A Comparison of OOA and OMT

Sally Shlaer

Project Technology, Inc.
10940 Bigge Street
San Leandro, California 94577-1123
510 567-0255
http://www.projtech.com

7 August 1992

The purpose of this paper is to compare Shlaer-Mellor's OOA/RD with the OMT method of Rumbaugh et al. The comparison is based on the published material only [see references at the end of the paper] and emphasizes the work products defined by these methods, the organization and relationships between the models, and the goals of each method.

1. Goals of the Analysis Methods

While both methods are aimed (at the analysis stage) at capturing information about a problem domain, the two groups (Shlaer-Mellor and Rumbaugh et al.) have taken quite different views as to what to value in the analysis models. These different views of analysis derive primarily from fundamentally different approaches to design (as discussed further in Section 9).

OMT. In general terms, OMT allows the analyst a lot of flexibility as to what information to capture in the models and how to represent that information. In OMT, readability of the models is highly valued, and the analyst is encouraged to make decisions as to what to represent and how to represent it so as to achieve maximum readability. In OMT, it is perfectly all right to omit detailed information if that will increase readability.

This view is consistent with OMT's approach to design, which consists of the successive introduction of additional detailed information on a class-by-class basis.

OOA. By contrast, OOA is a much stricter method. The analyst has relatively little choice in how to represent aspects of the real world/problem information; many rules and guidelines are provided. There are strict rules for completeness and consistency. The focus on completeness and consistency is required to support Recursive Design (RD): RD is based on the idea of literal transformation of the analysis models into design and implementation.

In OOA, readability is addressed through simplicity of the notations and organization of the work products. The analyst can enhance readability by good layout of the graphical models, but not by omission of information.

Goals of the Analysis Methods		
	OOA/RD	***OMT***
Capture of problem information	Capture all information explicitly	Capture as much information as analyst finds appropriate
Readability	Notations designed for readability.	Analyst controls readability by omitting information
Consistency	Many rules given by method	A few rules given by method
Completeness	Defined by method	Defined by analyst
Use of analysis models in design	Models will be mathematically transformed into design	Models will be elaborated by addition of detail to achieve design

2. Large Scale Partitioning

OOA. OOA requires that the analyst partition the system into separate subject matters known as domains. The domains are depicted on a Domain Chart. The domains are analyzed separately, and are connected together through traceable interfaces during design and implementation.

OOA permits a large domain to be broken down into pieces (called subsystems) for the purpose of managing the analysis. A special work product is defined to aid in scheduling, tracking, and estimating the work.

OMT. The OMT book refers to a concept of domain, but does not make use of the concept in the method. Information regarding different subject matters is added class by class during design.

OMT has a concept of module -- roughly equivalent to OOA/RD's subsystem. The module concept is not reflected in the OMT work products. There are no work products defined to aid in project management.

The following table shows concepts related to large scale partitioning of a system, and where those concepts appear in work products defined by each method.

Where large scale partitioning concepts are represented		
Concept	***OOA/RD***	***OMT***
Domains	Domain chart Project Matrix	Concept is not used
Portion of a domain (subsystem in OOA; module in OMT)	Project Matrix Subsystem level models (SRM, SCM, SAM)	Concept applies to application domain only and does not appear on any workproducts
Relationships between objects in different subsystems	Subsystem Relationship Model (SRM)	Not shown on any model
Asynchronous communication between subsystems	Subsystem Communication Model (SCM)	Not shown on any model
Synchronous communication between subsystems	Subsystem Access Model (SAM)	Not shown on any model

3. Analysis Concepts and Workproducts

The two methods have nearly equivalent textual work products: Documents describing the meaning of objects, attributes, and relationships. Both methods have process descriptions.

There are many more differences in the graphical and tabular work products. Both methods prescribe three fundamental analysis models (the first three listed in the table below), and a model depicting the thread of control that devolves from the arrival of an external event.

OOA also prescribes a number of derived models: models that can be mechanically generated from the fundamental models. The derived models help the analyst understand the operation of the subsystem or domain as a whole -- an issue not addressed by OMT.

Major Analysis Concepts and Work Products		
What is represented	***OOA/RD***	***OMT***
Conceptual entities and their interrelationships	Information Model	Object Model
Lifecycles of objects and relationships	State Models	State Models (not for relationships)
Processing	ADFDs	DFD
Completeness of state model	State Transition Table (partially derived)	no corresponding work product
Senders and receivers of events	Event List (derived)	no corresponding work product
Invocation of processes (who invokes whom)	State Process Table (derived)	no corresponding work product
Asynchronous communication between objects	Object Communication Model (derived)	no corresponding work product
Synchronous communication between objects	Object Access Model (derived)	no corresponding work product
Response to an external event	Thread of control chart	Event trace

4. Information Model vs. Object Model

OOA. The Information Model of OOA depicts objects, attributes, and relationships. The model is constructed from a small set of orthogonal constructs. As a result, the analyst has relatively few concepts to learn. Many rules and guidelines are provided so that analysts working together generally find it easy to agree on a common solution.

OMT. The Object Model contains the same basic information as the OOA IM[1]. The flavor of the model is, however, somewhat different: The analyst is provided with a large number of additional, non-orthogonal concepts and notations. The analyst has many more concepts to learn, and more choice as to how to express a fact about the problem. Few guidelines are provided as to which construct to use under what circumstances.

Identifiers and referential attributes. The most important difference between the two models has to do with identifiers and referential attributes. OMT takes the perspective that the existence of an instance (with a handle of some sort) is sufficient to identify it. Because OMT does not, in general, provide for identifiers, it cannot provide referential attributes.

OOA takes the perspective that referential attributes provide linkage information inherent in the problem domain, so that information must be captured at analysis time. OMT considers that this kind of linkage information is no different from linkage information inserted for implementation reasons (links between instances of the same object supplied to allow for searching), and so defers this issue to design.

When a set of relationships forms a loop on the Information Model (or Object Model), there may or may not be constraints (dependencies) on how instances are associated around the loop. OOA requires that all loops be analyzed and that the relationships making up the loop be formalized so as to express application constraints that must be carried forward into the design. Since OMT does not, in general, use referential attributes, it cannot be used to investigate dependencies in loops.

Information Model/Object Model Concepts		
Concept	***OOA***	***OMT***
Number of concepts	Few, orthogonal	Many, not orthogonal
Theory of data	Relational	Incomplete relational (no referential attributes)
Identifiers	Required for all objects	Omitted (unless "naturally occurring")
Referential attributes	Required	Used on occasion (no guidance as to when)
Formalization of relationships	All relationships must be formalized	Relationships can be formalized if (1) there are naturally occurring identifiers and (2) the analyst so chooses
Loops of dependent relationships	All loops must be analyzed for dependency	No concept of dependent loops
Methods	Appear on design models only (see Section 10)	Shown on Object Model

[1]The analyst can supplement the basic information on the OMT Object Model by adding text to indicate instance methods, class methods, signatures of the methods, attribute types, etc. This information is provided in associated work products in OOA, and is brought together in the Class Diagram (see section 10).

5. Instance Information

OMT provides for graphical diagrams representing specified instances of the analysis objects. Relationships between individual instances are represented by connectors on the diagrams. These diagrams are rarely used because they get big very quickly: If you have 6 objects with 10 instances each, you get a diagram with 60 boxes on it.

OOA represents the same information as filled out tables, as in a relational database. Here relationships between instances are shown through values of the referential attributes. OOA does not use a diagrammatic form, but captures the instance level information in a database so that it can be fed directly into the implementation. Simple database reports are used to provide instance level information in a compact form.

Instance Level Information		
	OOA	***OMT***
instance	row in a table	icon
association between instances	value of referential attributes	graphical connector
where recorded	in database	graphical model
when/how used	feed data values into implementation	rarely used

6. Comparison of State Models

OOA. In OOA, state models are used to formalize the lifecycle of each object and relationship on the Information model. The state models are planar (non-hierarchical), so the concept of state is a simple, single-valued one. An event causes a transition within this single-leveled diagram.

An action is associated with each state, and is executed on entry to the state. All rules regarding whether or not a transition takes place are expressed by the structure of the state model itself -- there are no extra conditions attached to the transition.

In OOA, a special Assigner state model is used to manage relationships that express competition or contention.

A number of rules and guidelines are provided to tell the analyst how to build the state models. A tabular form (the State Transition Table) is provided to allow analysis of completeness and correctness of the state model.

OMT. In OMT, state models are used to formalize the lifecycles of each object (only). State models are hierarchical, so the concept of state is a combinatorial one: An instance can be in one state at one level, another state on another level, and yet another state on yet another level. An event can cause transition in a single level or across levels. The analyst can attach conditions to a transition so that when an instance receives an event, the effect of that event will be determined not just by the combinatorial state of the instance, but also by any conditions associated with transitions.

At the analyst's choice, an action can be associated with entry to a state, exit from a state, or with a transition. In addition, an on-going activity can be conducted throughout the time that an instance remains in a state. These options provide a lot of choice for the analyst, but little guidance is provided as to which option to choose.

State Models		
	OOA	***OMT***
lifecycle of an object	state model	state model
lifecycle of a relationship	state model	no such concept
competitive relationship	Assigner state model	no such concept
form	flat	hierarchical
state	single-valued	combinatorial value
action	on entry to a state	on entry to a state, on exit from a state, on a transition. during a state (analyst can specify any or all)
transition	depends on state of instance	depends on state of instance and on all transition conditions
event	causes at most one transition within a level	causes transitions across multiple levels
event data	must carry identifier of target instance	carries no identifiers

7. Comparison of Process Models

OMT. Multi-layered process models are formed by functional decomposition. No guidelines are provided for partitioning the processes. It is asserted that each process on every level becomes a method of some object on the Object Model, but no rules are provided for making that connection. There is also no explicit association between the processes on the process models and the state models, so it is difficult to evaluate consistency between these two views of the problem.

Data stores are developed to suit the analysts needs: They can correspond to objects, to single attributes of multiple instances of the same object, or combinations of attributes of multiple objects.

OOA. An Action Data Flow diagram (ADFD) is constructed for each action in each state model. Because the system has already been partitioned into objects and then into actions for the states of these objects, the analyst proceeds directly to the elementary processes at the bottom of the system. Strict guidelines are given for factoring the action into processes, and for assigning the processes to objects on the information model.

Data stores correspond to objects on the IM.

Process Models		
	OOA	***OMT***
Form	One flat ADFD per action	Multi-level DFD applies to whole system
Level of processes to develop	Elementary processes only	All levels of processes
Guidelines for forming processes	fully defined	none
Guidelines for associating processes with objects	fully defined	none
Criteria for reuse of processes	fully defined	none
Work product to manage reuse	State Process Table	none
Association between Process Model and State Model	One flat ADFD per action	no explicit association
Data stores	Correspond to objects or to Current Time	Correspond to objects, a subset of attributes of an object, a collection of attributes of many objects, an external entity (actor) -- at analysts choice

8. Execution of the Analysis Models

OOA. The analysis models of OOA are executable: When combined with instance information, they contain all information required to simulate the functional behavior of the finished system.

OMT. The state models of OMT are based on Harel's Statecharts. Execution rules have been defined for State Charts, but since OMT does not define certain information, the models are not executable, either in the analysis or design rendition. The missing information is as follows:

- Execution rules relating events and processes are not defined
- Execution rules for the process models are undefined
- There are no identifiers by which one can determine which instance must respond to an event, or which instance is to be acted upon by a process.
- There are no referential attributes to allow you to trace through to instances of other objects to complete the processing

Factors affecting Executability of Analysis Models		
Factor	***OOA***	***OMT***
completeness of model	complete	referential attributes missing, identifiers missing, may be incomplete at analysts discretion
execution rules for state models	completely stated	complete (stated in Harel's papers)
execution rules for process models	completely stated	not stated
execution rules relating events and processes	completely stated	none

9. Viewpoints on Design and Implementation

OMT. OMT starts with the assumption that an object-oriented design is to be produced. The design is developed from the analysis models by successively adding information to the classes (objects on the Object Model) one by one, and by adding additional classes as the designer sees the need.

Because OMT is intended to support only an object-oriented design, no new work products are defined for design: The analysis notation (and only the analysis notation) continues to be used for design. When the designer is satisfied that a complete description of the design has been obtained, the system is implemented by hand-coding.

The advantage of this perspective is that there is only one notational system to learn. The primary disadvantages are that the notation supports only OOD, and that it does not support certain critical design concepts (multitasking, for example). In addition, the approach encourages handcrafting of classes (more of a disadvantage for large projects than for small ones).

OOA/RD. OOA/RD takes the perspective that an object-oriented design may or may not be appropriate for a given problem -- and the analyst/designer may not be able to determine that until the analysis is quite far along. Hence this method focuses on transforming the analysis models into a design of the users selection. The analysis models are not modified during this process; as a result, a designer can experiment with multiple transformations as required to achieve a design with adequate throughput, response times, etc. An important aspect of the transformation is code generation: the transformation can be developed so that the code is generated at the same time that the design is generated.

One advantage of this approach is that you are not limited to an object-oriented design. To achieve completeness and clarity in expressing the design, the designer is encouraged to use a design notation that supports the conceptual entities of the selected design.

Viewpoint on Design		
	OOA/RD	***OMT***
object-oriented designs	supported	supported
other design schemes	supported	not supported
how design is achieved	literal transformation of analysis models	addition of detail to analysis models, class by class
how implementation is achieved	most code generated by transformation of analysis models	intended for hand-coding
design notation	Not limited to a single notation -- recommends use of a notation that expresses conceptual entities of the selected design	Limited to OOD

10. Building an Object-Oriented Design

The process by which an object oriented design is produced in OMT was described in the previous section. The notation used is the same as the analysis notation.

OOA/RD produces an object-oriented design by transformation. One such transformation scheme has been published in [2]; others are available in [3]. An object-oriented design notation (OODLE) is provided to express the design.

Building an Object-Oriented Design		
What is represented	***OOA/RD***	***OMT***
external specification of a class	Class Diagram	Object Model
internal structure of a class	Class Structure Chart	not shown on any model
invocations between classes	Dependency Diagram	not shown on any model
inheritance relationships between classes	Inheritance Diagram	Object Model

---

References

[1] *Object-Oriented Modeling and Design*, James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen, Prentice Hall, 1990.

[2] *Object Lifecycles: Modeling the World in States*, Sally Shlaer and Stephen J. Mellor, , Prentice Hall, 1991.

[3] *Real Time Recursive Design* (training course), Project Technology, 1990.