Toward Formalized Object-Oriented Management Information Systems Analysis

SHOUHONG WANG

SHOUHONG WANG is an Associate Professor of Management Informaiton Sustems at the University of New Brunswick, Canada. He received his M.B.A. from Tsinghua University, China, in 1981, and his Ph.D. in information systems from McMaster University, Canada, in l990. His experience includes working as a production senior manager, teaching in MIS, and consulting as chief information systems analyst of the State Ecomonic Commission of China. His research interests include information systems analysis and design, artificial intelligence in management, and the human–computer interface. His papers have been published (or are forthcoming) in *Journal of Management Information Systems, Decision Sciences, IEEE Transactions on Systems, Man, and Cybernetics, Information Systems Management, IEEE Transactions on Pattern Analysis and Machine Intelligence, Computers and Operations Research, INFOR, Fuzzy Sets and Systems, Computational Intelligence, Management Science, European Journal of Operational Research, Canadian Journal of Administrative Science, International Journal of Information Management*, and others.

ABSTRACT: Object-oriented approaches have received attention in management information systems development due to the advantages over the traditional approaches claimed by the proponents of the object-oriented approaches. To describe how people actually perform object-oriented analyses, this paper formalizes an object-oriented systems analysis approach. Protocol analyses of seven systems analysts, who were experienced with structured analysis, were used to formalize procedures for object-oriented analysis. The protocol analyses revealed four fundamental types of object classes in a management information system. They are: input, output, physiomorphic, and event object classes. The identification of input, output, and event classes depends upon the problem domain being analyzed. Physiomorphic classes, on the other hand, are more likely to match a schema that is perceived by the analyst based on his a priori knowledge about the problem domain classes. The protocol analyses also revealed that object classes are identified in an ad hoc manner; however, when checking the analysis, depth-first or breadth-first searching methods are often used. These, in turn, are controlled by a global backward or forward tracing strategy. Tracing the origin of data in messages was the strategy used for checking the completeness of the analysis.

An experiment was conducted to compare the protocol-based object-oriented method and structured analysis. Thirty-two students who had no previous systems analysis experience were trained and then completed a problem using both techniques. The protocol-based method produced analyses that more closely matched the problem.

Acknowledgments: The author wishes to thank the students who participated in this study. This research was supported by a grant from the University of New Brunswick (90–352609) and a grant from the Social Sciences and Humanities Research Council of Canada (410930057). The author is indebted to two anonymous referees and an Associate Editor for their valuable comments in revising this paper.

Furthermore, it required less time to complete an analysis, and it was perceived as easier to use by the participants than the structured analysis method.

KEY WORDS AND PHRASES: management information systems analysis, object-oriented systems analysis, protocol analysis.

ALTHOUGH OBJECT-ORIENTED PROGRAMMING [35] WAS DEVELOPED IN THE MID–1960s, object-oriented system analysis (OOA) methods did not become popular until the late 1980s [4, 6, 10, 37]. Since then, OOA has gradually achieved acceptance in systems development [4, 55]. The philosophy of OOA is quite different from that of the structured systems analysis approach. Instead of using functional decomposition of the system, OOA focuses on identifying objects and their activities [37].

The computing society has demonstrated its extraordinary interest in object-oriented methods [8, 22, 23]. Several researchers assert that the OOA approach is beneficial for systems development [7, 13, 20, 25, 26, 41]. A recent information industry survey indicated that the adoption rate of object-oriented methods is increasing dramatically [34]. Organizations that embraced the approach have experienced significant cost savings in the systems development area [34]. Given the recency of its development, however, the methodology of OOA is far from mature. It is commonly accepted in the object-oriented research field that the identification of object classes remains an art that is highly dependent upon the problem domain [6, 7]. Although objects of a system are "just there for the picking" [28], there are no general guidelines for identifying objects. Moreover, the genuine OOA methods often suffer as a result of difficulties in process representation and function refinement in information systems analysis [17, 18, 37, 45]. There still is a need for an approach that merges functional, dynamic, and object-oriented methodologies for systems analysis [10, 13, 17, 37].

Given the information-intensive nature of the systems analysis process and the importance of representation for problem solving, basic research into the representation of systems analysis information is to be expected. On the other hand, there is little research on the cognitive processes underlying how an analyst abstracts a description for the information system. So far, much of the work on the process of systems analysis is prescriptive rather than being based on observations or experiments. We do not have enough experience with OOA to be sure where it fits best, how it can best be applied, or when it can be used to supplement more traditional techniques. This suggests that some important characteristics of systems analysis behavior need to be studied experimentally. In the case of developing an OOA method, this kind of research is especially helpful because the OOA field is still in its infancy.

The purpose of the present research is to demonstrate that an empirical approach for studying management information systems analysis is both possible and valuable. As such, this study first formalizes a method (or procedures) that could be used to analyze a system in the object-oriented context by using protocol analysis. It then tests the method empirically.

Using Protocol Analysis for Understanding OOA

Methodology—Protocol Analysis

PROTOCOL ANALYSIS [11, 31] IS A RESEARCH TECHNIQUE USED for studying management information systems development [24], process tracing in decision support systems [48], decision-making analysis [5], software psychology [42], and knowledge-based modeling [21]. In protocol analysis, participants are asked to "think aloud" and "talk aloud" while performing tasks and to verbalize their thoughts as they solve the problem. Protocol analysis is oriented toward extracting the mental behaviors inherent in verbal transcriptions, called protocols.

Protocol analysis is controversial. Two major literature reviews presented different viewpoints regarding the usefulness of verbal reports. Based on the evidence reviewed, Nisbett and Wilson [32] suggested that there may be little relationship between verbal reports and higher-order cognitive processes. It should be noted that all of the studies that were analyzed in Nisbett and Wilson's [32] review used retrospective verbalizations, and therefore their criticisms of protocol methodology should be limited to this particular type of protocol data collection [33]. The opposite view was provided by Ericsson and Simon [11], who proclaimed the validity of verbal reports. The central argument of protocol analysis, according to Ericsson and Simon [11], is that verbal reports are valuable data for understanding human information processing, even though some intermediate information may be missing. They argued that all data processing requires a transformation from an initial observation to a form in which a person's theories can be tested. Verbalization is one of the most critical elements of the human behavioral repertoire, and can be used as a vehicle for organizing, directing, and evaluating action toward a goal.

Although there is little research on the application of protocol analysis to information systems analysis, protocol methodology is considered a particularly promising method for use in MIS research [48, 49]. We believe that protocol analysis is particularly suited to developing an understanding of how information systems are analyzed. This research project was motivated by the success of Ericsson and Simon's approach and the consideration that a protocol may be the only source of data regarding the cognitive processes involved in systems analysis [50]. Protocol analysis is used in this research to trace how system analysts operate in modeling the real information system.[1]

The method of object-oriented systems analysis presented in this paper was based on the observation of behaviors in the context of the problem-solving theory of Newell and Simon [31]. It shares their view of problem solvers as information-processing systems. Because concurrent neutral-probing verbalizations are considered to be the most valid and reliable of methods utilized for formal protocol collection [12], they were selected for use in this study.

Participants

The selection of participants was difficult as this particular study attempted to solve a "chicken and egg" problem. On one hand, the experiments attempted to find an

appropriate method for OOA through a protocol analysis, which would provide insight into the analysts' cognitive processes during the performance of OOA. The participants for this study had to be representative of OOA analysts, as this would ensure the value of the conclusions of a protocol analysis. On the other hand, given the fact that OOA is still in the early stage of development, it was impossible to find skilled analysts with OOA experience in information systems organizations that were willing to participate in the study.

Another issue most often addressed in challenging the validity of verbal protocol analysis is the sample sizes employed. When the object of the research is to identify the process, strategy, or information used by the participants, and when there is no intent to make inferences about population parameters per se, the use of small samples does not adversely impact the analysis result [19]. In addition, the time-consuming nature of protocol analysis necessitates the use of smaller sample sizes than are usually employed in statistical tests of hypothesis. It is not unusual to carry out successful protocol analyses for only three or four participants (e.g., [16, 21]).

Taking the above factors into account, seven M.B.A. students (four male, three female), who had at least three years experience in systems analysis in industry, were selected[2] as the participants for the protocol analysis. Each participant learned OOA and completed a course project using the method. Textbooks [5, 37] were assigned to them to learn OOA; however, they were not required to follow a specific method or tool of OOA.

Experimental Design

Participants practiced "thinking aloud" while performing systems analysis prior to the protocol collection sessions. Following the practice of Ericsson and Simon [11, 12], participants were advised to take their time when verbalizing during the protocols, and not to be afraid of verbalizing too much. In protocol collection sessions, two sets of protocols were collected from each participant. One set was from a standard mini-case of a payroll system following the example in Senn [40, p. 701]. The second set of protocols was collected when the participant was analyzing his or her own project, such as a telephone company billing system or a lumber company purchasing system.

While working on systems analyses, the participants could use paper and pen to draw step-by-step diagrams. The participants were encouraged to use whatever notations and symbols they considered most appropriate to express their ideas. The participant was asked to "talk aloud" about what he or she was doing, and what the diagrams meant. The verbal statements were audiotaped, and the transcriptions of the tape recordings were analyzed together with the participant's draft diagrams. To obtain first-hand data and closely monitor the experimental procedure, the author was present during the verbalizations of each participant. His role was to operate the audio recorder and measure the draft diagrams. He did not give any instructions during the verbalizations.

The segmentation of spoken information into sentences in the transcript was made according to two rules. First, a break was made whenever the participant was judged to have paused in speaking. Second, if speech was relatively continuous, breaks were

made between major clauses. This segmentation into sentences in the protocol was intended to give an indication of the conceptual units used by the participant.

Retrospective questioning was driven by a checklist of different behaviors that were expected to occur during the analysis (e.g., "How did you find an object class in this situation?"). Following Ericsson and Simon's [11, 12] practice, only open-ended questions were presented to the participants. Finally, each of the participants completed a questionnaire identifying details of all previous systems analysis experience and his or her confidence in OOA.

Analysis

Protocol transcripts (see an example in appendix A) were analyzed by matching mental behaviors to oral sentences and written drafts of diagrams. Generally, two different analysis methods are employed in protocol studies [12]. The first method does not require the analysis of meanings for the observed verbalizations. This method is commonly used in cases where there is a prior agreement between the participant and the researcher regarding specific signals for their communication. The second type of coding does require the interpretation of meanings. This approach is more typical in the case of information systems analysis. In this type of coding, a protocol analysis must involve the researcher's prior knowledge or assumptions. Thus, the coding scheme—that is, the set of behavior categories—for protocols is not provided through analysis of the protocols themselves [12], but is defined a priori by the researchers. This was the approach used in this study.

A set of behavior coding categories was developed by evaluating the OOA processes in [3, 6, 10, 28, 37, 41] and the coding of preliminary data. Nine categories were generated and used in this study. In essence, each distinct category of behavior, such as identifying an object class, can be coded by mapping the verbalization to the category. For example, given the sentence "Customer must be an object class which provides necessary data for an order," the researcher would then code this as "identifying an object class." In protocol analysis for information systems analysis, it is important to divide the categories into mental and nonmental behaviors [46]. The nine categories and their divisions are:

Observed mental behaviors:

- *Identifying object classes*: Statement of an object class found in a protocol for the first time (e.g., "Employee is an object class").
- *Identifying attributes of a class*: Statement about data properties of a class (e.g., "Employee name, employee number, gender, address are the attributes of the employee class").
- *Identifying internal operations of a class*: Statement about information processing within a class (e.g., "Net pay is equal to gross pay minus tax deductions").
- *Identifying inheritance structures*: Statement of fact about the inheritance relationships between object classes (e.g., "Full-time employee and part-time employee are the subclasses of employee").

- *Identifying message sending*: Statement of fact about the communication between object classes (e.g., "Paycheck gets information about the employees, such as employee name and address, from the employee class").
- *Assertions of analysis strategies*: Verbalization about planning at a meta level (e.g., "Check over the diagram by tracing the sources of the data items in the paycheck class").
- *Iterations of processes*: Statement of control over the analytical process by repeating a process (e.g., "Do the same procedure to . . . as to . . . ").

Nonmental behaviors

- *Physical construction of diagrams*: Constructing and drawing an object-oriented systems analysis diagram, without any strict guideline.
- *Reading and searching information*: Searching for and retrieval of data in the problem text or existing diagrams (e.g., reading the problem statements of a system to be analyzed).

Each protocol segment was matched with the categories. For example, in appendix A, segment 2 was coded as "Identifying object classes," and segment 3 was coded as "Identifying attributes of a class." It was entirely possible for one statement to correspond to a few different coding categories. For example, according to the time-measured draft diagrams, segment 15 in appendix A was coded to "Identifying object classes" as well as "Identifying inheritance structure."

The protocol segments were then encoded into the Problem Behavior Graphs (PBG) [31]. A PBG describes the problem solver's behavior by representing the states of knowledge and their transformation during the course of problem solving. In this study, PBGs were used to aid us in identifying major components of the formalized OOA method. An example of the PBG encoded in this research program can be found in Wang and Archer [53]. In the final protocol analyses, we believe that a point was reached where the participants reliably reported their mental behaviors important for our purposes.

As a summary of the protocol analysis, the distribution of the categories to the mental behaviors for the payroll problem and other example cases are outlined in Table 1. In the table, the rows list mental behaviors, the columns list example cases, and each cell contains the number and percentage of categories that were associated with the behavior for that problem. Note that behaviors of recall (e.g., retrieve a defined object class), or redefining (e.g., rename a defined object class), that account for the other substantial portions of the protocols were normally found in checking the completeness of the analysis. These segments were not counted because they were related to the same mental behavior and did not make independent contributions to a system analysis method.

Validation of the Protocol Analysis

Each verbal report was transcribed from audio tape and segmented into short phrases by the researcher and an independent research assistant. Identified speech segments

Table 1. Distribution of the Categories to the Mental Behaviors for the Example Problems

Behaviors	Cases			
	The payroll system (average of 7 protocols)	Bird-watching club management	The purchasing system	The time and attendance system
Identifying object classes	10/73 (13.7 %)	6/91 (6.6%)	14/122 (11.5%)	12/165 (7.2%)
Identifying attributes of a class	9/73 (12.3%)	14/91 (15.4%)	15/122 (12.3%)	18/165 (10.9%)
Identifying internal operations of a class	7/73 (9.6%)	9/91 (9.9%)	20/122 (16.4%)	18/165 (10.9%)
Identifying inheritance structures	2 /73 (2.7%)	4/91 (4.4%)	2/122 (1.6%)	4/165 (2.4%)
Identifying the message sending	12/73 (16.4%)	12/91 (13.2%)	25/122 (20.5%)	18/165 (10.9%)
Assertions of analysis strategies	2/73 (2.7%)	5/91 (5.5%)	2/122 (1.6%)	4/165 (2.4%)
Iterations of processes	0/73 (0%)	1/91 (1.1%)	2/122 (1.6%)	0/165 (0%)

were then allocated to the categories listed previously in this paper. There was 98 percent agreement between the two researchers in the categorization of segments. The differences in categorization were discussed and an agreement was reached for those in which segmentation differed.

An OOA Method Based on the Protocol Analysis

IN THE PRESENT RESEARCH, A THINK-ALOUD PROTOCOL WAS EMPLOYED to assist in formalizing a method in OOA. Then, further experimental tests were designed to evaluate the formalized OOA method. The aim of constructing the OOA method was to produce a framework that describes individual episodes of systems analysis behavior. The protocol analyses of object-oriented systems analysis behaviors provided insights into the identification of object classes in OOA, and process of systems analysis.

Three major components of the OOA method were identified: four types of object classes, systematic control strategies, and matching inheritance structure. A matrix that maps the observed behaviors on the components of the method is presented in Table 2; the rows list the observed mental behaviors, and the columns list the major components of the OOA method. Each cell describes how the component of the method addressed the observed behavior.

Table 2. Relationships between the Mental Behaviors and the Components of the Method

Behaviors	Components of the method		
	Four fundamental types of object classes	Systematic control strategies: tracing and searching	Matching inheritance structure
Identifying object classes	All classes identified can be categorized	Some classes (especially event classes) are identified through tracing and searching	Some classes are identified by matching inheritance structures
Identifying attributes of a class	Event, input, output classes have their unique generic attributes (e.g. time in event classes)	Some attributes are identified through tracing and searching	Some attributes are identified via matching a priori frames
Identifying internal operations of a class	Event, input, output classes have their unique generic operations (e.g., print in output classes)	N/A	N/A
Identifying inheritance structures	Physiomorphic classes are more likely to match a priori frames	N/A	Inheritance structures are often identified by matching a priori frames
Identifying the message sending	Event, input classes have their unique generic message sending (e.g., trigger)	Tracing and searching are accomplished by following the message courses	N/A
Assertions of analysis strategies	Categorizing fundamental types of object classes can be an analysis strategy	Tracing and searching is an analysis strategy	Matching a priori frames is an analysis strategy
Iterations of processes	N/A	Tracing and searching are iterations	N/A
Searching memory	N/A	N/A	Searching a priori frames for matching inheritance structures

Four Types of Object Classes

There are four fundamental types of object classes in management information systems: physiomorphic, event, output, and input.

Physiomorphic Object

Physiomorphic object refers to a physically existing entity, including a person (e.g., customer, salesman), a property item (e.g., machine, building), and an organization unit (e.g., company, department). A system analyst usually has no difficulty in identifying those physiomorphic object classes around him or her; however, the attributes of each physiomorphic object class may not be identified completely at a glance. The operations on the object classes are even harder to uncover. It was found in the present protocol analysis that, for instance, an analyst of a marketing information system can easily identify CUSTOMER as an object class. Although the analyst is able to present a list of attributes for the CUSTOMER class, she or he may not be certain if these attributes are complete or redundant at the moment. She or he also cannot tell what operations would be involved without thinking about other object classes. As discussed later, to elaborate on physiomorphic object classes, iterations of refinement are usually needed.

Event

It was found that time perspectives were included in the schema of analysts. These time-dependent aspects of a system could be included in the operations of physiomorphic classes, but were often expressed in the form of event object classes. Because events are noticeable in the management environment, a system analyst is able to identify events based on his or her knowledge about day-to-day business operations and decision-making activities. Events are associated with state transitions of the system and explicitly express the system's dynamic properties. Some events could be routine operations or transactions such as paying, ordering, and scheduling. Others could be decision-making activities such as credit approval and personnel promotion.

An event is initiated by a trigger. In other words, one event is caused by or causes other events. A trigger could be a schedule (e.g., once every five days, 5 PM every day); the time when the state of an attribute of an object reaches a critical point (e.g., the inventory level reaches a reorder point); or an outside occurrence impacting the system (e.g., a telephone call).

A system analyst often discovers other object classes based on descriptions of events. For example, in an office system, the physiomorphic object class of SECRETARY is easy to identify but other object classes (e.g., MEETING-SCHEDULE) and their operations are less apparent unless the system analyst searches all the events happening to the SECRETARY.

Output

It was our observation that system analysts were knowledgeable about outputs of the systems. This is not only because the traditional structured methods, which emphasize identifying system outputs, have made a strong impact on the system development

philosophy of most analysts, but is also due to the nature of the objectives of systems development. For instance, questions such as "Why do I develop an information system?" and "What do I expect from the system?" were found in protocol segments.

An output object in MIS is usually a report. A sales summary, an invoice, or a credit certificate are some examples. It was found in the protocols collected that output was considered to be a fundamental type of object class based on the following facts. First, output objects have attributes describing the output properties, such as format and frequency. These attributes are called nontraceable attributes because they have little relationship to other object classes. Second, there are a significant number of operations in information systems that perform document writing/printing functions. These operations are associated with specific output reports so that object modules become more cohesive.

Input

As was the case with output object classes, system analysts were familiar with inputs of the system. Input objects are representations of the information entities that enter the system (e.g., order applications or government statistics for marketing forecasting). Input objects have their specific attributes, such as format and frequency, which must be included in the descriptions of the object classes.

Systematic Control Strategies: Tracing and Searching

It was found that at the early stage of a system analysis the analysts identified object classes in an ad hoc fashion. Some analysts started with physiomorphic object classes, but others' first move was with event, or input, or output. Each of the analysts, however, without exception, performed a global tracing procedure before she or he completed the analysis. There are two modes of global tracing, one of which is backward tracing. This mode uncovers (or checks) output classes and their attributes by tracing backward in three ways: (1) by finding the event that triggers the object class currently investigated; (2) by finding the origin of messages received and the destinations of messages sent to acquire data items; and (3) by defining the internal manipulations of the values of attributes.

For example, in a payroll system (see figures 3 and 4, in appendix B), suppose the analyst first identifies (or checks) the output object class of PAYCHECK and its attributes, such as EMPLOYEE-NAME, GROSS-PAY, INCOME-TAX, and NET-PAYMENT. The analyst then considers which event triggers PAYCHECK, and where the data items of these traceable attributes come from. The analyst will trace back to the PAYDAY event which triggers the PAYCHECK object class, as well as the EMPLOYEE object class where the data of EMPLOYEE-NAME come from. In the above example, it is found that an analyst will not stop a trace until the system boundary (e.g., the SYSTEM-CLOCK, REGISTRATION object class) or a terminal operation (e.g., INCOME-TAX-CALCULATION) is reached (see the example protocol lines 47–50 in appendix A).

The second global tracing mode is forward tracing. In contrast to backward tracing, the forward-tracing process begins with event and input object classes, and ends with output object classes. The protocol analysis indicated that the analysts often used a combination of the two tracing strategies.

Within a tracing process, there are two local searching strategies: breadth-first searching and depth-first searching. In breadth-first searching the analyst anchors an object class and searches all object classes that have direct connections (message sending) with the anchored class. For example, in a payroll system, suppose the PAYCHECK class is anchored, and then the following are searched: EMPLOYEE (which provides all information about the paycheck receiver), then TIME-CARD (which provides work hours), and finally PAYDAY which triggers PAYCHECK. In depth-first searching the analyst searches, in turn, the next object class along with the tracing path. In the payroll system example, suppose the backward tracing strategy is used. After PAYCHECK has been identified and anchored, the PAYDAY class that triggers PAYCHECK is identified. Next, PAYDAY is anchored, and SYSTEM-CLOCK which triggers PAYDAY is discovered.

Matching Inheritance Structure

Structuring the inheritance relationship between the object classes entails complex, open-ended cognitive analyses that are very difficult to characterize in a systematic way. Nevertheless, a number of mental behaviors in the identification of the inheritance structure of object classes were revealed in the protocol analysis. A priori, the analysts often had a schema or frame for representing a stereotypical inheritance structure of a familiar system [29, 39]. When the analyst tries to formulate the inheritance structure between the physiomorphic object classes, a schema is retrieved to match the identified physiomorphic object classes. This phenomenon is consistent with the findings in [15]. However, in the case of an unfamiliar problem that does not generate a schema in the analyst's mind and where the analyst fails to retrieve a schema, he or she is unlikely to formalize the inheritance relationships between the identified physiomorphic object classes.

The observed mental behaviors involved in generating the inheritance structures are illustrated by the following example. Assume that the analyst has identified a WORKER class based on the descriptions of the system. When the analyst identifies a class named LABOR, he may find that it is virtually the same as WORKER in the context of the system; therefore he could add an alias of LABOR to WORKER. If the analyst identifies a class named CUSTOMER, he may simply add it into the class set because he does not see any common properties shared by WORKER and CUSTOMER in that context. If the analyst identifies a class PART-TIME-WORKER, she is likely to decide to reorganize the structure of the class set so that the system has a superclass of EMPLOYEE, which extracts the common properties (attributes or operations) shared by both FULL-TIME-WORKER and PART-TIME-WORKER. This, in turn, subordinates FULL-TIME-WORKER and PART-TIME-WORKER to EMPLOYEE. It appears that these behaviors were accomplished by means of semantic analysis [43].[3]

Findings of the Protocol Analysis

Based on our protocol analyses, it was observed that an object-oriented systems analysis is a complicated information gathering and abstraction process. Given this finding, that it would be possible to summarize the protocols into a simple algorithm seems unlikely. Nevertheless, some aspects of commonality in OOA were observed in the protocol analyses. These refined components of object-oriented systems analysis can be formalized into OOA procedures, are described in figure 1. Four fundamental types of object classes might be identified by the systems analyst: input, output, physiomorphic, and event object classes. The identification of input, output, and event classes depends upon the problem domain being analyzed. Physiomorphic classes should be able to match a schema that is best perceived by the analyst in relation to the problem to be analyzed based on his or her a priori knowledge. To search object classes in a systematic way, depth-first or breadth-first searching methods can be used, which, in turn, are controlled by a global backward- or forward-tracing strategy.

The above formalized procedures can be employed as a method for OOA. This method for OOA has two unique features not supported by other OOA methods.

1. *A uniform paradigm of functional, dynamic and object-oriented methodologies.* Research by the object-oriented systems analysis community has often found that there is a lack of a uniform paradigm that merges functional, dynamic, and object-oriented methodologies [13, 17, 37]. The OOA method suggested in this paper integrates the descriptions of three aspects of an information system—namely, object, function, and dynamics—into the single object-oriented paradigm. This method not only deals with tangible physiomorphic objects in the usual way, but also describes dynamic (timing) and functional properties of a system by defining event object classes. The functional properties of a system are also delineated by specifying internal operations of the object classes and data flows (parameters of the messages) between the object modulars.
2. *An algorithmic analysis method.* The OOA method has outlined structured elements for system analyses, which include stereotypes of object classes, local searching, a global tracing process, and matching a priori inheritance structure. Even though the actual mental processes of a system analyst will always remain unknown, these infrastructural elements were often found to exist in the verbal protocols. As will be shown later in this paper, an approach composed of such algorithmic procedures appears to substantially reduce the requirement for artistic skills in object-oriented systems analyses.

To demonstrate the value of this method, the new features of this OOA method are contrasted with other currently available OOA techniques in Table 3. In the table, each cell gives a note to explain whether the currently available technique implements the component of the present OOA method.

Figure 1.

Table 3. Comparison of Object-Oriented Systems Analysis Methods

	Components of the new method		
Other OOA methods	Four types of object classes	Systematic control strategies: tracing and searching	Matching inheritance structure
Bailin object-oriented requirements specification [1]	Entity-relationship diagram focuses more on physiomorphic classes	Entity-relationship diagram does not specify data flows and processing sequences	The method does not support inheritance structures
Coad & Yourdon object-oriented analysis [6]	Object classes diagram layer–1 is for physiomorphic classes only	The method does not support data flow tracing	Formalize inheritance structure in bottom-up fashion
Shlaer & Mellor object-oriented analysis [41]	Event is not modeled in the object class term	The method does not provide a tracing device	The method does not specify how to construct an information structure diagram
Rumbaugh et al. object-oriented modeling [37]	Object classes are basically physiomorphic	Object paradigm does not integrate functional aspects	Formalize inheritance structure in bottom-up fashion
Embley et al. object-oriented analysis [10]	Physiomorphic and event are two fundamental types of classes	The approach does not ensure the completeness of an analysis	The method does not specify how to build inheritance structure

A reduction of the above OOA method has been developed as an OOA tool. The notations of diagrams for the OOA tool (see figure 3) and an example diagram of an OOA result (see figure 4) are demonstrated in appendix B.

Evaluation of the OOA Method

VALUABLE METHODS THAT SUPPORT INFORMATION SYSTEMS ANALYSIS are those that simultaneously ease the process of analysis and improve the usability of the analysis results [36]. An experiment was conducted to evaluate the potential of the formalized OOA method in performing this dual function. It was expected that the OOA method would be useful in business-process modeling [53] and MIS analyses [52, 54]. The remainder of this paper presents the results of an experiment designed to test the formalized OOA procedure.

Method

The method used in this experimental study compared the formalized OOA method with a data flow diagramming (DFD) [9, 14] method. Two considerations motivated these experiments. First, DFD methods are among the most popular for information

systems analysis [1], and thus provide a potential benchmark for evaluating a new systems analysis method. Second, as discussed earlier, the development of an OOA method is based mainly on the premise that the object-oriented paradigm is more beneficial than traditional methods in the whole life cycle of systems development. However, a complete evaluation of the paradigm requires comparison of analysis approaches.

Participants

During sampling, one consideration was that previous skills and knowledge of any particular information systems method would influence the participant's performance and represent uncontrolled biases to the experiment. Participants employed for this experiment must not have taken any MIS courses at the time of the project and have no previous experience in systems analysis. In addition, participants required business knowledge to understand the information system to be analyzed. Second- and third-year undergraduate business students have both of these characteristics.

Thirty-seven undergraduate business students were employed (paid $5 per hour) in the experiment. Five of these were unable to finish. Data from the remaining 32 participants (23 male and 9 female, with a GPA of at least 2.5) provided the basis for evaluating the usefulness of the protocol-based method.

Participants were asked to learn both the data flow diagram (DFD) method and the formalized OOA method, and to analyze an MIS mini-case using the two methods. Afterwards, participants were asked to complete a questionnaire.

Experimental Design

The independent variable was method with two categories (i.e., the DFD method and the formalized OOA method). The determination of dependent variables was not so straightforward. As pointed out earlier in this paper, because of the dearth of experimental comparisons of different MIS analysis methods, few guidelines exist to develop the dependent variables for such experiments. Based on a review of the research (i.e., Sutcliffe and Maiden [46]), and the proposition that the tool of the formalized OOA method is a "language" [30] for coding the real information processing world, the following dependent variables were evaluated: (a) time for learning the methods; (b) time for analyzing a system using the methods; (c) accuracy of the analysis results using the methods; and (d) subjective preference of the participants.

The first two are objective criteria. They are used to evaluate the ease and simplicity of the systems analysis method. These two criteria address surface structure or syntax characteristics of the method. The accuracy criterion is also objective. However, it mainly addresses the ability of the participants to use the analysis procedure in modeling the deep structure or semantic characteristics in the real world. The fourth criterion is subjective. It was primarily used to cross-check the experimental results on the first three objective criteria.

Experimental Procedure and Material

There were three experimental sessions for each participant. The first two were systems analysis sessions, and the third was a questionnaire session. In each of the systems analysis sessions the participants were asked to learn one of the methods (DFD or OOA) and to analyze a mini-case applying the method. To reduce interactive effects between the two sessions, participants were randomly divided into two groups, and each group was assigned one of the two methods first.

There were two subsessions in each systems analysis session. In the first subsession participants were given a six-page anonymous document describing the corresponding systems analysis method together with a simple example of it. The document for the DFD method was from Senn's textbook [40, pp. 701–706], while the document for the formalized OOA method was edited based on the present method (see [51] for a detailed outline of the document). The participants did not know the identity of the authors of the DFD and OOA documents. During the first part of an analysis subsection, each participant was requested to read the document until confident he or she understood the method. Meanwhile, they recorded the degree of their comprehension of the method on a seven-point scale at each interval of an hour spent reading (e.g., a sample question is: How much do you understand the systems analysis method?). Although no time limit was set for reading, all the participants finished within five hours.

In the second subsession of a systems analysis session, participants were requested to analyze a mini-case in management information systems analysis. The case came from McLeod's MIS textbook [27, pp. 419–420]. The same case was used for both approaches to ensure a fair comparison of the methods. The selection of the treatment was based on considerations such as problem scale and richness of semantic characteristics. In the first systems analysis session, participants were given sufficient time (about one hour) to read and fully understand the case. During the analysis subsession of both sessions, participants were given sufficient time to analyze the case using the two individual systems analysis methods—that is, the DFD method and the OOA method. The time spent analyzing the case was recorded for each participant.

After completion of the experimental task, a questionnaire session was administered to elicit subjective ratings of the systems analysis methods used. In the questionnaire session, participants were first requested to recall and briefly describe the two methods. Participants then answered closed-ended questions rated on a seven-point scale, as well as open-ended questions to elicit comments about the two methods.

Analysis and Findings

Time for Learning the Methods

The degree of participants' subjective comprehension (degree of understanding) of the two analysis methods was associated with the time used in learning the two methods. The analysis result is presented in figure 2. In terms of total time used for learning an individual method, no significant difference was found (see Table 4 for a

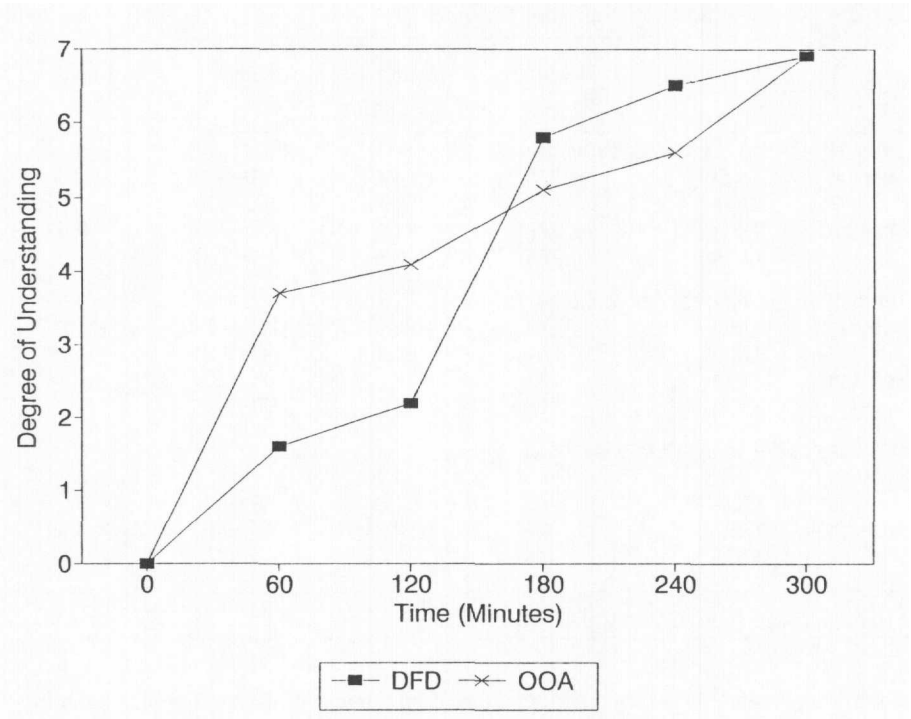

Figure 2.

summary). However, the patterns of the learning progress in the two methods were quite different. As shown in figure 2, in the early stage of the learning process, the OOA method seemed initially to be easier to comprehend than the DFD method. That is, it was found that the participants perceived that they learned OOA more quickly than DFD in the beginning, but the learning progress in OOA then slowed down.

This experiment showed "fatigue effects" due to the intensive time schedule. These effects were not taken into account in the comparison because they were expected to impact equally on the learning of both methods.

Time for Analyzing a System Using the Methods

A *t*-test showed that the OOA method was superior to the DFD method in that it saved time in systems analysis. The difference in time spent in analyzing a system using the two methods was significant (see Table 4 for a summary). According to our observations, participants spent an excessive amount of time developing artificial process modules and exploding from level to level when they were using the DFD method.

Accuracy of the Analysis Results Using the Methods

In evaluating the accuracy of systems analysis results, the focus is on semantic features of participants' solutions rather than on the syntax of the diagramming representation.

Table 4. A Summary of Statistics for the Test

		Means	Standard deviation	Ranges	t-test
Time for learning the methods (minutes)					
Session 1	DFD	283	64.4	210–300	$t_{30} = 0.36^*$
	OOA	291	59.8	220–300	
Session 2	DFD	273	71.3	195–300	$t_{30} = 0.92^*$
	OOA	295	63.5	215–300	
Time for analyzing a system (minutes)					
Session 1	DFD	172	25.9	145–185	$t_{30} = 2.51^{**}$
	OOA	151	21.1	140–180	
Session 2	DFD	124	18.5	95–135	$t_{30} = 1.40^*$
	OOA	115	17.8	90–140	
Accuracy of the analysis results (scores)					
Session 1	DFD	4.5	2.40	2–8	$t_{30} = 3.11^{***}$
	OOA	6.9	1.94	4–10	
Session 2	DFD	4.9	2.81	2–8	$t_{30} = 2.17^{**}$
	OOA	6.8	2.08	3–11	

* Not significant; ** $p < 0.05$; *** $p < 0.01$.

A marking scheme was first developed based on the text of the treatment case. Since natural language is unstructured but powerful in semantic expression [44], a narrative sentence is recognized as an independent semantic statement for the information system and is considered to be a component of a description of the system. Fifteen sentences were identified to be relevant to the information processing. Each was a component of the description of the system. Accordingly, the fifteen components formed a semantic scoring. Participants received a score of one if a component was correctly modeled in the resulting diagram, or zero if it was incorrectly modeled. Correctness scores were considered an appropriate measure of accuracy of analysis. A t-test showed that the OOA method was significantly better than the DFD method in terms of accuracy in modeling an information system (see Table 4 for a summary). A qualitative examination of the participants' analysis reports revealed that the DFD method was defective in modeling timing properties of the systems.

Subjective Preference Based on the Participants' Judgment

Subjective preference based on the participants' judgments of the two systems analysis methods was analyzed. According to the Mann-Whitney statistical test on the ratings, the OOA method was easier to use, the meanings captured in OOA diagrams were more clear, and the unbiased students preferred the OOA method. However, the two methods showed little difference in perceived ease of learning and usefulness. A summary of the attributes evaluated and their ratings is presented in Table 5.

On the basis of the results, it was concluded that the formalized OOA method is effective for systems analysis compared with the data flow diagram method. Given

Table 5. Mann-Whitney Test Result on Questionnaire Data

Attribute	Median*	Mann-Whitney significance	Average rank OOA	Average rank DFD	Point estimate for $ETA_{OOA}-ETA_{DFD}$ and confidence interval (95%)	W (value of Wilcoxson test)
Easy to learn	4.0	N.S.	—	—	—	—
Easy to use	4.0	0.048	19.7	13.3	1.0 (–0.00, 2.00)	315
Clarity of the diagrams	4.0	0.046	19.6	13.3	1.0 (–0.00, 1.00)	315
Usefulness in modeling systems	5.0	N.S.	—	—	—	—
Overall preference	4.0	0.001	22.7	10.3	2.0 (1.00, 3.00)	363

* A higher score means agreement with the statement.
All items were rated on a 7-point scale.

the limited scope of our experiments, this conclusion may have limited applicability to complex systems analysis tasks. There is good reason to believe, however, that the results could be used for generating a priori hypotheses for research investigating more comprehensive systems analysis tasks.

Discussion and Conclusions

A PARTICULAR SYSTEMS DEVELOPMENT APPROACH IS USUALLY ASSOCIATED with a philosophy of computer programming, and a full evaluation of a systems development approach should have a much broader coverage than a discussion of its analysis method. It is rarely possible to prove that one system development approach is better than the others in all aspects. Nevertheless, from our point of view, the "goodness" of a systems development approach could not be concluded without investigating the approach at the systems analysis level.

This research has come one step toward demonstrating the effectiveness and efficiency of the object-oriented systems development approach. The objective of this research program was to contribute to the development of a practical systems analysis method in the object-oriented paradigm. To obtain knowledge about behaviors during systems analysis, a protocol analysis method was applied.

Protocols are a form of data, and can be effectively used to formalize a procedure to describe, in our case, how systems analysts develop models of information systems using object-oriented techniques/methods. However, protocol analysis always involves errors. From this point of view, a method based on the protocol analysis always

has its limitations. In this study, as has been recognized, novice object-oriented systems analysts were employed for the protocols given the fact that sophisticated object-oriented systems analysts are scarce at this time. Consequently, the method formalized in this paper may not be applicable for general cases. Further, the formalized method was compared with the data flow diagrams method based on the performance of undergraduate business students who were naive about information systems analysis. Given such a limited scope of testing, it is not known whether the formalized method is effective and efficient for other cases. Nevertheless, we believe that the formalized method is a good start for the development of object-oriented management information systems analysis methods.

There are potential directions for future research on the problem. One is to strive to improve the present formalized procedures based on independent protocol analyses. It will be possible to collect protocols from experienced object-oriented systems analysts in the future. Another is to test the present research result intensively by employing different types of analysts. It is expected that a comprehensive guideline for object-oriented systems analysis will be developed.

NOTES

1. We are also aware that there are a number of disadvantages of protocol analysis. For example, participants vary in the depth and amount of information they provide [46], and the large volume of verbal data requires a great deal of time for analysis [37].

2. Note that these M.B.A. students were selected for the experiment to illustrate the process of OOA methods, and demonstrate the behaviors involved in their application. It was not the intention of the present study to make generalizations from this experiment.

3. The reader is referred to [53] for a detailed discussion about the identification of inheritance structure in object-oriented systems analysis.

REFERENCES

1. Bailin, S.C. An object-oriented requirements specification method. *Communications of the ACM, 32*, 5 (May 1989), 608–623.

2. Bansler, J.P., and Bodker, K. A reappraisal of structured analysis: design in an organizational context. *ACM Transactions on Information Systems, 11*, 2 (April 1993), 165–193.

3. Barsalou, L. W. Ad hoc categories. *Memory & Cognition, 11*, 3 (May 1983), 211–227.

4. Booch, G. Object-oriented development. *IEEE Transactions on Software Engineering, SE–12*, 2 (February 1986), 211–221.

5. Bouwman, M.J.; Frishkoff, P.A.; and Frishkoff, P. How do financial analysts make decisions? a process model of the investment screening decision. *Accounting, Organizations and Society, 12*, 1 (January 1987), 1–29.

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

7. Coleman, D., et al. *Object-Oriented Development: The Fusion Method*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

8. Crawford, D., ed. Special issue on object-oriented software testing. *Communications of the ACM, 37*, 9 (September 1994).

9. DeMarco, T. *Structured Systems Analysis and Design*. New York: Yourdon, 1978.

10. Embley, D.W.; Kurtz, B.D.; and Woodfield, S.N. *Object-oriented Systems Analysis: A Model-Driven Approach*. Englewood Cliffs, NJ: Yourdon Press, 1992.

11. Ericsson, K.A., and Simon, H.A. Verbal reports as data. *Psychological Review, 87*, 3 (May 1980), 216–251.

12. Ericsson, K.A., and Simon, H.A. *Protocol Analysis: Verbal Reports as Data*. Cambridge, MA: MIT Press, 1984.

13. Fichman, R.G., and Kemerer, C.F. Object-oriented and conventional analysis and design methodologies. *IEEE Computer, 25*, 10 (October 1992), 22–39.

14. Gane, C., and Sarson, T. *Structured Systems Analysis: Tools and Techniques*. Englewood Cliffs, NJ: Prentice-Hall, 1979.

15. Gentner, D. Structure-mapping: a theoretical framework for analogy. *Cognitive Science, 7*, 1 (January–March 1983), 155–170.

16. Guindon, R. Designing the design process: exploiting opportunistic thoughts. *Human-Computer Interaction, 5*, 2–3 (Summer 1990), 305–344.

17. Henderson-Sellers, B., and Constantine, L.L. Object-oriented development and functional decomposition. *Journal of Object-Oriented Programming, 3*, 1 (January 1991), 11–16.

18. Jalote, P. Functional refinement and nested objects for object-oriented design. *IEEE Transactions on Software Engineering, 15*, 3 (March 1989), 264–270.

19. Klersey, G.F., and Mock, T.J. Verbal protocol research in auditing. *Accounting, Organization and Society, 14*, 1–2 (January 1989), 133–151.

20. Korson, T., and McGregor, J.D. Understanding object-oriented: a unifying paradigm. *Communications of the ACM, 33*, 9 (September 1990), 40–64.

21. Krishnan, R.; Li, X.; and Steier, D. A knowledge-based mathematical model formulation system. *Communications of the ACM, 35*, 9 (September 1992), 138–146.

22. Maurer, J., ed. Special issue on object-oriented design. *Communications of the ACM, 33*, 9 (September 1990).

23. Maurer, J., ed. Special issue on next-generation database systems. *Communications of the ACM, 34*, 10 (October 1991).

24. Mantei, M.M., and Teorey, T.J. Incorporating behavioral techniques into the systems development life cycle. *MIS Quarterly, 13*, 3 (September 1989), 257–274.

25. Martin, J. *Principles of Object-Oriented Analysis and Design*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

26. McIntyre, S.C., and Higgins, L.F. Object-oriented systems analysis and design: methodology and application. *Journal of Management Information Systems, 5*, 1 (Summer 1988), 25–35.

27. McLeod, R., Jr. *Management Information Systems*, 5th ed. New York: Macmillan, 1993.

28. Meyer, B. *Object-Oriented Software Construction*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

29. Minsky, M.L. A framework for representing knowledge. In J. Haugeland (ed.), *Mind Design*. Cambridge, MA: MIT Press, 1981, pp. 95–128.

30. Morris, C.W. Foundations of the theory of signs. *International Encyclopedia of Unified Science*, vol. 1, no. 2. Chicago: University of Chicago Press, 1955.

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

32. Nisbett, R.E., and Wilson, T.D. Telling more than we know: verbal reports on mental processes. *Psychological Review, 84*, 3 (May 1977), 231–259.

33. Payne, J.W.; Braunstein, M.L.; and Carroll, J.S. Exploring predecisional behavior: an alternative approach to decision research. *Organizational Behavior and Human Performance, 22*, 1 (August 1978), 17–44.

34. Pei, D., and Cutone, C. Object-oriented analysis and design. *Information Systems Management, 12*, 1 (Winter 1995), 54–60.

35. Rentsch, T. Object oriented programming. *SIGPLAN Notices, 17*, 9 (September 1982), 51–61.

36. Rosson, M.B., and Alpert, S.R. The cognitive consequences of object-oriented design. *Human-Computer Interaction, 5*, 4 (Winter 1990), 345–379.

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

38. Sanderson, P. M.; James, J.M.; and Seidler, K.S. SHAPA: an interactive software environment for protocol analysis. *Ergonomics, 32*, 11 (November 1989), 1271–1302.

39. Schank, R.C. *Dynamic Memory*. Cambridge: Cambridge University Press, 1982.
40. Senn, J.A. *Information Systems in Management*, 4th ed. Belmont, CA: Wadsworth, 1990.
41. Shlaer, S., and Mellor, S.J. *Object-Oriented Analysis: Modeling the World in Data*. Englewood Cliffs, NJ: Yourdon Press, 1988.
42. Shneiderman, B. *Software Psychology: Human Factors in Computer and Information Systems*. Cambridge, MA: Winthrop, 1980.
43. Sowa, J.F. *Conceptual Structures: Information Processing in Mind and Machine*. Reading, MA: Addison-Wesley, 1984.
44. Sowa, J.F. *Principles of Semantic Networks*. San Mateo, CA: Morgan Kaufmann, 1991.
45. Sutcliffe, A.G. Object-oriented systems development: survey of structured methods. *Information and Software Technology, 33*, 6 (1991), 433–442.
46. Sutcliffe, A.G., and Maiden, N.A.M. Analysing the novice analyst: cognitive models in software engineering. *International Journal of Man-Machine Studies, 36*, 5 (May 1992), 719–740.
47. Sweeney, M.; Maguire, M.; and Shackel, B. Evaluating user-computer interaction: a framework. *International Journal of Man-Machine Studies, 38*, 4 (April 1993), 689–711.
48. Todd, P., and Benbasat, I. Process tracing methods in decision support systems research: exploring the black box. *MIS Quarterly, 11*, 4 (December 1987), 493–512.
49. Vinze, A.S.; Sen, A.; and Liou, S.F.T. AEROBA: a blackboard approach to model formulation. *Journal of Management Information Systems, 9*, 3 (Winter 1992–93), 123–143.
50. Vitalari, N.P., and Dickson, G.W. Problem solving for effective systems analysis: an experimental exploration. *Communications of the ACM, 26*, 11 (November 1983), 948–956.
51. Wang, S. Object-oriented systems analysis: a tool for MIS. *Data Resource Management, 3*, 4 (Fall 1992), 12–21.
52. Wang, S. The advantages of an object-oriented CASE environment. *Software Engineering Strategies, 1*, 3 (July–August 1993), 14–21.
53. Wang, S. Object-oriented modeling of business processes. *Information Systems Management, 11*, 2 (Spring 1994), 36–43.
54. Wang, S., and Archer, N.P. Identifying inheritance structure in object-oriented systems analysis—a pattern matching approach. *Journal of Object-Oriented Programming, 7*, 2 (May 1994), 47–55.
55. Wirfs-Brock, R.J., and Johnson, R.E. Surveying current research in object-oriented design. *Communications of the ACM, 33*, 9 (September 1990), 104–124.

APPENDIX A: Segmented Raw Protocol of an OOA

1. I'm going to analyze the payroll processing system.
2. Obviously, employee is an object class.
3. Let's say employee number, employee name, and employee address are the attributes.
4. The operations on these attributes are not uncovered yet,
5. but I would say that "return data" is a generic operation.
6. I'm going to find another object class.
7. umm . . . Paycheck is an object.
8. Pay date, employee name, gross pay, taxes, and net pay are its attributes.
9. At this moment,
10. "print record," "calculate gross pay," "determine tax," and "calculate net pay" are the operations.
11. Paycheck should send a message to employee,
12. to get information about employee,
13. such as employee name, pay rate, and so on.

14. Oh, we may consider two types of employees,
15. Full time employee and part time employee.
16. The difference between them is that
17. different pay rates are applied to them.
18. Flat pay rate for full time employee,
19. and hour pay rate for part time employee.
20. These data will be sent back to paycheck
21. in response to the message.
22. so we have an inheritance relationship
23. between employee and full time employee and part time employee.
24. I think there is a need to keep the payment record in the payroll department.
25. So let's say "payment history" is an object class.
26. Date, employ number, and paid amount should be kept in the record.
27. "Write record" is a generic operation of it.
28. Payment history can be created by paycheck.
29. But, what creates paycheck? . . .
30. Well, let's say, "pay day" is an object class that creates and prints paychecks.
31. "Pay day" also writes "payment history."
32. Now for pay day . . .
33. I think "system clock" will create a "pay day" object,
34. in accordance with the system definition.
35. Let's check the system description . . .
36. "Time card" is an object class,
37. that has employee number and work hours.
38. A generic operation is "return data."
39. "Paycheck" gets information about work hours from time card
40. to calculate the gross pay.
41. I think that "verify time card" is an operation included in "paycheck."
42. We need "tax table" to determine tax.
43. "Tax table" is an object class,
44. and consists of various tax rates.
45. What else . . .
46. Okay . . . Let's check it over.
47. I start with paycheck,
48. because it is important to this system.
49. Paycheck is created by pay day.
50. Pay day is in turn created by system clock.
51. Date comes from pay day.
52. Employee name and employee number come from employee.
53. Gross pay is calculated by an internal operation,
54. using data of pay rate which comes from employee,
55. and data of work hours which comes from time card.
56. Net pay is calculated

57. based on gross pay and tax table.
58. Here, we have also included "verify time card."
59. "Pay day" also creates "pay history" to keep record.
60. Date comes from pay day.
61. Employee number comes from employee
62. through the messages between employee and paycheck,
63. and between pay day and paycheck.
64. Net pay comes from paycheck through pay day.
65. It seems to me that every data item is completed.
66. Let's check inheritance structure.
67. Full-time employee and part-time employee inherit from employee.
68. I think that's it.

APPENDIX B: Notations of the OOA Method and an Example of OOA Result

According to this OOA method, the object class modulars, inheritance structures, messages, and parameters of the messages are all specified in a single diagram. One may find that the diagrams (figures 3 and 4) contain both the system analysis and system logical design features of the traditional structured methods. Such analysis diagrams could be used to convert into object-oriented programs directly [50, 51].

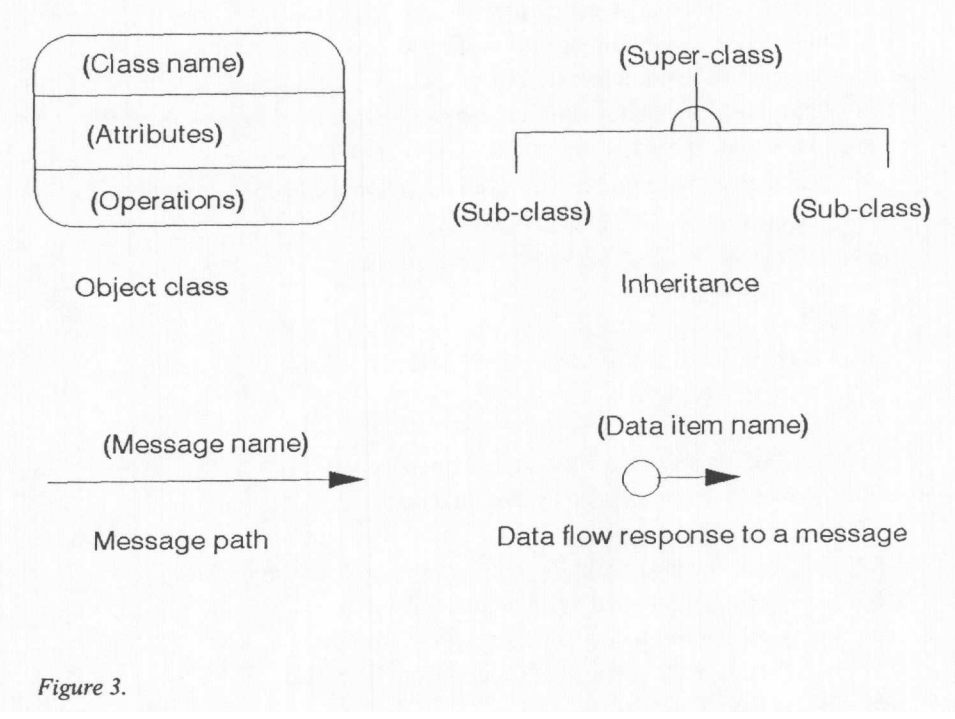

Figure 3.