Comparing and Contrasting 6 Methodologies Currently Being Used for Object Oriented Analysis and Design

By: Morteza Abdolrahim Kashi

Computer Science Department

Concordia university, Montreal, Quebec, Canada

Abstract:

In this article, I compare and contrast 6 formal approach of object oriented analysis and design methodologies. First, each methodology is explained by its meta-data and meta-process model. Second, the comparison of those six methodologies is performed considering each methodology's explanation. Finally, the similarity and differences between the 6 methodologies are given in tables. Looking at these tables, one can compare and contrast the 6 methodologies easily.

1-Introduction:

In this article, I compare and contrast 6 methodologies currently being used in Object Oriented Analysis and Design. I use the abbreviation of OOADM for "Object Oriented Analysis and Design" all across this article. There are more than 12 popular OOADMs but none of them is said to be the best OOADM. The reason for that is that each OOADM has different specifications and functionality compare with others. For example, some OOADMs allow objects to change their class membership where some OOADMs do not allow objects to change their class membership. But one may ask himself or herself that what OOADMs he or she has to choose. To answer to this question, one must have a good mentality about the correct comparison among the

available OOADMs. This comparison sounds a little hard since each OOADM has its own set of concepts and notations. The comparison of methodologies, also, depends on the vision of the person who wants to do the comparison.

The best way is to compare OOADMs when we look at all OOADMs in a standard way and a uniform view.

First, I will give a short description of each 6 OOADM I want to compare and contrast.

Second, I give the Meta modeling of the considered 6 OOADMs [23]. For each methodology, I consider its aim, concept, steps, technique, and graphical notation. According to all information I give, I will bring two Meta models as following:

- **Meta-process model** that shows the design and analysis used by each methodology
- **Meta-data model that** shows the techniques and concepts belong to each methodology

Third, I will use the Meta models of the considered methodology to compare those methodologies in the following respects:

- **The concepts**
- **The analysis and design steps**
- **The techniques is used in each methodology**

Forth, I will bring my conclusion about comparing and contrasting the chosen 6 methodologies to be discussed.

Finally, I will give the references and full bibliography for my paper.

2-Object Oriented Analysis and Design Methodologies' Description

I have chosen 6 methodologies to compare and contrast. These six methodologies are given by Booch [2], Coad and Yourdon[7,8], Martin and Odell[17], Rumbaugh et al.[19], Shlaer and Mellor[20] and Wirfs-Brock[24] et al. These methodologies are chosen since they are accepted as real Object Oriented Analysis and Design methodologies. I will discus these 6 methodologies as following:

2-1 Object Oriented Design with Applications (OODA) by Booch [2]:

This method is basically for the design stage of projects. Booch explains few specifications of general properties of well-structured complex systems. All systems which are made by Object Oriented Design Analysis Methodology should have all of those specifications. In Object Oriented Design Analysis, the problem domain is modeled from two different respects. These two respects include the logical structure of the system and the physical structure of the system. Both static and dynamic semantics are modeled in each respect. Object Oriented Design Analysis provides variety of procedures to do these two important jobs.

2-2 Object Oriented Analysis and Object Oriented Design by Coad & Yourdon [7, 8]:

This methodology depends on a number of general principles for managing the complexity of systems. During the analysis phase, the problem is divided into five layers in which classes, objects, the inheritance structures, relationships, message connections, and other things are included.

In the design phase, these five layers are changed and improved according to four components: a Problem Domain component, a Data Management component, a Task Management component and a Human Interaction component. Graphical notations are available for showing the five-layer model of the problem domain, the dynamic behavior of objects, and the functional structures.

2-3 Object Oriented Analysis and Design (OOAD) by Martin & Odell [17]:

This methodology takes advantage of the set theory and logic. This methodology stresses on describing the behavior of objects. There are a lot of techniques developed to specify objects and their relationships in this concern in order to describe the dynamic behavior of objects and to capture the high level business processes.

2-4 Object Modeling Technique (OMT) by Rumbaugh, et al. [19]

This methodology focuses on data instead of functions in order to make very stable programs. This methodology consists of three phases including Analysis to explain the problem domain, Systems Design to design the overall structure of the system, and Object Design to refine the Object structures for an efficient implementation. This methodology has methods to explain the problem domain from three different perspectives including: the static structure of Classes and Objects, the dynamic behavior of Objects, and the functional structures.

2-5 Object Oriented Systems Analysis (OOSA) by Shlaer & Mellor [20]

This methodology includes object-oriented analysis and gives a methodology to solve some problems we have in the Structured Analysis approach. The most important job of this methodology is to analyze the static specifications of Objects. All Techniques in this

methodology are given for modeling the static, the dynamic, and the functional specifications of objects.

2-6 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.

3- Meta-Modeling of OOADMs

Meta-models are conceptual models of modeling methodologies or techniques. There are two figures of a systems development methodology: the processes consist of the steps with related input and output products and the principles that are used to make the representation of the intermediate and final products of the methodology. In structured analysis, for instance, the processes give the steps leading an analyst to make 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].

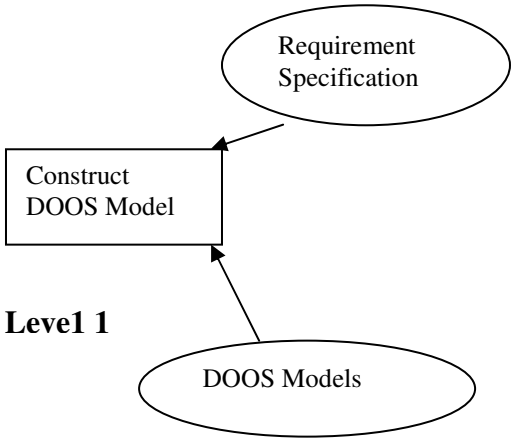
In this article, the processes of each of the six OOADMs is brought in a meta-process model, while its concepts and the associations among them, as applied in the various diagrammatic and textual techniques of the methodology, are described by a meta-data model. The meta-process models and methodologies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

Since I have limitation in the length of this article, I explain the meta-modeling approach by explaining one meta-process model and two meta-data models. But the readers of this article can find the complete set of meta-models of these 6 OOADMs in [23].

3-1- Meta-Process Model

Figure 3-1 shows the incomplete meta-process model for the methodology (DOOS) given by Wirfs-Brock, et al. [24]. The activities of the methodology are shown in rectangles and the intermediate and final products are shown in Ovals. Arrows are used to show the output dependencies between activities.

Figure 3-1 is given on the next page.



Leve1 1

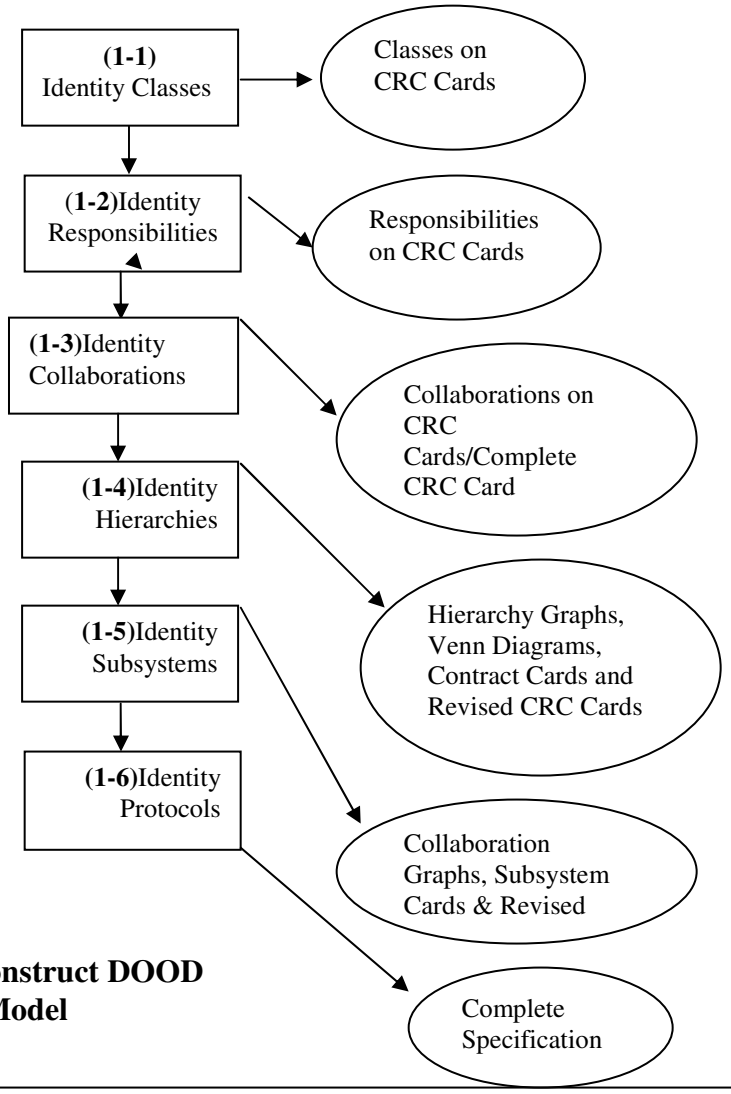
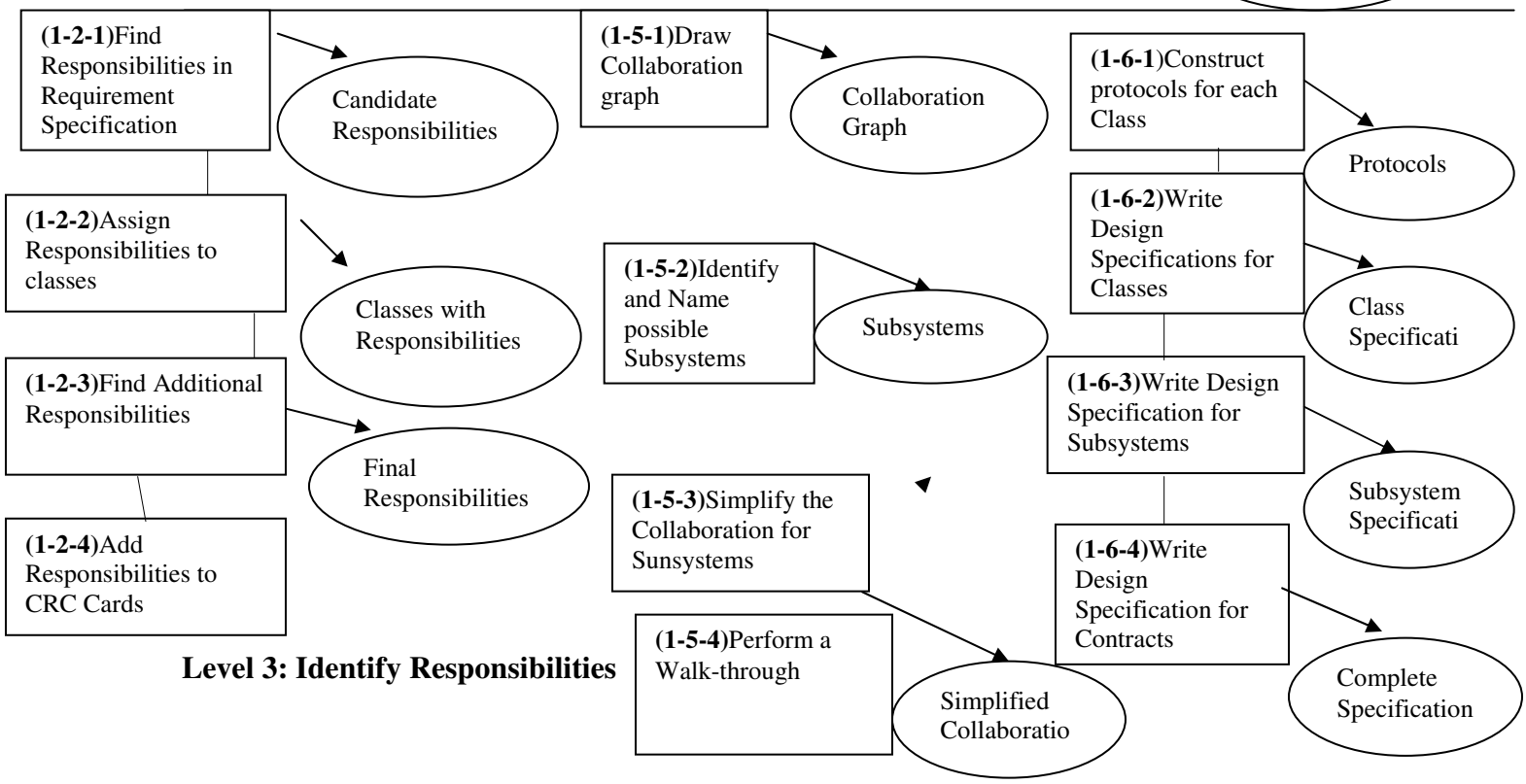


Figure 3-1: Meta-Process Model of DOOS Methodology

Level 2: Construct DOOD Model



Level 3: Identify Responsibilities

The activities DOOS are shown in multiple levels. You can see the Construct DOOS Model generic activity at **level 1**.For example, at **Level 1**, there is one generic activity called **Construct DOOS Model** .This activity consists of six activities at **Level 2**. Activities at **Level 2** are decomposed into many activities at **Level 3**. Each activity has a unique identifier. This identifier is used as a reference for the activity in the next section in which I compare the activities of all OOADMs.

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, I dropped 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 used to show the meta-process model is known as Task Structure Diagrams and was made for the SOCRATES meta-CASE environment **[5, 6, and 22].**

3-2- Meta-Data Models

Figures 3.2 and 3.3 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, I 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 3.3, 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 sub-concept 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 3.2 by the IS-A relationships, i.e., Abstract Class and Concrete Class are both sub-concepts 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 3.3 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.

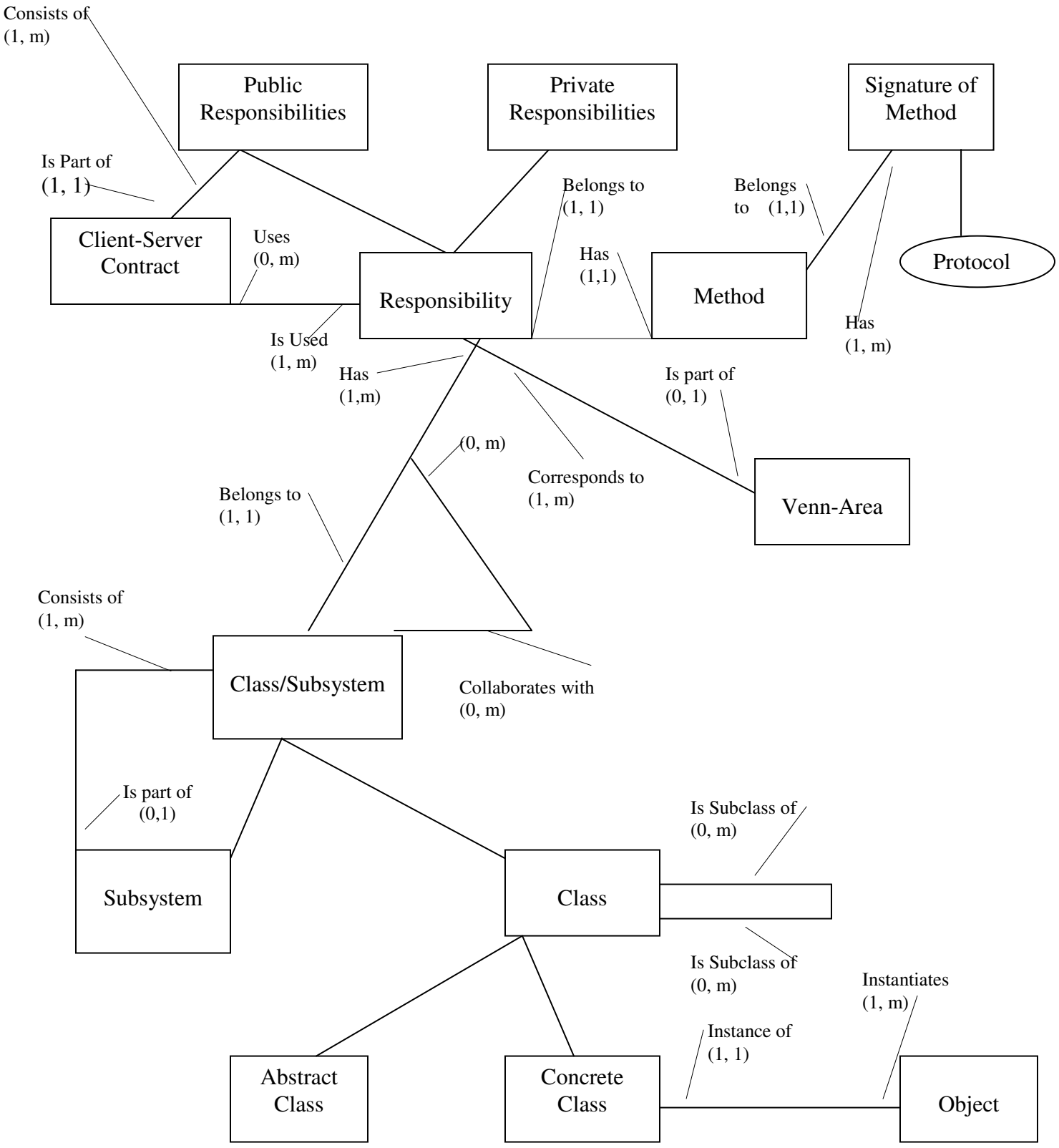


Figure3-2: Meta-Data Model of the DOOS Methodology

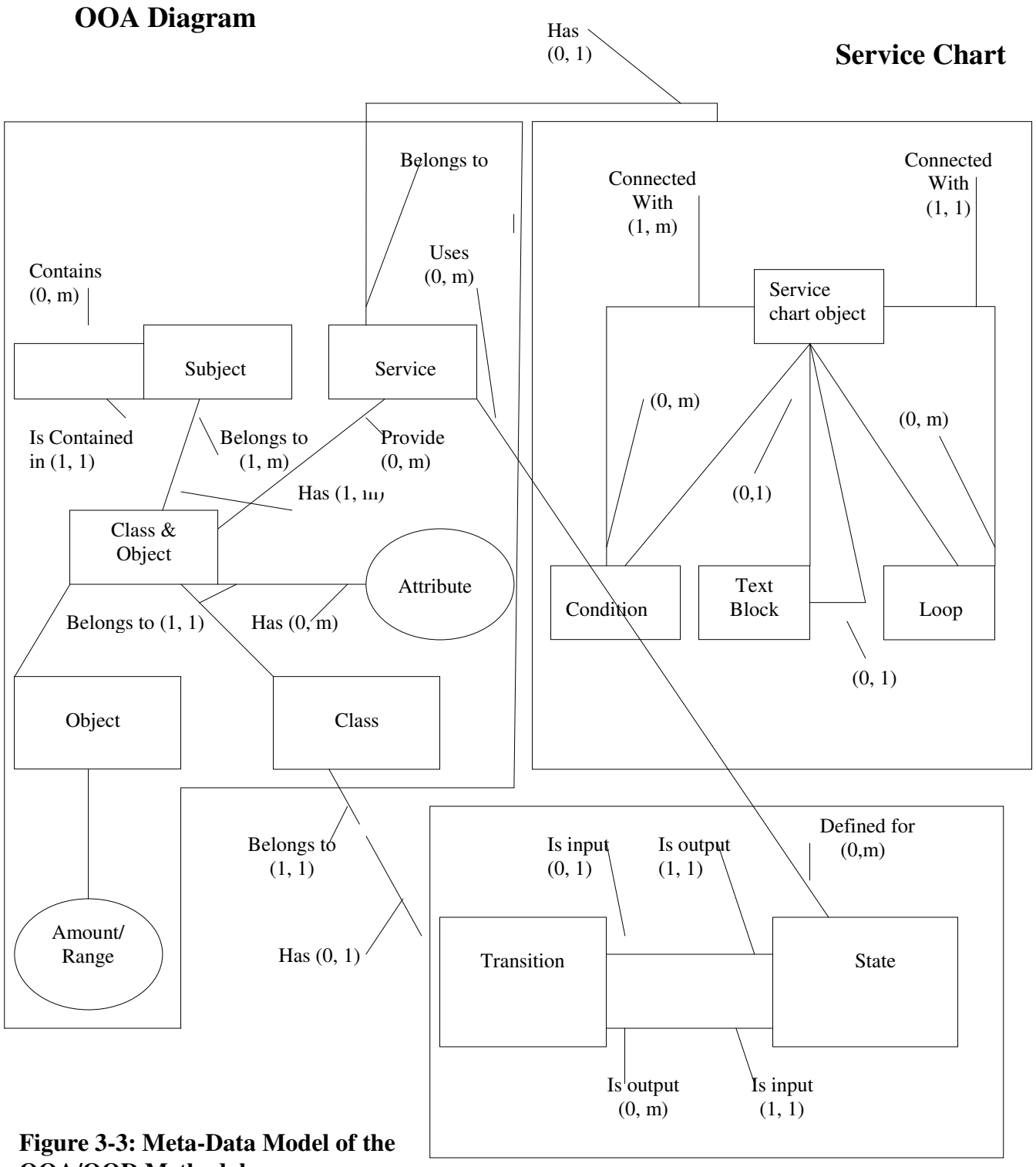


Figure 3-3: Meta-Data Model of the OOA/OOD Methodology

4: Comparison of the Methodologies

In this paper the comparison of the six OOADMs is 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, I present a part of results of the comparison and refer the reader to the complete results in [23]. At the end of this section, I also provide a short discussion of the implementation issues when these OOADMs are used.

4.1: Comparison of the Processes

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, I take the approach of creating a so-called super methodology as the target to compare.

This super-methodology 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 super-methodology 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 super-methodology, it is omitted.

- A comparison indicator that compares an activity "S" of the super-methodology to an activity m of an OOADM as follows:
- S '=' the activity s is equivalent to the activity m.
- S '>' the activity s does more than the activity m.
- S '<' the activity s does less than the activity m.
- S '><' m A part of the activity overlaps a part of the activity m and the other parts of both activities do not overlap.

This activity is absent from the OOADM. For instance, Activity 1.2.1 (Identify objects and Classes) of the super-methodology is equivalent to Activity 1.1.2 of DOOS as shown in Figure 3.1, this activity of DOOS tries to find candidate classes abstracted from various objects. This super-methodology 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, I 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 my objective in this paper is to demonstrate the formal comparison approach, I omitted the discussion of the comparison remarks. Please see [23] for the details.

4.2: 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

Activity of Super-methodology	OOA/ OOD	DOOS	OMT	OODA	OOSA	OOAD
1.Construct the model						
1.1 Study Requirements	=1.1.1	=1.1.1	=1.1			<1.1.1
1.1.1 Understand Requirements						
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 Well	=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		=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		=1.6.1				
2.3.2 Choose Algorithms =3.2.1			=3.2.1			
2.3.3 Choose Data Structures =3.2.2			=3.2.2			

Table1: Comparison of activities

I 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 super-methodology 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 super-methodology 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.

4.3: 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.

Concepts of Super-methodology	OOA/ OOD	DOOS	OMT	OODA	OOSA	OOAD
MAIN ONCEPTS						
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 Caregory		
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	…	…	…	…	…	…

Table II: Comparison of Concepts

Naturally, there is a question on whether the same technique, say state transition diagram, provided by different OOADMs is precisely the same. However, in this project, I do not attempt to address this issue which is beyond our research.

4.4: Implementation Issues

To find out the smoothness of the transition from an OOADM to the implementation, I 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, I 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. I think that it is better 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.

5: Conclusion

The main contribution of this paper 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 misinterpretation of methodologies can be detected and, therefore, can be avoided during the comparison process.

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/Diag, Hierarchy Graphs	Object Diagram	Class Diagram	Informat-ion 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 Table	(1)	State Diagrams	State Diagrams	State Transition diagram 2	Event Schema
5. System Dynamics				Timing diagrams		
6. Functional behavior	Service Charts		Data Flow Diagram		Data Flow Diagram(2)	
7. Implementation properties			Subsyste ms	Module/Pro cess Diagrams		
8. Communication between Classes/ objects	Message connec-tions OOA model	Collaboration s graphs	Event Flow Diagrams	Synchroniza tion on Object Diagram		

TABLE III: COMPARISON OF TECHNIQUES

Secondly, my 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]. I purposely avoided to rate these methodologies.

First, a standard on what is a good OOADM would be required for rating these OOADMs, which is not possible because of the current state-of-the-art of and the divergent views on object orientation. I 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 I might draw would quickly become invalid. Secondly, the quality of a methodology should be measured from 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 I did not compare the guidelines and rules provided by each OOADM. A formal system must be employed for this purpose. I have spent a lot of time 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, I did 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.

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

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

[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.