The Shlaer-Mellor Metamodel

Here is an overview map of the model subsystems defining the Shlaer-Mellor Metamodel.
Taken together, these subsystems define executable, platform independent modeling semantics that can be expressed with a variety of notations including, but not exclusive to Executable UML and SysML.

The content here is maintained and described in the github modelint/shlaer-mellor-metamodel repository

You'll find all of the documentation in the repository wiki

Underneath each subsystem rectangle is a brief description with a list of its key classes to the right

CLASS MODEL

Class and Attribute

Class definition
Attributes and Identifiers
Other Attribute roles

How we define classes and attributes

Relationship

Relationship definition
Types of relationships:
Association with Perspective, and Association Class
Generalization with Facet and Lineage
Ordinal
Constrained Loop

The different kinds of relationships among classes

Formalization

How we use referential attributes to formalize Association and Generalization Relationships

DOMAIN MODEL

Domain

System
Modeled Domain
Realized Domain
Subsystem
Element

How we organize an entire system and identify subysstem and spanning model elements

STATE MODEL

State

Lifecycle and Assigner
State
Transition
Event Reponse

How we define a state model

Event

Event Specification
Monomorphic event (normal events)
Polymorphic event

How an event is specified

ACTION MODEL

Activity

State Activity
Method
Signature
Parameter

How Activities are defined and associated with model elements

Action

Action
Instance / Relation Action
Flow Dependency
Wave

All of the Action building blocks that can be assembled to define an Activity as well as the flow dependencies forming the flow graph of that Activity

Flow

Control Flow
Instance Flow
Relation Flow
Scalar Flow

Types of data and control that flows in an Activity

Navigation

Traverse Action
Path
Hop

Defines the Traverse Action and the anatomy of a Path through the Relationships of a class model

Computation

Computation Action
Computation Input

Defines the Computation Action which inputs multiple Data Flows, applies a mathematical or logical expression and outputs a single Scalar result

Signal Action

Signal Action
Initial Signal Action
Signal Assigner Action
Interaction Signal Action
Delivery Time

Defines the Signal Action which emits a signal to various kinds of destinations

Create Delete

Create Action
Attribute Initialization
Delete Action

Defines actions to create and delete an instance of a Class

Attribute Access

Read / Write Action

Defines actions to read and write the value of an Attribute

External Service

External Service
External Event
External Operation

How we define operations and events triggered in an external domain

Select Action

Select Action
Identifier Select
Many Select

Defines the Select Action which selects one or more instances of a Class

Conditional Logic

Sequence Flow
Decision Action
Switch Action
Case
Control Dependency

Defines Actions that output Control Flows to enable other Actions

Method Call

Method Call
Method Call Output
Method Call Parameter

Defines an Action that invokes a Method on some instance

Relational Action

Restrict Action
Project Action
Restriction Condition
Rename Action
Set Action
Extend Action

Defines the supported Relational Actions

Iteration

Iteration Action
Sequence
Sorting Attribute

Defines an action that iterates through a sequence of model elements (no, these are not for loops!)

Type

Type
Type Action
Type Operation
Scalar
Table Attribute

Type definition is outside the scope of Shlaer-Mellor semantics, but here we do register the externally defined names of types as well as supported type operations along with any invocations of those operations

EXECUTION

Execution

State Machine
Instance
Activity Execution
Dispatched Event

All of the other subsystem specify static model elements, but here is where execution on all of those elements is defined

With all the hard work done in those in that multitude of semantic specification subsystems, our work here is greatly simplified to the point we need only one small subsystem to define the platform independent execution mechanics.

October 5, 2025 / v2.0.1
Model Integration, LLC