A Formal Approach to the Comparison of Object-Oriented Analysis and Design Methodologies

Shuguang Hong
Georgia State University
Dept. of Computer Information Systems
P.O. Box 4015
Atlanta, GA 30302-4015
(404)651-3880
cisssh@gsusgi2.gsu.edu

Geert van den Goor[1]
University of Nijmegen
Dept. of Information Systems
P.O. Box 9010
6500 GL Nijmegen
The Netherlands
geertg@cs.kun.nl

Sjaak Brinkkemper
University of Twente
Dept. of Computer Science
P.O. Box 217
7500 AE Enschede
The Netherlands
sjbr@cs.utwente.nl

Abstract

This paper presents a formal approach to the comparison of six object-oriented analysis and design methodologies. For each of the methodologies, a formal representation of it is constructed as a meta-process model and a meta-data model. These two meta models of a methodology represent the steps of the analysis and design, the concepts, and the techniques provided by this methodology. Based on this uniform representation, an extensive comparison of these six methodologies is then performed. The results are given as a set of tables in which the similarity and differences of these methodologies are exhibited. Adopting this formal approach, we can avoid errors caused by misunderstanding or misinterpretation of these methodologies. Consequently, an accurate and unbiased comparison is made possible.

1: Introduction

We have witnessed the overwhelming popularity of object-oriented analysis and design in recent years. This phenomenon is evidenced by the number of articles and papers published in various journals, conference proceedings, books, and other forms. We expect to see an even large incoming stream of publications in the next few years. Several large projects have adopted object-oriented technology [15, 25]. The positive results from those projects have convinced more and more organizations that want to adopt this new paradigm.

However, the object-oriented analysis and design methodology (OOADM) is still in its growing phase and continues to evolve. There are more than a dozen popular OOADMs that have been widely circulated, but no single one of them has gained a wide acceptance. Several fundamental issues have not yet been agreed upon by the community. For example, some OOADMs (e.g., [17]) permit an object to dynamically change its class memberships while other OOADMs (e.g., [2, 19]) do not allow an object to switch its class memberships over time.

Given this early state of the development of the OOADM, an organization that is planning for a transition to object-oriented technology faces one critical question: *which OOADM should be chosen?* The answer to this question demands a systematic comparison of available OOADMs before any commitment can be made to one of them. Unfortunately, the comparison of these methodologies is complicated by the fact that the currently popular OOADMs are composed of informal descriptions [10], and each methodology has its own set of definitions of the concepts, techniques, and notations. A comparison of the methodologies is, therefore, subject to the perceptions and interpretations of the person who performs the task.

To be accurate and objective, OOADMs should be compared based on a uniform, formal, and unbiased basis. This is the approach that we take in our study of six popular OOADMs [23]. The research is divided into two phases. The first research phase is to perform meta modeling of the selected six methodologies. For each methodology, we extract its goal and approach as well as its concepts, techniques, steps, and graphical notation. Based on the collected information, we construct the following two meta models:

- *Meta-process model* that describes the analysis and design steps advocated by each methodology. It also shows the input to and the output from each step.
- *Meta-data model* that represents the concepts and techniques provided by each methodology. It depicts both the definitions of the concepts and the relationships among them.

Using the same meta modeling constructs for all methodologies, we can obtain the uniform and formal representation of these methodologies.

Using the meta-models of these methodologies, the second research phase compares these methodologies in the following categories:

- The analysis and design steps,
- The concepts, and
- The techniques provided.

The results of this research phase include a set of tables that provide extensive comparison of these methodologies. Based

[1] Geert van den Goor is now with Andersen Consulting, Stadhoudersplantsoen 24, 2517 JL The Hague, The Netherlands

on the meta-models and the tables, we can understand the similarities and differences of these OOADMs. We also identify a number of questions that need to be addressed, but for which there is no sufficient information available yet for the methodologies.

This paper presents the research approach of this project while the complete description of the project can be found in [23]. Section 2 discusses the related research that influenced our approach. Section 3 provides a short description of each OOADM to be compared. The meta-modeling of the six OOADMs is given in Section 4, and the comparison of these methodologies is shown in Section 5. Finally, Section 6 concludes the comparison and points out further research directions.

2: Related Research

De Champeaux and Faure at Hewlett Packard Laboratories has initiated a systematic comparison of OOADMs [10]. By surveying more than ten object-oriented analysis methodologies, de Champeaux compared the common features and the major differences of the chosen methodologies. His article provides an excellent tutorial for object-oriented analysis, but his comparison of the methodologies is somewhat abbreviated.

In the same direction, research has been done in Hewlett Packard Laboratories by Arnold and his colleagues [1]. They defined several comparing criteria and performed an extensive comparison of these methodologies. The results were presented in a set of tables.

Wirfs-Brock and Johnson have provided an excellent survey about several research projects of object-oriented design in academia and in industry [25]. Although no direct comparison was performed, they highlighted the major concepts, processes, and notations of the methodologies proposed by the research projects.

The above research has influenced our decision on the selection of OOADMs to use in our comparisons and the criteria we should use to compare these methodologies. However, we take a different research path by building a formal representation for each methodology and comparing these methodologies based on the uniform representation. In contrast, no uniform and formal basis was developed in previously published research. The comparisons were based on the informal description of the methodologies.

The technique we use to construct the formal representation of an OOADM is meta-modeling. Meta modeling has been used to develop theoretical foundations for information systems. Brinkkemper has adopted this technique to build a framework for the formalization of information system modeling [3, 4]. The analysis and design of information systems can be viewed as the conceptual modeling of the systems, and the development of an information system is the modeling process that starts with the Universe of Discourse and ends at the information system. Hence, various analysis and design methodologies can be viewed as specific modeling processes that apply a set of predefined techniques. He proposes a formal framework for information system modeling and addresses the issues of what is the best way to construct a model and how one particular model is related to the other models of an information system.

Due to the lack of a common core of theory in information systems, Ahituv [Ahituv 87] built a meta model that views an information system as the data flow that transfers from one state to another. He used this meta model to show that some existing models can be considered as emanating from the meta model and demonstrated its advantages.

Addressing the similar concern, meta modeling has been used to build database systems that support multiple data models [9, 18]. In these systems, a meta data model was built to understand the modeling concepts of different data models. Recently, meta modeling has been applied to the object-oriented data modeling [13, 14]. In this research, a meta data model is used to represent the concepts of various object-oriented data models so that database system software that is specific to the data models can be automatically generated.

The research above greatly influenced this project in performing meta modeling of the OOADMs. However, our goal in using meta modeling is to build a uniform and formal basis for an accurate and objective comparison of the OOADMs.

3: Object-Oriented Analysis and Design Methodologies

The six OOADMs we chose to compare are the ones proposed by Booch [2], Coad and Yourdon [7, 8], Martin and Odell [17], Rumbaugh et al. [19], Shlaer and Mellor [20] and Wirfs-Brock et al. [24]. The selection of these methodologies was based on the following two criteria:

- The methodologies must be accepted by the community as real object-oriented analysis and/or design methodologies, and
- They should be published in textbook form so that sufficient information is available.

All of the chosen methodologies satisfy both criteria [1, 10]. These criteria might exclude recent developments in the OOADM field or some well-known OOADMs, but we view maturity, sufficiency, and general acceptance of methodologies as the primary requirements for the I/S development practice. To facilitate discussions in the following sections, we provide a short profile for each of the six methodologies in the following paragraphs.

Object Oriented Design with Applications (OODA) by Booch [2]

Booch's method is mainly intended for the design stage of a project. Booch describes a number of general properties of well-structured complex systems. Systems built with the OODA methodology should satisfy these properties. In OODA the problem domain is modeled from two different perspectives: the logical structure of the system and the physical structure of the system. For each perspective both static and dynamic semantics are modeled. OODA describes

various techniques to accomplish these tasks, and provides a rich set of graphical notations.

Object Oriented Analysis and Object Oriented Design (OOA/OOD) by Coad & Yourdon [7, 8]

This methodology is based on a number of general principles for managing the complexity of systems. During the analysis phase the problem domain is modeled in five layers in which objects, classes, relationships, the inheritance structures, message connections, and others are captured. In the design phase these layers are refined according to four components: a Problem Domain component, a Human Interaction component, a Task Management component and a Data Management component. Graphical notations are available for depicting the five-layer model of the problem domain, the dynamic behavior of objects, and the functional structures.

Object Oriented Analysis and Design (OOAD) by Martin & Odell [17]

This methodology is based on the logic and set theory. The authors give a very extensive introduction to all concepts and discuss the management of the complexity of the system as their main objective. The methodology strongly focuses on describing the behavior of objects. Techniques have been developed to identify objects and their relationships, to describe the dynamic behavior of objects and to capture the high level business processes.

Object Modeling Technique (OMT) by Rumbaugh, et al. [19]

This methodology aims to create stable programs by placing the emphasis on data instead of functions. The methodology is divided into three stages: Analysis to describe the problem domain, Systems Design to design the overall structure of the system, and Object Design to refine the Object structures toward an efficient implementation. The methodology provides techniques to describe the problem domain from three different perspectives: the static structure of Objects and Classes, the dynamic behavior of Objects, and the functional structures.

Object Oriented Systems Analysis (OOSA) by Shlaer & Mellor [20]

This methodology covers object-oriented analysis and provides a methodology to overcome problems that are faced in the Structured Analysis approach. The main focus of the methodology is to describe the static aspects of Objects; the dynamic and functional aspects are discussed briefly. Techniques are proposed for modeling the static, the dynamic and the functional aspects of objects.

Designing Object Oriented Software (DOOS) by Wirfs-Brock, et al. [24]

This methodology covers mainly the analysis phase of the systems development life cycle. Two major concepts, abstraction and encapsulation, are used to manage the real-world complexity. The DOOS methodology describes the problem domain as a set of collaborating objects. A system is developed in two stages. During the initial exploratory phase objects, their responsibilities and the necessary collaborations to fulfill these responsibilities are identified. The detailed analysis phase streamlines the results of the first phase. Two graphical techniques are introduced for the second phase. One technique is to show classes and class structures and the other is to depict classes, subsystems and client-server relationships.

4: Meta-Modeling of OOADMs

Meta-models are conceptual models of modeling methodologies or techniques. There are two aspects of a systems development methodology: the *processes* that are the steps with corresponding input and output products and the *concepts* that are used to construct the representation of the intermediate and final products of the methodology. In structured analysis, for example, the *processes* correspond to the steps that lead an analyst to build data flow diagrams from requirements specification, and the input and output products are the results of each analysis step, such as data flow diagrams at different levels. The *concepts*, in this example, consist of data store, process, data flow, etc. These two aspects, processes and concepts, are analogous to the well-known dichotomy of control and data of software systems [12].

In this project, the processes of each of the six OOADMs is captured in a *meta-process model*, while its concepts and the associations among the concepts, as applied in the various diagrammatic and textual techniques of the methodology, are described by a *meta-data model*. The meta-process models and meta-data models provide the blueprints of all these methodologies and are used side by side for an extensive comparison, which is discussed in the next section.

Due to limitations of space, this paper will illustrate the meta-modeling approach with one meta-process model and two meta-data models and refer interested readers to the complete set of meta-models of these six OOADMs in [23]. This meta-modeling approach is similar to that applied to the comparison of information planning methodologies in [3]. In the following discussion, we treat the terms *step* and *activity* interchangeably.

4.1: Meta-Process Model

The incomplete meta-process model for the methodology (DOOS) by Wirfs-Brock, et al. [24] is shown in Figure 4.1. The activities of the methodology are denoted in rounded rectangles and the intermediate and final products are in parallelograms. The output dependencies between activities are indicated by arrows. Note that the arrows between activities should not be interpreted as the indication of the activity sequence; they merely indicate the logical sequence for constructing the output products. All methodologies explicitly stress that the object-oriented analysis and design activities should be iterative and any sequential order should not be imposed on them.

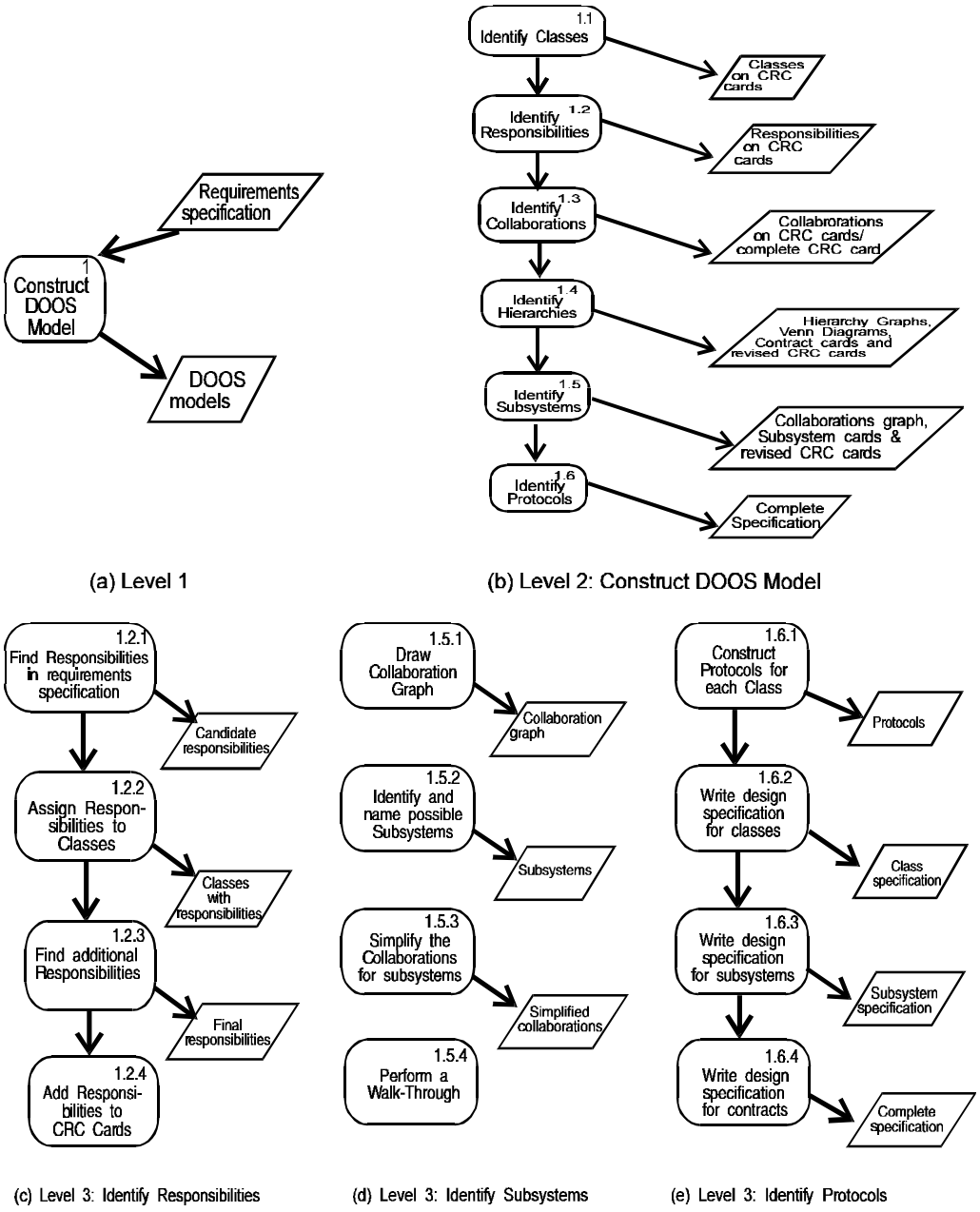

Figure 4.1: Meta-Process Model of DOOS Methodology (Excerpt)

The activities DOOS are depicted in multiple levels of details. For example, at Level 1 there is one generic activity *Construct DOOS Model* that consists of six activities at Level 2. Each of activities at Level 2 is decomposed further into activities at Level 3. Every activity is given a unique identifier that is used as a reference of it in the next section where the activities of all OOADMs are compared.

The meta-process only shows the products generated by each activity. For example, the activity *1.2.2: Assign Responsibilities to Classes* results in a list of *Classes with Responsibilities*. For simplicity and readability of the diagram, we deliberately omitted the input arrows from intermediate products to activities because these methodologies all assumed that output from any activity is globally accessible by all other activities.

The notation adopted to show the meta-process model is called *Task Structure Diagrams* and was developed for the SOCRATES meta-CASE environment [5, 6, 22]. Task structure diagrams allow further modeling of the decomposition of activities and may include decisions as separate meta-modeling notions.

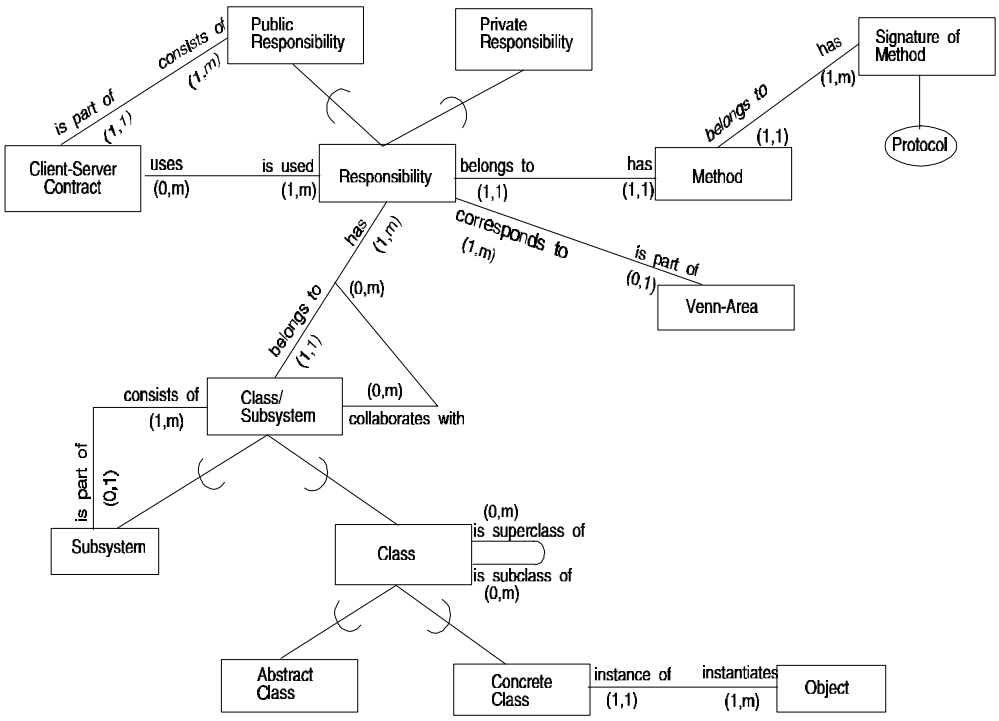

Figure 4.2 Meta-Data Model of the DOOS Methodology

4.2: Meta-Data Models

Figures 4.2 and 4.3 depict the meta-data model for the methodology (DOOS) by Wirfs-Brock, et al. [24] and the meta-data model for the methodology (OOA/OOD) by Coad and Yourdon [7, 8], respectively. As shown in these two figures, we adopt the Extended Entity Relationship (ER) model as proposed in [11]. In these figures, concepts of the methodologies are mapped to entity types such as *Class* and *Object*. Associations and constructs of the concepts are modeled as relationships with the cardinality constraints. In Figure 4.3, for instance, the construct of *Inheritance* (generalization-specialization structure) in OOA/OOD is represented as a relationship, *is-generalization-of -- is-specialization-of*, between two *Classes* with (0,m) cardinality. In a methodology, a concept may be the subconcept of other concept. Such a relationship is also modeled in the meta-data model. For example, DOOS has the concepts: *Class, Abstract Class*, and *Concrete Class*. These concepts are related in Figure 4.2 by the *IS-A* relationships, i.e., *Abstract Class* and *Concrete Class* are both subconcepts of *Class*.

If, at the end of the analysis/design, several final products are yielded, the concepts and associations that are used to represent the products are grouped into clusters (modules) with thick border lines in the meta-data model. The connections among the products are represented as relationships across clusters. For example, in the OOA/OOD methodology there are three final products, *Service Chart, OOA Diagram*, and *Object State Diagram*. These three products were shown in Figure 4.3 as three clusters, respectively. One of them, the *Object State Diagram*, contained two concepts, *Transition* and *State*, and two relationships between them. There are two relationships that link an *OOA Diagram* to an *Object State Diagram*; one is from *Class* to the *Object State Diagram* cluster, and the other is between *Service* and *State*. The former indicates that a class could have a state diagram that describes the states of its objects over time, while the latter indicates that the state behaviors are defined by *Service*.

Some remarks should be made on the meta-modeling approach. Although in principle an OOADM should be suitable, none of the proposed OOADMs could have been used for meta-modeling in this research project, since this would create a prejudice towards one of the methodologies. Also, we could have included more details in the meta-data models and meta-process models. However, it was not necessary for this particular comparison project because the information captured by these models is sufficient for an extensive comparison.

5: Comparison of the Methodologies

In this project the comparison of the six OOADMs was performed in three categories: the process, the concepts, and the techniques the methodologies provided. The comparison drew information mainly from the meta-process models and meta-data models as discussed in the previous section. Limited by the space, we present a part of results of the comparison and refer the reader to the complete results in [23]. At the end of this section, we also provide a short discussion of the implementation issues when these OOADMs are employed.

5.1: Comparison of the Processes

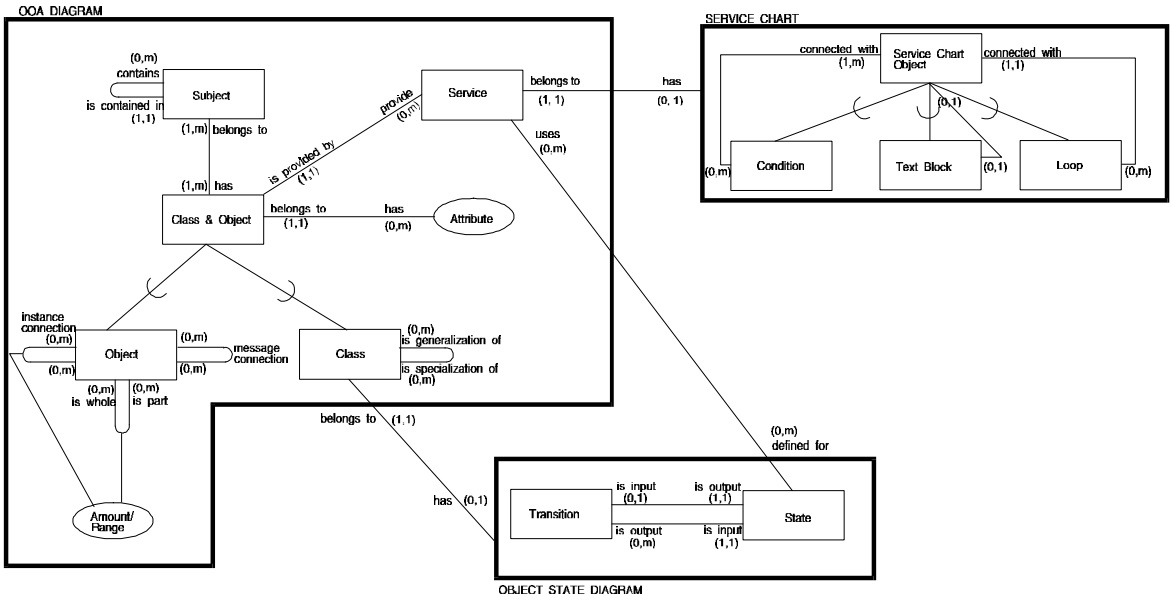

Figure 4.3 Meta-Data Model of the OOA/OOD Methodology

The comparison of the processes is performed by aligning the steps of the OOADMs side by side and revealing the similar and different activities of the analysis and design. There are several approaches of comparison, such as comparing all OOADMs to one of them or creating an entirely new methodology to which these OOADMs are compared. After carefully evaluating the possible alternatives based on the principle of unbiased comparison, we take the approach of creating a so-called *supermethodology* as the target to compare. This supermethodology is defined as the smallest common denominator of all activities depicted in the meta-process models of the OOADMs. The partial results of the comparison are listed in Table I; while the complete table contains over 100 rows. In Table I, the activities of the supermethodology are listed in the leftmost column, and each OOADM occupies one column. The following notations are used in the table:

- The activity identifier in its meta-process model. If this identifier is the same as that of the supermethodology, it is omitted.
- A comparison indicator that compares an activity *s* of the supermethodology to an activity *m* of an OOADM as follows:

  s '=' *m* The activity *s* is equivalent to the activity *m*.

  s '>' *m* The activity *s* does more than the activity *m.*

  s '<' *m* The activity *s* does less than the activity *m.*

  s '><' *m* A part of the activity *s* overlaps a part of the activity *m* and the other parts of both activities do not overlap.

  Blank This activity is absent from the OOADM.

For instance, Activity 1.2.1 (Identify Objects and Classes) of the supermethodology is equivalent to Activity 1.1.2 of DOOS as shown in Figure 4.1; this activity of DOOS tries to find candidate classes abstracted from various objects. This supermethodology activity does more than that of Activity 1.1.1 of OOSA [20] which identifies only objects. Note that this activity is absent from OOAD [17].

From Table I, we have drawn extensive conclusions about the similarity and differences in these OOADMs. For instance, this table shows that OOAD [17] lacks detailed analysis and design steps although OOAD provided very extensive discussion about the analysis and design concepts and constructs. Because our objective in this paper is to demonstrate the formal comparison approach, we omitted the discussion of the comparison remarks. See [23] for details.

5.2: Comparison of the Concepts

The concepts of the six OOADMs are compared in the following categories:

- The main concepts, such as Class, Object, etc.
- The main relationships such as Inheritance and Whole-Part Structures.
- The built-in operations such as Read, Write, etc.
- The types of communications between objects.
- The kinds of concurrency mechanisms.

TABLE I: COMPARISON OF ACTIVITIES (Excerpt)

ACTIVITY OF SUPERMETHODOLOGY	OOA/ OOD	DOOS	OMT	OODA	OOSA	OOAD
1. CONSTRUCT THE MODEL						
1.1. Study Requirements						
1.1.1 Understand Requirements Specification	= 1.1.1	= 1.1.1	= 1.1			< 1.1.1
1.2. Find Objects and Classes						
1.2.1 Identify Objects and Classes	= 1.1.2	= 1.1.2	= 1.2.1	= 1.1.1	> 1.1.1	
1.2.2 Name Classes and Objects Properly	= 1.1.3	< 1.1.3			= 1.1.3	
1.2.3 Describe Classes and Objects	= 1.6.2	= 1.1.6	= 1.2.2	>1.2.1 < 1.3.4	= 1.1.2	
1.2.4 Apply Guidelines to Control Classes & Objects	= 1.1.4	<1.1.3			=1.1.4	
1.2.5 Identify Abstract Classes		=1.1.4				
1.2.6 Search for Missing Classes		=1.1.5				
1.3 Identify Relationships						
1.3.1 Identify Inheritance Relationships	= 1.2.1	= 1.4.4	= 1.2.5	= 1.3.2	> 1.1.9	
1.3.2 Identify Part-of Relationships	= 1.2.2		< 1.2.3			
1.3.3 Identify Multiple Structures	=1.2.3					
1.3.4 Identify Associations	=1.4.3		<1.2.3	=1.3.1	=1.1.8	
...						
1.4 Define Attributes						
1.4.1 Identify Attributes	=1.4.1		=1.2.4	<1.3.4	<1.1.5	
1.4.2 Position Attributes	=1.4.2				<1.1.5	
1.4.3 Check Attributes	<1.4.4					
1.4.4 Describe Attributes	<1.4.5				=1.1.7	
...						
2. REFINE THE MODEL						
2.1 Write Design Specification						
2.1.1 Write Design Spec for Classes		=1.6.2				
2.1.2 Write Design Spec for Subsystems		=1.6.3				
2.1.3 Write Design Spec for Contracts		=1.6.4				
2.2 Define Modules						
2.2.1 Rearrange Classes & Operations			=3.5.1			
2.2.2 Abstract out Common Behavior			=3.5.2			
2.2.3 Use Delegation to Share Implementation			=3.5.3			
...						
2.3 Refine Methods						
2.3.1 Construct Protocols		=1.6.1				
2.3.2 Choose Algorithms			=3.2.1			
2.3.3 Choose Data Structures			=3.2.2			
...	...	...	...	...	...	...

We follow a similar approach as that for comparing the processes. A super set of concepts is derived from the meta-data models of these six OOADMs and is used as the comparison criteria for the concepts of these OOADMs. The results of the comparison form a table with over 100 rows. A subset of this table is extracted and displayed in Table II in which the concepts of the supermethodology are shown in the leftmost column. The notations that are different from that of TABLE I are as follows:

'*Strings*' ── This concept is equivalent to that of the supermethodology but the term '*String*' is used.

'(*number*)' ── It provides a footnote to the concept compared.

In Table II, for example, OOA/OOD [7, 8], OMT [19], OODA [2] and OOAD [17] have the *Whole-Part* relationship concept except that OOAD calls this relationship as *Composition* relationship, but DOOS [24] and OOSA [20] have no similar relationship.

5.3: Comparison of the Techniques

Eight different techniques are provided by these OOADMs to help an analyst/designer capture objects, classes, partitioning of the analysis, object dynamics, system dynamics, functional behavior, communication between objects, and implementation properties. The comparison results are shown in Table III. In this table the concept to which a technique is applied is listed in the leftmost column. Each entry of the table provides the name of the technique used by an OOADM.

As shown in Table III, different methodologies may use different techniques to model the same concept. For example, to model the dynamic aspect of objects, OOA/OOD [7, 91b], OMT [19], OODA [2], and OOSA [20] use a technique similar to the state transition diagram, while OOAD [17] uses event schema. Naturally, there is a question on whether the same

TABLE II: COMPARISON OF CONCEPTS (Excerpt)

Concepts of Supermethodology	OOA/ OOD	DOOS	OMT	OODA	OOSA	OOAD
MAIN CONCEPTS						
Class & Objects	=					
Class	=	=	=	=	Object	Object Type
Abstract Class		=	=	=		
Meta Class			=	=		
Object	=	=	=	=		=
Passive Object				=		
Active Object				=		
Attribute	=		=	Field	=	
Derived Attribute			=			
Attribute Constraint	=		=	=	=	
Method	Service	Responsibility	Operation	Operation		Operation
Method Signature	Parameter	=	=	Operation Parameter		
Subject	=	Subsystem	Module	Class Category		
…						
RELATIONSHIPS						
Inheritance	Gen-Spec	Super/ subclass	Super/ subclass	Super/ subclass	Super/ subtype (1)	Super/ subtype
Multiple Inheritance	=	=	=	=		=
Whole-Part relationship	=	(2)	=	=		Composition
Association	Instance connection		=	Using relationship	relationship	Relation
Derived association			=			Computed functions
Message connection	=	Collaboration		Message relationship		
Instantiation relationship				=		
…	…	…	…	…	…	…
OPERATIONS …	…	…	…	…	…	…
COMMUNICATION …	…	…	…	…	…	…
CONCURRENCY …	…	…	…	…	…	…

(1) OOSA only supports inheritance of Attributes. (2)DOOS does not really have a whole-part concept.

technique, say state transition diagram, provided by different OOADMs is precisely the same. However, in this project, we do not attempt to address this issue which is beyond our research.

5.4: Implementation Issues

To find out the smoothness of the transition from an OOADM to the implementation, we decided to compare the concepts of the six OOADMs to the concepts of the six most popular object-oriented programming languages (OOPLs). This comparison should not be mistaken as the evaluation of the degree of coupling between OOPLs and OOADMs. Instead we compare how they might be matched. As pointed out by de Champeaux [10], an OOADM should be independent of any implementation details. Table IV shows a subset of the table for DOOS [24]. Complete tables can be found in [23].

Database management systems are very important for the implementation of an information system. We believe that it is worthwhile to go one step further and to survey which OOPL is supported by object-oriented database management systems (OODBMS). After surveying ten commercial OODBMS products[2] C++ is the only OOPL that is supported by all OODBMS vendors[3], while a few OODBMS also support Smalltalk.

6: Conclusion

The main contribution of this research is the use of the meta-modeling technique to build a formal representation of six OOADMs and the comparison of the OOADMs based on their uniform representation. This approach enables us to perform a more accurate, unbiased, and extensive comparison as shown in this paper. In this way, errors of misunderstanding or

2 G-Base/GTX from Object Databases, GemStone from Servio Logic, ITASCA from Itasca Systems, O2 from O2 technology, Object-Base from Versant Object Technology, ObjectStore from Object Design, Objectivity/DB from Objectivity, ONTOS from Ontologic, OpenODB from Hewlett-Packard, and UNISQL/X from UNISQL.

3 Some OODBMS's do not directly support persistent C++ objects, conversion from the C++ objects to the database objects is required.

misinterpretation of the methodologies can be detected and, therefore, can be avoided during the comparison process.

TABLE III: COMPARISON OF TECHNIQUES

Technique to capture:	OOA/OOD	DOOS	OMT	OODA	OOSA	OOAD
1. Objects	Object layer OOA model			Object Diagram		
2. Classes/Class Structures	Class layer OOA model	CRC cards; Venn diagr.; Hierarchy Graphs	Object Diagram	Class Diagram	Information model	Object schema
3. Partitioning of the Class and Class Structures	Subject layer OOA model	Subsystem Cards	Modules	Class Category Diagrams		Object Flow Diagram
4. Object Dynamics	Object State Tables	(1)	State Diagrams	State Diagrams	State Transition diagrams (2)	Event Schema
5. System Dynamics				Timing Diagrams		
6. Functional behavior	Service charts		Data Flow diagrams		Data Flow diagrams (2)	
7. Implementation properties			subsystems	Module/ Process Diagrams		
8. Communication between classes/ objects	Message connec-tions OOA model	Collaborations graphs	Event Flow Diagrams	Synchronization on Object Diagram		

(1) The dynamic behaviors of object is not explicitly specified.

(2) This technique is provided but it is described very briefly.

Secondly, our research results provide information system professionals an extensive survey of these six OOADMs and can assist information system professionals in the evaluation and study of these methodologies. Furthermore, these results are a valuable resource for organizations that are planning for a transition to object-oriented technology. The meta-models and comparison tables provide blue-prints to correlate the present I/S practice with some alternatives for this new technology. Finally, the formal representation of these methodologies can be used to build a CASE tool that would support multiple OOADMs. The multi-methodology CASE tool concept is originated from the research area called *methodology engineering* [16].

We deliberately avoided to rate these methodologies. First, a standard on what is a good OOADM would be required for rating these OOADMs, which is not feasible because of the current state-of-the-art of and the divergent views on object orientation. We feel that none of these methodologies has reached its mature stage and they will continue to evolve. Because of the rapid advance of object-oriented technology, any conclusion we might draw would quickly become invalid. Secondly, the quality of a methodology should be measured from the all perspectives, such as the complexity of and the scale of applications and the I/S development practice in an organization that wants to adopt an OOADM. This issue itself is a separate research topic.

A limitation of this research is that we do not compare the guidelines and rules provided by each OOADM. A formal system must be employed for this purpose. We have put a great effort in building the meta-models so that they are as accurate as possible. However, limited by the Entity Relationship model, several concepts of some OOADMs are very difficult to represent. Consequently, the accuracy of the comparison results may be affected. A better meta model might be used to overcome the problem. Finally, and most importantly, we do not compare how an OOADM guides the user to design a better software system and to take the maximal benefits of object-oriented technology, such as reusability. These issues demand further research.

References

[1] Arnold, P., Bodoff, S., Coleman, D., Gilchrist, H., Hayes, F., *An Evolution of Five Object Oriented Development Methods*, Research report, HP Laboratories, June 1991.

[2] Booch, G., *Object-Oriented Design with Applications*, The Benjamin/Cummings Publishing Company Inc., Redwood City, CA, 1991.

[3] Brinkkemper, S., Geurts, M., van de Kamp, I., Acohen, J., "On a Formal Approach to the Methodology of Information Planning," In: *Proceedings of the First Dutch Conference on Information Systems*, R. Maes (Ed.), 1989.

[4] Brinkkemper, S., *Formalisation of Information SystemsModelling*, Thesis Publishers, Amsterdam, The Netherlands, 1990.

[5] Brinkkemper, S., M. de Lange, R. Looman and F.H.G.C. van der Steen, "On the Derivation of Method Companionship by Meta-Modelling,". In: *Advance Working Papers, Third International Conference on Computer Aided Software Engineering*, Ed. J. Jenkins, Imperial College, London, UK, July 1989. pp. 266-286. Also in: *Software Engineering Notes*, journal of the Special Interest Group on Software Engineering of the ACM, vol. 15, nr. 1, January 1990, pp. 49-58.

[6] Brinkkemper, S., A.H.M. ter Hofstede, T.F. Verhoef and G.M. Wijers, "A Meta-Modeling Based CASE Shell to Support Customized Domain Modeling," In *the Proceedings of the ICSE Workshop on Domain Modeling*, Eds. N. Iscoe, G.B. Williams and G. Arango, Austin, TX, USA, May 1991, pp. 31-36.

[7] Coad, P., Yourdon, E., *Object Oriented Analysis* (2nd Edition), Yourdon Press, Englewood Cliffs, N.J., 1991.

[8] Coad, P., Yourdon, E., *Object Oriented Design*, Yourdon Press, Englewood Cliffs, N.J., 1991.

[9] Demurjian, S.A. and Hsiao, D.D., "Towards a Better Understanding of Data Models Through the Multilingual Data Systems," *IEEE Transactions on Software Engineering*, (14, 7) July 1988, pp. 946-958.

TABLE IV: Concepts Matching Between OOPLs and DOOS (Excerpt)

Major Concepts of DOOS	Smalltalk	Objective- C	Eiffel	Object Pascal	C++	CLOS
Class	=	=	=	Object type	=	=
Object	=	=	=	Object	=	Instance
Abstract Class	< Class	< Class	< Class	< Object type	< Class	< Class
Concrete Class	Class	Class	Class	Object type	Class	Class
Subsystem		> Files	?	>Unit	>Files	Package
Public Responsibility	Public method	Public method	Exported method	Function/ Procedure	Public member function	Public method
Private Responsibility	Private method	Private method	(Default)		Private member function	
Contract						
Single Inheritance	Super-subclass	Super-subclass	Ancestor-descendant	Ancestor-descendant	Derived classes	Superclass
Multiple Inheritance			=		Derived classes	=
Collaboration	> Message	> Message	> Routine call	> Function procedure call	Function calls	Function calls

[10] de Champeaux, D. and Faure, P., "A Comparative Study of Object Oriented Analysis Methods," *Journal of Object-Oriented Programming (JOOP)*, March/April, 1992, pp. 21-33.

[11] Elmasri, R. and Navathe, S, *Fundamentals of Database Systems*, The Benjamin/Cummings Publishing Company Inc., 1989.

[12] Harel, D., "Biting the silver bullet: toward a brighter future for system development," *IEEE Computer*, Vol. 25, no. 1, January 1992, pp. 8-20.

[13] Hong, S. and Maryanski, F., "Using a Meta Model to Represent Object-Oriented Data Models," *Proceedings of IEEE Six International Conference on Data Engineering*, Feb. 1990, pp. 11-19.

[14] Hong, S. and Maryanski, F., "Representation of Object-Oriented Data Models," *Information Sciences*, 52, Dec. 1990, pp. 247-284.

[15] Korson, T., et. al., "Managing the Transition to Object-Oriented Technology (Panel)," in *Proceedings of OOPSLA*, Oct. 1992, pp. 355-358.

[16] Kumar, K. and Welke, R., "Methodology Engineering: A Method for Situation Specific Methodology Construction," in *Systems Analysis and Design: a Research Agenda*, eds. W.W. Cotterman and J.A. Senn, forthcoming.

[17] Martin, J., Odell, J., *Object Oriented Analysis and Design*, Draft manuscript, 1992.

[18] Morgenstern, M., "A Unifying Approach for Conceptual Schema to Support Multiple Data Models," *Entity-Relationship Approach to Information Modeling and Analysis*, Editor P.P. Chen, North-Holland Corp., 1983, pp. 279-297.

[19] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., *Object Oriented Modelling and Design*, Prentice-Hall, Englewood Cliffs, N.J., 1991.

[20] Shlaer, S., Mellor, S.J., *Object-Oriented Systems Analysis: Modeling the World in Data*, Yourdon Press, Englewood Cliffs, N.J., 1988.

[21] Sorenson, P., J.-P. Tremblay and A. McAllister, "The Metaview system for many specification environments," *IEEE Software*, vol. 5, no. 2, March 1988, pp. 30-38.

[22] Verhoef, T.F., Hofstede, A.H.M. ter, and G.M. Wijers, "Structuring Modelling Knowledge for CASE Shells," In: *Proceedings of the CAiSE 91 Conference*, Trondheim, Norway, *Lecture Notes in Computer Science*, Springer Verlag, Berlin, Germany, May 1991.

[23] Goor, G. van den, Brinkkemper, S., Hong, S., *Formalization and Comparison of Six Object Oriented Analysis and Design Methods*, Master Thesis, Method Engineering Institute, University of Twente, 1992.

[24] Wirfs-Brock, R., Wilkerson, B., Wiener, L., *Designing Object Oriented Software*, Prentice-Hall, Englewood Cliffs, N.J., 1990.

[25] Wirfs-Brock, R.J. and Johnson, R.E., "Surveying Current Research in Object-Oriented Design," *The Communications of ACM*, (33, 9) Sept. 1990, pp. 104-1124.