

ISE

Quick Guide for SysML Diagrams and Symbols

SysML Structure diagrams

A Practical Guide to SysML. Sanford Friedenthal, Alan Moore and Rick Steiner. Elsevier 2008.

SysML Reference Guide: Table A3, A4, A5, A8 and A9.

Table A.3 Block Definition Diagram Nodes for Representing Block Structure and Values


Diagram Element	Notation	Description	Section
Block Node	<div>«block» <Name></div> <div>parts <Part>:<Block>[<Multiplicity>]</div> <div>references <Reference>:<Block>[<Multiplicity>]</div> <div>values <ValueProperty>:<ValueType>=<ValueExpression></div> <div>operations <Operation>(<Parameter>,...):<Type></div> <div>receptions «signal»<Signal>(<Parameter>,...)</div>	<p>The block is the fundamental modular unit for describing system structure in SysML.</p> <p>Compartments are used to show structural features (parts, references, values) and behavioral features (operations, receptions) of the block. See the following tables in this section for more block compartments.</p> <p>Additional properties on blocks are {encapsulated, abstract}. Abstract may also be indicated by italicizing the <Name>.</p> <p>Additional properties on structural features include: {ordered, unordered, unique, nonunique, subsets <Property>, redefines <Property>}.</p> <p>A forward slash (/) before a property name indicates that it is derived.</p>	6.2, 6.3, 6.5.2
Dimension and Unit Nodes	<div>«unit» <Name></div> <div>dimension = <Dimension></div> <div>«dimension» <Name></div>	A dimension identifies a physical quantity such as length, whose value may be stated in terms of defined units, such as meters or feet. A unit must always be related to a dimension.	6.3.3
Value Type Node	<div>«valueType» <Name></div> <div>values <ValueProperty>:<ValueType>=<ValueExpression></div> <div>operations <Operation>(<Parameter>,...):<Type></div> <div>dimension=<Dimension> unit=<Unit></div>	A value type is used to provide a uniform definition of a quantity with units that can be shared by many value properties.	6.3.3
Enumeration Node	<div>«enumeration» <Name></div> <div><EnumerationLiteral></div>	An enumeration defines a set of named values called literals.	6.3.3
Actor Node	<div>«actor» <Name></div> <div> <Name></div>	An actor is used represent the role of a human, an organization, or any external system that participates in the use of some system being investigated.	11.3

Table A.4 Block Definition Diagram Nodes for Representing Interfaces








Diagram Element	Notation	Description	Section
Flow Specification Node	<div>«flowSpecification» <Name></div> <div>flowProperties <Direction> <FlowProperty>:<Item></div>	A flow specification defines the set of input and/or output flows for a noncomposite flow port. <Direction> may be one of: in, out, or inout.	6.4.3
Interface Node	<div>«interface» <Name></div> <div>operations <Operation>(<Parameters>,...):<Type></div> <div>receptions «signal»<Signal>(<Parameter>,...)</div>	An interface is used to specify the set of behavioral features either required or provided by a standard (service-based) port.	6.5.3
Port Compartments for Block Node	<div>«block» <Name></div> <div>standardPorts <Port>:<Interface></div> <div>flowPorts <Direction> <Port>:<Type></div>	Ports can be shown in separate compartments labeled flow ports and standard ports. <Direction> may be one of: in, out, or inout. Non-atomic flow ports do not have a direction but may have the keyword {conjugated}.	6.4.3, 6.5.2
Nonatomic Flow Port Node	<div><Name>:<FlowSpecification>[<Multiplicity>] </div> <div><Name>:<FlowSpecification>[<Multiplicity>] </div>	A nonatomic flow port describes an interaction point where multiple different items may flow into or out of a block. A shaded symbol implies a conjugate port.	6.4.3
Atomic Flow Port Node	<div><Name>:<Item>[<Multiplicity>] </div> <div><Name>:<Item>[<Multiplicity>] </div> <div><Name>:<Item>[<Multiplicity>] </div>	An atomic flow port describes an interaction point where an item can flow into or out of a block, or both, as indicated by the direction of the arrow in the Atomic Flow Port Node.	6.4.3
Standard Port Node	<div><Interface>  <Name>[<Multiplicity>]</div> <div><Interface>  <Name>[<Multiplicity>]</div>	A standard port defines the service-based interaction points on the interface to a block. The shape of the <Interface> symbol indicates whether services are required (socket) by or provided by (ball) the block.	6.5.3
Interface Realization Path	----->>	A realization dependency asserts that a block will declare a behavioral feature for each behavioral feature in an interface.	6.5.3
Usage Dependency Path	----->	A uses dependency asserts that a block requires a set of behavioral features defined by an interface.	6.5.3

Table A.5 Block Definition Diagram Paths

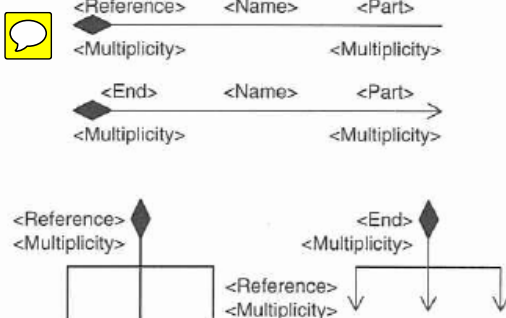
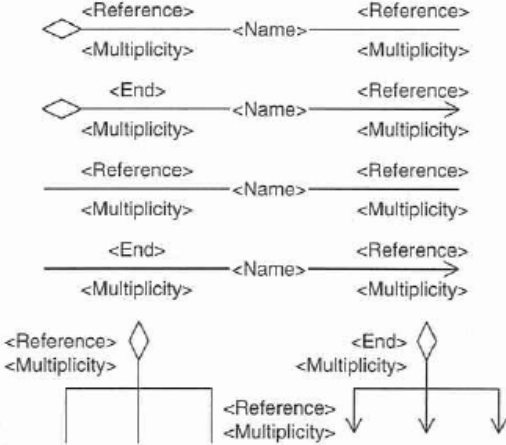
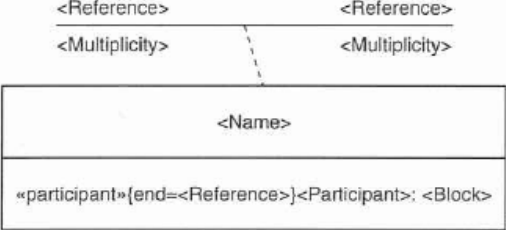

Diagram Element	Notation	Description	Section
Composite Association Path	 <p>The notation shows three examples of composite association paths. The first example is a path with a diamond at the whole end and an arrow at the part end, with labels <Reference>, <Name>, <Part>, <Multiplicity>, <End>, <Name>, <Part>, and <Multiplicity>. The second example is a path with a diamond at the whole end and a diamond at the part end, with labels <Reference>, <Multiplicity>, <End>, <Name>, <Part>, and <Multiplicity>. The third example is a path with a diamond at the whole end and a diamond at the part end, with labels <Reference>, <Multiplicity>, <End>, <Name>, <Part>, and <Multiplicity>.</p>	<p>A composite association relates a whole to its parts showing the relative multiplicity at both whole and part ends. A composite association always defines a part property in the whole (indicated by <Part>).</p> <p>Where there is no arrow on the nondiamond end of the association it also specifies a reference property to the whole in the part (indicated by <Reference>).</p> <p>Otherwise when there is an arrow, the name at the whole end simply gives a name to the association end (indicated by <End>).</p>	6.3.1
Reference Association Path	 <p>The notation shows five examples of reference association paths. The first example is a path with a diamond at the whole end and a diamond at the part end, with labels <Reference>, <Name>, <Reference>, <Multiplicity>, <End>, <Name>, <Reference>, and <Multiplicity>. The second example is a path with a diamond at the whole end and a diamond at the part end, with labels <Reference>, <Name>, <Reference>, <Multiplicity>, <End>, <Name>, <Reference>, and <Multiplicity>. The third example is a path with a diamond at the whole end and a diamond at the part end, with labels <Reference>, <Name>, <Reference>, <Multiplicity>, <End>, <Name>, <Reference>, and <Multiplicity>. The fourth example is a path with a diamond at the whole end and a diamond at the part end, with labels <Reference>, <Name>, <Reference>, <Multiplicity>, <End>, <Name>, <Reference>, and <Multiplicity>. The fifth example is a path with a diamond at the whole end and a diamond at the part end, with labels <Reference>, <Name>, <Reference>, <Multiplicity>, <End>, <Name>, <Reference>, and <Multiplicity>.</p>	<p>A reference association can be used to specify a relationship between two blocks. A reference association can specify a reference property on the blocks at one or both ends.</p> <p>The white diamond is the same as no diamond, but profiles can be used to differentiate them by specifying additional constraints.</p>	6.3.2
Association Block Path and Node	 <p>The notation shows two examples of association block paths and nodes. The first example is a path with a diamond at the whole end and a diamond at the part end, with labels <Reference>, <Name>, <Reference>, <Multiplicity>, <End>, <Name>, <Reference>, and <Multiplicity>. The second example is a path with a diamond at the whole end and a diamond at the part end, with labels <Reference>, <Name>, <Reference>, <Multiplicity>, <End>, <Name>, <Reference>, and <Multiplicity>.</p>	<p>An association block, as the name implies, is a combination of an association and a block, so it can relate two blocks together but can also have internal structure and other features of its own.</p> <p>Participants are placeholders that represent the blocks at each end of the association block, and are used when it is desired to decompose a connector.</p>	6.3.2
Generalization Path	 <p>The notation shows two examples of generalization paths. The first example is a path with a diamond at the whole end and a diamond at the part end, with labels <Reference>, <Name>, <Reference>, <Multiplicity>, <End>, <Name>, <Reference>, and <Multiplicity>. The second example is a path with a diamond at the whole end and a diamond at the part end, with labels <Reference>, <Name>, <Reference>, <Multiplicity>, <End>, <Name>, <Reference>, and <Multiplicity>.</p>	<p>A generalization describes the relationship between the general classifier and specialized classifier. A set of generalizations may either be {disjoint} or {overlapping}. They may also be {complete} or {incomplete}.</p>	6.6

Table A.8 Internal Block Diagram Nodes

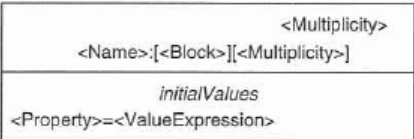

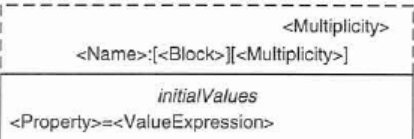
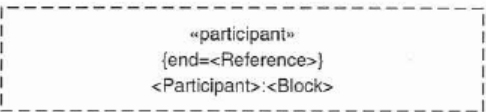
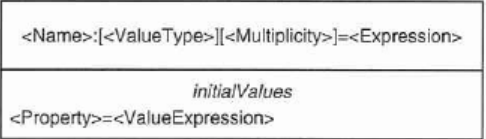
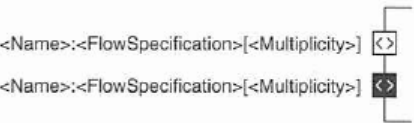
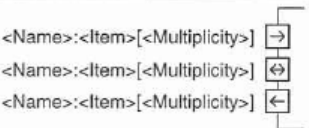
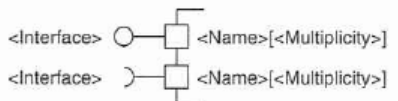
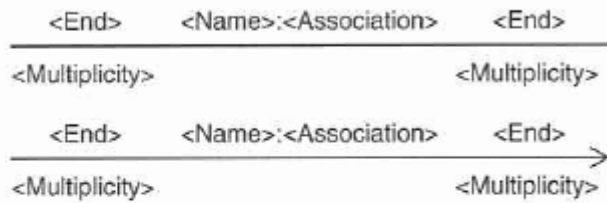
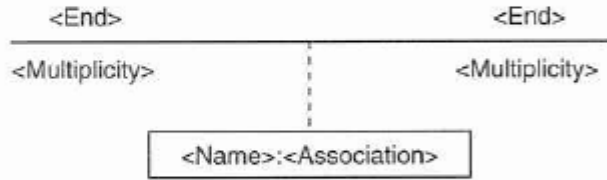
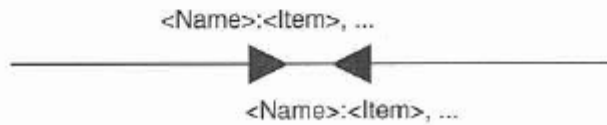
Diagram Element	Notation	Description	Section
Part Node		<p>A part is a property of an owning block that is defined (typed) by another block. The part represents a usage of the defined block in the context of the owning block.</p> <p>Note that a Part Node may have the same compartments as a Block Node. [<Block>] represents a property-specific type.</p>	6.3.1, 6.6.5
Actor Part Node	 <Name>:<Actor>[<Multiplicity>]	<p>An actor part is a property of an owning block that is defined (typed) by an actor.</p>	11.5
Reference Node		<p>A reference property of a block is a reference to another block.</p> <p>Note that a Reference Property Node may have the same compartments as a Block Node. [<Block>] represents a property-specific type.</p>	6.3.2
Participant Property Node		<p>A participant property represents one end of an association block. Using a participant property, a modeler can show the relationship between the internal structure of the association block and the internal structure of its related ends.</p>	6.3.2
Value Property Node		<p>A value property describes the quantitative characteristics of a block.</p> <p>Note that a Value Property Node may have the same compartments as a Value Type Node. [<ValueType>] represents a property-specific type.</p>	6.3.3
Nonatomic Flow Port Node		<p>A nonatomic flow port describes an interaction point that allows multiple different items to flow into or out of a block. A nonatomic flow port is typed by a flow specification. A shaded symbol implies a conjugate port that reverses the items' allowable in and out flow direction.</p>	6.4.3
Atomic Flow Port Node		<p>An atomic flow port describes an interaction point where an item can flow into or out of a block, or both, as indicated by the direction of the arrow in the Atomic Flow Port Node.</p>	6.4.3
Standard Port Node		<p>A standard port describes a service-based interaction point on a block. A standard port is defined by its interface. The shape of the <Interface> symbol indicates whether services are required by or provided by the block.</p>	6.5.3

Table A.9 Internal Block Diagram Paths

Diagram Element	Notation	Description	Section
Connector Path		A connector is used to bind two parts (or ports) and provides the opportunity for those parts to interact, although the connector says nothing about the nature of the interaction.	6.3.1
Connector Property Path and Node		More detail can be specified for connectors by typing them with association blocks. An association block, as the name implies, is a combination of an association and a block, so it can relate two blocks together but can also have internal structure and other features of its own.	6.3.2
Item Flow Node		An item flow is used to specify the items that flow across a connector in a particular context. An item flow specifies the type of the item that is flowing and the direction of flow. It may also be associated to a property, called an item property, of the enclosing block to identify a specific usage of an item in the context of the enclosing block.	6.4.2

SysML Behavior diagrams

A Practical Guide to SysML. Sanford Friedenthal, Alan Moore and Rick Steiner. Elsevier 2008.

SysML Reference Guide: Table A11, A12, A13, A14, A15, A16, A18, A19 and A20.

Table A.11 Activity Diagram Structural Nodes

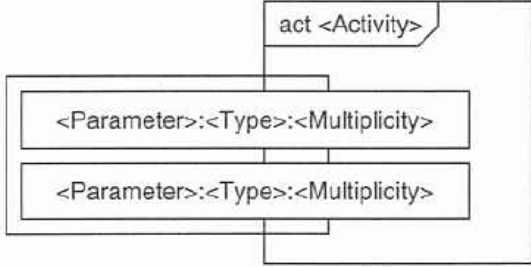

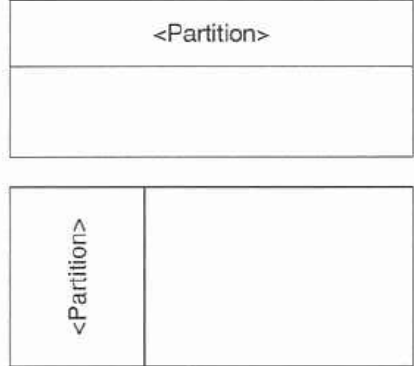

Diagram Element	Notation	Description	Section
Activity Parameter Node		<p>Activity parameter node symbols are rectangles that straddle the boundary of the activity frame.</p> <p>Other annotations include: «noBuffer», «optional», «overwrite», «continuous», «discrete», {rate=<Expression>}.</p> <p>Parameters can be organized into parameter sets, indicated by a bounding box around the parameters in the set. Parameter sets may overlap, and may have an annotation: {probability=<Expression>}.</p>	8.4.1
Interruptible Region Node		<p>An interruptible region groups a subset of the actions within an activity and includes a mechanism for stopping their execution. Stopping the execution of these actions does not effect other actions in the activity.</p>	8.8.1
Activity Partition Node		<p>A set of activity nodes can be grouped into an activity partition (also known as a swimlane) that is used to indicate responsibility for execution of those nodes. <Partition> may be the name of a block or name and type of a part/reference. Partitions may overlap in a grid pattern.</p>	8.9.1
Activity Partition in Action Node		<p>An alternative representation for an activity partition for call actions is to include the name of the partition or partitions in parentheses inside the node above the action name. This can make the activity easier to layout than when using the swimlane notation.</p>	8.9.1

Table A.12 Activity Diagram Control Nodes

Diagram Element	Notation	Description	Section
Merge Node		A merge node has one output flow and multiple input flows—it routes each input token received on any input flow to its output flow. Unlike a join node, a merge node does not require tokens on all its input flows before offering them on its output flow. Rather it offers tokens on its output flow as soon as it receives them.	8.5.1, 8.6.1
Decision Node		A decision node has one input flow and multiple output flows—an input token can only traverse one output flow. The output flow is typically established by placing mutually exclusive guards on all outgoing flows and offering the token to the flow whose guard expression is satisfied. A decision node can have an accompanying decision input behavior, which is used to evaluate each incoming object token and whose result can be used in guard expressions.	8.5.1, 8.6.1
Join Node		A join node has one output flow and multiple input flows—it has the important characteristic of synchronizing the flow of tokens from many sources. Its default behavior can be overridden by providing a join specification, which can specify additional control logic.	8.5.1, 8.6.1
Fork Node		A fork node has one input flow and multiple output flows—it replicates every input token it receives onto each of its output flows. The tokens on each output flow may be handled independently and concurrently.	8.5.1, 8.6.1
Initial Node		When an activity starts executing a control token is placed on each initial node in the activity. The token can then trigger the execution of an action via an outgoing control flow.	8.6.1
Activity Final Node		When a control or object token reaches an activity final node during the execution of an activity, the execution terminates.	8.6.1
Flow Final Node		Control or object tokens received at a flow final node are consumed but have no effect on the execution of the enclosing activity. Typically they are used to terminate a particular sequence of actions without terminating an activity.	8.6.1

Table A.13 Activity Diagram Object and Action Nodes

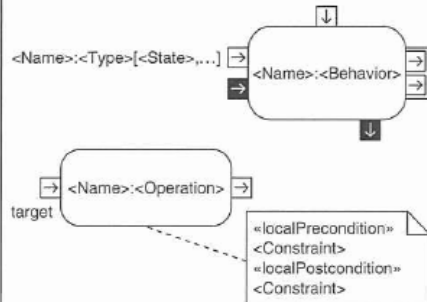
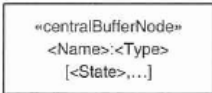
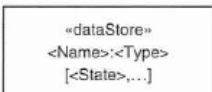
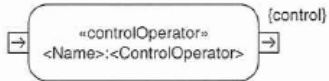

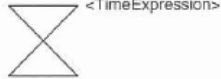

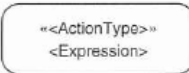
Diagram Element	Notation	Description	Section
Call Action Node		<p>Call actions can invoke other behaviors either directly or through an operation, and are referred to as call behavior actions and call operation actions, respectively. A call action must own a set of pins that match in number and type of the parameters of the invoked behavior/operation. A called operation requires a target. Streaming pins may be marked as {stream} or filled (as shown).</p> <p>Where the parameters of the called entity are grouped into sets, the corresponding pins are as well. Pre- and postconditions can be specified that constrain the action such that it cannot begin to execute unless the precondition is satisfied, and must satisfy the postcondition to successfully complete execution.</p>	8.1, 8.3, 8.4.2
Central Buffer Node		A central buffer node provides a store for object tokens outside of pins and parameter nodes. Tokens flow into a central buffer node and are stored there until they flow out again.	8.5.3
Datastore Node		A datastore node provides a copy of a stored token rather than the original. When an input token represents an object that is already in the store, it overwrites the previous token.	8.5.3
Control Operator Action Node		A control operator produces control values on an output parameter, and is able to accept a control value on an input parameter (treated as an object token). It is used to specify logic for enabling and disabling other actions.	8.6.2
Accept Event Action Node		An activity can accept events using an accept event action. The action has (sometimes hidden) output pins for received data.	8.7
Accept Time Event Node		A time event corresponds to an expiration of an (implicit) timer. In this case the action has a single (typically hidden) output pin that outputs a token containing the time of the accepted event occurrence.	8.7
Send Signal Action		An activity can send signals using a send signal action. It typically has pins corresponding to the signal data to be sent and the target for the signal.	8.7
Primitive Action Node		Primitive actions include: object access/update/manipulation actions, which involve properties and variables, and value actions, which allow the specification of values. The <Expression> will depend on the nature of the action.	8.12.1

Table A.14 Activity Diagram Paths

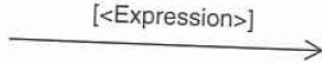
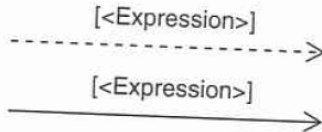
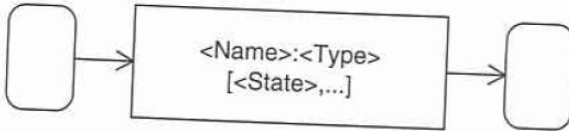
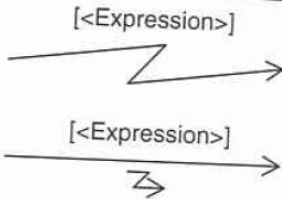
Diagram Element	Notation	Description	Section
Object Flow Path		Object flows connect inputs and outputs. Additional annotations include «continuous», «discrete», {rate=<Expression>}, {probability=<Expression>}.	8.1, 8.5
Control Flow Path		Control flows provide constraints on when, and in what order, the actions within an activity will execute. A control flow can be represented using a solid line, or using a dashed line to more clearly distinguish it from object flow.	8.1, 8.6
Object Flow Node		When an object flow is between two pins that have the same characteristics, an alternative notation can be used where the pin symbols are elided and replaced by a single rectangular symbol called an object node symbol.	8.5
Interrupting Edge Path		An interrupting edge interrupts the execution of the actions in an interruptible region. Its source is a node inside the region and its destination is a node outside it.	8.8.1

Table A.15 Sequence Diagram Structural Nodes

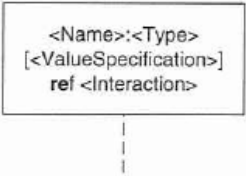
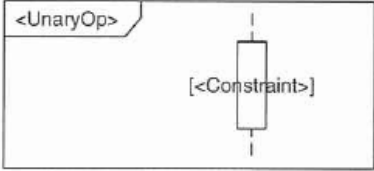
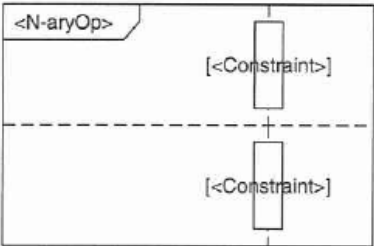

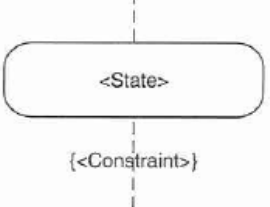
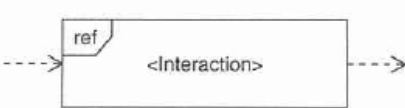
Diagram Element	Notation	Description	Section
Lifeline Node		A lifeline represents the relevant lifetime of an instance that is part of the interaction's owning block, which will either be represented by a part property or a reference property.	9.4
Single-compartment Fragment Node		A combined fragment can be used to model complex sequences of messages. A number of combined fragments have operators with only a single compartment for all operands, shown as <UnaryOp>. These are: seq, opt, break, strict, loop, neg, assert, critical.	9.7.1, 9.7.2
Multi-compartment Fragment Node		Two combined fragments have operators with a compartment per operand, shown as <N-aryOp>. These are par and alt. The lifelines that participate in the fragment overlay on top of the fragment (i.e., are visible) and lifelines that don't participate are obscured behind the fragment. (Note: This is also true of Single-Compartment Fragment Nodes.)	9.7.1
Filtering Fragment Node		There are two combined fragments with filter operators: consider and ignore, shown as <FilterOp>. Inside such a construct, messages that have been explicitly ignored (or not considered) may be interleaved with valid traces.	9.7.2
State Invariant Symbol		A state invariant on a lifeline is used to add a constraint on the required state of a lifeline at a given point in a sequence of event occurrences. The invariant constraint can include the values of properties or parameters, or the state of a state machine.	9.7.3
Interaction Use Node		An interaction use allows one interaction to reference another as part of its definition. The lifelines that participate in the interaction are obscured behind the fragment, and lifelines that don't participate overlay on top of the fragment (i.e., are visible).	9.8

Table A.16 Sequence Diagram Paths and Activation Nodes

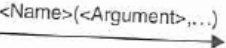
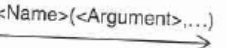
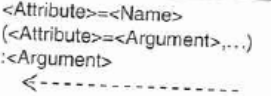
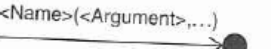
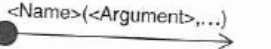
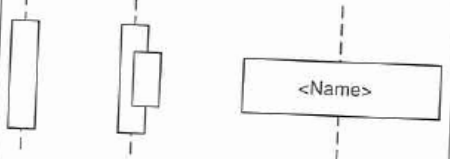
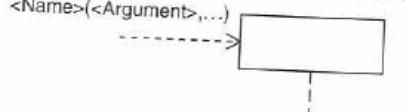


Diagram Element	Notation	Description	Section
Synchronous Message		A synchronous message corresponds to the synchronous invocation of an operation, and is generally accompanied by a reply message.	9.5.1
Asynchronous Message		Asynchronous messages correspond to either the sending of a signal or to an asynchronous invocation (or call) of an operation, and do not require a reply message.	9.5.1
Reply Message		A reply message shows a reply to a synchronous operation call, together with any return arguments.	9.5.1
Found Message Path		A lost message describes the case where there is sending event for the message but no receiving event.	9.5.2
Lost Message Path		A found message describes the case where there is receiving event for the message but no sending event.	9.5.2
Focus of Control (Activation) Node		Focus of control bars or activations are overlaid on lifelines and correspond to executions; they begin at the execution's start event, and end at the execution's end event. When executions are nested, the focus of control bars are stacked from left to right. An alternate notation for activations is a box symbol overlaid on the lifeline with the name of the behavior or action inside.	9.5.4
Create Message Path		The creation of an instance is indicated by the receipt of a create message.	9.5.5
Destroy Event Node		An instance's destruction is indicated by the occurrence of a destroy event.	9.5.5
Coregion Symbol		Within a coregion, there is no implied order between any messages sent or received by the lifeline.	9.7.1

Table A.18 State Machine Diagram State Nodes

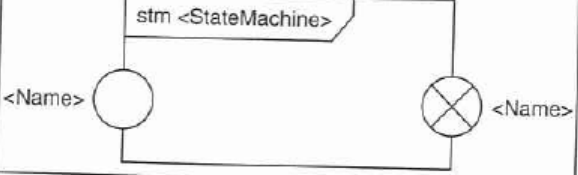
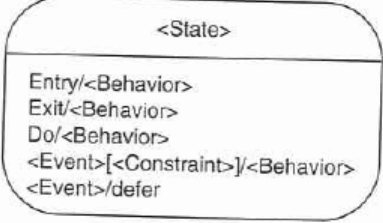
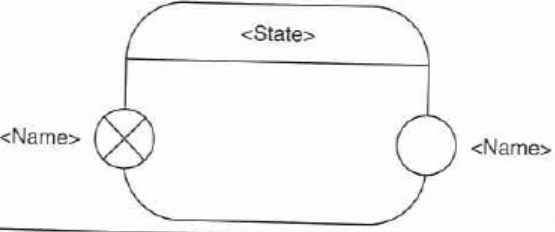
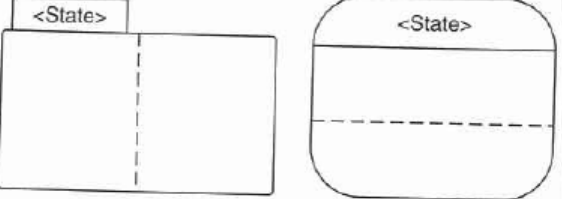
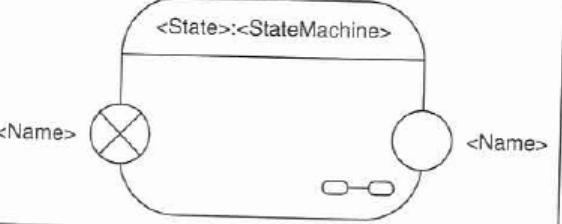
Diagram Element	Notation	Description	Section
State Machine with Entry- and Exit-Point Pseudostate Nodes		A state machine may have entry- and exit-point pseudostates, which are similar to junctions. On state machines, entry-point pseudostates can only have outgoing transitions and exit-point pseudostates can only have incoming transitions.	10.6.5
Atomic State Node		A state represents some significant condition in the life of a block, typically because it represents some change in how the block responds to events. Each state may have entry and exit behaviors that are performed whenever the state is entered or exited, respectively. In addition, the state may perform a do activity that executes once the entry behavior has completed and continues to execute until it completes or the state is exited.	10.3
Composite State with Entry- and Exit-Point Pseudostate Nodes		A composite state is a state with nested regions; the most common case is a single region. A composite state may have entry- and exit-point pseudostates that act like junction pseudostates. Entry points have incoming transitions from outside the state and exit points have the opposite.	10.6.1
Composite State Node with Multiple Regions		A composite state may have many regions, which may each contain substates. These regions are orthogonal to each other and so a composite state with more than one region is sometimes called an orthogonal composite state.	10.6.2
Sub-State Machine Node with Connection Points		A state machine may be reused using a kind of state called a submachine state. A transition ending on a submachine state will start its referenced state machine. Transitions may also be connected to connection points on the boundary of the state.	10.6.5

Table A.19 State Machine Diagram Pseudostate and Transition Nodes



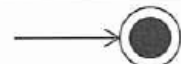


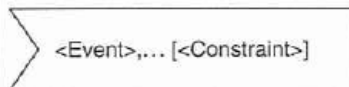

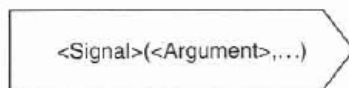
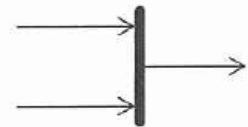


Diagram Element	Notation	Description	Section
Terminate Pseudostate Node		If a terminate pseudostate is reached, then the behavior of the state machine terminates.	10.3
Initial Pseudostate Node		An initial pseudostate specifies the initial state of a region.	10.3
Final State Node		The final state indicates that a region has completed execution.	10.3
Choice Pseudostate Node		The outgoing transitions of a choice pseudostate are evaluated once it has been reached.	10.4.2
Junction Pseudostate Node		A junction pseudostate is used to construct a compound transition path between states.	10.4.2
Trigger Node		This node represents all the transition's triggers, with the descriptions of the triggering events and the transition guard inside the symbol.	10.4.3
Action Node		<EffectExpression> describes the effect of the transition, either the name of a behavior or the body of an opaque behavior.	10.4.3
Send Signal Node		This node represents a send signal action. The signal's name, together with any arguments that are being sent, are shown within the symbol.	10.4.3
Join Pseudostate Node		A join pseudostate has a single outgoing transition and many incoming transitions. When all of the incoming transitions can be taken, and the join's outgoing transition is valid, then all the transitions happen.	10.6.2
Fork Pseudostate Node		A fork pseudostate has a single incoming transition and many outgoing transitions. When an incoming transition is taken to the fork pseudostate, all of the outgoing transitions are taken.	10.6.2
History Pseudostate Node		A history pseudostate represents the last state of its owning region, and a transition ending on a history pseudostate has the effect of returning the region to the state it was last in.	10.6.4

Table A.20 State Machine Diagram Paths

Diagram Element	Notation	Description	Section
Time Event Transition Path	$\text{after } \langle \text{TimeExpression} \rangle [\langle \text{Constraint} \rangle] / \langle \text{Behavior} \rangle \longrightarrow$ $\text{at } \langle \text{TimeExpression} \rangle [\langle \text{Constraint} \rangle] / \langle \text{Behavior} \rangle \longrightarrow$	Time events indicate either that a given time interval has passed since the current state was entered (after), or that a given instant of time has been reached (at). The transition can also include a guard and effect.	10.4.1
Signal Event Transition Path	$\langle \text{Signal} \rangle (\langle \text{Attribute} \rangle, \dots) [\langle \text{Constraint} \rangle] / \langle \text{Behavior} \rangle \longrightarrow$	Signal events indicate that a new asynchronous message has arrived. A signal event may be accompanied by a number of arguments, which may be assigned to attributes. The transition can also include a guard and effect.	10.4.1
Call Event Transition Path	$\langle \text{Operation} \rangle (\langle \text{Attribute} \rangle, \dots) [\langle \text{Constraint} \rangle] / \langle \text{Behavior} \rangle \longrightarrow$	Call events indicate that an operation on the state machine's owning block has been requested. A call event may also be accompanied by a number of arguments, which may be assigned to attributes. The transition can also include a guard and effect.	10.5
Change Event Transition Path	$\text{when } \langle \text{Expression} \rangle [\langle \text{Constraint} \rangle] / \langle \text{Behavior} \rangle \longrightarrow$	Change events indicate that some condition has been satisfied (normally that some specific set of attribute values hold). The transition can also include a guard and behavior/effect.	10.7