraction to another. When faced with a difficult issue at the higher levels of abstraction, subjects appeared to shift attention to lower and more concrete levels and then shift back to a higher level to put the details in the right perspective. Again, this is in concert with findings on general problem solving (Duncker 1945).

We expected to find a general heuristic of decomposing a problem into subproblems and dealing with the subproblems one at a time. Although this was never articulated in the subjects' verbal protocols, this heuristic is apparent from their actual behavior – it was used constantly. Furthermore, notice that some of the heuristics implicitly assume that the problem has been decomposed. More importantly, our general impression is that decomposition is relatively easy, but relating some of the decomposed subproblems proved to be more difficult. Future research should look more closely into how decomposition can be structured to facilitate better integration.

The strategy of testing throughout the modeling process instead of delaying testing to the end appears to pay dividends. Indeed, the three superior performers tested subproblems before they had the entire representation.

Moreover, this is also in accordance with explanations of the anchoring bias (Tversky and Kahneman 1974) and the premature conversion on a solution (Janis and Mann 1977).

4.3 Other Implications

This work suggests that continued research may result in practical implications for training and developing modeling aids (including computerized aids). Effective heuristics, such as early testing, should be encouraged in training system designers. With regard to other heuristics that are used effectively by some designers but not by others, it will be useful to identify the possible disadvantages of these heuristics. Trainers should be careful not to change individual styles, but it may be profitable to show the pitfalls of certain heuristics.

The proposed model and evidence suggests that data-modeling aids should facilitate an easy transition from one level of abstraction to another. For example, it may be effective to design computerized modeling aids that allow the user to simultaneously view multiple levels and thus be able to see the impact of changes at one level on lower or higher levels.

5. CONCLUSION

The main conclusion that we draw from this research is that end users can use semantic modeling effectively but do so employing a wide range of cognitive control strategies. It appears that the systematic use of an abstraction mechanism does indeed have payoffs. In other words, the abstractions must be used for more than mere subproblem identification. They must be used as a fundamental problem structuring mechanism in order to proficiently represent a complex problem. This is an aspect that must be built into the design of modeling aids, ensuring that the interfaces encourage the user to work with high levels of abstraction until the problem is fairly well represented before delving into details.

The use of the cognitive model in mapping user behavior proved to be useful. We identified specific heuristics and constraints used and encountered by the users and how, in turn, modeling activities and performance were affected. The results obtained from non-abstraction based heuristics have important implications for training end users. It seems that intermediate testing of representations results in higher quality models and more accurate queries. In this experiment, the relative merits of other heuristics did not emerge clearly. Future research may result in the revision of the cognitive model that we used to better identify appropriate heuristics. It is important to outline successful strategies so that, in training end users, these are stressed. Future research is also needed to develop the preliminary model presented in Figure 2. In particular, the link between heuristics and the use of the data model needs to be studied.

Finally, a comment on using process tracing as a methodology. The emphasis on the process at a micro level of behavior necessitated the use of this approach. While this may be appropriate at an exploratory stage, careful experimentation may be warranted to study the impact of different training methods and the effect of different types of constraints on user performance. Ultimately, we wish to encourage end users to engage in application development using semantic modeling. It is important to recognize that such development is a secondary activity as far as end users are concerned; making them experts is not the objective.

6. REFERENCES

Ballay, J. M. "An Experimental View of the Design Process." In W. B. Rouse and K. R. Boff (eds.) *System Design*. New York: North-Holland, 1987.

Batini, C., and Ceri, S. "Database Design: Methodologies, Tools and Environments." *Proceedings of the ACM-SIGMOD Conference*, 1985, pp. 148-150.

Batra, D.; Hoffer, J. A.; and Bostrom, R. P. "Comparing Representations with Relational and EER Models." *Communications of the ACM*, Volume 33, Number 2, February 1990, pp. 126-139.

Borgida, A.; Greenspan, S.; and Mylopoulos, J. "Knowledge Representation as the Basis for Requirements Specification." *Computer*, Volume 18, 1985, pp. 82-91.

Bouwman, M. "The Use of Accounting Information: Expert Versus Novice Behavior." In G. Ungson and D. Braunstein (eds.), *Decision Making: An Interdisciplinary Inquiry*. Boston: Kent Publishing Company, 1982, pp. 134-167.

Brodie, M. L. "On the Development of Data Models." In M. L. Brodie, J. Mylopolous, and J. W. Schmidt (eds.), *On Conceptual Modeling*. New York: Springer Verlag, 1984.

Chen, P. P. "The Entity Relationship Model: Toward a Unified View of Data." *ACM Transactions of Database Systems*, Volume 1, Number 1, March 1976, pp. 9-36.

Duncker, K. "On Problem Solving." In J. F. Dashiell (ed.), *Psychological Monographs*. Washington, DC: The American Psychological Association, 1945.

Elmasri, R.; Hevner, A.; and Weeldreyer, J. "The Category Concept: An Extension to the Entity-Relationship Model." *Data and Knowledge Engineering*, 1985, Volume 1, Number 1, pp. 75-116.

Ericsson, K. A., and Simon, H. A. "Verbal Reports as Data." *Psychological Review*, Volume 87, Volume 3, May 1980, pp. 215-251.

Greeno, J. G., and Simon, H. A. *Problem Solving and Reasoning*. Pittsburgh: University of Pittsburgh, LRDC, 1984.

Guindon, R. "A Model of Cognitive Processes in Software Design: An Analysis of Breakdowns in Early Design Activities by Individuals." *MCC Technical Report STP-283-87*, 1987.

Hammer, M., and McLeod, D. "Database Descriptions with SDM: A Semantic Database Model." *ACM Transactions on Database Systems*, Volume 6, Volume 3, September 1981, pp. 351-386.

Janis, I. L., and Mann, L. *Decision Making*. New York: Free Press, 1977.

Jarvenpaa, S. L., and Machesky, J. J. "Data Analysis and Learning: An Experimental Study of Data Modeling Tools." *International Journal of Man-Machine Studies*, Volume 31, 1989, pp. 367-391.

Jeffries, R.; Turner, A.; Polson, P. G.; and Atwood, M. E. *The Processes Involved in Designing Software*. Englewood, Colorado: Science Applications, 1980.

Kent, W. "Limitations of the Record Based Information Models." *ACM Transactions on Database Systems*, Volume 4, Number 1, March 1979, pp. 107-131.

Mantha, R. W. "Data Flow and Data Structure Modeling for Database Requirements Determination: A Comparative Study." *MIS Quarterly*, December 1987, pp. 531-545.

Munro, M. C.; Huff, S. L.; and Moore, C. "Expansion and Control of End User Computing." *Journal of MIS*, Volume 4, Number 3, Winter 1987-88, pp. 5-27.

Newell, A., and Simon, H. A. *Human Problem Solving*. Englewood Cliffs, New Jersey: Prentice-Hall, 1972.

Rasmussen, J. "Skills, Rules, Knowledge: Signals, Signs, and Symbols and Other Distinctions in Human Performance Models." *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13 (1983), pp. 257-269.

Reisner, P. "Human Factor Studies of Database Query Languages." *ACM Computing Surveys*, Volume 13, Number 1, March 1981, pp. 13-31.

Reitman, J. S. "Skilled Perception in GO: Deducing Memory Structures from Interresponse Times." *Cognitive Psychology*, 1976, Volume 8, pp. 336-356.

Rockart, J. F., and Flannery, L. S. "The Management of End User Computing." In C. A. Ross (ed.), *Proceedings of the Second International Conference on Information Systems*, Cambridge, Massachusetts, December 1981, pp. 351-363.

Rouse, W. B. "On the Value of Information in System Design: A Framework for Understanding and Aiding Designers." *Information Processing and Management*, Volume 22, 1986, pp. 217-228.

Rouse W. B., and Boff K. R. (eds.). *System Design*. New York: North-Holland, 1987.

Schmid, H. A., and Swenson, J. R. "On the Semantics of the Relational Data Model." *Proceedings of the 1975 SIGMOD Conference*, San Jose, California, 1975, pp. 211-213.

Smith, J. M., and Smith, D. C. P. "Database Abstractions: Aggregations and Generalizations." *ACM Transactions on Database Systems*, Volume 2, Number 2, June 1977, pp. 105-133.

Stefik, M., and Bobrow, D. G. "Object Oriented Programming: Themes and Variations." *The AI Magazine*, January 1986, pp. 40-62.

Stonebraker, M. *Readings in Database Systems*. Morgan Kauffman Publishers, 1988, pp. 369-372.

Teorey, T.; Yang, D.; and Fry, J. F. "A Logical Design Methodology for Relational Databases Using the Extended Entity Relationship Model." *ACM Computing Surveys*, Volume 18, Number 2, June 1986, pp. 197-222.

Todd, P., and Benbasat I. "Process Tracing Methods in Decision Support System Research: Exploring the Black Box." *MIS Quarterly*, December 1987, pp. 493-512.

Tsichritzis, D. C., and Lochovsky, F. H. *Data Models*. Englewood Cliffs, New Jersey: Prentice Hall, 1982.

Tversky, A., and Kahneman, D. "Judgment Under Uncertainty: Heuristics and Biases." *Science*, Volume 185, 1974, pp. 1124-1131.

Vassiliou, Y.; Jarke, M.; Stohr, E. A.; Turner, J. A.; and White, N. H. "Natural Language for Database Queries: A Laboratory Study." *MIS Quarterly*, December 1983, pp. 47-61.

Zloof, M. M. "Query By Example: A Database Language." *IBM Systems Journal*, Volume 16, Number 4, 1975, pp. 324-343.

6. ENDNOTES

1. Note that this definition also allows us to represent unary relationships such as the bill of materials problem as well as to build dependency chains where one composite object has another as one of its details.