

What is State Machine Diagram?

[What is State Machine Diagram?](#)

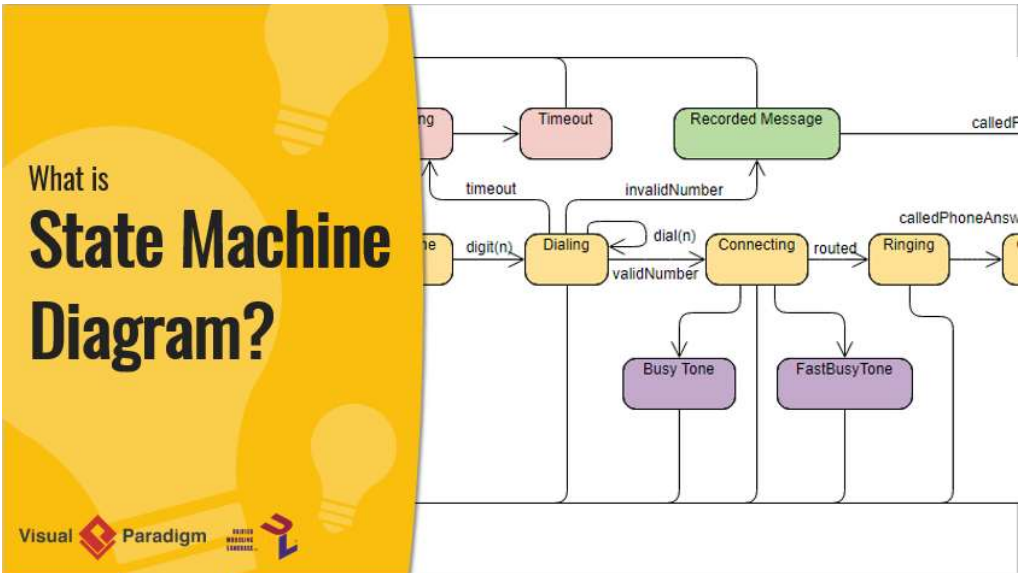
[Why State Machine Diagrams](#)

[Basic Concepts of State Machine Diagram](#)

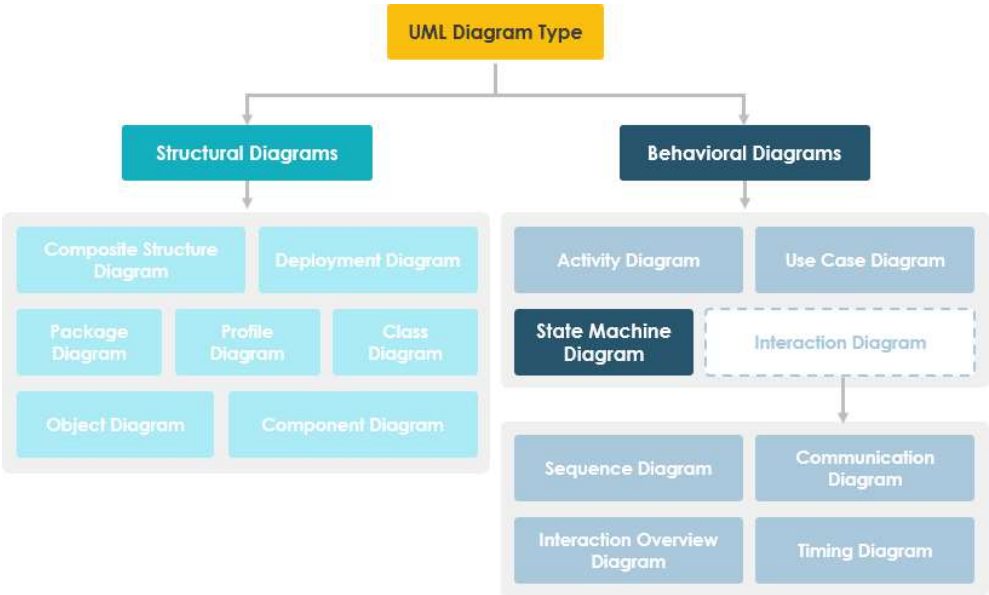
[Characteristics of State Machine Notations](#)

[Simple State Machine Diagram Notation](#)

[Related Links](#)



The behavior of an entity is not only a direct consequence of its inputs, but it also depends on its preceding state. The past history of an entity can best be modeled by a finite state machine diagram or traditionally called automata. [UML \(https://en.wikipedia.org/wiki/Unified_Modeling_Language\)](https://en.wikipedia.org/wiki/Unified_Modeling_Language) State Machine Diagrams (or sometimes referred to as state diagram, state machine or state chart) show the different states of an entity. State machine diagrams can also show how an entity responds to various events by changing from one state to another. State machine diagram is a UML diagram used to model the dynamic nature of a system.



Learn UML Faster, Better and Easier

Are you looking for a Free UML tool for learning UML faster, easier and quicker?
Visual Paradigm Community Edition is a UML software that supports all UML diagram types. It is an international award-winning UML modeler, and yet it is easy-to-use, intuitive & completely free.

[Free Download \(/download/community.jsp\)](#)

Why State Machine Diagrams?

State machine diagram typically are used to describe state-dependent behavior for an object. **An object responds differently to the same event depending on what state it is in.** State machine diagrams are usually applied to objects but can be applied to any element that has behavior to other entities such as: actors, use cases, methods, subsystems systems and etc. and they are typically used in conjunction with interaction diagrams (usually sequence diagrams).

For example:

Consider you have \$100,000 in a bank account. The behavior of the withdraw function would be: `balance := balance - withdrawAmount;` provided that **the balance after the withdrawal is not less than \$0**; this is true regardless of how many times you have withdrawn money from the bank. In such situations, the withdrawals do not affect the abstraction of the attribute values, and hence the gross behavior of the object remains unchanged.

However, if the **account balance would become negative after a withdrawal**, the behavior of the withdraw function would be quite different. This is because the state of the bank account is changed from positive to negative; in technical jargon, a transition from the positive state to the negative state is fired.

The abstraction of the attribute value is a property of the system, rather than a globally applicable rule. For example, if the bank changes the business rule to allow the bank balance to be overdrawn by 2000 dollars, the state of the bank account will be redefined with condition that the balance after withdrawal must not be less than \$2000 in deficit.

Note That:

- A state machine diagram describes all events (and states and transitions for a single object)
- A sequence diagram describes the events for a single interaction across all objects involved

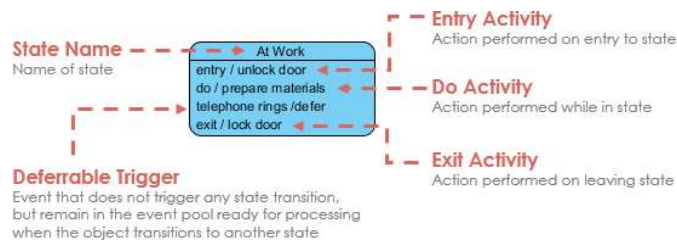
Basic Concepts of State Machine Diagram

What is a State?

Rumbaugh defines that:

"A state is an abstraction of the attribute values and links of an object. Sets of values are grouped together into a state according to properties that affect the gross behavior of the object."

State Notation



Characteristics of State Machine Notations

There are several characteristics of states in general, regardless of their types:

- A state occupies an interval of time.
- A state is often associated with an abstraction of attribute values of an entity satisfying some condition(s).
- An entity changes its state not only as a direct consequence of the current input, but it is also dependent on some past history of its inputs.

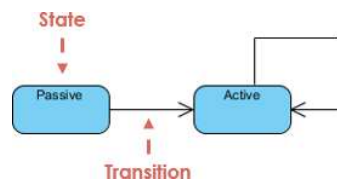
State

A state is a constraint or a situation in the life cycle of an object, in which a constraint holds, the object executes an activity or waits for an event.

A state machine diagram is a graph consisting of:

- States (simple states or composite states)
- State transitions connecting the states

Example:



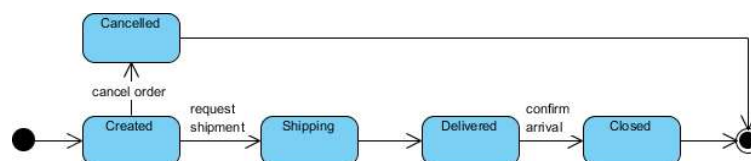
Characteristics of State

- State represent the conditions of objects at certain points in time.
- Objects (or Systems) can be viewed as moving from state to state
- A point in the lifecycle of a model element that satisfies some condition, where some particular action is being performed or where some event is waited

Initial and Final States

- The **initial state** of a state machine diagram, known as an initial pseudo-state, is indicated with a solid circle. A transition from this state will show the first real state
- The **final state** of a state machine diagram is shown as concentric circles. An open loop state machine represents an object that may terminate before the system terminates, while a closed loop state machine diagram does not have a final state; if it is the case, then the object lives until the entire system terminates.

Example:



Events

An event signature is described as Event-name (comma-separated-parameter-list). Events appear in the internal transition compartment of a state or on a transition between states. An event may be one of four types:

1. Signal event - corresponding to the arrival of an asynchronous message or signal
2. Call event - corresponding to the arrival of a procedural call to an operation
3. Time event - a time event occurs after a specified time has elapsed
4. Change event - a change event occurs whenever a specified condition is met

Characteristics of Events

- Represents incidents that cause objects to transition from one state to another.
- Internal or External Events trigger some activity that changes the state of the system and of some of its parts
- Events pass information, which is elaborated by Objects operations. Objects realize Events
- Design involves examining events in a state machine diagram and considering how those events will be supported by system objects

Transition

Transition lines depict the movement from one state to another. Each transition line is labeled with the **event** that causes the transition.

- Viewing a system as a set of states and transitions between states is very useful for describing complex behaviors
- Understanding state transitions is part of system analysis and design
- A Transition is the movement from one state to another state
- Transitions between states occur as follows:
 1. An element is in a source state
 2. An event occurs
 3. An action is performed
 4. The element enters a target state
- Multiple transitions occur either when different events result in a state terminating or when there are guard conditions on the transitions
- A transition without an event and action is known as automatic transitions

Actions

Action is an executable atomic computation, which includes operation calls, the creation or destruction of another object, or the sending of a signal to an object. An action is associated with transitions and during which an action is not interruptible - e.g., entry, exit

Activity

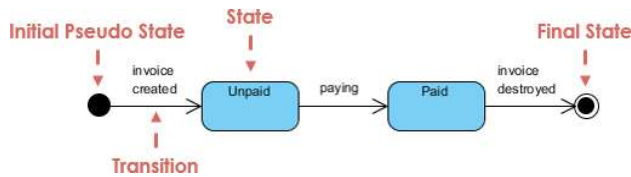
Activity is associated with states, which is a non-atomic or ongoing computation. Activity may run to completion or continue indefinitely. An Activity will be terminated by an event that causes a transition from the state in which the activity is defined

Characteristics of Action and Activities

- States can trigger actions

- States can have a second compartment that contains actions or activities performed while an entity is in a given state
- An action is an atomic execution and therefore completes without interruption
- Five triggers for actions: On Entry, Do, On Event, On Exit, and Include
- An activity captures complex behavior that may run for a long duration - An activity may be interrupted by events, in which case it does not complete occur when an object arrives in a state.

Simple State Machine Diagram Notation



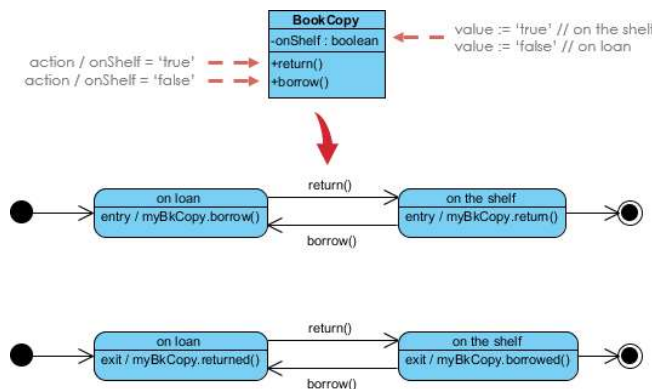
Entry and Exit Actions

Entry and Exit actions specified in the state. It must be true for every entry / exit occurrence. If not, then you must use actions on the individual transition arcs

- **Entry Action** executed on entry into state with the notation: **Entry / action**
- **Exit Action** executed on exit from state with the notation: **Exit / action**

Example - Entry / Exit Action (Check Book Status)

This example illustrates a state machine diagram derived from a Class - "BookCopy":



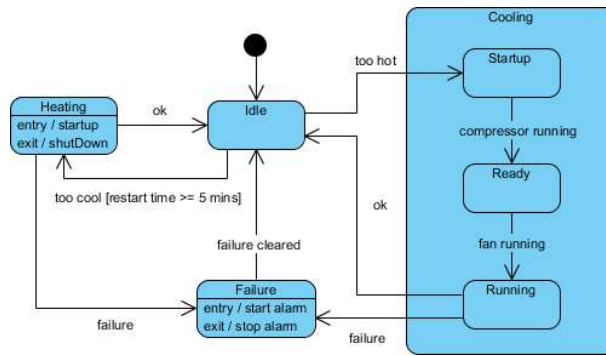
Note:

1. This state machine diagram shows the state of an object myBkCopy from a BookCopy class
2. Entry action : any action that is marked as linked to the entry action is executed whenever the given state is entered via a transition
3. Exit action : any action that is marked as linked to the exit action is executed whenever the state is left via a transition

Substates

A simple state is one which has no substructure. A state which has substates (nested states) is called a composite state. Substates may be nested to any level. A nested state machine may have at most one initial state and one final state. Substates are used to simplify complex flat state machines by showing that some states are only possible within a particular context (the enclosing state).

Substate Example - Heater

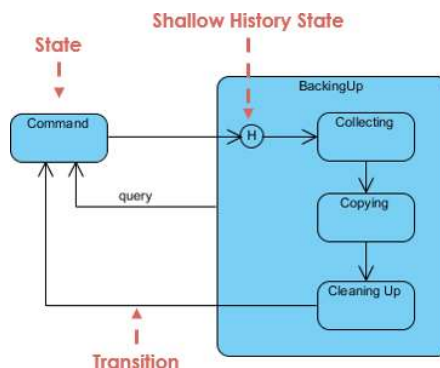


State Machine Diagrams are often used for deriving testing cases, here is a list of possible test ideas:

- Idle state receives Too Hot event
- Idle state receives Too Cool event
- Cooling/Startup state receives Compressor Running event
- Cooling/Ready state receives Fan Running event
- Cooling/Running state receives OK event
- Cooling/Running state receives Failure event
- Failure state receives Failure Cleared event
- Heating state receives OK event
- Heating state receives Failure event

History States

Unless otherwise specified, when a transition enters a composite state, the action of **the nested state machine starts over again at the initial state** (unless the transition targets a substate directly). History states allow the state machine to **re-enter the last substate that was active prior to leaving** the composite state. An example of history state usage is presented in the figure below.

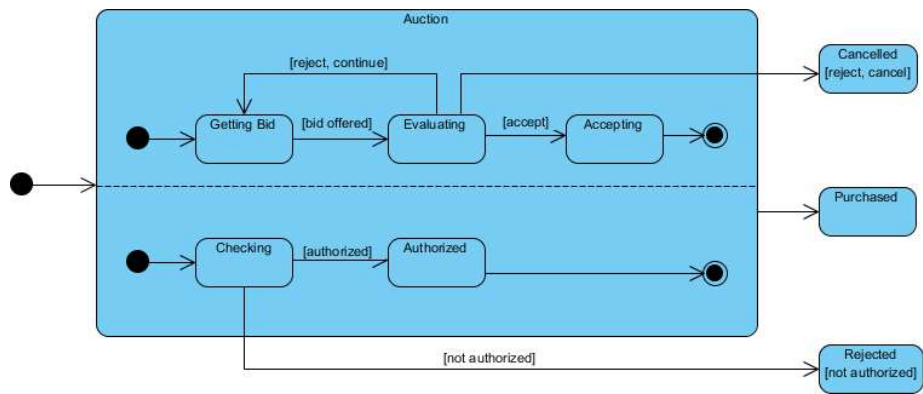


Concurrent State

As mentioned above, states in state machine diagrams can be nested. Related states can be grouped together into a single composite state. Nesting states inside others is necessary when an activity involves concurrent sub-activities. The following state machine diagram models an auction with two concurrent substates: processing the bid and authorizing the payment limit.

Concurrent State Machine Diagram Example - Auction Process

In this example, the state machine first entering the Auction requires a fork at the start into two separate start threads. Each substate has an exit state to mark the end of the thread. Unless there is an abnormal exit (Canceled or Rejected), the exit from the composite state occurs when both substates have exited.



Try to Draw UML State Machine Diagram Now

You've learned what a State Machine Diagram is and how to draw a State Machine Diagram. It's time to draw a State Machine Diagram of your own. Get Visual Paradigm Community Edition, a free UML software, and create your own State Machine Diagram with the free State Machine Diagram tool. It's easy-to-use and intuitive.

[Free Download \(/download/community.jsp\)](/download/community.jsp)

Related Links

1. [What is Unified Modeling Language? \(/guide/uml-unified-modeling-language/what-is-uml/\)](/guide/uml-unified-modeling-language/what-is-uml/)
2. [Professional UML diagram tool \(/features/uml-tool/\)](/features/uml-tool/)

Turn every software project into a successful one.

[Try Visual Paradigm Free \(/download/\)](/download/)

..(.)

Product	Support	Learn	Company
Features (/features/)	Forums	Community Circle	About Us (/aboutus/)
Editions (/editions/)	(http://forums.visual-paradigm.com/)	(https://circle.visual-paradigm.com/)	Newsroom
Try Now (/download/)	Request Help	Know-how	(/aboutus/newsreleases/)
Pricing (/shop/)	(/support/#support-form)	(https://knowhow.visual-paradigm.com/)	YouTube Channel
Visual Paradigm Online	Customer Service	Demo Videos	Academic
(https://online.visual-paradigm.com/)	(https://cs.visual-paradigm.com/)	(/features/demo/)	Partnership
		Tutorials (/tutorials/)	(/partner/academic/)
		Documents	
		(/support/documents/)	

(<https://twitter.com/visualparadigm>)



(<https://www.facebook.com/Visual-Paradigm-822068561487170/>)



(<https://www.linkedin.com/company/visual-paradigm>)



(<https://www.pinterest.com/visualparadigm/>)

(<https://www.youtube.com/user/VisualParadi>)



(<https://www.instagram.com/visualparadigm>)