RT-OMT: A Real-time Object Modeling Technique for Designing Real-time Database Applications: A Position Paper

Bhavani Thuraisingham and Alice Schafer

The MITRE Corporation
Burlington Road, Bedford, MA 01730

ABSTRACT

This position paper describes a methodology called RT-OMT (Real-time Object Modeling Technique) for designing real-time database applications. RT-OMT adapts the OMT (Object Modeling Technique) methodology for this purpose. OMT is one of the more popular object-oriented methodologies for designing applications for complex information systems. These include database systems, executive/enterprise information systems, collaborative computing systems, medical information systems, and hypermedia systems. This paper proposes an enhanced real-time OMT, by defining a real-time object model, a real-time dynamic model, and a real-time functional model for modeling and analysis of real-time database applications.

1. INTRODUCTION

Designing a real-time database application involves (i) analyzing the system requirements to identify the data, the processing, the time-constraints, and the levels of criticality, as well as detecting potential inconsistencies, (ii) determining the features of potential hardware, firmware, and software infrastructure (including potential DBMSs) in order to specify the actual time behavior of the required processing, (iii) designing the real-time database, which consists of data which may be assigned different criticality values, integrity constraints to be enforced, transactions, and timing constraints on those database transactions, and (iv) designing the modules of the automated system (which models the slice of the real world of interest). These tasks are not performed purely linearly as described above, but instead, iteratively, as the project team achieves a deeper understanding of the system task and of the real-world resources actually available with which to implement that system.

The automated system may utilize a real-time database management system (RT-DBMS) to manage the real-time database(s). Such a DBMS must ensure that users access and share these real-time databases in such a way that the timing and priority constraints are met. Much of the focus to date has been on the design and development of RT-DBMSs without paying much attention to the overall application design process. As a result, an extra burden is placed on the real-time database application designer when utilizing an RT-DBMS because the analysis tools which are currently available simply do not deal with real-time issues. To reduce overall development time and to verify the real-time consequences of the design, tools must be developed to aid the designer in analyzing requirements, in designing the real-time database, the integrity constraints and transactions, and then in designing the modules of the automated system. This paper is a proposal for such a tool.

Designing a real-time database application is a complex process. First of all, the entities must be represented using an appropriate model unambiguously. It is important to capture the semantics as accurately and completely as possible, as well as the expected activity against the database. The requirements must be analyzed in order to detect possible inconsistencies. Therefore a design tool should capture the specification of the application and then conduct an analysis of that specification. The designer should be informed of any potential problems. In addition, a real-time database application design tool must also take into consideration the operational requirements. The tool must analyze the various operational scenarios and determine whether there could be potential timing or priority inconsistencies. In order to successfully design such a tool, we believe that a powerful modeling and analysis methodology is needed. This paper describes preliminary aspects of such a methodology, called RT-OMT (Real-time Object Modeling Technique), which we have developed.

RT-OMT enhances the OMT (Object Modeling Technique) Methodology [RUMB91] with extensions for designing real-time database applications. OMT is one of the more popular methodologies for analysis and design of applications for database systems, executive/enterprise information systems, collaborative computing systems, medical information systems, and hypermedia systems. By adapting an existing methodology, we can take advantage of the various tools that have already been developed for that methodology. We chose Rumbaugh et al's OMT Methodology because it was available, because it was developed specifically for modeling and reasoning about complex applications for information systems, and especially because it includes event traces and state diagrams, which can be adapted to support real-time analysis. There are a number of other major O-O analysis methodologies which have similar characteristics and accompanying tools, and which could be adapted for real-time in the same manner as described in this paper for OMT. Such methodologies include those formulated by Shlaer-Mellor [SHLA92] [SHLA88], Martin/Odell [MART92], and Jacobson [JACO92].

OMT, as a software engineering methodology, encompasses three viewpoints of the system to be analyzed: the object model, the dynamic model, and the functional model. As stated in [RUMB91], the object model describes the static structure of the objects in an application and their relationships. The dynamic model describes the aspects of the application that change over time. It is used to specify and implement the control aspects. The functional model describes the data value transformation within an application. The OMT methodology consists of an analysis phase during which the application requirements are

0-8186-6375-8/94 $04.00 © 1994 IEEE

analyzed, a system design phase during which the database and the system processes are generated, and an object design phase during which the algorithms and the interfaces are generated.

An object-oriented (O-O) methodology such as OMT was chosen as the basis for our work because O-O models are capable of modeling the real world of interest more naturally than models which don't consider the behavior of entities. Moreover, a carefully partitioned O-O design will enable new functionality and objects to be added with minimum perturbation after initial design is completed. In addition, the resultant O-O Design will map more easily into an O-O-based implementation. These are the strongest advantages of such a model over relational and the original entity-relationship models. Object models also provide an intuitive graphical representational scheme which an application designer can use to communicate with the client.

Since OMT was not developed for real-time database applications, the methodology has had to be adapted for this purpose. While there is other work on adapting OMT for real-time applications (see for example [KUUS93]), our approach focuses on real-time database applications. Like OMT, RT-OMT consists of three phases: the analysis phase, the system design phase, and the object design phase. The analysis phase combines the three related viewpoints. The real-time object model of RT-OMT represents the structural aspects of the application. The real-time dynamic model of RT-OMT represents the control aspects of the application. The intent of this dynamic model is to capture these interactions and identify potential problematic situations in the automated system with respect to timing constraints. The real-time functional model of RT-OMT represents the transformational aspects of the application. One can think of the functional model as representing the functional behavior of the objects. The system design phase of RT-OMT designs the real-time database, the integrity constraints, the transactions, the modules of the automated system. The details of the automated system are determined during the object design phase.

This position paper provides an overview of RT-OMT. A more detailed discussion of RT-OMT is given in [THUR94]. The organization of this paper is as follows. The analysis phase of RT-OMT is given in section 2, which consists of a brief description of the object, dynamic, and functional models. In section 3, the system and object design phases of RT-OMT are discussed. Related work is discussed in section 4. The paper concludes in section 5 with a discussion of future work. It is assumed that the reader is familiar with the essential points in object-oriented data models. For a discussion we refer to [RUMB91].[1]

[1]The ideas in this paper have been influenced by our earlier work on applying OMT for designing multilevel database applications [SELL93].

2. OVERVIEW OF THE ANALYSIS PHASE OF RT-OMT

2.1 THE REAL-TIME OBJECT MODEL

The real-time object model captures the static aspects of the application. That is, it represents all of the entities of the application, their associated criticality values (see below), and the built-in relationships between them. We use the term entities to represent not only the objects of the object model, but also the classes, associations, relationships, links, events, and activities. The real-time object model of RT-OMT is described in detail in [THUR94] with examples. The concepts include objects, classes, attributes, operations and methods, links and associations, composite objects, class hierarchy and inheritance, and metadata and constraints.

As a real-time enhancement to OMT, each object shall have a criticality measure associated with it which can be used to affect the priority the system gives to actions the object generates or to actions made upon it. Some of the objects will have natural built-in criticality values (CVs), e.g., a Commander-in-Chief or the emergency shut-off valves in a nuclear plant would have higher criticality values than most other objects in that system. CVs can be changed, depending upon the role to which the object is currently assigned, e.g., when an airplane is identified as an attacking aircraft or when a traffic light is designated as out-of-order, the CV of that object will be upgraded to ensure that it will be processed in sufficient time for a favorable outcome. Another purpose of assigning criticality values is that in the presence of time-constrained query/update processing, those objects with higher critical values may be chosen to be accessed first. In summary, each object has associated with it a criticality value (CV) which asserts how critical the object is at the current time.

RT-OMT includes the specification of the general constraints of OMT as well as additional constraints. General constraints include application independent constraints, application specific constraints, and exception constraints. The additional constraints are called real-time constraints. One type of a real-time constraint is one which is enforced on the methods and even on classes and object instances. For example, the aircraft instance, AAA, must be serviced by 8/1/94. Constraints could also be enforced on the all of the instances of a class, e.g., each aircraft must be serviced by 8/13/94. Methods must have static timing constraints which permit the verification, during analysis, of whether or not certain required processing is temporally possible. For example, the minimum time that a method could take to execute is 10 seconds; the maximum is 11.9 seconds. If the system requires that the results of the method be retrieved in less than 10 seconds, the minimum time constraint is violated. Another type of RT-OMT constraint is one which assigns criticality values for objects. An example of such a constraint is: all aircraft which fly to country X must have a criticality value > 5.0.

We will discuss the essential points of the three models of the analysis phase of RT-OMT with a simple example application. In this application, commanders reserve aircraft to carry out missions. The classes generated during the

construction of the object model is illustrated in figure 1. These classes are COMMANDER, AIRCRAFT, MISSION, and MISSION-PLAN. There are associations between the classes. As can be seen in figure 1, the associations "reserves" and "orders" are many-many while the association "used-in" is many-one between AIRCRAFT and MISSION. The association between MISSION and MISSION-PLAN is one-one. For this simple example, let us assume that the CVs of all of the objects are equal.[2] The dynamic and functional models for this application will be described in sections 2.2 and 2.3, respectively.

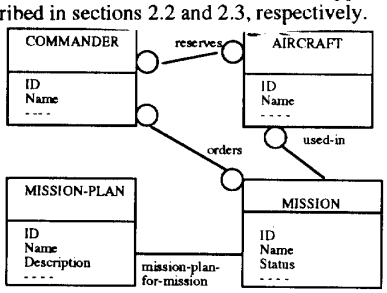

Figure 1. Classes and Associations

2.2 THE REAL-TIME DYNAMIC MODEL

The aspects of a real-time application that deal with time and state changes are represented by the real-time dynamic model. That is, the changes to the objects and their relationships are captured by this model. The sequence of operations that must respond to events form the basis of this model. An event is an individual stimulus (message) from one object to another. Since the dynamic model of OMT does not capture timing constraints, it needs to be adapted for this purpose. That is, the purpose of the real-time dynamic model is to analyze each scenario and determine potential problems that could occur. A scenario is a sequence of related events such as a commander reserving some aircraft and then requesting a mission to be carried out or a customer inserting a card into an automatic teller machine and withdrawing cash from a bank account. Once the designers of a system are informed by the tool of potential problems, they could then discuss with the customer ways to rectify the problems or they could design the system in such a way as to minimize the problems that could occur and provide fall-back positions when they do occur.

Developing a real-time dynamic model consists of identifying the active objects (objects that can stimulate other objects), passive objects, events, and states, and then developing event trace diagrams, event flow diagrams, and state diagrams. These diagrams are used to detect whether there could be inconsistent timing constraints. We discuss some of the essential points here.

[2]We have not considered CVs in this example as there are still some outstanding issues that need to be resolved in assigning CVs. For example, does it make sense to assign CVs to classes and what is the relationship between the CV of a class and the CV of its instances? Some of the issues are discussed in [THUR94].

RT-OMT groups the objects into two categories: active objects and passive objects. The active objects may stimulate each other or stimulate passive objects causing some changes to occur. Passive objects do not stimulate any other objects; they can only respond to the object which stimulated them. From a real-time execution point of view, the active objects carry out activities that have operational values (OV) associated with them. One can regard the OV of an activity of an active object to be the priority level at which the object carries out the activities.[3] In developing a dynamic model, the first step is to determine which of the objects are active and which of the objects are passive. Subsequently, OVs must be assigned to the activities carried out by an active object. Determining whether an object should be passive or active may not be straightforward. For example, a mission itself may be an active object while the mission-plan may be a passive object.[4]

Event analysis is the process used to detect potential timing problems. Consider our example of a commander requesting to reserve some aircraft to carry out a mission. The active objects are Commander and Mission while Aircraft and Mission-plan are passive objects. Suppose the requests generated by the commander have OV values of P-Cmdr and the requests generated by Mission have OV values of P-Msn. Suppose the timing constraint assigned for the entire activity is T-Msn-Total minutes. That is, the operation must be carried out in T-Msn-Total minutes. Also, suppose from a timing analysis it is determined that the time to reserve an aircraft must be at most T-Res-Rqd minutes. Further assume that the method associated with aircraft has a constraint that it take T-Res-Min minutes to reserve an aircraft (which is the minimum time that is required to reserve an aircraft).[5] Figures 2, 3, and 4 show event trace diagrams.

[3]The question is, should all of the activities of an active object have the same OV or could they be within a range? This issue needs to be examined further.

[4]The relationship between OVs of the activities carried out by an active object and the CV of the object are not clear at this point. For example, do we need both CVs and OVs or do we need just one type of value? Can OV be generated based on the CV of the objects involved and the CV of the transaction? These issues need to be examined further.

[5]Note that whenever the minimum time for an activity exceeds the required time for that activity, then the system detects a timing violation. Also, if the maximum time for an activity exceeds the required time, there could be a potential timing violation. In this example we consider only minimum time to carry out an activity.

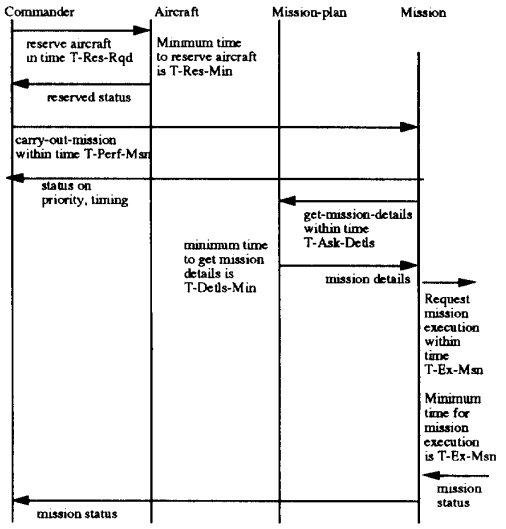

Figure 2. Event Trace Diagram I: No Inconsistency

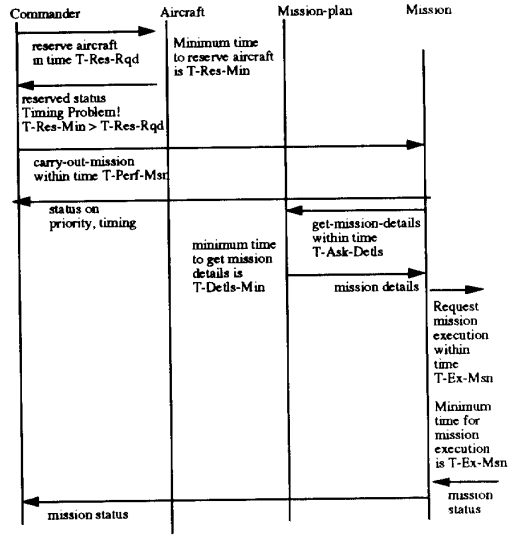

Figure 3. Event Trace Diagram II: Timing Inconsistency

Figure 2 illustrates the case where there are no timing or priority inconsistencies when a commander reserves an aircraft to carry out a mission. That is, it is true that T-Msn-Total>= T-Res-Rqd>=T-Res-Min and P-Cmdr>= P-Msn (assuming that a commander whose activities are at a certain priority cannot request a higher priority mission to be carried out). Suppose T-Perf-Msn is the time constraint imposed by the commander to perform the mission, T-Ask-Detls is the time constraint imposed by the Mission object to execute the mission, and T-Ex-Min is the minimum time it will take to execute the mission. For the operation to be successful, the following must hold: T-Msn-Total>= T-Res-Min+ T-Perf-Msn; T-Perf-Msn >= T-Ask-Detls + T-Ex-Msn; T-Ask-Detls >= T-Detls-Min; and T-Ex-Msn >= T-Ex-Min.

Figure 3 illustrates the case where there is a timing problem for reserving an aircraft, assuming T-Res-Rqd >= T-Res-Min.

Figure 4 illustrates the case where the priority levels are in conflict, assuming P-Cmdr <= P-Msn. When the designer is alerted to the potential problems and given possible suggestions by the tool, the designer could discuss the alternatives with the client and make some decisions. Based on the feedback given by the client, the designer could then proceed with the remaining tasks.[6]

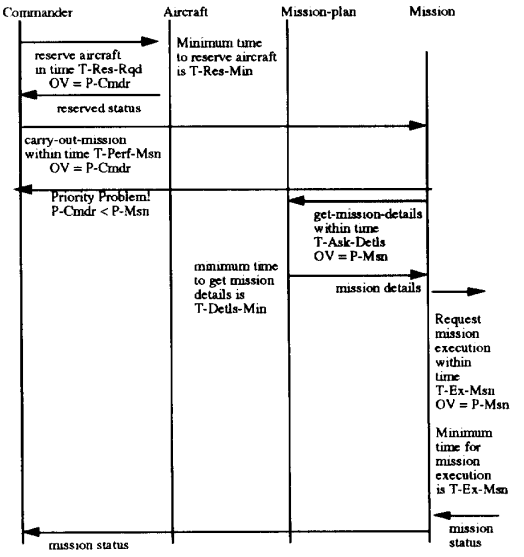

Figure 4. Event Trace Diagram III: Priority Inconsistency

2.3 REAL-TIME FUNCTIONAL MODEL

The real-time OMT functional model generates the methods of a class and the priority levels (which can be regarded to be the OV) for the execution of each method. It does this by examining the object model, the dynamic model, and subsequently performing a data flow analysis. Once the methods (which can be regarded as the functional behavior of the objects of the application) are determined, then for each method its OV is also generated for each scenario as determined by the event analysis process.

For the example described in section 2.2, the functional model would subsequently generate the method *update-*

[6]The times given here are a rough estimate. We have not carried out a formal analysis here.

reserve-status for AIRCRAFT, the method *retrieve-mission-details* for MISSION-PLAN, and the method *execute-mission* for MISSION. The OVs of *update-reserve-status, retrieve-mission-details,* and *execute-mission* are P-Cmdr, P-Msn, and P-Msn respectively. The pseudo code for the three methods are given below.

```
CLASS AIRCRAFT::update-reserve-status
        (OUTPUT: STRING)
    DECLARE STRING status-msg;
    BEGIN
        IF my->status == RESERVED
        THEN status-msg := "Already Reserved";
        ELSE
        my->status := RESERVED;
        status-msg := "OK";
        ENDIF;
        RETURN (status-msg);
    END;

CLASS  MISSION-PLAN::
    retrieve-mission details
    (OUTPUT: MISSION-DETAILS)
    BEGIN
        RETURN(my->msn-details);
         /* no error checking is shown*/
    END;

CLASS MISSION::execute-mission
    (INPUT: SET-OF{AIRCRAFT} the-aircraft,
    OUTPUT: STRING)
    DECLARE MISSION-DETAILS temp-msn-details;
    DECLARE STRING status-msg;
    BEGIN
        temp-msn-details := my->mission-plan->
        retrieve-mission-details();
        status-msg := my->carry-out-mission
                    (the-aircraft, temp-msn-details);
        /*carry-out-mission is a method of the mission
        object that will carry  out the mission.
        The actual process could be automatic or an
        operator could be notified to carry out the
        mission*/
    RETURN (status-msg);
    END;
```

Now, the methods must take timing constraints into consideration. For example, the timing constraints for the execution of the methods *update-reserve-status, retrieve-mission-details,* and *execute-mission* are T-Res-Rqd, T-Ask-Detls, and T-Perf-Msn respectively. These constraints have to be passed as input (which has not been shown in the code). The minimum time for the methods *update-reserve-status* and *retrieve-mission-details* to execute are T-Res-Min and T-Detls-Min respectively. So, these methods will have to check whether the timing constraints can be satisfied. If not, it has to be reflected in the status returned. Also, the method *execute-mission* has a timing constraint of T-Perf-Msn. It must check whether this constraint can be met (i.e., whether T-Perf-Msn >= T-Ask-Detls + T-Ex-Msn). The method *carry-out-mission* has a timing constraint of T-Ex-Msn. However, the minimum time to execute this method is T-Ex-Min. So, it has to check whether the timing constraint can be met.

3. THE NEXT STEPS

As can be seen, the three models are related to one another. Each describes some aspect of the application utilized by another. The object model describes data structures and the CVs of these structures. The dynamic model describes the control structures of the objects and specifies activities, when these activities happen, and their OVs. The functional model specifies what the functions to be executed on the objects are and their OVs derived from the dynamic model.

The three models of RT-OMT will be used in the analysis phase of the application design. The output of this phase will be the objects, classes, and the various types of associations, a description of the events and the changes to the states of the objects, the functions which would implement the activities, and also the associated CVs and OVs of all of the objects, events and functions. At this time the designers will be alerted to the potential problems and inconsistencies with the design which they would rectify or limit, possibly by consulting with the client. The initial real-time analysis phase, especially where priorities, criticality values, and timing relationships that must be met are concerned, can be carried out independently of the implementation. For example, for a real-time database application, the analysis phase is not concerned whether the database is relational or object-oriented and also whether the system is general purpose or needs to be specially constructed. However, when actual timing considerations must be taken into account, it will become crucial to know what the hardware/software platform is capable of delivering.

The question may arise: is it necessary to construct all three models for the design of a specific application? This would depend on what the customer wants. If only a relational database schema needs to be generated, then the object model structure, without methods, would suffice. If the system is to be designed in addition to the database, then the dynamic aspects of the application need to be analyzed. If in addition, the transactions are to be generated, then the functional model comes into play. Therefore, the level of detail depends on what the customer expects from the application designer.

Once the analysis phase is completed, what are the next steps? The subsequent phases are the system and object design phases. During system design, the focus on *what* needs to be done is shifted to *how* it is done. For example, in the case of a real-time database application, the steps in the system design phase would include:

(i) mapping the object model into a real-time database,
(ii) determining the integrity constraints to be enforced,
(iii) mapping the dynamic and functional models into transactions,

(iv) determining the timing constraints to be enforced on the transactions,
(v) designing the modules of the automated system.

While the analysis phase determines what the implementation must do, and the system design phase determines how it should be done, the details are determined by the object design phase. This will include algorithms and interface specifications for the implementation. It is during this phase, that the implementation details as to the specific RTDBMS to be selected and other packages to be used are determined. Rumbaugh et al. [RUMB91] have developed various tools (for example OMTool) for performing the OMT analysis and design. Such tools need to be adapted for real-time database applications.

4. COMPARISON TO OTHER APPROACHES

In comparing RT-OMT to other approaches, there are two issues to be considered. First of all, what are the advantages of OMT with competing models such as structured analysis, Jackson structured diagrams, information modeling based on entity relationships diagrams, and other object-oriented approaches (such as those by Booch and Yourdan). The advantages of OMT have been described in detail in [RUMB91] as well as in special articles in some of the recent issues in the Journal of Object-oriented Programming. It has also been pointed out that while some of the other approaches may be better for a specific type of application, OMT is general enough to be applicable to a variety of applications.

The second aspect is to compare RT-OMT with other approaches to real-time object-oriented analysis and design. It appears that most of the approaches have been focusing on developing real-time operating systems. Designing the database, the application, and the transactions have not been considered. The work by Kuusela and Awad [KUUS93] has adapted OMT for developing real-time systems. The earlier version of this methodology was called OMT+ which was then refined to OMT/RT. Its object model implements a program as a set of object groups which communicate with each other. There are four groups, each with a priority level. A group can interrupt processing in groups with lower priorities. The dynamic model consists of interaction diagrams for each operation. Also, the object group which has to handle the operation is determined first. The functional model collects information about each object into class template. It also has information about asynchronous objects and mail messages. Neither OMT+ nor OMT/RT discuss issues such as criticality values, timing constraints, detecting inconsistencies, and database issues such as designing the real-time database and the transactions.

5. SUMMARY AND FUTURE CONSIDERATIONS

This paper has described a modeling methodology called RT-OMT to design real-time database applications. The main focus has been on the analysis phase of the methodology. It consists of object, dynamic, and functional models. The methodology not only captures the structural aspects of the application, but also the dynamic and functional aspects. The paper has also briefly discussed the system design and object design phases of RT-OMT.

There are several areas that need further work. First of all, RT-OMT has to be reviewed to determine if it could be enhanced further. For example, the CV assignment schemes that we have discussed need to be examined to see if they are applicable to a wide class of real-time applications. Also, should there be separate CVs and OVs or should each object and activity have only CVs? We also need to test the object, dynamic, and functional models with realistic applications. Another area for further work is to develop a methodology to generate 'good' real-time databases from the object model. Next, the system design and object design phases need to be developed so that transactions as well as algorithms for transaction execution are generated for the real-time system. In addition, the timing constraints on the transactions as well as transaction scheduling algorithms have to be determined. Determining schedules for concurrent execution of transactions is a major issue. Finally, the techniques developed have to be incorporated into a tool to be used by the real-time database application designer.

REFERENCES

[JACO92] Jacobson, I., et al. *Object -Oriented Software Engineering*, 1992, ACM Press Addison-Wesley, N.Y.

[KUUS93] Kuusela, J., and M. Awad, OMT/RT, OOPSLA-93 Conference Workshop on Object-Oriented Real-time Systems, Washington, D.C., September 1993.

[MART92] Martin, J., and J. Odell, *Object -Oriented Analysis and Design*, 1992, Prentice Hall, Englewood Cliffs, N.J.

[RUMB91] Rumbaugh, J., et. al. *Object -Oriented Modeling and Design*, 1991, Prentice Hall, Englewood Cliffs, N.J.

[SELL93] Sell, P., and B. Thuraisingham, 1993, Applying OMT for Designing Multilevel Database Applications, Proceedings of the 7th IFIP Working Conference in Database Security, Huntsville, AL.

[SHLA88] Shlaer S. and S. Mellor, 1988, *Object-Oriented Systems Analysis: Modeling the World in Data*, Englewood Cliffs, New Jersey, Yourdon Press.

[SHLA92] Shlaer S. and S. Mellor, 1992, *Object Lifecycles: Modeling the World in States*, Englewood Cliffs, New Jersey, Yourdon Press.

[THUR94] Thuraisingham, B. RT-OMT: A Real-Time Object Modeling Technique for Designing Real-time Database Applications, Technical Report, The MITRE Corporation, in preparation.