Shlaer-Mellor Object-Oriented Analysis Rules

Neil Lang

Project Technology, Inc.
10940 Bigge Street
San Leandro, CA 94577-1123
510-567-0255

http://www.projtech.com

1. Abstract

Shlaer-Mellor Object Oriented Analysis provides a complete and executable description of a problem domain using a set of formal analysis models: an information model, a set of state models, and a set of process models. In this paper we present the conditions that must be satisfied by a valid Shlaer-Mellor Object Oriented Analysis.

2. Introduction

In Shlaer-Mellor Object Oriented Analysis [1, 2] the analyst describes a problem domain by constructing a set of formal models (see figure 1): An Information Model to define the conceptual entities (objects) of the domain and the relationships between those objects, a State Model to formalize the behavior of each object, and a set of process models in the form of Action Data Flow Diagrams to specify the processing required in the state models. A formal analysis consists of other derivative models: an Object Communication Model summarizing the asynchronous communication among the objects, and an Object Access Model summarizing the synchronous access among the objects.

This paper lists the conditions that must be satisfied by a valid Shlaer-Mellor Object Oriented Analysis. The "rules" are descriptive; that is, they describe what a valid analysis must contain but they do not prescribe how to actually produce a valid analysis. The rules are designed for both practitioners (analysts, designers, and implementers) as well as for toolmakers developing CASE tools.

3. Shlaer-Mellor Object Oriented Analysis Rules

1. A large domain may be analyzed by partitioning it into a set of subsystems. Each subsystem must have a unique name.

2. The complete information model for the domain will be partitioned into multiple smaller information models. Each information model is assigned to a separate subsystem.

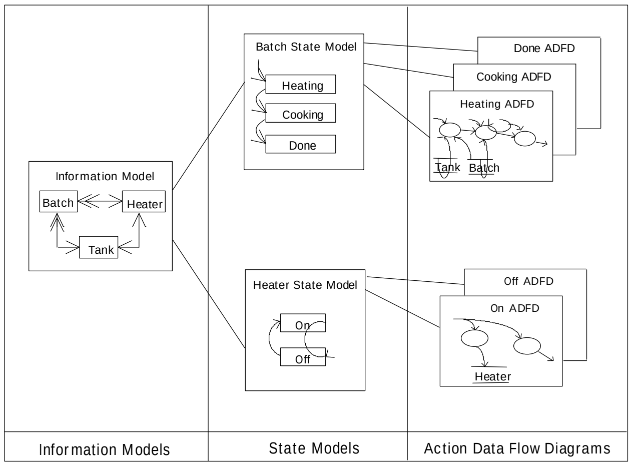

Figure 1: Structure of the primary models of Shlaer-Mellor Object-Oriented Analysis

3. Each object belongs to one and only one subsystem. If a relationship exists between objects in different subsystems, then a copy of the object from the foreign subsystem can be shown on the IM for this subsystem, but the foreign object must marked distinctively to indicate that it has been imported from another subsystem.

4. Each object in a subsystem must have a unique name. Objects must also be numbered uniquely to aid organizing the documentation.

5. Each object must have a unique KeyLetter. The KeyLetter is a user-specified short character string and is used in event labels and process identifiers.

6. Each attribute within an object must have a unique name. The full name of an attribute has the form <object>.<attribute>. Note that the <attribute> part of the full attribute name can be the same in several objects.

7. Each object must have at least one identifier i.e. an attribute or set of attributes whose values uniquely distinguish each instance of the object. The attribute(s) of the identifier of choice for an object must be indicated.

8. Each relationship in a subsystem must have a unique label of the form Rn (where n is an integer).

9. Each relationship in a subsystem must be formalized either through referential attributes or through composition.

10. Referential attributes (i.e. attributes used to formalize a relationship between objects) are indicated by appending the label of the relationship to the attribute.

11. The domain of a referential attribute must be the same as the domain of the corresponding attribute of the associated object.

12. A relationship Ri formed by composition (i.e. Ri exists as a logical consequence of other relationships Rj and Rk in the IM) must be indicated as Ri = Rj + Rk.

13. Each object, attribute, and relationship in a subsystem must be fully defined or described to complete the documentation for the IM for the subsystem. However foreign objects in a subsystem do not have object or attribute descriptions in the IM for that subsystem.

14. A relationship between objects in different subsystems will appear in the IM for both subsystems. However the relationship will be described in only one of the subsystems.

15. An object may have one state model formalizing the lifecycle of its instances. The name of the state model is the name of the object. The KeyLetter of the state model is the KeyLetter of the object.

16. An associative object may have an additional state model to manage the creation of instances of the associative object. The name of this state model is <object>_ASSIGNER where <object> is the name of the associative object. The KeyLetter of this state model is <KL>_A where <KL> is the KeyLetter of the associative object. The analyst must avoid creating another object in the IM with the same name or KeyLetter as those of the assigner state model.

17. As a consequence of the previous two rules, each state model will have a unique name and each state model will have a unique KeyLetter.

18. Each state within a state model must have a unique name. The full name of a state has the form <state_model>.<state>. Note that the <state> part of the full state name can be the same in several state models.

19. Each state within a state model is also assigned a unique number.

20. External entities (also known as terminators) are those entities that are not part of the subsystem but do generate events to and receive events from state models. Each external entity is also assigned a KeyLetter which must be unique over all external entities, objects, and state models within the subsystem.

21. An object representing timers is part of the OOA formalism. The object is named TIMER and is assigned the KeyLetter TIM; both the name and KeyLetter are reserved and cannot be used for other objects in the IM. The attributes of the object are timer_id, instance_id, event label, time remaining, and timer status. The TIMER object is not shown on the IM. The object has a state model which receives events to set and reset a timer, and generates an event when the timer has expired.

22. Events are generated by state models and external entities.

23. Events are received by state models and external entities.

24. An event is directed towards one and only one state model or external entity. Further more if the state model formalizes the lifecycle of an object, then the event is directed to a single specific instance of the object.

25. An event is specified by i) an event name which captures the real world incident being abstracted, ii) the name of the state model or external entity that the event is directed to, and iii) the event data. Every event specification is assigned a unique event label.

26. An event label has the following form: KLi where KL is the KeyLetter of the state model or external entity that receives the event, and i is an integer.

27. Events are represented in state models using the following form:
    <Event_label>:<Incident_name>
    (<event_data>)

28. If an event can cause an existing instance of an object to transition to a new state, the event data carried by that event must include an identifier of the instance.

29. All events causing a transition into the same state must carry exactly the same event data.

30. The following events involving timers are defined as part of the OOA formalism:
    TIM1: Set Timer(timer_id, ELx, instance_id, time_interval) and
    TIM2: Reset Timer(timer_id)
    where timer_id is the identifier of the timer to be set or reset, ELx is the label of the timer expired event to be generated upon expiration of the timer, instance_id is the instance of the object to which the event ELx is directed, and time_interval is the interval to wait before generating the timer expired event.

31. State models and external entities that generate or receive events must appear on the Object Communication Model.

32. An event which is generated by an external entity or state model and received by another state model must appear on the Object Communication Model.

33. The TIMER state model and events generated and received by the TIMER are not shown on the OCM.

34. Each state in a state model has an action associated with it.

35. An action modifying descriptive data for an instance must ensure that the all data for that instance is self-consistent upon completion.

36. An action creating or deleting instances of its own object must ensure that any relationships involving those instances are made consistent with the rules and policies in the IM.

37. An action must leave subtypes and supertypes consistently populated.

38. Except for the action of a deletion state, an action in the state model formalizing the behavior of an object must always update the current state attribute to indicate the new current state of the instance.

39. A process model represents the desired processing for the action associated with a state. The name of process model is the name of the associated state.

40. Process models consist of one or more processes and one or more data flows. Control flows and data stores may also appear in process models.

41. Data stores are labelled “Timer”, “Current Time” or with the name of an object in the IM.

42. A data store labelled “Timer” represents the time left on each timer in the system.

43. A data store labelled “Current Time” represents data describing current time.

44. A data store labelled with the name of an object in the IM represents persistent data associated with all instances and all attributes of that object.

45. A given data store may appear in multiple places in a single process model or in multiple process models.

46. Conditional control flows in a process model are labelled with the circumstances under which the control flow is generated.

47. Unconditional control flows in a process model are unlabelled.

48. Both conditional and unconditional data flows in a process model must be labelled with the names of the data elements they carry.

49. A data flow between a process and an object data store is labelled with the attribute(s) read or written by the process. The <object> part of the full attribute name can be omitted on such data flows.

50. Data flows between processes may represent either attributes of objects or transient data (i.e. data intended for use by another process but not retained after the action has completed execution).

51. If a data flow between processes represents transient data, the data flow is labelled with an appropriate variable name and further labelled with (transient).

52. If a data flow between processes represents persistent data, the data flow is labelled with the name of the attribute(s) comprising the flow. The label on the data flow will have the form ObjectP.attribute1=ObjectC.attribute2 where ObjectP.attribute1 is the full name of the attribute from the perspective of the process producing the data and ObjectC.attr2 is the full name of the attribute from the perspective of the process consuming the data. If the attribute part of each full attribute name is the same, then the flow may be labelled with simply <attribute>.

53. Event data associated with the event that causes a transition into the state executing the action being modeled is shown as a data flow from nowhere into the processes requiring the event data. Each such data flow is labelled with the names of only those attributes that are required by the process.

54. An event generated by a process is shown as a data flow directed away from the process to nowhere. The data flow is labelled with event’s label, meaning, and event data just as it appears in the state model and OCM.

55. If a process deletes an instance of an object, then the data flow from that process to the corresponding object data store is labelled with (delete). No attribute names are shown on the data flow.

56. A process executes when all of its data inputs and control inputs are available.

57. The data outputs and control outputs of a process are available once the process completes executing.

58. Event data associated with the event that caused the transition into the state whose action is being modeled are always available.

59. Data from data stores are always available.

60. Each process in a process model must be identified by a process identifier and a meaningful name describing the purpose or function of the process.

61. Complex processes should be partitioned into simpler processes of the following types: accessors, transformations, event generators, and tests. An accessor is a process whose only purpose is to access data in a single data store. A transformation is a process whose purpose is to perform calculations on or transform the input data. An event generator is a process whose purpose is to produce exactly one event. A test is a process that tests a condition and produces one of several conditional control outputs.

62. Processes that carry out the same function, read/write the same attributes from/to the same data stores, accept the same input from other processes or events, produce the same output to other processes, produce the same events, and produce the same conditional control outputs are the same process and are to be labelled with the same process identifier and name.

63. The process identifier has the form KL.i where KL is the KeyLetter of an object, state model or external entity and i is an integer unique within all processes assigned that KeyLetter.

64. An accessor is assigned the KeyLetter of the object corresponding to the data store accessed.

65. The following accessors involving the TIMER data store are defined as part of the OOA formalism:

TIM.3	Create Timer
TIM.4	Delete Timer
TIM.5	Read Time Remaining

66. A transformation process is assigned the KeyLetter of the state model in which it appears. If the transformation process is part of the action of a state model formalizing the lifecycle of an object, the transformation process may also access the data store for that object.

67. An event generator is assigned to the KeyLetter of the object or external entity to which it is directed. An event generator may not access any data stores.

68. The following event generators involving the TIMER data store are defined as part of the OOA formalism:

TIM.1	Generate event TIM1
TIM.2	Generate event TIM2

69. A test process is assigned the KeyLetter of the state model in which it appears. If the test process is part of an action of a state model formalizing the lifecycle of an object, the test may also access the data store for that object.

70. If the action for some state in a state model (with KeyLetter X) includes a process that accesses the data store of an object with another KeyLetter (with KeyLetter Y) then both state models X and Y must appear on the Object Access Model. The access is represented by an arrow drawn from X to Y. The arrow is labelled with the corresponding process identifier.

71. All subsystems in a domain must appear on the Subsystem Communication Model.

72. An event which is generated by a state model in one subsystem and received by a state model in another subsystem must appear on the Subsystem Communication Model.

References

1. Shlaer, S. and Mellor, S. J. *Object-Oriented Systems Analysis: Modeling the World in Data,* Prentice Hall, Englewood Cliffs, N.J., 1988.

2. Shlaer, S. and Mellor, S. J. *Object Lifecycles: Modeling the World in States,* Prentice Hall, Englewood Cliffs, N.J., 1992.