Exploring the Difficulties of Learning Object-Oriented Techniques

STEVEN D. SHEETZ, GRETCHEN IRWIN, DAVID P. TEGARDEN,
H. JAMES NELSON, AND DAVID E. MONARCHI

STEVEN D. SHEETZ is an Assistant Professor of Accounting Information Systems at the Pamplin College of Business at Virginia Tech. He received his Ph.D. in information systems from the University of Colorado. His research interests include the cognitive complexity of developing information systems, learning and use of object-oriented development techniques, participatory design, and the application of group support systems technology. He has published articles in *Decision Support Systems, Journal of Management Information Systems*, and *Object-Oriented Systems*.

GRETCHEN IRWIN is a lecturer at the University of Auckland, New Zealand. She received her Ph.D. and M.S. in information systems from the University of Colorado, Boulder. Dr. Irwin's research interests include the cognitive aspects of software reuse, the cognitive aspects of object-oriented systems development, and the teaching and learning of systems development methods and technology.

DAVID P. TEGARDEN is an Assistant Professor in Accounting Information Systems in the Pamplin College of Business and a Fellow in the Center for Human–Computer Interaction at Virginia Tech. He received a Ph.D. in information systems from the University of Colorado. His research emphases are in the areas of object-oriented software engineering, the development of cognitively based software complexity measurements, psychology of programming, and the application of virtual reality and visualization technologies in business. He has published articles in *Object-Oriented Systems, Journal of Management Information Systems, Decision Support Systems*, and *Software Quality Journal*.

H. JAMES NELSON is a Lecturer of Information Systems at the University of Kansas. He received his B.S. in computer science from California Polytechnic State University, San Luis Obispo, and his M.S in information systems from the University of Colorado, Boulder. He is currently ABD in Information Systems from the University of Colorado. His research interests include developing theoretically grounded models and metrics for business processes and investigating the problems people have shifting to emerging technologies.

DAVID E. MONARCHI is a Professor of Information Systems at the University of Colorado, Boulder. His research interests include conceptual modeling and the design of object-oriented systems.

ABSTRACT: Object-oriented (OO) analysis, design, and programming techniques have emerged as potential solutions to the software crisis. However, learning OO techniques can be a difficult process. This study investigates students' perceptions of the diffi-

Journal of Management Information Systems / Fall 1997, Vol. 14, No. 2, pp. 103–131

0742–1222 / 1997 $9.50 + 0.00

culties in learning and using OO techniques. Two groups of students who had recently completed a sixteen-week course on OO systems development participated in the study. Cognitive mapping techniques implemented with group support system (GSS) technology were used to gather information on the students' perceptions. The groups used the GSS to identify what was difficult about learning and using OO techniques, classify these concepts into categories, rank the relative importance of each category, and determine the relationships among the categories. Importance rankings of the categories show that learning basic object concepts was most important to and most difficult for the students, followed by design issues, and programming techniques. Composite cognitive maps of the shared perceptions of group members suggest that educators and industry trainers can reduce the difficulties of learning OO concepts by teaching simplified methodologies and controlling difficulties of the programming environment.

KEY WORDS AND PHRASES: cognitive mapping, group support systems, learning, object-oriented systems.

OBJECT-ORIENTED (OO) SYSTEMS DEVELOPMENT TECHNIQUES (e.g., OO analysis and design methods, OO programming languages and tools) have emerged as useful ways to address the development needs of information systems (IS) groups and organizations [3, 8, 9, 24, 33]. Many IS professionals and researchers consider OO techniques a potential solution to the perennial problem of delivering timely and cost-effective systems that meet users' needs [25]. Leading journals, such as the *Communications of the ACM* [10, 11, 12], and the popular press (e.g., *Business Week* [37]), are also promoting object orientation as a potential and partial solution for what currently ails software development.

The OO approach to systems development uses problem-oriented representations that attempt to reduce the cognitive distance between problem-domain constructs and computational structures [3, 8, 24, 32, 33]. Proponents of this approach claim that OO development may lead to higher-quality, more understandable, and more maintainable systems than traditional structured techniques [29, 32]. However, learning and using OO techniques have not proven to be easy, and research is needed to improve understanding of the learning process [31, 34, 38]. Identifying learners' difficulties with OO techniques can provide a basis for enhancing academic teaching and industry training in this area.

The focus of this study is on identifying and understanding the issues that contribute to the difficulties of learning and using OO techniques. We examine the perceptions of two groups of students who recently completed a sixteen-week course on OO systems development. We used a cognitive mapping approach implemented with group support system (GSS) technology to capture the students' perspectives on the difficulties of learning and using OO techniques. Our results show that learning basic object concepts was most important to and most difficult for the students, followed by design issues and programming techniques. Composite cognitive maps of the shared perceptions of group members suggest that educators and industry trainers can reduce

the difficulties of learning OO concepts by teaching simplified methodologies and controlling difficulties of the programming environment.

Research Method

COGNITIVE MAPPING IS A SET OF TECHNIQUES for studying and recording people's perceptions of the world around them. Cognitive mapping techniques have been used to investigate diverse areas, including politicians' beliefs [2], musician's views of performances [7], information systems requirements [27], knowledge acquisition [20], distributed artificial intelligence [40], and analysis of cognitive maps as decision-making aids [5, 17, 21, 23, 39]. Cognitive mapping usually begins by asking participants a question to elicit their perceptions. The questions in many studies are open-ended and focus on exploring the perceptions of individuals in a particular environment [2, 17, 23]. This use of cognitive mapping is consistent with the use of cognitive mapping in our study, where an open-ended question was used to explore students' perceptions of their OO learning experience.

An analysis of cognitive mapping techniques [2, 7, 17, 23] shows that most techniques consist of three major activities: (1) eliciting concepts, (2) refining concepts, and (3) identifying relationships between concepts. The Self-Q technique [4, 5, 6, 7, 39] is a specific cognitive mapping technique designed to reduce researcher bias by obtaining directly from the participants both the concepts and the relationships between the concepts. The Self-Q technique consists of the following steps:

1. Self-questioning to elicit concepts associated with the domain;
2. Grouping the concepts from self-questioning into categories;
3. Developing or defining the categories;
4. Ranking the categories by importance;
5. Determining relationships between categories.

In the fifth step of the Self-Q technique, each participant creates a cause map—a special type of cognitive map that connects concepts to each other with unidirectional arrows, where an arrow represents an assertion that one concept affects another.

We used a group support system (GSS) to implement the cognitive mapping procedures similar to those defined in the Self-Q technique. Several studies suggest using GSS technology to implement the procedures of a research methodology, as opposed to using a GSS as a decision-making tool [1, 15, 22, 36]. These studies indicate that GSS is useful for recording the views of participants and controlling data collection procedures. We had a threefold purpose in using the technology: (1) to facilitate the participants' tasks, (2) to expedite data collection, and (3) to reduce researcher-introduced bias. Researcher-introduced bias was reduced by having the groups, rather than the researcher/facilitator, identify the concepts and categories. The facilitator neither directed nor influenced the content; he provided only procedural guidance by keeping track of time and introducing new activities during the session. Throughout the process, the facilitator followed prewritten instructions and did not participate in discussion or determination of concepts and categories.

Study Environment and Participants

The study was conducted at a major western university that has GSS technology and a course in OO systems development. It provided a setting that had both the needed technology and participants familiar with learning OO systems development techniques.

The participants in the GSS session were seven undergraduate and graduate students pursuing degrees in information systems. Participants had recently completed a sixteen-week course that included building a nontrivial OO prototype. The course required students working in small groups to develop an OO analysis and design and to implement the design in Smalltalk V/Windows (an OO programming language and environment by Digitalk). The use of students in this study was appropriate because our intention was to explore what novices find difficult about learning and using OO techniques. These students were near the end of their degree programs and had considerable experience taking courses and learning systems development techniques. They were OO novices who nonetheless had prior experience learning other, more traditional, systems analysis, design, and programming approaches. Thus, they were able to reflect on their experience in the OO course and to compare it with their experiences learning other systems development techniques.

Table 1 describes the systems development experience of the participants at the time of the study. There were four participants in group 1 (participants 1–4) and three participants in group 2 (participants 5–7).

Data were collected using the VisionQuest software of Collaborative Technologies Corporation. The environment is configured as a teaching room with eight straight rows of five 386 PCs on an LAN, with a public viewing screen. VisionQuest provides the ability to structure group activities. In this case, the structure implements the cognitive mapping procedures necessary to reveal the participants' perceptions of the difficulties in using OO techniques.

Group Support System (GSS) Sessions

Table 2 outlines the activities performed during the GSS sessions and shows the time elapsed, VisionQuest tool used, and data collected for each activity. The data collection approach used in this study is consistent with that used by Sheetz et al. [36]. Each session lasted three to four hours, including a ten-minute break roughly halfway through the session.

Two GSS sessions were conducted, one for each group. Participation in the study was voluntary. One of the authors contacted students roughly one month after they had completed the OO course. The students were invited to participate and told that the results would help the researchers better understand OO learning difficulties and improve the OO course. Students were paid for their participation. Seven students volunteered; they were split into two groups. Group membership was determined according to the participants' time availability.

Participant input was captured by the logging program of the VisionQuest software. In addition, the session was audiotaped to provide additional documentation of the

Table 1. Profile of Participants

Group	Participant no.	Programming experience	Management experience	No. of programming languages	Analysis experience
1	1	18	0	3	8
	2	12	0	1	20
	3	36	0	2	10
	4	48	0	2	10
2	5	36	0	5	12
	6	20	4	3	14
	7	8	3	4	4

Experience includes coursework and/or professional work and is reported in months.

timing of exercises, category definitions, and the activities of the facilitator. Participants were asked to respond (using the GSS) to the following statement:

> We are interested in understanding the process of using object-oriented systems analysis, design, and programming techniques. It would be useful to know what you believe is difficult to understand about using these techniques.

This statement was intentionally broad in scope, using the term "object-oriented techniques" to cover analysis, design, and/or programming representations and tools. The students had been exposed to many aspects of object orientation in the OO course [28], from the OO "mindset" and concepts, to analysis and design methods and tools, to the Smalltalk programming environment. Since we had no reason *a priori* to believe that one of these areas was particularly easy or difficult for the students, we did not want to direct them to a particular aspect of object orientation. This broad definition reduces researchers' bias introduced into the study, but it also leaves room for multiple interpretations and can lead to a large variation in results. We felt that this tradeoff was acceptable at this point in our research.

Results

THE GSS SESSIONS CAPTURED PARTICIPANTS' PERCEPTIONS at the concept, category, and relationship (cause map) level.

Concept Level

Concept-level data results from the concept identification procedure (see Table 2). Concepts are the "things" that participants perceive to be difficult about learning and using OO techniques. Participants identified a total of 105 concepts. Group 1, with four members, identified 65 concepts; Group 2, with three members, identified 40 concepts.[1]

The concepts can be loosely grouped into four areas, based on the content and structure of the course. These areas are: (1) OO concepts (i.e., terminology and foundations); (2) modeling (OO analysis and design activities and representations);

Table 2. Steps in the GSS Cognitive Mapping Approach

Activity	Description	Duration (min)	VisionQuest tool
1. Elicit concepts			
Introduction	Describe system use; sign informed consent form; discuss framing statement and stall diagram.	30	Comment Cards and Brainwriting
Concept identification	Elicit characteristics, concepts, and/or issues that contribute to (increase or decrease) the difficulties of using OO techniques.	30	Brainwriting
2. Identify categories			
Category identification	Elicit categories to group concepts by similarity; agree on category definitions and names.	30	Verbal for participants, Facilitator uses Compactor
(Break)		10	
3. Classify concepts			
Concept categorizations	Classify the concepts into (top) ten categories.	45	Compactor
Discussion	Discuss classification.	10	Compactor
4. Rank categories			
Category rating step 1	Rate each category on a 7-point scale, from important to extremely important.	5	Rating
Discussion	Group means are calculated and discussed.	15	Verbal Discussion
Category rating step 2	Same as category rating step 1.	5	Rating
Discussion	Group means are calculated and discussed.	15	Verbal Discussion
Category rating step 3	Same as category rating step 1.	5	Rating
5. Define relationships			
Identify relationships	Each individual is given a comparison matrix of the categories in a rating task. Rate each comparison on a scale of –3 to +3, from strong negative influence to strong positive influence of one category on another category.	30	Scoring
Debriefing	Enter comments on the GSS procedures and results of the process.	10	Comment Cards

(3) programming (specifically, Smalltalk); and (4) other (i.e., topics either not covered in the course or covered in a cursory manner depending on available time and/or student questions). One of the authors (also a teaching assistant in the course) classified each concept into one of these areas without knowledge of the participants' placements of the concept into the categories they defined. The results are summarized in Table 3.

Table 3. Summary of Concept Identification Results

	Group 1		Group 2	
Topic area	No. (%) concepts	Examples	No. (%) concepts	Examples
OO concepts and terms	10 (15)	Concepts of overloading and polymorphism; Tons of new vocabulary, it is hard to keep it all straight; Understanding class versus instance methods; Differences between generalization and abstraction.	10 (25)	Different use of terms by different experts; Encapsulation versus information hiding versus overloading; Distinguishing between class and instance attributes.
Modeling	18 (28)	The modeling and diagramming techniques—is there a standard; Creating a model that is semantically correct; Transforming something intangible in the real world into a somewhat tangible model.	11 (27.5)	Designing and diagramming—no clear methodologies to use; Translating from concepts to implementation; Defining abstract ideas as "objects."
Programming	27 (42)	What methods have already been written and finding them; So many classes and methods that you can't get a good feel for which ones to use; Finding the right object to inherit from. Understanding one message without understanding all embedded messages; Mapping the flow of execution in an OO program.	8 (20)	Finding out if a class exists and locating it; Reuse can be very difficult and time consuming unless you know the library of classes. Understanding message passing.
Other	10 (15)	Why functions in procedural [languages] were so different from objects in object [oriented languages]; the code is harder to follow than traditional code like COBOL.	11 (27.5)	Programming with structured programming frame of mind; transition to OO not as easy as it seems; trying to distinguish objects from program modules; syntax of OO language very different from structured programming language. Integrating 2 people's work can take more coordination than in structured approaches; Time needed to gain experience using the OO language.

Both groups identified concepts within each area that were difficult. From the concept examples shown in Table 3, it is evident that the groups were thinking similarly about these issues. Learning OO terms, managing the large class library in Smalltalk, grasping the distributed nature of OO programming, and making the shift from procedural to OO thinking were difficulties identified by both groups.

Despite these similarities, there were also areas where the groups differed in the amount and nature of the concepts generated. Group 1, for example, identified considerably more programming-specific issues than did group 2 (42 percent versus 20 percent of all concepts, respectively). Group 1 identified more difficulties with respect to programming than with any other area, while the programming area for group 2 contained the fewest difficulties. Group 2 identified more high-level difficulties than did group 1, particularly with respect to "buying in" to the OO paradigm. For example, group 2 identified issues associated with obtaining management support for and investment in OO technology and wading through the media hype to determine how, where, and why OO techniques should be used.

To determine whether the group similarities and differences were due to one or two members dominating each session, we examined the contribution of individuals within each group. Table 4 shows the number of concepts generated by each participant.

All participants contributed to the concept identification task, although not all contributed equally. Each group had two members who contributed over 70 percent of all concepts—P1 and P3 in group 1, and P6 and P7 in group 2. P6 and P7 identified all of the "management-related" issues (e.g., buying into the OO paradigm, managing OO projects, startup costs of switching paradigms) in group 2. P6 and P7 were also the only two participants in the study who had any management experience (see Table 1). This may account for some of the difference in focus between the groups, since management issues were not explicitly addressed in the course content.

The difference in emphasis on programming issues also may be explained partially by the different programming backgrounds between the groups. Group 2 members knew, on average, four programming languages, while group 1 members knew only two programming languages on average. All three participants in group 2 knew C; P6 and P7 in group 2 had also tried C++, an OO programming language. Participants in group 1 had prior training in only structured programming languages (COBOL, Basic, or Pascal). This difference in programming language experience may have contributed to the different emphasis each group placed on programming issues. For example, the additional programming languages known in group 2 may have given participants greater confidence, and hence increased their self-efficacy, in their ability to learn a new programming language such as Smalltalk. This interpretation would be consistent with prior research on computer training that shows a positive relationship between individuals' self-efficacy and performance working with new computer applications after completing a training course [13]. If group 2 members were more confident in their ability to learn Smalltalk, this confidence may have enabled them to focus on other nonprogramming issues that make learning OO techniques difficult.

To summarize the concept-level results, both groups identified difficulties with respect to analysis and design techniques, programming, and concept/terminology

Table 4. Number and Percentage of Concepts Identified by Each Participant

Participant in group 1	No. (%) of concepts identified	Participant in group 2	No. (%) of concepts identified
1	20 (31)	5	10 (25)
2	9 (14)	6	18 (45)
3	26 (40)	7	12 (30)
4	10 (15)		
	65 (100)		40 (100)

issues. In other words, some aspects of all topics covered in the OO course were difficult for the participants. This is not surprising given the view that OO thinking and OO development require a "paradigm shift" from traditional structured system development [26]. In both groups, 25–30 percent of the concepts relate to similar OO analysis and design difficulties, suggesting agreement on the difficulties associated with OO modeling and diagramming techniques. The most noticeable difference between the groups is the relative emphasis group 1 placed on programming issues and the relative emphasis group 2 placed on higher-level issues such as management commitment and training. Again, this difference may be explained partially by the different education and work experience of members in each group.

Category Level

Category-level data results from the three activities following concept identification: category definition, concept classification, and category ranking (see Table 2).

Category Definition and Concept Classification

Category identification resulted in the category names and definitions presented in Table 5. Each group identified ten categories, which was the maximum number of categories permitted by the VisionQuest GSS. Group 1's categories were Smalltalk Language (ST), Smalltalk Front End (FE), Analysis and Design Techniques (AD), Object Concepts (OC), Object Interaction (OI), Working with Graphical User Interfaces (WG), Event Programming (EP), Choosing Words (CW), Lack of Standards (LS), and Organizing Information Load (IL). Group 2's categories were Reuse (RE), Paradigm Shift (PS), Modeling Reality (MR), Terms (T), Design Issues (DI), Inheritance (IN), Message Passing (MP), Design Tools (DT), Commitment and Investment (CI), and Generalization/Specialization and Aggregation (GS).

Once the categories were identified and defined, participants classified each concept into one of the categories. To determine the cohesiveness of the categories from the participants' perspectives, we calculated the level of group agreement for each concept classified. Most concepts—69 percent (72/105)—were classified consistently (i.e., placed in the same category) by a majority of the participants. In group 1, 60 percent

(39/65) of the concepts were classified consistently by three or four group members. In group 2, 83 percent (33/40) of the concepts were classified consistently by two or three group members. The overall K, Kappa, for all concept classification is 0.48 for group 1 and 0.58 for group 2, indicating a moderate level of agreement [35] that is statistically significant (group 1, $Z = 22.00, p < 0.001$; group 2, $Z = 17.02, p < 0.001$). This agreement suggests that participants had a shared understanding of the categories.

Individual group member contribution to the classification activity is summarized in Table 6, which shows which categories each participant used most often for the concepts he or she identified earlier in the GSS session. Here again, we see similarities and differences between the groups. Programming-related categories (e.g., Smalltalk Language, Smalltalk Front-End) were heavily used in group 1, which was not the case for group 2. Analysis and design categories (e.g., Analysis and Design Techniques, Choosing Words, Design Issues, Modeling Reality) were used in both groups.

The similarity and differences in the concepts generated by the two groups were discussed earlier. Given the amount of overlap in the concepts identified, it is not surprising that there is also considerable overlap in the categories identified by each group. The similarity in categories may not be immediately obvious from the category names, but it becomes more clear when we consider the category definitions and the concepts grouped into them. Figure 1 presents a matrix where the rows represent group 1's categories and the columns represent group 2's categories. An "X" in a cell indicates that the two corresponding categories overlap. Overlap was determined using a content analysis approach that identified common words and meanings in the definitions of categories and text of the concepts placed in the categories. Two of the authors independently completed the comparisons, then combined their responses to produce this matrix.

The overlap in category definitions implies that students in both groups have similar views of the difficulties of using OO techniques. For example, group 1's Smalltalk Front End (FE) category overlaps with group 2's Reuse (RE) category. Both of these categories deal with using existing classes such as those in Smalltalk's class library (see the definitions in Table 5). The difference between the categories is reflected in their names—the Smalltalk Front End category groups the class library with other Smalltalk tools (e.g., the debugger), while the Reuse category groups class library issues with broader reuse issues (e.g., reuse of other people's ideas). In both cases, however, becoming familiar with and properly utilizing the class library were an important concern. Other common areas of difficulty for the groups were clarifying and understanding concepts such as generalization/specialization, inheritance, aggregation, and message passing (group 1's categories Object Concepts and Object Interaction; group 2's categories Generalization/Specialization/Aggregation, Inheritance, Terms, and Message Passing).

Design categories and OO analysis were identified and used as well. The difficulties here fall into two areas—conceptually modeling a domain using OO concepts and specifically representing that model using OO analysis and design methods and notations. The groups perceived capturing a semantically rich and valid model of the world as difficult (see group 1's Choosing Words and Analysis and Design Techniques

Table 5. Category Names and Definitions Provided

Category name (code)	Definition
	A. Group 1
Smalltalk Language (ST)	Syntax issues particular to Smalltalk such as capitalization and punctuation.
Smalltalk Front End (FE)	The program development environment including the class library. Finding things and moving around. Using the debugger and other parts.
Analysis and Design Techniques (AD)	Object diagramming and representing the problem space versus the (solution) space. Learning OO design tools as opposed to the programming language.
Object Concepts (OC)	The terminology of OO techniques (e.g., abstraction, generalization, polymorphism).
Object Interaction (OI)	The interaction of objects including message passing.
Working with Graphical User Interfaces (WG)	Using graphical user interfaces to implement OO systems.
Event Programming (EP)	Understanding and using event-driven programming techniques.
Choosing Words (CW)	Choosing the right words to semantically represent the problem, to discuss objects, and to name objects and methods.
Lack of Standards (LS)	No standards for analysis, diagramming, design, and programming.
Organizing Information Load (IL)	Organizing the bulk of information associated with object-orientation, including OO concepts, programming language, and graphical user interfaces.

categories and group 2's Modeling Reality category). They also felt that the abundance and use of specific OO analysis and design diagrams and tools added to the difficulty of OO techniques (see group 1's Analysis and Design Techniques and Lack of Standards categories and group 2's Design Tools categories).

Figure 1 also demonstrates where the two groups differed in terms of categories. Shaded columns and rows represent categories identified by one group that do not overlap with the other group's categories. Again, group 1 was more focused on programming issues, and group 2 was more focused on higher-level issues. For example, group 1 identified five categories—Smalltalk Language, Smalltalk Front End, Working with Graphical User Interfaces, Event Programming, and Object Interaction—that involve programming-specific (Smalltalk specific and Microsoft Windows specific) concepts. Three of these categories have no counterpart in group 2's categories. On the other hand, group 2 focused more on high-level, management or organizational issues than group 1 did. Group 2 identified two categories, Commitment and Investment and Paradigm Shift, that have no counterpart in group 1's categories. These two categories address issues of management "buy-in" to OO techniques and changing the way programmers think about developing systems.

The results of the category definition and concept classification activities reinforce

Table 5. *Continued*

b. Group 2

Reuse (RE)	Adapting existing code and classes. Using other peoples' ideas.
Paradigm Shift (PS)	Moving from thinking in terms of structured programming to OO concepts.
Modeling Reality (MR)	The process of mapping concepts in reality into objects in the system. Breaking reality up into pieces. Defining the boundaries of the problem and solution space.
Terms (T)	Understanding and reconciling the various sets of terminology. The labels as separated from the concepts and their meanings.
Design Issues (DI)	Deciding how to structure a program. Classes and how they relate to each other. Organizing the class hierarchy — what goes where, inheritance, etc.
Inheritance (IN)	How attributes and behavior waterfall through the [class] hierarchy.
Message Passing (MP)	The mechanism that allows objects to communicate and function as a system.
Design Tools (DT)	The methodologies and techniques for OO analysis and design or lack thereof.
Commitment and Investment (CI)	Commitment and investment required to trust, learn and implement OO concepts. Time and effort of individuals; money and commitment at the organization level. Belief that the investment will pay off.
Generalization, Specialization, and Aggregation (GS)	The definitions of and distinguishing between the concepts of generalization, aggregation, specialization, and abstraction.

the findings from the concept identification activity discussed earlier. This suggests that the categories adequately represent the beliefs, facts, and issues contained in the concepts.

Category Ranking

The preceding discussion focused on the definition and number of categories identified by each group. To understand the relative importance of issues from the participants' perspectives, a category ranking activity was performed in the GSS sessions (see Table 2). Category importance ratings indicate the participants' perceptions of the contribution of the categories to the difficulties of using OO techniques. The higher the rating, the more the category contributes to the difficulty of learning OO techniques. Table 7 presents the mean category importance ratings and the level of agreement, the Kendall coefficient of concordance [35], attained by the participants for each of three rating steps.

As Table 7 shows, the participants in group 1 reached a moderate (0.50) level of agreement by the third rating step. Rankings of the categories were essentially the same over all three rating steps. Group 2's category rankings were not as stable over the three ratings steps as were those of the first group. The Generalization/Specialization category fell in perceived importance, while the Paradigm Shift category increased in importance. The Inheritance and Terms categories also traded places from the second rating step to the third rating step. Overall, the group reached a high (0.73)

	RE	PS	MR	T	DI	IN	MP	DT	CI	GS
ST										
FE	X				X	X				
AD			X		X			X		
OC				X		X				X
OI							X			
WG										
EP										
CW			X							
LS								X		
IL										

Figure 1. Overlap In Group 1 and Group 2 Category Definitions
Group 1 Category Codes (rows): ST = Smalltalk Language; FE = Smalltalk front End; AD = Analysis and Design Techniques; OC = Object Concepts; OI = Object Interaction; WG = Working with Graphical User Interfaces; EP = Event Programming; CW = Choosing Words; LS = Lack of Standards; IL = Organization Information Load.
Group 2 Categories (columns): RE = Reuse; PS = Paradigm Shift; MR = Modeling Reality; T = Terms; DI = Design Issues; IN = Inheritance; MP = Message Passing; DT = Design Tools; CI = Commitment and Investment; GS = Generalization, Specialization, and Aggregation.
Note: An "X" indicates overlap in the category definitions for the corresponding row and column. Shaded rows and columns indicate categories identified by only one group.

Table 6. Category Usage by Participant (Only for Concepts Identified by the Participant)

Participant	No. of concepts identified	Categories used by a participant to classify 2 or more of the concepts he or she generated
		Group 1
P1	20	Smalltalk Language (6), Analysis and Design Techniques (5), Object Concepts (5)
P2	9	Smalltalk Front End (3), Object Interaction (2)
P3	26	Smalltalk Language (4), Smalltalk Front End (4), Analysis and Design Techniques (3), Object Interaction (3), Event Programming (3), Choosing Words (3)
P4	10	Smalltalk Language (3), Object Concepts (3)
		Group 2
P5	10	Design Issues (3), Paradigm Shift (2)
P6	18	Generalization and Specialization (4), Modeling Reality (3), Design Issues (3), Reuse (2)
P7	12	Paradigm Shift (5), Modeling Reality (3)

Numbers in parentheses are the number of concepts generated by the participant and assigned to that category by the participant.

level of agreement by the third rating step.

Group 1 ranked Object Concepts and Analysis and Design Techniques as the most important categories (mean ratings for the third rating step were 7 and 6, respectively). For group 2, the most important categories were Commitment/Investment and Modeling

Table 7. Category Ratings

Category name	Means rating 1	Means rating 2	Means rating 3
a. Group 1			
Object Concepts	7.00	6.75	7.00
Analysis and Design Techniques	6.25	6.25	6.00
Object Interaction	6.25	6.00	5.75
Smalltalk Language	6.00	5.50	5.50
Smalltalk Front End	5.00	4.75	4.75
Working with GUIs	4.25	4.25	4.00
Choosing Words	4.25	4.25	4.50
Event Programming	4.00	3.50	3.75
Organizing Information Load	4.00	3.50	3.25
Lack of Standards	2.25	2.75	3.00
Total			
Group agreement	0.67	0.50	0.50
Kendall coef. of concordance (W) significance	$p < 0.05$	$p < 0.05$	$p < 0.05$
b. Group 2			
Commitment/Investment	6.33	6.33	6.33
Modeling Reality	5.67	6.33	6.33
Design Issues	5.00	5.00	5.33
Design Tools	5.00	4.00	4.33
Paradigm Shift	3.67	3.67	3.67
Generalization/Specialization/Aggregation	4.67	3.00	2.67
Reuse	4.33	3.00	3.00
Inheritance	3.33	2.00	1.67
Terms	2.33	1.67	2.00
Message Passing	2.33	1.33	1.00
Total			
Group agreement	0.52	0.70	0.73
Kendall coef. of concordance (W) significance	p Ns.	$p < 0.05$	$p < 0.05$

Reality (mean ratings for the third rating step were 6.33 and 6.33). Thus, both groups perceived of analysis and design issues as relatively important (or difficult). Group 2's focus on managerial or organizational issues is evident in the high ranking of the Commitment category. This is consistent with our earlier analysis of this group's concept and category identification activities. Group 1, however, did not rank programming categories as most important, although these participants identified many programming-related concepts and categories.

A classification activity analysis also was performed to determine the relative importance of the categories. A classification activity can be defined as a participant placing a concept into a category. The rightmost column in Table 8 shows the total number of classification activities for each category.

Table 8. Number of Concepts per Category for each Participant

a. Group 1

Category name	P1	P2	P3	P4	Total
Object Concepts	22	13	8	12	55
Smalltalk Language	15	10	10	15	50
Analysis and Design Techniques	9	10	13	8	40
Object Interaction	6	6	5	6	23
Smalltalk Front End	1	8	6	7	22
Working with GUIs	4	4	4	4	16
Choosing Words	3	4	5	3	15
Organizing Information Load	0	3	4	5	12
Lack of Standards	2	5	1	4	12
Event Programming	3	2	5	1	11
Total	65	65	61	65	256

Note: Participant 3 did not classify four of the group's 65 concepts.

b. Group 2

Category name	P5	P6	P7	Average
Design Issues	10	4	7	21
Paradigm Shift	6	8	3	17
Modeling Reality	5	5	5	15
Reuse	5	5	4	14
Commitment and Investment	3	3	5	11
Generalization/Specialization and Aggregation	2	3	6	11
Terms	4	5	2	11
Design Tools	3	3	2	8
Message Passing	2	1	2	5
Inheritance	0	0	2	2
Total	40	37	38	115

Note: Participants 6 and 7 did not classify three and two of the group's 40 concepts, respectively.

Group 1 used the Object Concepts, Smalltalk Language, and Analysis and Design Techniques categories most often in the classification exercise; group 2 used the Design Issues, Paradigm Shift, Modeling Reality, and Reuse categories most often. The ranking by total category usage from this analysis is consistent with the category importance rankings. For group 1, there was a very high level of agreement between the importance rankings and usage rankings (Kendall Coefficient of Concordance (W) = 0.96, p = 0.0451). Comparison of Table 7a and Table 8a shows the consistency of these results. The importance and usage rankings for group 2 also showed a very high level of agreement (Kendall Coefficient of Concordance (W) = 0.84, p = 0.0860). Thus, the more concepts placed in a category, the more important that category was perceived to be.

To summarize, the category-level analysis is consistent with the concept-level

results discussed earlier. Both groups identified difficulties with respect to OO analysis and design, programming, and concepts or terminology. The strongest similarity between the groups is the importance of and complexity associated with analysis and design. The strongest difference between the groups is group 1's focus on programming issues and group 2's focus on managerial or organizational issues. However, the first group's emphasis on programming may be slightly mitigated by the fact that the Object Concepts category ranked as more important than any of the group's five programming-related categories. Thus, while these participants may perceive a large number of programming-related obstacles, they may also believe these are more surmountable than the initial indoctrination into "object think" and vocabulary.

Cognitive Maps

In the last portion of the GSS sessions, participants defined causal relationships between the categories (see Table 2). Each relationship represents an individual's perception of how the difficulty associated with learning one category or topic influences the difficulty of learning another category or topic. Participants scored relationships using a scale from –3 (strongly decreasing) to +3 (strongly increasing). The number of relationships identified by the individual participants ranged from 38 to 72 with an average of 50 of the 90 possible relationships. Individuals in group 1 identified an average of 44 relationships among their 10 categories while individuals in group 2 identified an average of 59 relationships among their 10 categories.

Cause maps of individual perceptions of the difficulty of using OO techniques were derived from these relationships. Several of the participants' maps showing relationships of strength greater than or equal to two (i.e., –3, –2, 2, or 3) are presented in the appendix.[2]

A cause map can be analyzed for several purposes: to identify the relative importance of concepts in the map [18]; to determine if concepts are perceived as goals, inputs, or strategies for using inputs to accomplish goals [4, 7, 39]; and to identify shared perceptions of group members [5]. These analyses are performed by computing each concept's cognitive centrality [18], performing a Givens-Means-Ends analysis [39], and averaging the strengths of common relationships in individual cognitive maps [39], respectively.

Cognitive Centrality Analysis

Cognitively central concepts indicate a focus of an individual's attention. The number of causal relationships connected to a concept in a cognitive map is a measure of cognitive centrality [18]. More important concepts are more cognitively central—that is, individuals believe concepts that influence (or are influenced by) many other concepts are more important than concepts that influence (or are influenced by) few other concepts. Thus, a ranking of the categories by cognitive centrality should be consistent with the category importance rankings and the rankings of category usage. The categories were ranked in order of average cognitive centrality. The results of this ranking are shown in Table 9.

Table 9. Category Rankings by Average Cognitive Centrality

a. Group 1

Category name	P1	P2	P3	P4	Average
Smalltalk Language	12	10	3	11	9.00
Object Interaction	6	10	4	4	6.00
Lack of Standards	3	11	1	8	5.75
Analysis and Design Techniques	10	3	3	6	5.50
Smalltalk Front End	7	7	0	6	5.00
Object Concepts	6	4	3	7	5.00
Event Programming	2	6	4	7	4.75
Working with GUIs	2	7	3	5	4.25
Organizing Information Load	1	3	0	10	3.50
Choosing Words	7	0	5	2	3.50

b. Group 2

Category name	P5	P6	P7	Average
Commitment and Investment	7	11	8	8.67
Design Issues	2	16	4	7.33
Paradigm Shift	3	16	3	7.33
Modeling Reality	1	13	5	6.33
Inheritance	2	9	5	5.33
Generalization/Specialization and Aggregation	2	12	2	5.33
Design Tools	1	11	0	4.00
Message Passing	2	7	2	3.67
Reuse	2	7	0	3.00
Terms	1	7	1	3.00

Both groups showed a high level of agreement between the group category importance rankings, group category usage rankings, and average cognitive centrality rankings (group 1 Kendall Coefficient of Concordance (W) = 0.78, p = 0.0121; group 2 Kendall Coefficient of Concordance (W) = 0.65, p = 0.0411). Thus, the categories perceived as more important were generally the more cognitively central categories. The levels of agreement attained on the rankings of the categories obtained from three different data collection methods support the validity of the relative importance of the categories from the students' perspectives.

As Table 9 shows, analysis and design categories had high cognitive centrality scores for both groups (e.g., the Analysis and Design and Lack of Standards categories for group 1 and the Design Issues and Modeling Reality categories for group 2). Group 1's most cognitively central category was the Smalltalk Language, again reflecting the emphasis on programming difficulties for these participants. Group 2's most cognitively central category was Commitment and Investment, again confirming these participants' emphasis on higher-level issues.

Givens-Means-Ends Analysis

Givens-means-ends analysis provides a systematic approach for understanding cognitive maps. Givens represent inputs, constraints, or expectancies with which individuals in the environment must contend. A givens category has more outflows than inflows of causal influence in a cognitive map; it primarily *influences* other categories. Ends represent goals or outcomes that an individual attempts to achieve. An end category has more inflows than outflows of causal influence; it is primarily *influenced by* other categories. Means represent strategies, practices, or approaches that are attempts to manage constraints to achieve outcomes. A means category has approximately the same number of inflows and outflows of causal influence. The ratio of inflows to outflows determines whether a category is a given, means, or end. Ratios less than one are givens; ratios greater than one are ends; and ratios roughly equal to one are means. Each individual map was evaluated using givens-means-ends analysis [7, 18, 39]. The individual cognitive maps are shown in the appendix. Table 10 shows the categories classified as givens, means, and ends from the participants' perspectives. Italicized rows indicate categories that were perceived as either givens, means, or ends by a majority of group members.

In group 1, a majority (75 percent) of the participants identified the Object Concepts and Lack of Standards categories as ends, the Working with Graphical User Interfaces and Analysis and Design categories as givens, and the Choosing Words category as means. Fifty percent of the participants in group 1 identified Event Programming (P1 and P2), Object Interaction (P1 and P3), and Organizing the Information Load (P1 and P4) categories as givens and programming issues, including issues in the Smalltalk Language (P1 and P2), Smalltalk Front End (P1 and P2), and Event Programming (P3 and P4) categories, as means.

The givens, means, and ends identified by the individuals in group 1 indicate that they perceive that learning OO techniques starts with OO analysis; design, and implementation ideas (the Analysis and Design, Working with GUIs, and Object Interaction categories). Learning these ideas occurs through the identification of objects that are semantically consistent with a problem domain (Choosing Words category) and implementing those objects using OO programming (Smalltalk Language, Smalltalk Front-End, and Event Programming categories). The expected outcome of the learning process is an understanding of OO concepts (Object Concepts category). The Lack of Standards category also was perceived as an end. This category was strongly related to the Analysis and Design Techniques category. We speculate that the students were looking for standards as a way to understand OO analysis and design.

A majority (66 percent) of the participants in group 2 (see Table 10b) identified the Generalization-Specialization-Aggregation category as an end and the Commitment and Investment and Design Tools categories as givens. All participants in group 2 viewed the Design Issues category as a given and the Inheritance category as a means. The givens, means, and ends identified by the individuals in group 2 indicate that they perceive that learning OO techniques starts with a commitment (Commitment and

Table 10. Category Givens, Means, and Ends for Participants
(Cells contain category codes)

a. Group 1

	P1	P2	P3	P4
Givens (ratio of inflows to outflows < 1)	EP OI IL	*WG* *AD* EP	*WG* *AD* OI LS	*WG* *AD* IL ST
Means (ratio of inflows to outflows between 1 and 1.5)	*CW* ST FE WG OC AD	ST FE OI	*CW* EP	*CW* EP
Ends (ratio of inflows to outflows > 1.5)	*LS*	*OC* *LS* IL	*OC* ST	*OC* *LS* FE OI

b. Group 2

	P5	P6	P7
Givens (ratio of inflows to outflows < 1)	*DI* *DT* T MR GS	*DI* *DT* *CI* MP RE	*DI* *CI* PS
Means (ratio of inflows to outflows between 1 and 1.5)	*IN* RE MP	*IN* PS MR T	*IN*
Ends (ratio of inflows to outflows > 1.5)	PS CI	*GS*	*GS* MR MP T

Investment category) to OO analysis and design tools and techniques (Design Issues and Design Tools categories) and ends with the understanding of the OO concepts such as generalization and aggregation (Generalization category). This is consistent with the goal of learning object concepts identified by participants in group 1.

Composite Cognitive Maps

Figure 2 shows composite cognitive maps for each group. These maps provide an indication of group perceptions of category interrelationships. An arrow indicates that the participants in the group shared a perception that an increase in difficulty due to

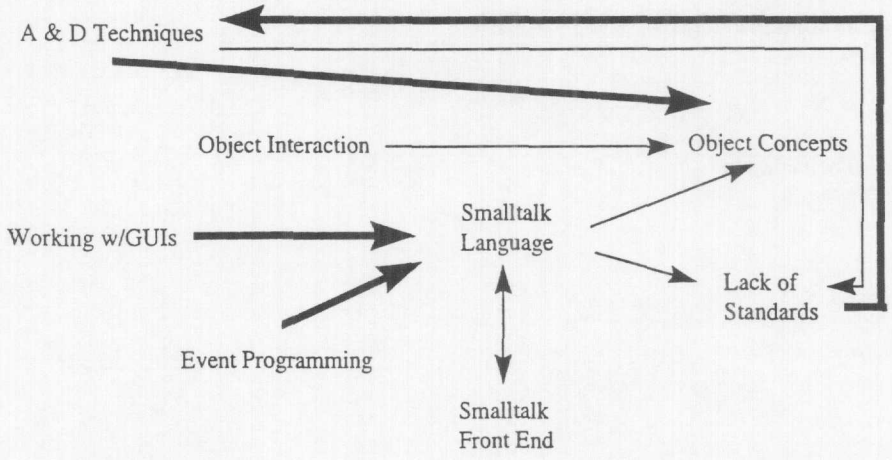


Figure 2a. Group 1 Composite Cognitive Map

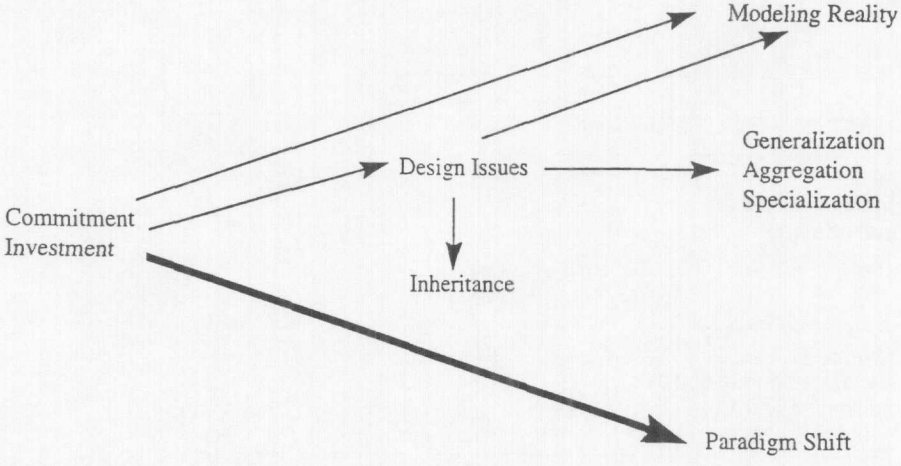


Figure 2b. Group 2 Composite Cognitive Map

the originating category affects the difficulty due to the terminating category. Thick arrows represent strong relationships (i.e., the average of the participants' ratings of the relationship was 1.5 or higher). Thin arrows indicate moderately strong relationships (i.e., average ratings between 1 and 1.5). The maps do not show weak relationships (i.e., average ratings less than 1). The maps in figure 2 are organized with givens on the left, means in the middle, and ends on the right. This organization shows causal influence flowing from left to right in the diagram.

For group 1 (figure 2a), working with graphical user interfaces and event programming strongly affect the perceived difficulty of using Smalltalk, which in turn affects the difficulty of understanding OO concepts. This perception may be due to the fact that Smalltalk was not only the students' first OO programming experience, it was

also (in most cases) their first experience developing an application with a graphical user interface in a Microsoft Windows environment. Thus, the difficulty of OO programming in Smalltalk was increased because the students also had to learn about GUI design, windows and panes, mouse events, and so on. User interface development is one of the most seductive and complex parts of the Smalltalk environment [30]. One way to address this difficulty is to assign student projects that require a simple user interface in order to reduce the scope and (hence) the difficulty of the event programming and GUI programming aspects of Smalltalk. This, in turn, should make Smalltalk more manageable, which, given the group 1 cognitive map, should also decrease the difficulty of understanding OO concepts. Thus, instruction that provides materials such as tutorials, templates, and examples for window design and event handling should facilitate learning OO concepts.

Group 1 participants also perceived analysis and design and object interaction as a positive influence on their understanding of object concepts. The analysis and design and object interaction categories address using OO diagramming techniques and CASE tools to represent various aspects of an OO model. The CASE tool used in the course (ObjectMaker) may have added another dimension of complexity for the students that detracted from, rather than enhanced, their understanding of OO concepts. Thus, for these students, instruction that reduces the difficulties of analysis and design techniques would reduce the difficulties in understanding object concepts. Inadvertently increasing analysis and design difficulty (e.g., through the use of a CASE tool that has its own associated learning curve) makes it more difficult to learn object concepts.

The means of implementing analysis and design ideas identified by the givens-means-ends analysis, that is, choosing semantically correct words, was not involved in any relationships with an average strength greater than one in the composite group map. Thus, the Choosing Words category does not appear to influence the difficulty of understanding object concepts. Based on group 1's perceptions, instructors should select a simple OO methodology, albeit one that includes the essential object concepts, to reduce the difficulties of understanding object concepts.

Group 1's cognitive map shows a bidirectional relationship between the analysis and design category and the lack of (analysis and design) standards category. These students perceived that analysis and design difficulties made it more difficult to find appropriate standards. Conversely, they perceived that the lack of standards made analysis and design more difficult. One way to address this problem is to use instruction that emphasizes the agreed-upon standard characteristics of OO analysis and design techniques (e.g., problem orientation, abstraction, inheritance, and message passing). Introduction of more advanced OO methods then could build on these fundamentals, ideally in a subsequent course. These steps should facilitate learning the object concepts.

Group 2's composite cognitive map, shown in figure 2b, illustrates their somewhat "higher level" of thinking about OO techniques. The strongest relationship in the map is the influence of the Commitment and Investment category on the ability to make the paradigm shift that is required by OO techniques. In addition, group 2 perceived

that commitment and investment affect the difficulty of learning OO design. The design issues category is a given in the composite cognitive map for group 2. This is consistent with the givens-means-ends analysis where all three participants identified design issues as a given. Design issues influence the difficulty of learning inheritance, generalization, and aggregation.

Rather than focusing on programming and graphical user interfaces, group 2's cognitive map indicates that commitment to OO technology strongly affects the perceived difficulty of moving from thinking about problems in a structured manner to an OO manner. This would indicate that, for group 2, "buying in" to object orientation is an important first step to shifting from structured to OO programming. As with group 1, the cognitive map for this group is consistent with their category importance rankings. The primary given (Commitment and Investment) and end (Modeling Reality) categories in the composite cognitive map were also the two most important categories in the category ranking activity described earlier.

The goals (ends) of group 2 have similarities and differences with the goals (ends) of group 1. Both groups were attempting to learn generalization, aggregation, and inheritance concepts that are essential for understanding OO techniques. Furthermore, both groups indicate that simplifying design issues will make learning OO concepts less difficult. The major differences between the groups can be summed up by recognizing the "lower-level" or programming emphasis of group 1 and the "higher-level" emphasis of group 2.

Summary of Cognitive Map Findings

Results from the cognitive maps indicate that category rankings by cognitive centrality are consistent with importance rankings developed through the earlier Delphi-like process [14], and rankings of category usage during concept classification. Consistency of the relative importance of the categories implies that the perceptions of the participants were congruent.

Givens-means-ends analysis shows that participants in both groups had similar goals (object concepts for group 1, generalization and aggregation for group 2) and expectations (analysis and design for group 1, design issues and design techniques for group 2). Programming issues contributed to learning for group 1 more than they did for group 2. Group 2 was concerned about making the commitment and changing their way of thinking from structured techniques to an OO approach. Composite cognitive maps show that simplified concepts should make learning less difficult for both programming and analysis issues. Providing rationales for buy-in to an OO approach also should reduce the difficulty of the learning process.

Discussion

A MAJOR THEME THAT ARISES FROM THIS STUDY is that there are many issues that make learning OO techniques difficult. One of the biggest training issues raised in prior research is the paradigm shift from structured to OO thinking [26]. While none

of our participants was a long-time procedural programmer, they had learned structured analysis, design, and programming techniques in prior courses and/or work experience, and they struggled with the change to an OO mindset. These struggles are evidenced in a number of the concepts identified by each group (e.g., "functions in procedural [languages] were so different from objects"; "the code is harder to follow than traditional code like COBOL"). Group 2 identified a separate category (the Paradigm Shift category) for these issues, and this was one of the group's most cognitively central categories.

The students also had difficulty with reuse or class library usage. Reuse in Smalltalk can be difficult because the class library is large and overwhelming to the OO newcomer, who may feel as though everything in the library has to be learned before anything can be done [30]. In our study, several students identified difficulties grasping the contents of the class library and deciding when and where to reuse (see Table 3). Each group created a category that largely focused on reuse and class library issues (the Smalltalk Front End category for group 1 and the Reuse category for group 2). Although these categories were not perceived as the most important (i.e., most difficult) to the students, they were perceived as having a direct impact on the most cognitively central category in each group. In group 1, three of the four members' cognitive maps show that reuse directly influenced the difficulty associated with the Smalltalk Language category (and vice versa). In group 2, two of three members' cognitive maps show that reuse directly influenced the difficulty associated with the Commitment/Investment category. Thus, for both groups, the potential benefits of reuse were somewhat mitigated by the learning curve associated with the class library, and reuse difficulties made it more difficult to learn OO programming (for group 1) and more difficult to "buy in" to the OO paradigm (for group 2).

Message passing, a core OO concept, was also challenging for the students. Both groups identified difficulties understanding the flow of execution or sequence of message passing in an OO program (see Table 3), and both created separate categories for these issues (the Object Interaction category for group 1 and the Message Passing category for group 2). Message passing can be difficult to understand because it changes the traditional concept of "a program." Instead of a program printout with a clear beginning and end, an OO program exists in small chunks (methods and messages) distributed among many objects in a class hierarchy [30]. Rosson and Carroll describe this difficulty:

> [P]rogrammers expect computer programs to have a beginning and an end; one understands a program by reading through it. These notions do not map well in Smalltalk. . . . understanding a program means understanding what the objects are and how messages are passed among the objects to accomplish tasks. . . . functionality in a system is incredibly distributed. [30, p. 76]

Finally, OO analysis and design issues were very important sources of difficulty for the students. Both groups had multiple categories for analysis and design issues, and many of these categories were ranked as highly important and cognitively central. Many of the students' analysis and design issues referred to complexities in diagramming notations and difficulties with the CASE tool used. The lack of a standard

accepted set of analysis and design diagrams and the introduction of the ObjectMaker CASE tool likely increased the students' struggles in this area.

On the basis of our results, we recommend thethe OO analysis, design, and programming environment be considerably simplified for pedagogical purposes. The OO approach represents a significant change from traditional approaches to systems development. The students in our study had not only to learn new vocabulary and concepts, but also to incorporate these concepts into new analysis and design models and a new programming environment. The composite cognitive maps for both groups show that analysis, design, and programming difficulties increased the difficulty of grasping OO concepts. Thus, it appears that learning "OO think" may be complicated, rather than reinforced, by the use of multiple OO analysis and design diagrams, OO CASE tools, and/or projects with a heavy emphasis on GUI programming. Simplifying the instructional tools and technology (where possible) should facilitate an emphasis on core OO concepts and their manifestation in analysis, design, and programming.

There are several ways to simplify the instructional tools and technology. For example, Rosson and Carroll [31] advocate a minimalist approach to teaching OO programming in Smalltalk. They reduce some of the complexity of the Smalltalk environment by creating a browser that restricts the view of the class hierarchy to only those classes relevant to a particular programming task. Our results suggest that a minimalist approach may be useful not only for programming, but also for reducing the difficulties associated with learning OO analysis and design techniques. A complete set of OO analysis and design diagrams could be "pruned" to a smaller core set, and/or each diagram could use a restricted subset of the complete notation. This would reduce some of the complexity in the analysis, design, and programming environments that detracts from learning the key OO concepts.

The cognitive maps for each group suggest other ideas for improving OO teaching. Each group's composite map provides an indication of how an educator might untangle the students' confusion about OO techniques. For example, from group 1's perspective, event programming and working with graphical user interfaces contributed to the difficulty of learning Smalltalk. Thus, separating event programming and graphical user interface design (at least conceptually) from Smalltalk, and addressing the former first (or reducing its importance in class projects) might provide a better foundation for learning OO programming concepts. Group 2's cognitive map suggests that a high-level discussion of the benefits of OO development over more traditional approaches would be a good starting point. If these students can themselves "buy into" the OO approach, they may be in a better position to learn OO analysis, design, and programming techniques.

Finally, as OO techniques continue to mature and gain in acceptance, other avenues for addressing the students' difficulties become available. While there are still dozens of published OO analysis and design techniques, many of the technique authors are now working together toward a *de facto* standard [3, 24, 33]. Many traditional systems analysis and design textbooks are also moving toward incorporating OO modeling concepts and techniques, so that students may be introduced to "OO think" and OO analysis and design models earlier in their degree programs (e.g., [16]). Introductory

programming courses that incorporate OO concepts and use OO programming languages are available at many universities. To the extent that OO modeling and programming concepts are introduced in other courses, the enormity of new material the students encounter in a single "dedicated" OO course will diminish.

Limitations and Future Research Directions

THIS STUDY HAS SEVERAL LIMITATIONS THAT DESERVE MENTION. First and foremost, we had a small convenience sample, which limits the ability to generalize our findings. All of our participants had learned OO techniques by taking the same course, with one instructor, and one set of OO tools (e.g., Smalltalk, ObjectMaker). These students' perceptions are certainly influenced by the content and delivery of the course. These students were also, by and large, novice programmers. Thus, our findings cannot be generalized to experienced structured programmers or to individuals who have learned OO techniques through other means. However, many of our findings are consistent with other research and experiential reports on OO learning (e.g., [26, 28, 31]). Future studies that incorporate larger and more varied samples are certainly warranted. Many important factors, such as individual and group differences, instructor characteristics, and course structure and content, have not yet been systematically explored.

Second, the study was conducted after the students had completed the course. We got their impressions of learning not as it was occurring, but after the fact. This required the students to reflect on their OO learning experience and report their perceptions. Such recall is often imperfect and biased when feelings are elicited; however, when identifying issues or facts about experience, participants are capable of accurate responses [19]. This was deemed acceptable in view of our focus on the issues the students perceived as difficult versus their feelings about the class or OO in general.

Third, the very broad nature of the research question limits the detailed information available from the study pertaining to any one category of issues. We wanted to get the students' perceptions of OO difficulties to identify stumbling blocks from their perspective. We gathered perception data rather than a more objective assessment of what OO knowledge they took away from the course to avoid artificially restricting the range of their responses. We felt this was acceptable because of the exploratory nature of our research and the consistency of this approach with other cognitive mapping studies. Certainly, other studies that investigate OO learning difficulties in a longitudinal manner and from multiple perspectives (e.g., students' perceptions, "objective" testing/evaluation, instructor/coworker perceptions) are warranted and will enrich our early understanding of OO learning.

The results of this study contribute to our understanding of the difficulties in learning OO techniques and can sensitize educators in classroom and corporate training environments to some of the obstacles learners encounter. In particular, we found that the shift from procedural to OO thinking can be easily confounded in two ways. First, the use of less mature and often more complex analysis and design techniques and CASE tools can detract from students' understanding of OO concepts, rather than reinforcing those concepts. Second, the fact that many OO programs are GUI-driven

means that students must learn event programming and user interface design, in addition to OO programming. Teaching approaches that reduce the "noise" associated with analysis, design, and programming should reduce the difficulties students encounter when learning OO systems development. Future research needs to explore the effectiveness of these and other teaching approaches on OO learning.

Finally, we believe that the use of cognitive mapping with GSS is a promising technique for exploring OO learning. One avenue for future work would be to conduct several GSS sessions during the course of OO learning, or during an organization's adoption and assimilation of OO technology, to observe the evolution of individual and group cognitive maps over time, as experience and knowledge are gained. Another direction is to compare and contrast the cognitive maps of expert and novice OO developers. These approaches can further improve our understanding of OO learning difficulties and the impact of teaching/training methods and experience on the learning process.

NOTES

1. A complete listing of the concepts can be obtained from the first author upon request.
2. Cognitive maps of each participant in the study can be obtained from the first author upon request.

REFERENCES

1. Anson, R.G.; Fellers, J.W.; Bostrom, R.P.; and Chidambaram, L. Using group support systems to facilitate the research process. *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, vol. 4, 1992, pp. 70–79.

2. Axelrod, R. *Structure of Decision—The Cognitive Maps of Political Elites*. Princeton, NJ: Princeton University Press, 1976.

3. Booch, G. *Object-Oriented Design with Applications*. Redwood City, CA: Benjamin/Cummings, 1991.

4. Bougon, M.G. Uncovering cognitive maps—the self-Q technique. In G. Morgan (ed.), *Beyond Method: Strategies for Social Research*. Beverly Hills, CA: Sage, 1983, pp. 173–188.

5. Bougon, M.G. Congregate cognitive maps: a unified dynamic theory of organization and strategy. *Journal of Management Studies*, *29*, 3 (May 1992), 369–389.

6. Bougon, M.G.; Baird, N.; Komocar, J.M.; and Ross, W. Identifying strategic loops: the self Q interviews. In A.S. Huff (ed.), *Mapping Strategic Thought*. Chichester, UK: John Wiley, 1990, pp. 327–354.

7. Bougon, M.G.; Weick, K.; and Binkhorst, D. Cognition in organizations: an analysis of the Utrecht Jazz Orchestra. *Administrative Science Quarterly*, *22* (December 1977), 606–639.

8. Coad, P., and Yourdon, E. *Object-Oriented Analysis*, 2d ed. Englewood Cliffs, NJ: Yourdon Press, 1991.

9. Coad, P., and Yourdon, E. *Object-Oriented Design*. Englewood Cliffs, NJ: Yourdon Press, 1991.

10. *Communications of the ACM*. Special issue: analysis and modeling in software development. *Communications of the ACM*, *35*, 9 (1992).

11. *Communications of the ACM*. Special issue: concurrent object-oriented programming. *Communications of the ACM, 36*, 9 (1993).

12. *Communications of the ACM*. Special issue: object-oriented software testing. *Communications of the ACM, 37*, 9 (1994).

13. Compeau, D.R., and Higgins, C.A. Application of social cognitive theory to training for computer skills. *Information Systems Research, 6*, 2 (1995), 118–143.

14. Dalkey, N., and Rourke, D. *Experimental Assessment of Delphi Procedures with group Value Judgment R–612-ARP*. Santa Monica, CA: RAND Corporation, 1971.

15. Davis, G.B.; Nunamaker, J.F.; Watson, H.J.; and Wynne, B.E. The use of a collaborative work system for the study of the key issues facing information systems managers: a comparison of issues and data collection methods from previous studies. *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, vol. 4, 1992, pp. 46–59.

16. Decker, R., and Hirshfield, S. *The Object Concept: An Introduction to Computer Programming*. Boston: PWS, 1995.

17. Eden, C. On the nature of cognitive maps. *Journal of Management Studies*, *29*, 3 (May 1992), 261–265.

18. Eden, C.; Ackermann, F.; and Cropper, S. The analysis of cause maps. *Journal of Management Studies*, *29*, 3 (May 1992), 309–324.

19. Ericsson, K.A., and Simon, H.A. Verbal reports as data. *Psychological Review*, *87* (May 1980), 215–251.

20. Finch, L.; Landry, J.; Monarchi, D.E.; and Tegarden, D.P. A knowledge acquisition methodology using cognitive mapping and information display boards. *Proceedings of the Twentieth Hawaii International Conference on System Sciences*, vol. 3, 1987, pp. 470–477.

21. Fiol, M., and Huff, A.S. Maps for managers: where are we? where do we go from here? *Journal of Management Studies*, *29*, 3 (May 1992), 267–285.

22. Hoffer, J.A.; Anson, R.; Bostrum, R.P.; and Michaelc, S.J. Identifying the root causes of data systems planning problems: an application of the PLEXSYS electronic meeting support system. *Proceedings of the Twenty-Third Hawaii International Conference on System Sciences*, vol. 3, 1990, pp. 30–39.

23. Huff, A.S. *Mapping Strategic Thought*. Chichester, UK: John Wiley, 1990.

24. Jacobson, I.; Christerson, M.; Jonsson, P.; and Overgaard, G. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Wokingham, UK: Addison-Wesley, 1992.

25. Kalkota, R.; Rathman, S.; and Whinston, A.B. The role of complexity in object-oriented systems development. *Proceedings of the Twenty-Sixth Hawaii International Conference on System Sciences*, vol. 2, 1993, pp. 759–768.

26. Manns, M.L., and Nelson, H.J. An exploration of schema development in procedure-oriented programmers learning object-oriented technology. *Proceedings of the Fourteenth International Conference on Information Systems*, 1993, pp. 385–386.

27. Montazemi, A.R., and Conrath, D.W. The use of cognitive mapping for information requirements analysis. *MIS Quarterly*, *10*, 1 (1986), 44–55.

28. Puhr, G.P.; Nelson, H.J.; and Monarchi, D.E. Teaching object-oriented systems development: challenges and recommendations. *Journal of Object-Oriented Systems*, *2* (1995), 135–154.

29. Rosson, M.B., and Alpert, S.R. The cognitive consequences of object-oriented design. *Human-Computer Interaction*, *5* (1990), 345–379.

30. Rosson, M.B., and Carroll, J.M. Climbing the smalltalk mountain. *SIGCHI Bulletin*, *21*, 3 (1990), 76–79.

31. Rosson, M.B.; Carroll, J.M.; and Bellamy, R.K.E. A case study in minimalist instruction. *Human Factors in Computing Systems, CHI '90 Conference*, 1990, pp. 423–429.

32. Rosson, M.B., and Gold, E. Problem-solution mapping in object-oriented design. *OOPSLA–89 Conference Proceedings, SIGPLAN Notices*, *24*, 10 (1989), 7–10.

33. Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; and Lorensen, W. *Object-Oriented Modeling and Design*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

34. Scholtz, J.; Chidamber, S.; Glass, R.; Goerner, A.; Rosson, M.B.; Stark, M.; and Vessey, I. Object-oriented programming: the promise and the reality. *Journal of Systems Software*, *23* (1993), 199–204.

35. Siegel, S., and Castellan, N.J., Jr. *Nonparametric Statistics for the Behavioral Sciences*. New York: McGraw-Hill, 1988.

36. Sheetz, S.D.; Tegarden, D.P.; Kozar, K.A.; and Zigurs, I. A group support systems approach to cognitive mapping. *Journal of Management Information Systems*, *11*, 1 (Summer 1994), 31–57.

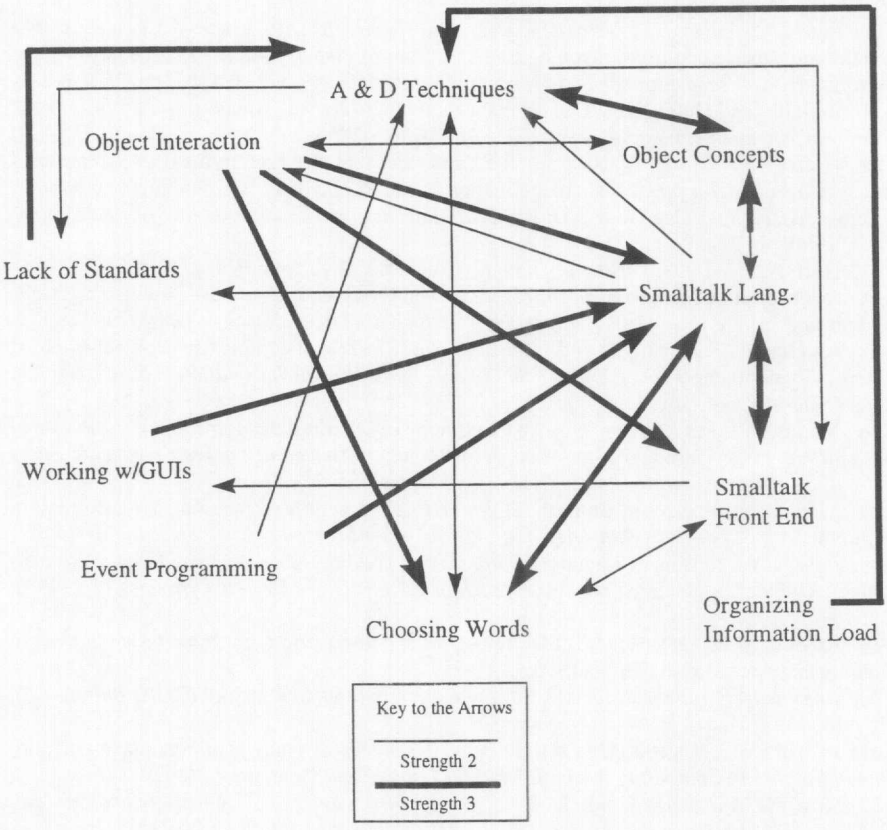
37. Verity, J.W., and Schwartz, E.I. Software made simple: will object-oriented programming transform the computer industry? *Business Week* (September 30, 1991), 92–100.

38. Vessey, I., and Conger S. Requirement specification: learning object, process, and data methodologies. *Communications of the ACM*, *37*, 5 (1994), 102–113.

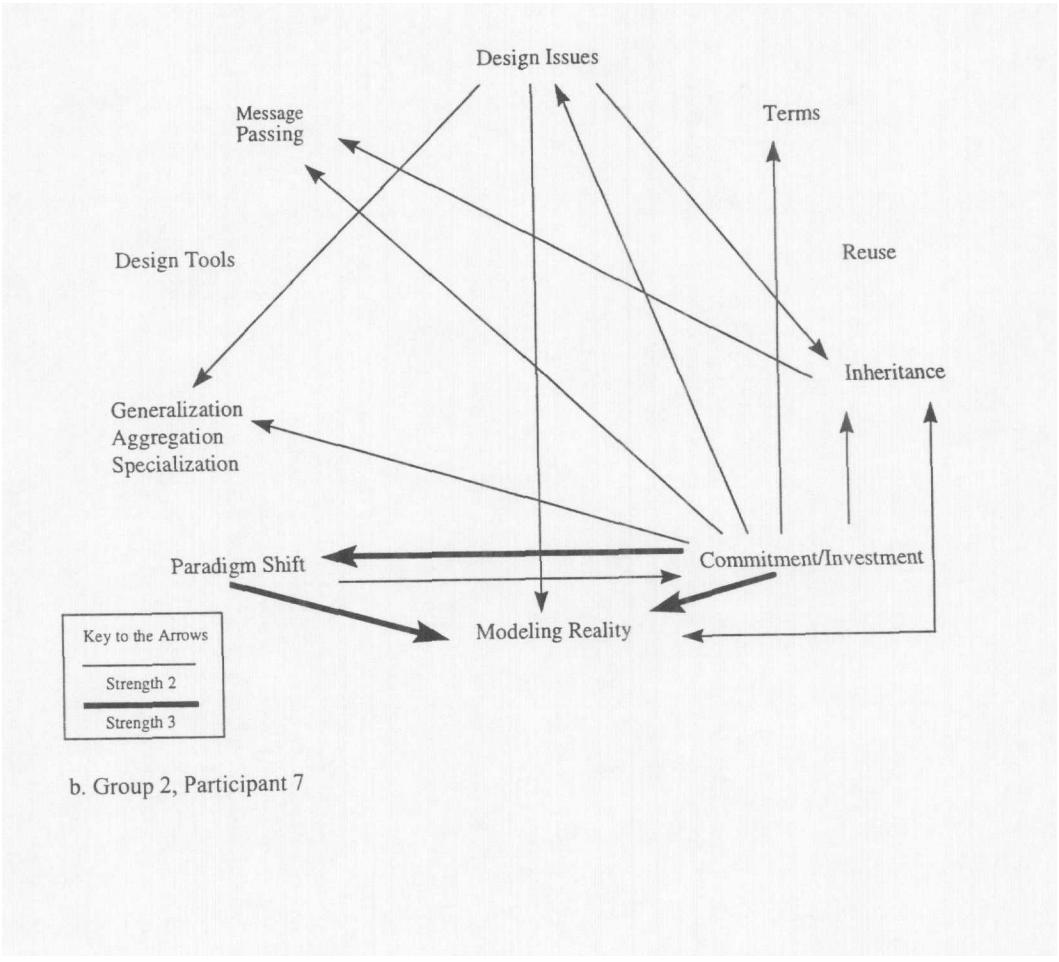
39. Weick, K.L., and Bougon, M.G. Organizations as cognitive maps: charting ways to success and failure. In H. Sims and D. Gioia (eds.), *The Thinking Organization: Dynamics of Organizational Cognition*. San Francisco: Jossey-Bass, 1986.

40. Zhang, W.; Chen, S.; Wang, W.; and King, R.S. A cognitive-map-based approach to the coordination of distributed cooperative agents. *IEEE Transactions on Systems, Man, and Cybernetics*, *22*, 1 (1992), 103–114.

APPENDIX : Examples of Individual Participant Cognitive Maps



A. Group 1, Participant 1



b. Group 2, Participant 7