UML State Machine Diagrams

Introduction

- The state machine diagrams include two types of diagrams:

Behavioral State Machines

Shows the behavior of the elements implementation.

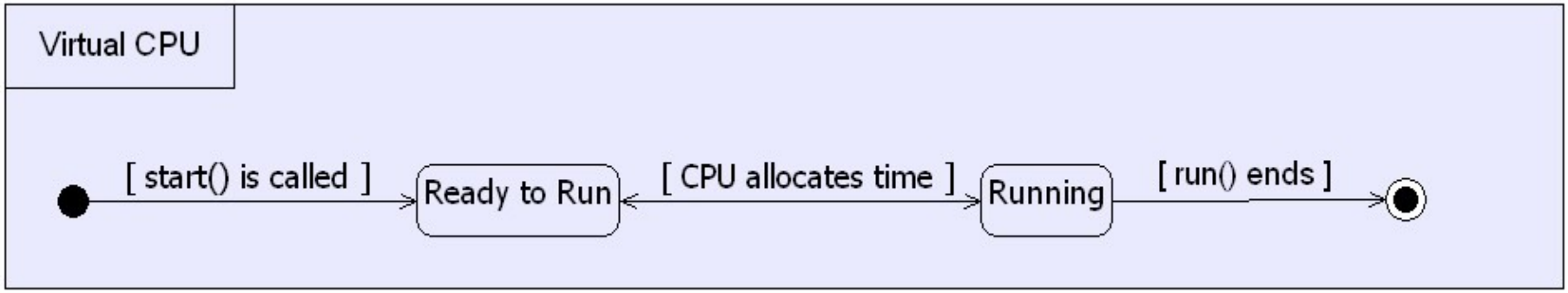
Protocol State Machines

Shows the behavior of a protocol.

Behavioral State Machines

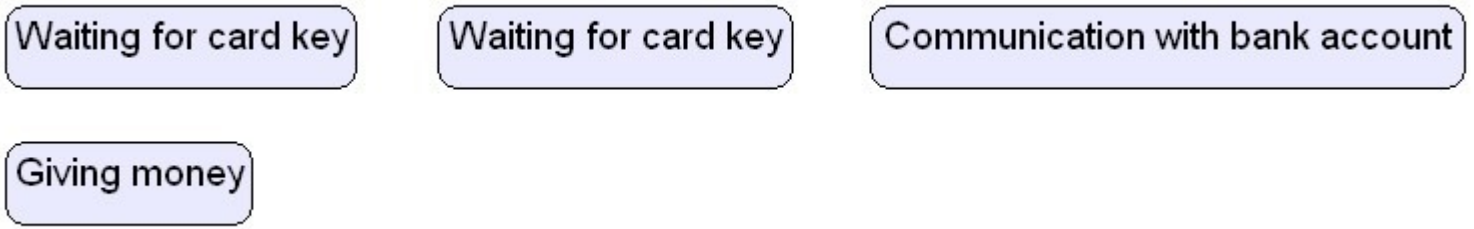
Introduction

- The behavioral state machines shows the behavior of the elements implementation.
- Simple basic rectangle notation with the name of the machine inside the top compartment surrounds the diagram and tells us which machine the diagram describes.



Simple States

- A “simple state” models a specific situation in the machine behavior. The “simple state” is defined using a condition (e.g. “Running”, “Waiting to run”, “Waiting to get the lock flag” etc.).
- The “simple state” notation is a rectangle with rounded corners. The name of the state is written inside.



Simple States

- The rounded rectangle can be divided into the following three compartments:

  Name

  The name of the simple state.

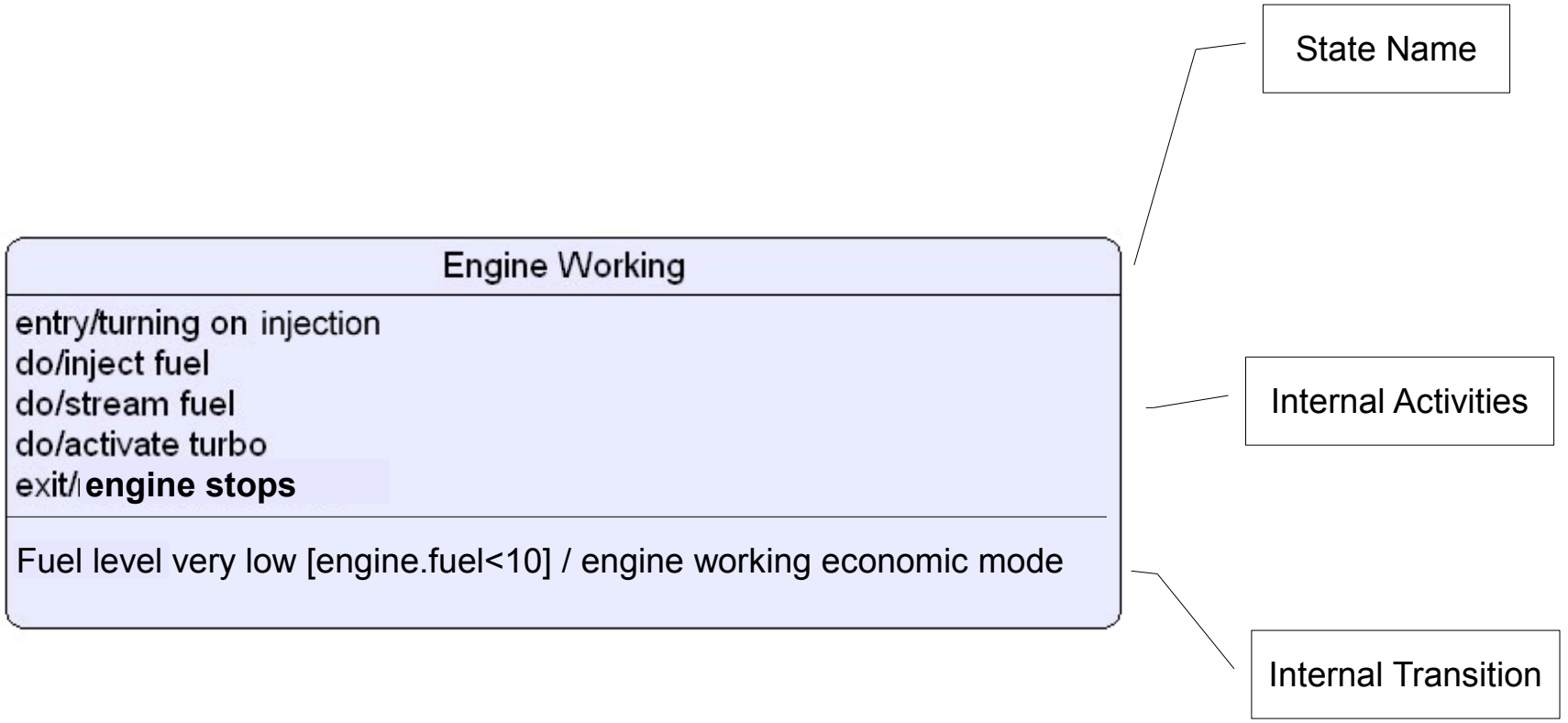
  Internal Activities

  Includes a list of available internal activities performed as long as the machine is in this simple state. Three types of internal activities: entry, do & exit.

  Internal Transitions

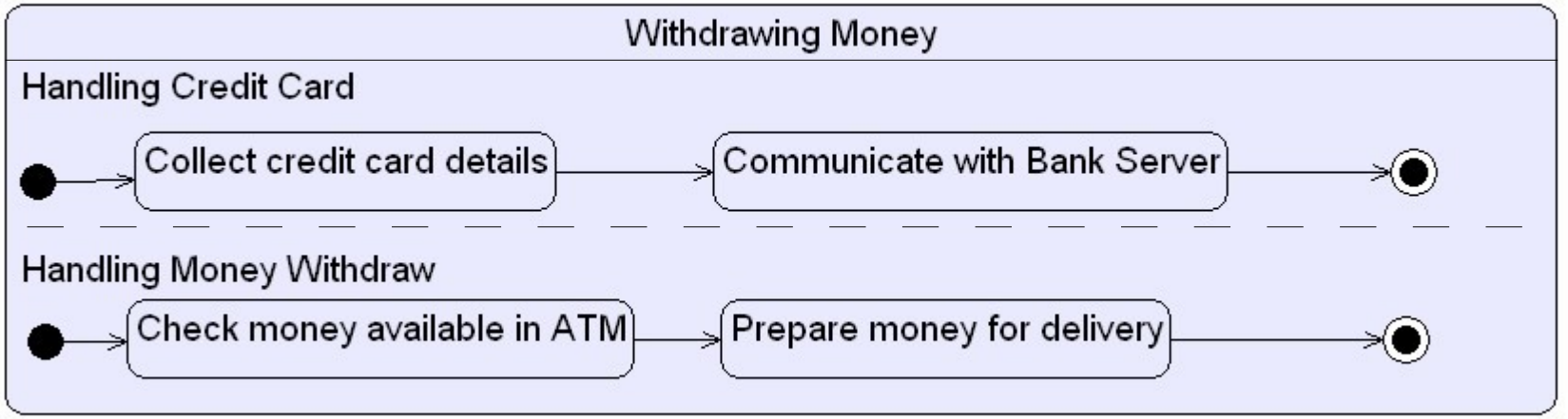
  Includes a list of internal transitions and events that trigger them. This part uses the following pattern: event (attributes list) [guard condition] / transition

Simple States



Composite States

- A “composite state” is a state that includes sub states within inner region\s.



Submachine States

- A “submachine state” is a state contained by another state.
- A "submachine state" notation is a rounded rectangle that includes the name of the outer state, followed by a colon (:) and the name of the submachine state.

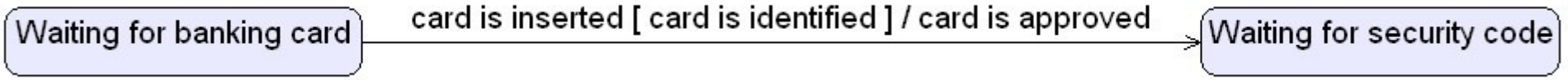
  You can add the ∞ icon to indicate that the sub machine is defined in a separated diagram.

  “Purchase Item” is the name of the outer state. “Verify Credit Card” is the name of the inner state.

States Transition

- A transition between two states represents the change that takes place in the state machine.
- A transition is depicted as a line between two states with an arrowhead pointing to the destination state.
- The transition details are specified using the following syntax:
  trigger [guard] / effect

States Transition



- The transition details are specified using the following syntax:

  trigger [guard] / effect

  trigger

  This is the condition that might cause this transition to occur.

  guard

  This is an additional condition that should be true in order to allow the transition.

  effect

  This is a description of what happens when the transition completes.

Transition Types

- The possible transitions are of the following types:

  Compound Transition

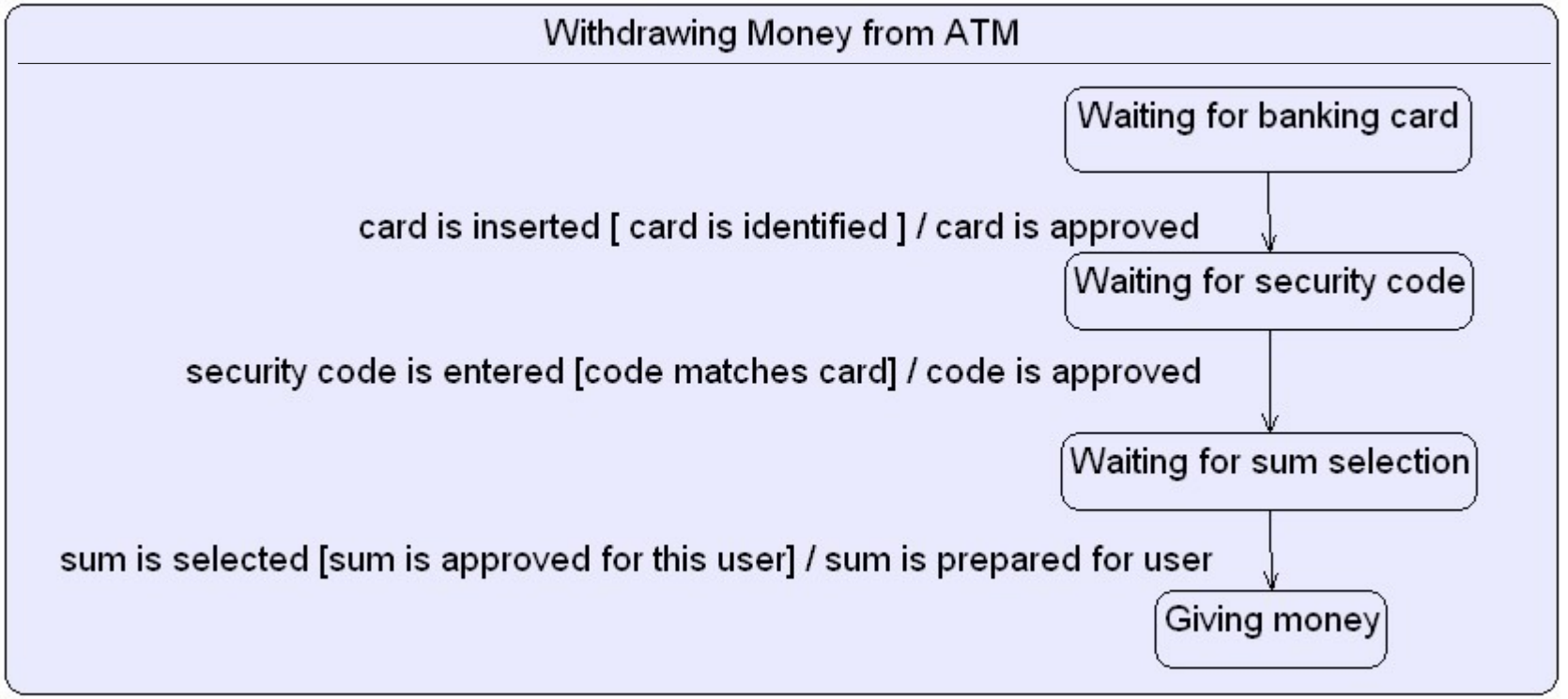
  Completion Transition

  Internal Transition

  High Level Transition

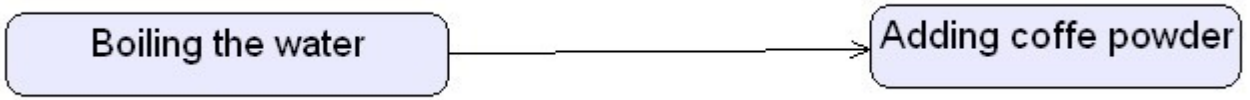
Internal Transition

- An internal transition is a transition within the same composite state without getting out of it.



Completion Transition

- A simple transition from a state that finishes its activities as a consequence of the termination of its activities.



Compound Transition

- A compound transition is a set of transitions leading from one complete state to another.

High Level Transition

- A high level transition is a set of transitions leading from one composite state outside of it (while having all states - both the composite one we exit... as well as any other one inside it - be exited).

Activities

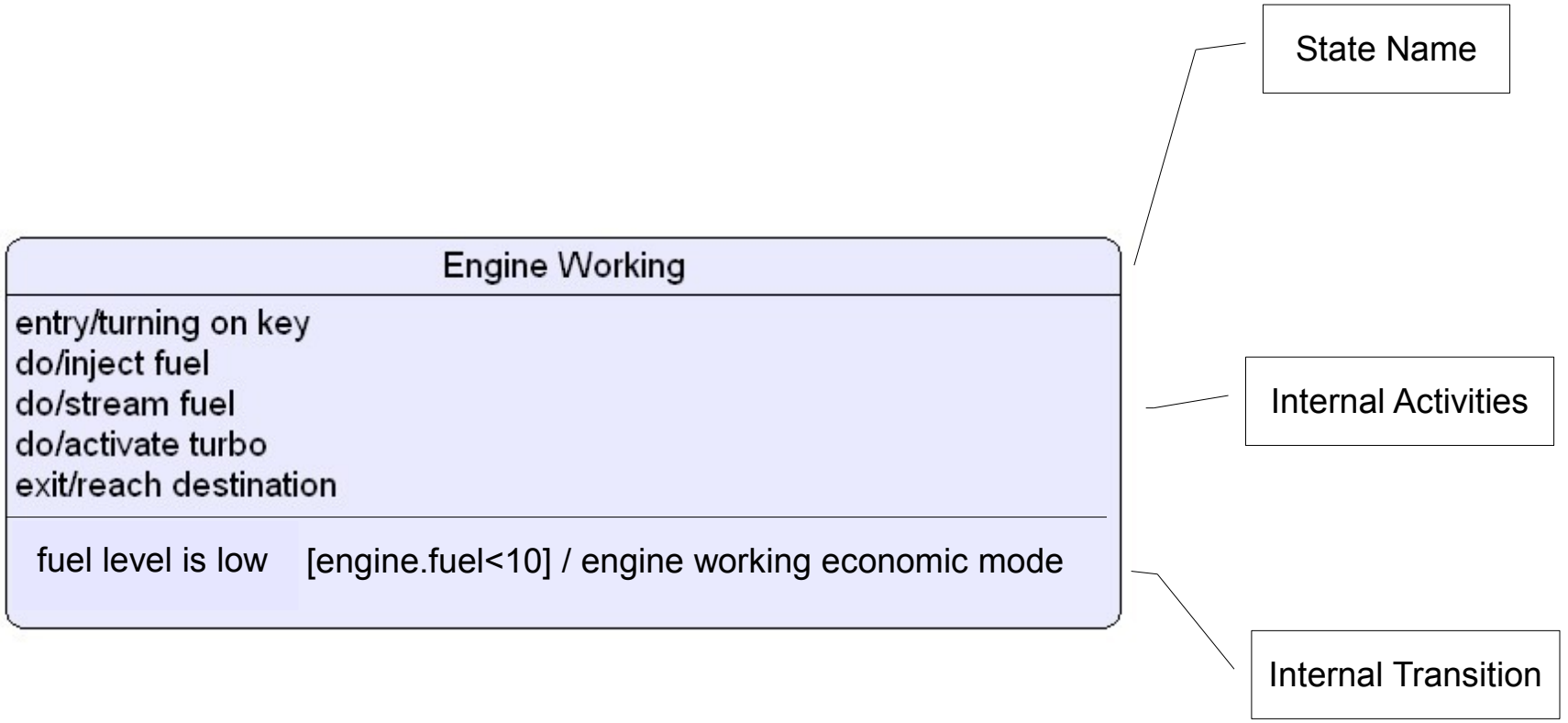
- Each state has activities that are executed as long as the state is active.
- The notation that represents activity is written in the following pattern:

  label / activity expression

  label is one of the following: entry, exit or do.

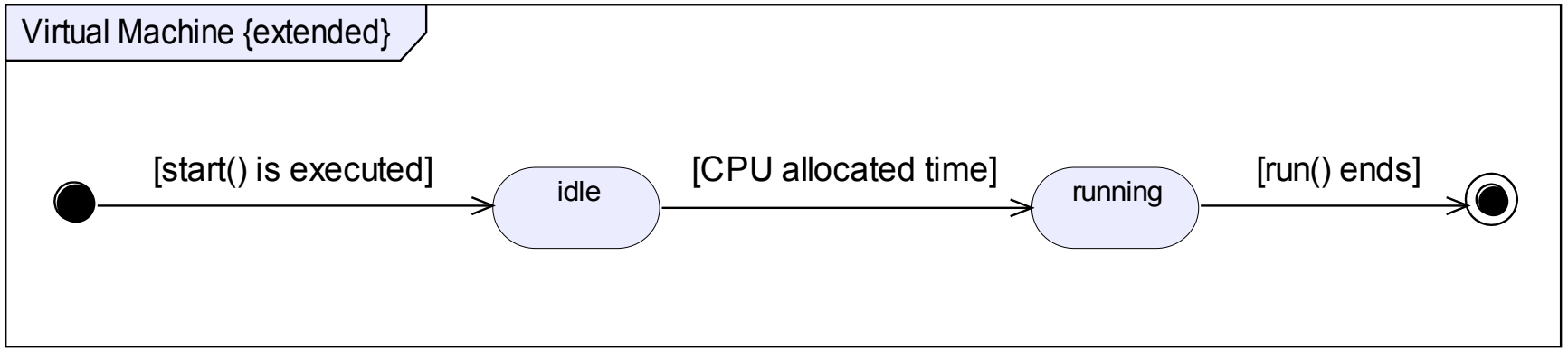
  activity expression describes the activity.. it can be either natural language or code.

Activities



State Machine Diagram Extension

- The state machine diagram can be specialized detailed for specific situation.
- When doing so, we can add {extended} to the state machine diagram title.



Pseudo States

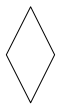
- A pseudo state is a special type of state that represents a specific behavior.
- The available pseudo states types are:

Initial Pseudo State

This is the starting point of the state machine diagram.

Final Pseudo State

This is the ending point of the state machine diagram.



Choice

Using guard conditions the execution chooses between the available states.

Pseudo States

Deep History

Used within a state region. When getting to this pseudo state from outside the region, it takes us to the last sub state it was in within that region.

Exit Point

Represents a possible transition outside of the composite state.

Junction

Allows several possible transitions to get into one pseudo state. Allows the same when leaving the junction to other states.

Pseudo States

- Entry Point

  Possible target for a transition that moves into a composite state.
  Must be labeled.

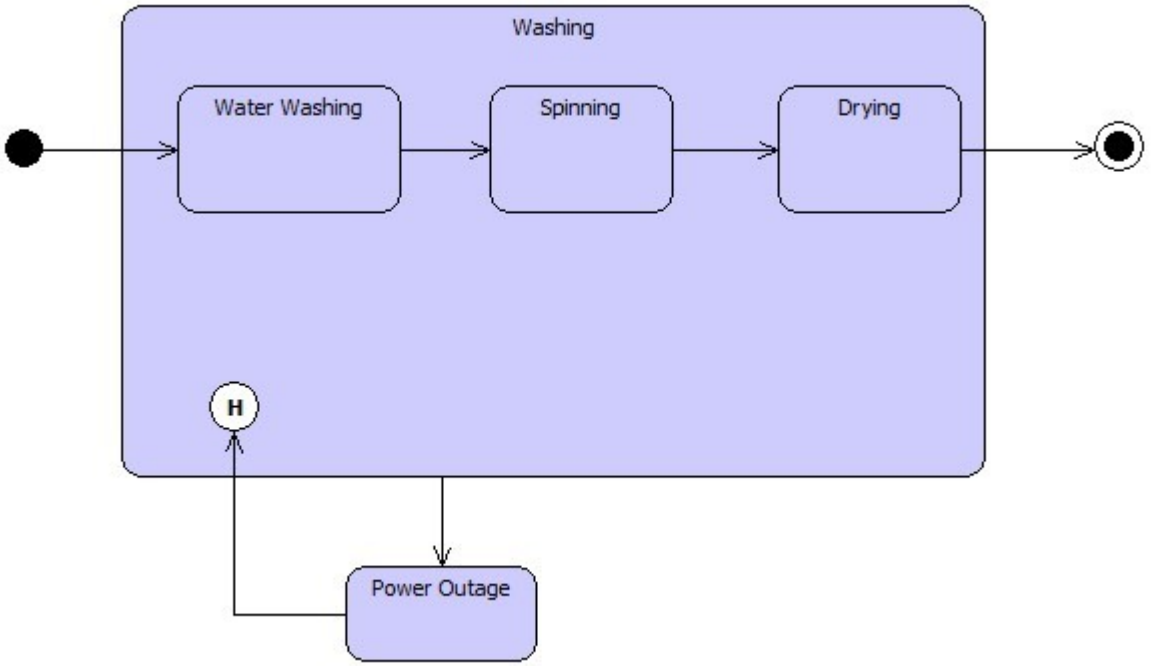
- Shallow History

  The same as deep history with the following difference. The last substate must be at the same level as the history pseudostate.

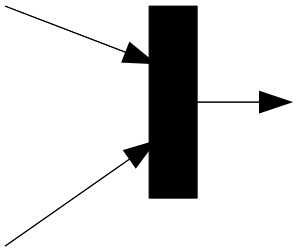
- Terminate Node

  The state machine terminates.

Pseudo States



Pseudo States

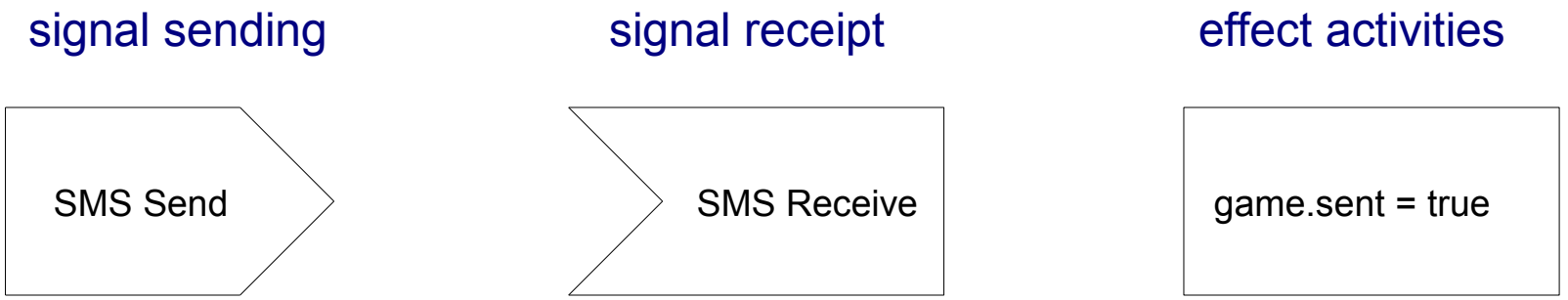


Fork & Join

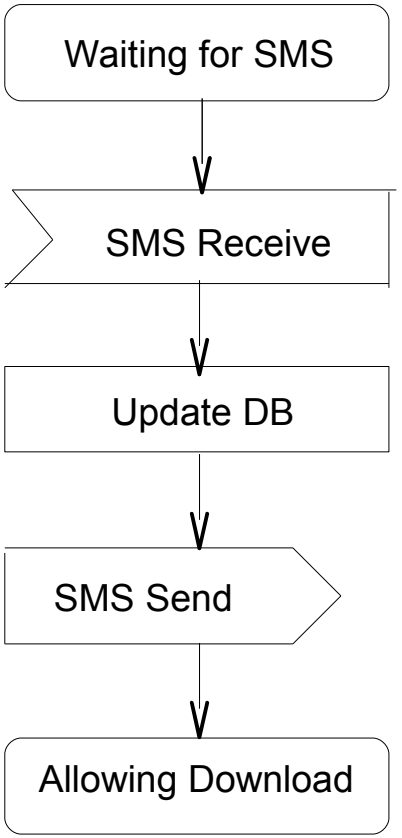
The fork represents a split in the execution into orthogonal regions. The join represents a unite of separated regions into a single transition. This unite won't take place till all separated regions have transitioned to the join pseudo state.

Signal Symbols

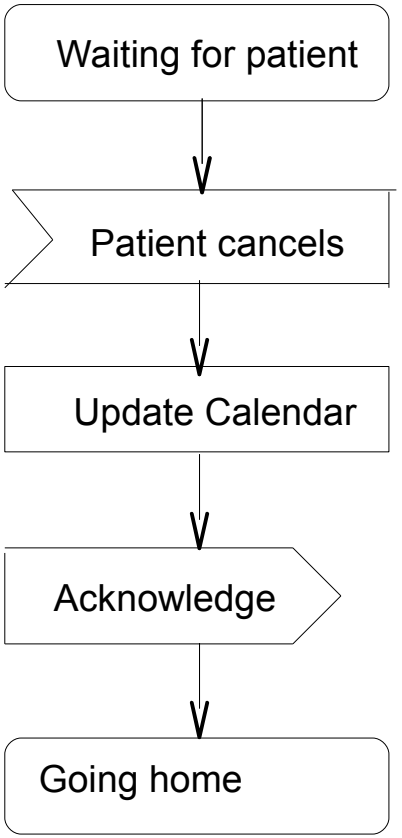
- Signal symbols are explicit icons we can use to show the following transition types:



Signal Symbols



Signal Symbols



Transitions Types

- A transition from one state to another can be of several possible types.

Default Entry

Transition from an external state to the border of a composite state. Once this type of transition takes place the entry activity of the composite state is executed and a default transition to the sub-state takes place.

Explicit Entry

Transition from an external state to specific sub-state of a composite state. The entry activity of the composite state is executed before the system moves to that specific sub-state.

Orthogonal Composite State

- An orthogonal composite state is a composite state with two or more regions, that execute concurrently with each other.
- When exiting a composite state the exit activities execute 'in side out'. The composite state exits after the exit activity of each one of the regions' last active state completes.

Protocol State Machines

Introduction

- The Protocol State Machine diagram represents the behavior of a protocol (such as HTTP, UDP etc..) while ignoring the operations executed by the system.

  The protocol state diagram ignores the implementation.

- In order to differentiate protocol state machine diagram from a standard state machine diagram we add "`{protocol}`" to the diagram title.

Introduction

- The notation for a protocol transition is:

  `[precondition] operation / [postcondition]`

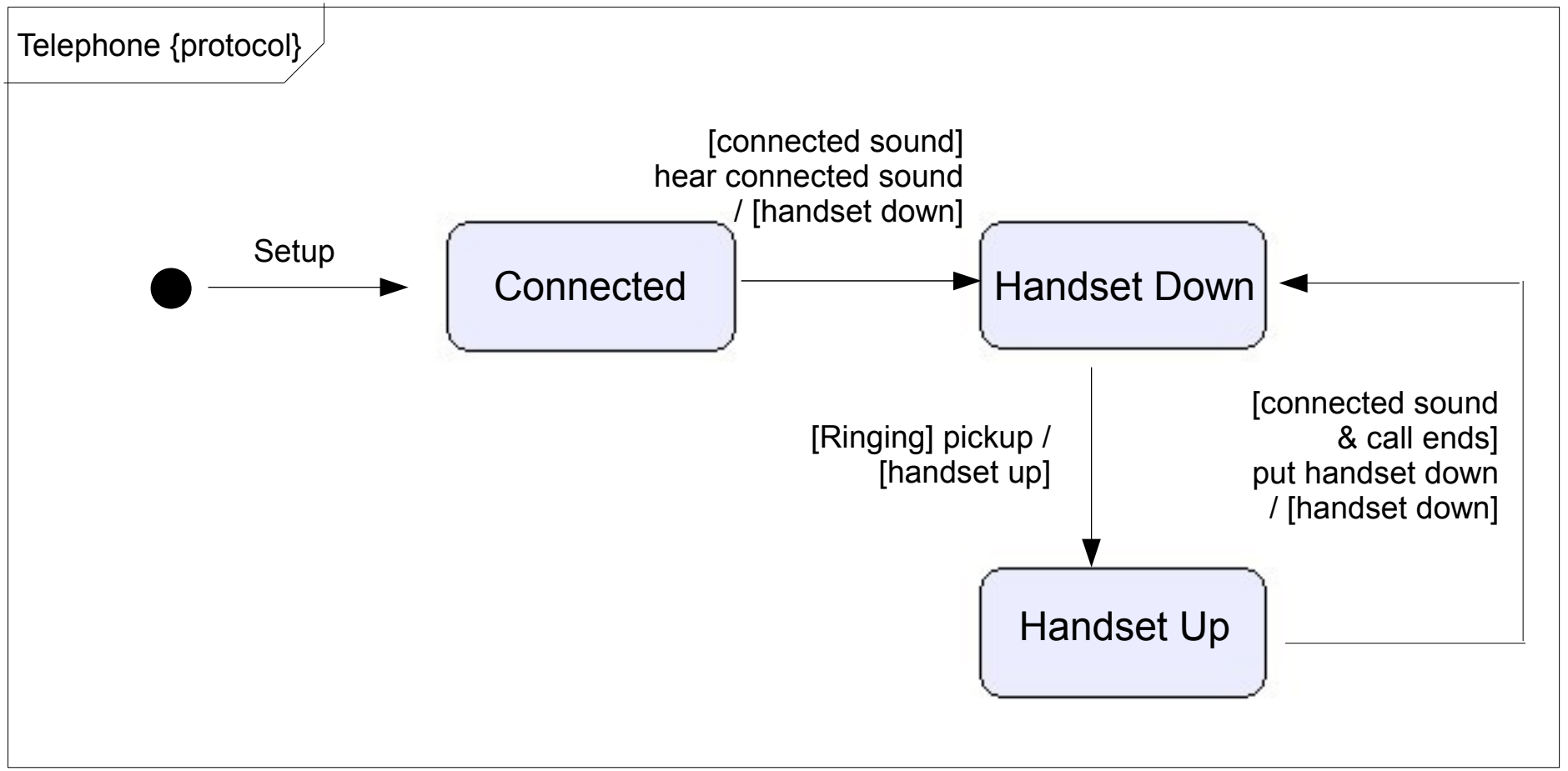
  This notation comes in place of the notation state machine diagrams use.

- Each protocol transition is associated with 0 (or 1) operation.

  The transition guarantees the precondition will be true before the operation is called and that the post condition will be true after wards.

- Unlike the State Machine diagram, when depicting the Protocol State Machine diagram we don't have the entry/exit/do activities.

Sample



UML State Machine Diagrams

Introduction

- The state machine diagrams include two types of diagrams:

  Behavioral State Machines

  Shows the behavior of the elements implementation.

  Protocol State Machines

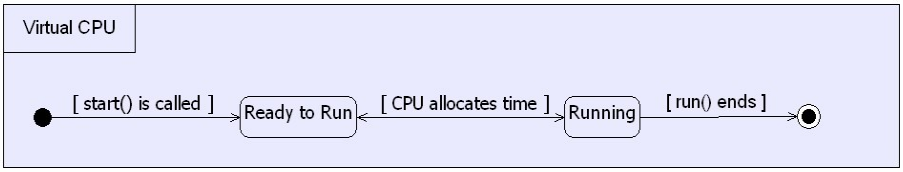
  Shows the behavior of a protocol.

The protocol state machines diagram is a special case of Behavioral state machines.

Behavioral State Machines

Introduction

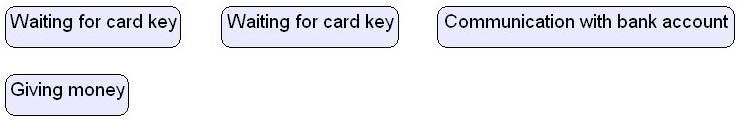
- The behavioral state machines shows the behavior of the elements implementation.
- Simple basic rectangle notation with the name of the machine inside the top compartment surrounds the diagram and tells us which machine the diagram describes.



Moving from one state to another happens in accordance with the states' connection points, which are either entry or exit pseudo states

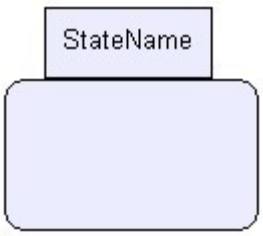
Simple States

- A “simple state” models a specific situation in the machine behavior. The “simple state” is defined using a condition (e.g. “Running”, “Waiting to run”, “Waiting to get the lock flag” etc.).
- The “simple state” notation is a rectangle with rounded corners. The name of the state is written inside.



The state can be either a static situation (e.g. “waiting for...”) or a dynamic one (e.g. “running”).

Alternatively to writing the state name inside the rounded rectangle it is also possible to write it outside of the rounded rectangle within a tab notation attached the rounded rectangle above it. Both options are available. Yet, it is better to stick one of them in order to keep our diagram clear and understandable.



Simple States

- The rounded rectangle can be divided into the following three compartments:

  Name

  The name of the simple state.

  Internal Activities

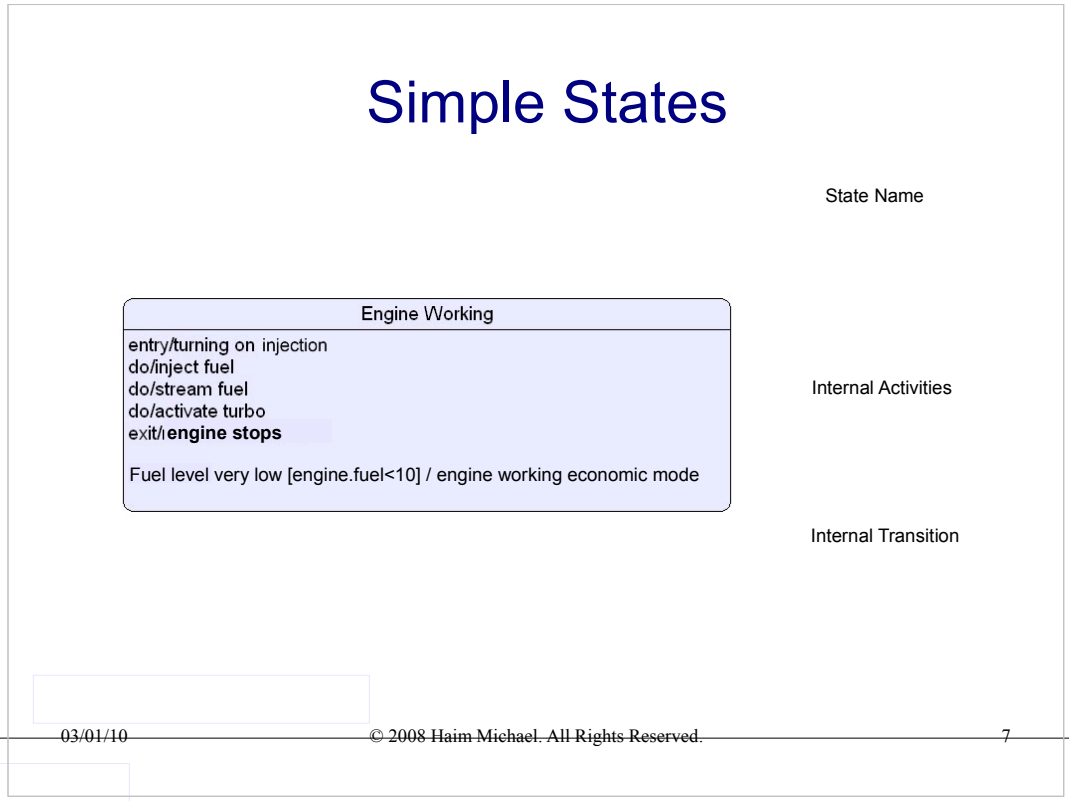
  Includes a list of available internal activities performed as long as the machine is in this simple state. Three types of internal activities: entry, do & exit.

  Internal Transitions

  Includes a list of internal transitions and events that trigger them. This part uses the following pattern: event (attributes list) [guard condition] / transition

The internal transitions can repeat the same event as well as the same transition more than once as long as each entry has a different unique guard condition.

Internal activities includes three types of activities: entry, do & exit. Each internal activity starts with one of the following three words: entry, do or exit.



Three types of internal activities are feasible.

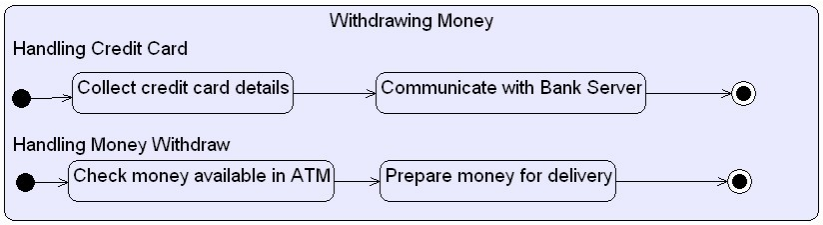
Entry
This is an activity that starts when entering in a new state.

Exit
This is an activity that starts when exiting/leaving a state.

Do
This is an activity that executes as long as the state is active.

Composite States

- A “composite state” is a state that includes sub states within inner region\s.



This composite state includes two inner regions. Each of the inner regions includes a decomposition compartment (detailed view of a composite state).

A dashed line separates between the regions. Each region can have its own title.

Having a composite that includes more than one region means that each region executes concurrently with the others. No need for all regions to finish at the same time. Once all regions have completed the composite state completes as well.

Submachine States

- A “submachine state” is a state contained by another state.
- A "submachine state" notation is a rounded rectangle that includes the name of the outer state, followed by a colon (:) and the name of the submachine state.

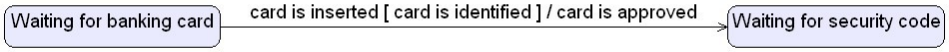
  You can add the ∞ icon to indicate that the sub machine is defined in a separated diagram.

  “Purchase Item” is the name of the outer state. “Verify Credit Card” is the name of the inner state.

States Transition

- A transition between two states represents the change that takes place in the state machine.
- A transition is depicted as a line between two states with an arrowhead pointing to the destination state.
- The transition details are specified using the following syntax: trigger [guard] / effect

States Transition



- The transition details are specified using the following syntax:

  trigger [guard] / effect

  trigger

  This is the condition that might cause this transition to occur.

  guard

  This is an additional condition that should be true in order to allow the transition.

  effect

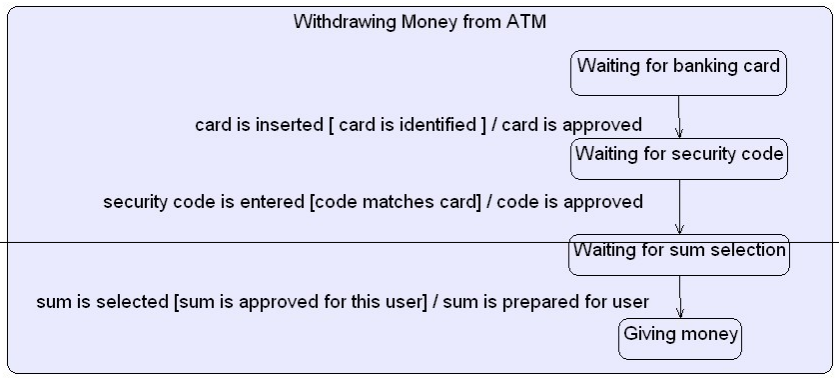
  This is a description of what happens when the transition completes.

Transition Types

- The possible transitions are of the following types:
  - Compound Transition
  - Completion Transition
  - Internal Transition
  - High Level Transition

Internal Transition

- An internal transition is a transition within the same composite state without getting out of it.



A transition from one inner state to another inner state in another composite state is not allowed.

Completion Transition

- A simple transition from a state that finishes its activities as a consequence of the termination of its activities.

Compound Transition

- A compound transition is a set of transitions leading from one complete state to another.

High Level Transition

- A high level transition is a set of transitions leading from one composite state outside of it (while having all states - both the composite one we exit... as well as any other one inside it - be exited).

Activities

- Each state has activities that are executed as long as the state is active.
- The notation that represents activity is written in the following pattern:

  label / activity expression

  label is one of the following: entry, exit or do.

  activity expression describes the activity.. it can be either natural language or code.

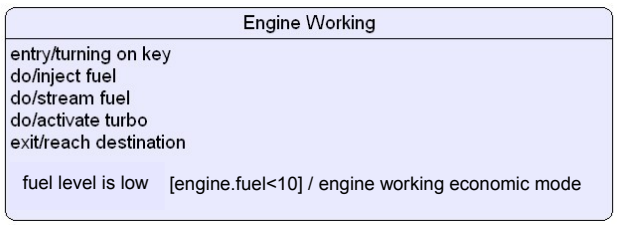
UML defines three possible labels:

Entry
This is an activity that starts when entering into a new state.

Exit
This is an activity that starts when exiting/leaving a state.

Do
This is an activity that executes as long as the state is active. The 'Do' activity executes after the 'Entry' activity and lasts until it completes or as long as the state machine hasn't left current state.

Activities



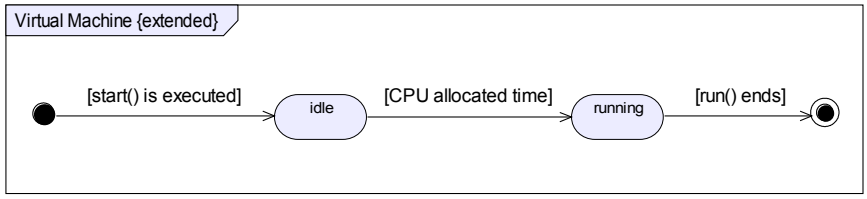
State Name

Internal Activities

Internal Transition

State Machine Diagram Extension

- The state machine diagram can be specialized detailed for specific situation.
- When doing so, we can add {extended} to the state machine diagram title.



The new diagram can include just the parts that are being extended.

Pseudo States

- A pseudo state is a special type of state that represents a specific behavior.
- The available pseudo states types are:

Initial Pseudo State
This is the starting point of the state machine diagram.

Final Pseudo State
This is the ending point of the state machine diagram.

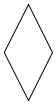
Choice
Using guard conditions the execution chooses between the available states.

03/01/10 20

Moving from one state to another happens in accordance with the states' connection points, which are either entry or exit pseudo states

Initial Pseudo State
The starting point of the state machine diagram.

Final Pseudo State
The ending point of the state machine diagram.



Choice
This special pseudo state allows the execution of a state machine to choose between several states.

Pseudo States

Deep History

Used within a state region. When getting to this pseudo state from outside the region, it takes us to the last sub state it was in within that region.

Exit Point

Represents a possible transition outside of the composite state.

Junction

Allows several possible transitions to get into one pseudo state. Allows the same when leaving the junction to other states.

Deep History
This special pseudo state means be forward to the last sub state it was in within this region.

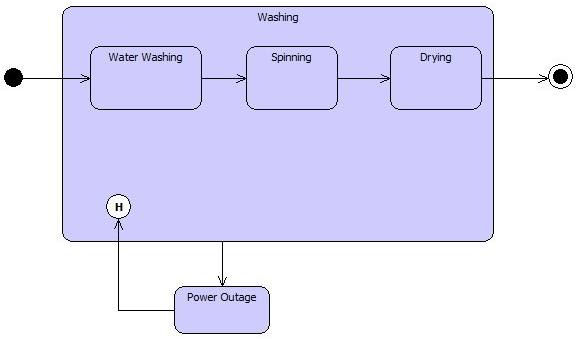
Pseudo States

- Entry Point
  Possible target for a transition that moves into a composite state.
  Must be labeled.

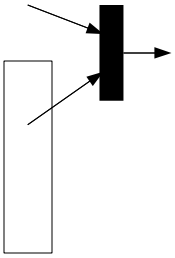
- (H) Shallow History
  The same as deep history with the following difference. The last substate must be at the same level as the history pseudostate.

- X Terminate Node
  The state machine terminates.

Pseudo States



Pseudo States



Fork & Join

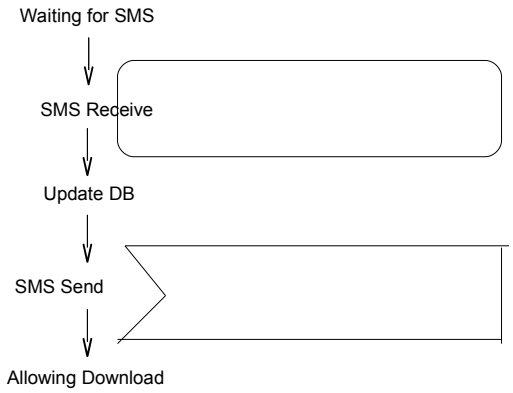
The fork represents a split in the execution into orthogonal regions. The join represents a unite of separated regions into a single transition. This unite won't take place till all separated regions have transitioned to the join pseudo state.

Signal Symbols

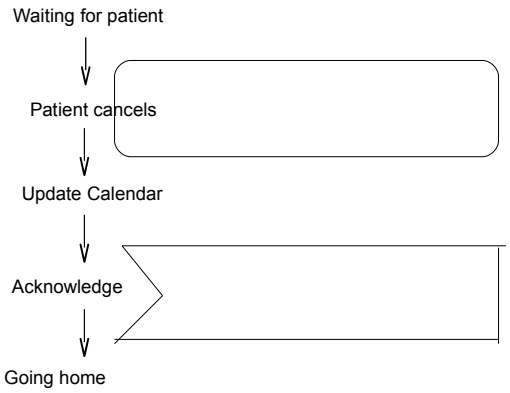
- Signal symbols are explicit icons we can use to show the following transition types:

signal sending	signal receipt	effect activities
SMS Send	SMS Receive	game.sent = true

Signal Symbols



Signal Symbols



Transitions Types

- A transition from one state to another can be of several possible types.

  Default Entry

  Transition from an external state to the border of a composite state. Once this type of transition takes place the entry activity of the composite state is executed and a default transition to the sub-state takes place.

  Explicit Entry

  Transition from an external state to specific sub-state of a composite state. The entry activity of the composite state is executed before the system moves to that specific sub-state.

Orthogonal Composite State

- An orthogonal composite state is a composite state with two or more regions, that execute concurrently with each other.
- When exiting a composite state the exit activities execute 'in side out'. The composite state exits after the exit activity of each one of the regions' last active state completes.

Protocol State Machines

Introduction

- The Protocol State Machine diagram represents the behavior of a protocol (such as HTTP, UDP etc..) while ignoring the operations executed by the system.

  The protocol state diagram ignores the implementation.
- In order to differentiate protocol state machine diagram from a standard state machine diagram we add "`{protocol}`" to the diagram title.

Moving from one state to another happens in accordance with the states' connection points.

Introduction

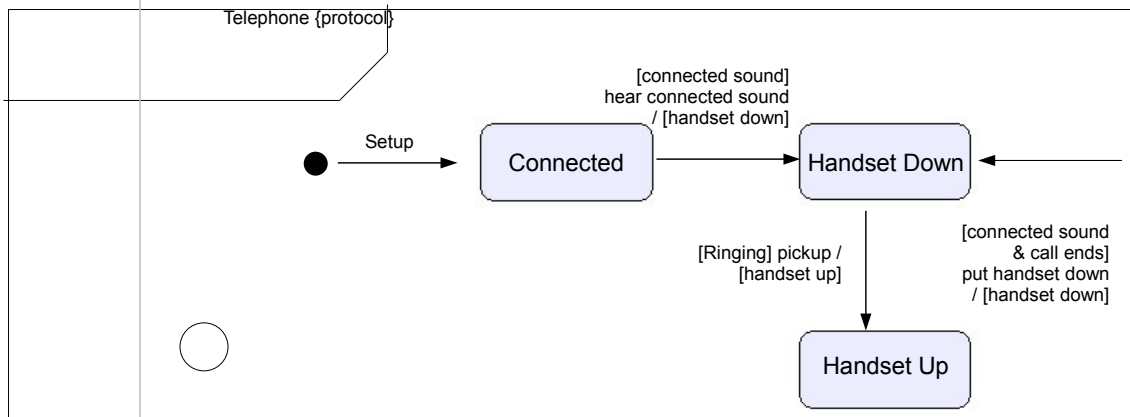
- The notation for a protocol transition is:

  `[precondition] operation / [postcondition]`

  This notation comes in place of the notation state machine diagrams use.
- Each protocol transition is associated with 0 (or 1) operation.

  The transition guarantees the precondition will be true before the operation is called and that the post condition will be true after wards.
- Unlike the State Machine diagram, when depicting the Protocol State Machine diagram we don't have the entry/exit/do activities.

Sample



The protocol reflected in this diagram is separated from the operations underneath, which separates it from the possible implementations (e.g. digital land line phone, internet phone, mobile GSM phone and others).

