Short note of the following in UML with suitable examples:

- Events and signals,
- State machines,
- Processes and Threads,
- time and space,
- State chart diagrams.

Events and Signals

- An **event** is nothing but the action taken by triggering something either by user or system.
- An event contains the properties of time and space to denote where the action is taking place.
- Kinds of Events

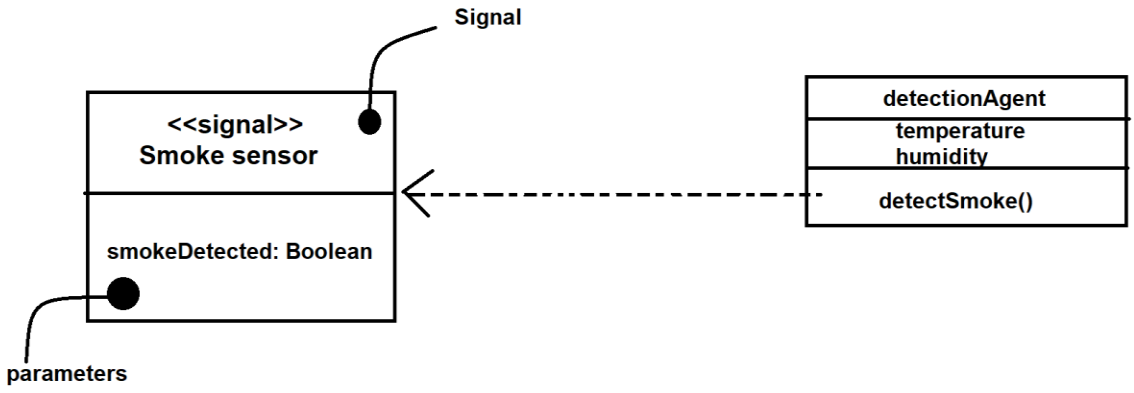
  Events may be **external** or **internal**.

  External events are those that happen between the system and its actors, here the users.

  For example, the pushing of a button is an external event where a user performs the action of hitting the button.

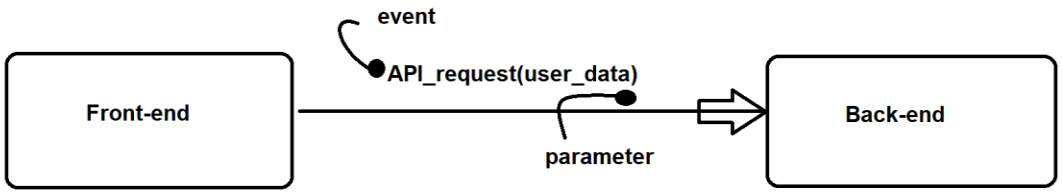
  Another example can be an alarm of a smoke sensor in a building to detect fire.

  Internal events are those that happen between let's say the objects that exist inside the programs and modules of the codebase of a project. So here, if an overflow exception is raised by the compiler then it is an example of an internal event.



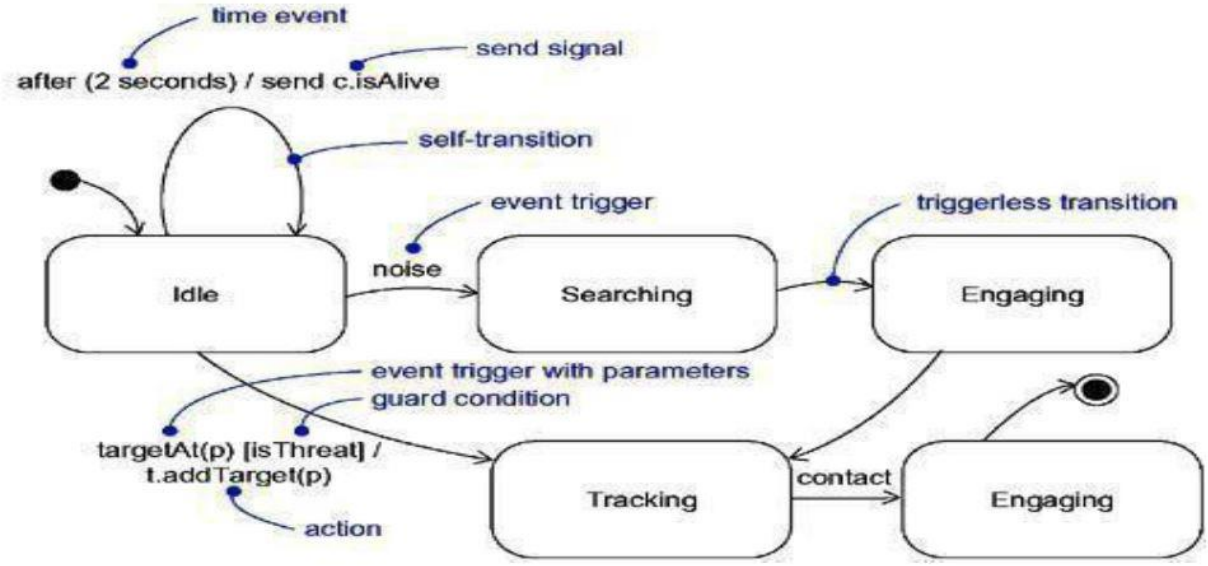
Signals

- A **signal** is a kind of event that represents the specification of an asynchronous stimulus communicated between instances.
- A signal represents a named object that is dispatched (thrown) asynchronously by one object and then received (caught) by another.
- For example, sending an API request from front-end to back-end is a signal. Then the response that we receive from the back-end can also be accounted as a signal which can be a success message 200 or error message 404.



State Machines

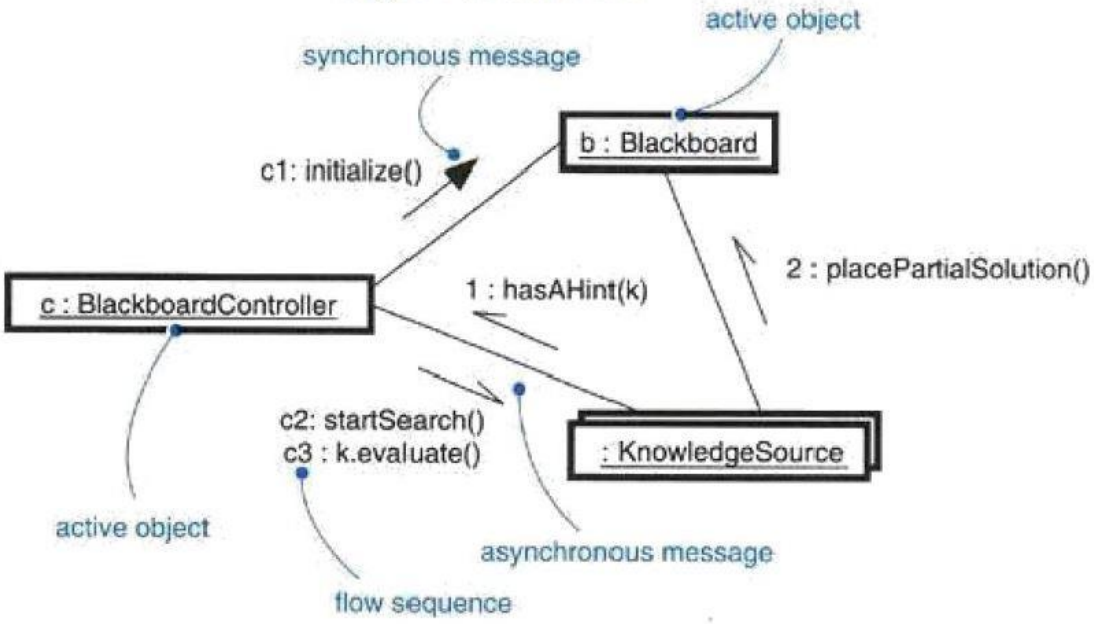
- A **state machine** is a behavior that specifies the sequences of states an object goes through during its lifetime in response to events, together with its responses to those events.
- In other words, a state machine is like a blue-print of all the states present in the lifecycle of a project which maps the events, its responses in a formal format.
- A **state** is a condition or situation during the life of an object during which it satisfies some condition, performs some activity, or waits for some event.
- An **event** is the specification of a significant occurrence that has a location in time and space. In the context of state machines, an event is an occurrence of a stimulus that can trigger a state transition.
- A **transition** is a relationship between two states indicating that an object in the first state will perform certain actions and enter the second state when a specified event occurs and specified conditions are satisfied.
- An **activity** is ongoing nonatomic execution within a state machine.
- An **action** is an executable atomic computation that results in a change in state of the model or the return of a value. Graphically, a state is rendered as a rectangle with rounded corners. A transition is rendered as a solid directed line.



Processes and thread

- An active object is an object that owns a process or thread and can initiate control activity.
- An active class is a class whose instances are active objects.
- A process is a heavyweight flow that can execute concurrently with other processes.
- A thread is a lightweight flow that can execute concurrently with other threads within the same process.
- Graphically, an active class is rendered as a rectangle with thick lines. Processes and threads are rendered as stereotyped active classes (and also appear as sequences in interaction diagrams).

Figure Communication



Time and Space

- A timing mark is a denotation for the time at which an event occurs.
- Graphically, a timing mark is formed as an expression from the name given to the message (which is typically different from the name of the action dispatched by the message).
- A time expression is an expression that evaluates to an absolute or relative value of time.
- A timing constraint is a semantic statement about the relative or absolute value of time.
- Graphically, a timing constraint is rendered as for any constraint that is, a string enclosed by brackets and generally connected to an element by a dependency relationship.
- Location(Space) is the placement of a component on a node.
- Graphically, location is rendered as a tagged value, that is, a string enclosed by brackets and placed below an element's name, or as the nesting of components inside nodes.

Figure Space(location)

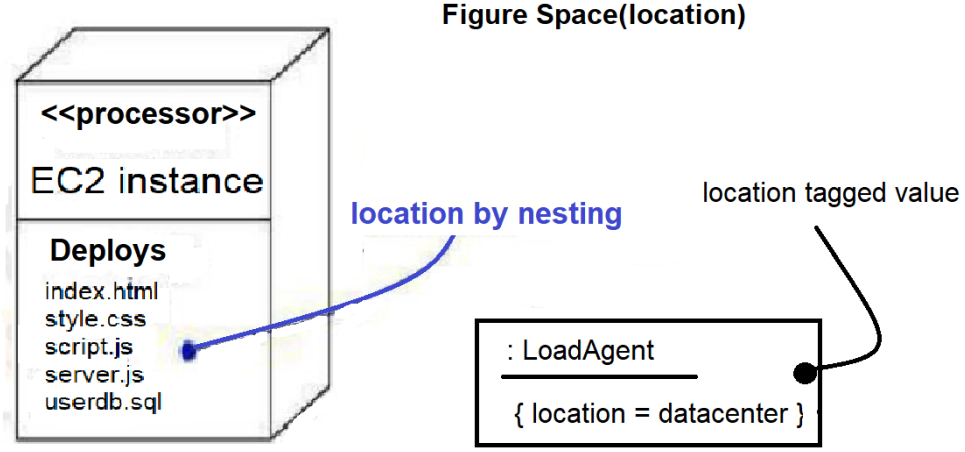
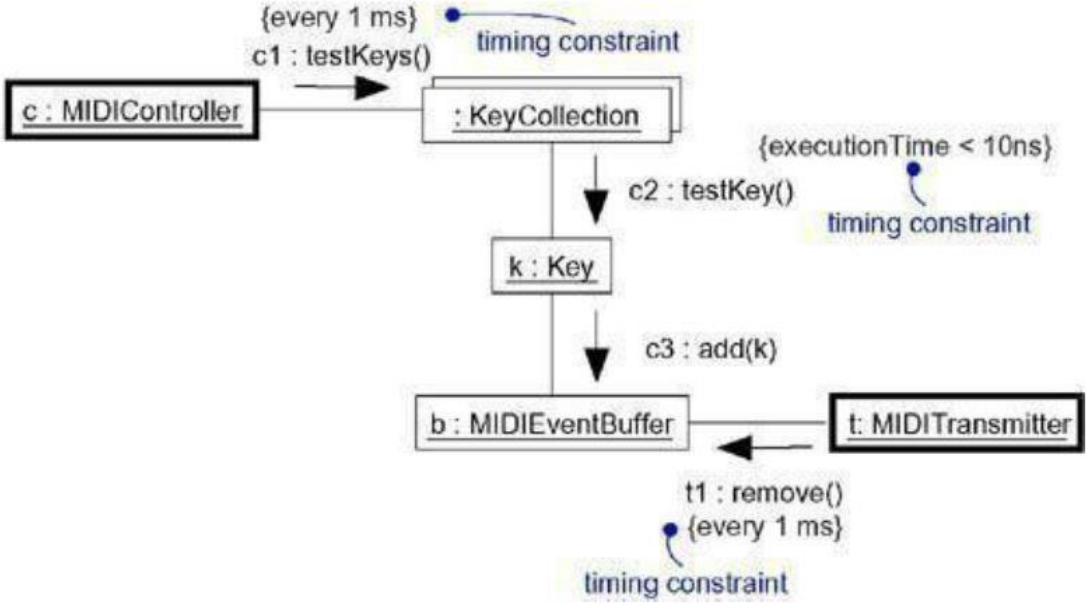


Figure Time


{every 1 ms}
timing constraint
c1 : testKeys()
c : MIDIController
: KeyCollection
{executionTime < 10ns}
c2 : testKey()
timing constraint
k : Key
c3 : add(k)
b : MIDIEventBuffer
t: MIDITransmitter
t1 : remove()
{every 1 ms}
timing constraint