* **Documenting System Interactions**

In previous assignments you developed the functionality and structural model for the “BUNDLE’nBANK” supermarket.

1.1 Identify Interesting System Behaviour

Some concepts of your conceptual analysis model will exhibit reactive behaviour,

i.e., will be easy to describe as transitioning between states when receiving events from their context.

A concept worth documenting the reactive behaviour of, will react differently to (and/or allow) one and the same event depending on which state it is in.

Identify an interesting concept with such a behaviour.

You may also choose the whole system as the subject of your reactive behaviour description.

In any case, your example should allow a broad spectrum of state diagram features to be exploited in a meaningful way.

1.2 Create a State Diagram

Draw a state diagram showing the states and events for the concept (or system) you have chosen above.

Make sure that you include guards and transition actions.

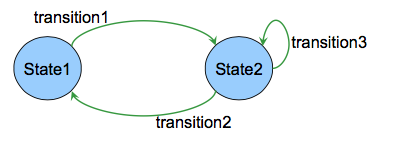
Try to use as many more state diagram features (actions, activities, superstates, concurrent states) as meaningfully possible.

As always, make sure to use correct UML syntax and use meaningful, self-documenting names for the modelling elements you include.

If you are aiming at full marks, use as much of the advanced state chart notation and features that were introduced in lectures as you can.

Key idea: • If a system has finite set of possible states, then you can list them,

and identify each of the possible things that could happen in each state.



modelling a Door:

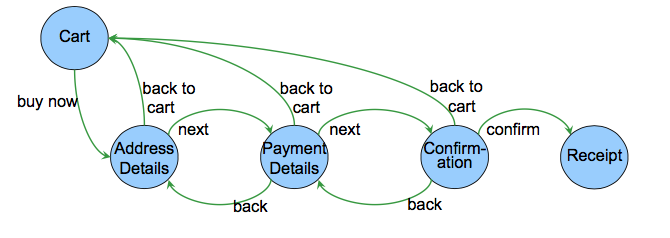
Possible states: open,closed, closed&Locked

Possible transitions: push, pull, turn key (Take system from one state to another or back to same state)

Eg: an ecommerce website:

states: the series of pages in the purchase sequence

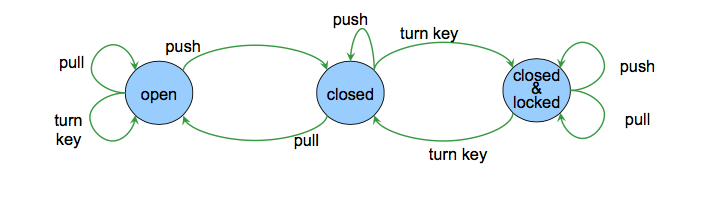
transitions: the buttons: buy now, next, back, confirm, back to cart



--Passive system works

•states the system can be in,

• actions on the system that change its state.



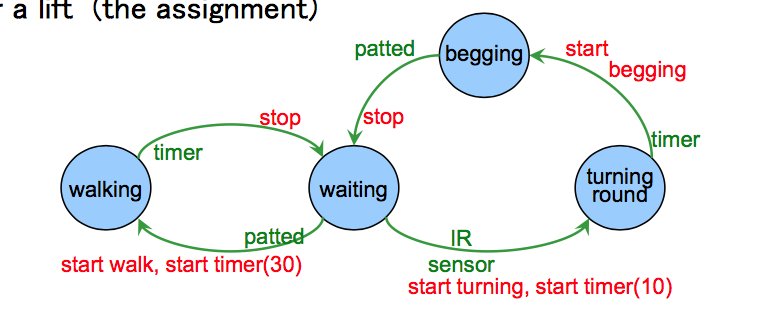
--active system works

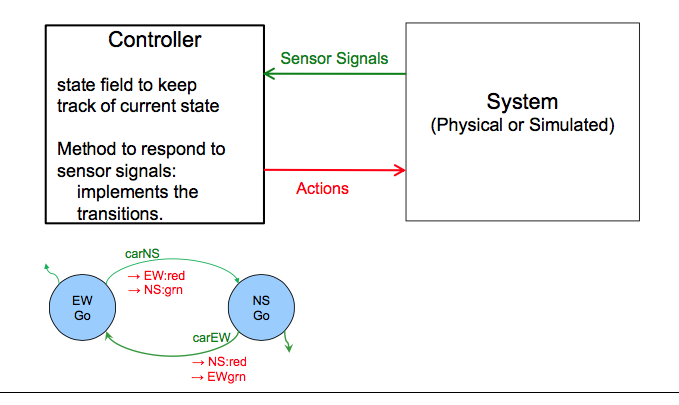
• states the system can be in

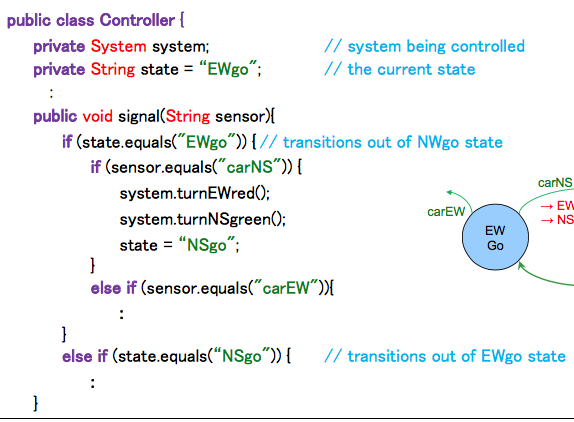
• input/signals/sensors that the system may respond to

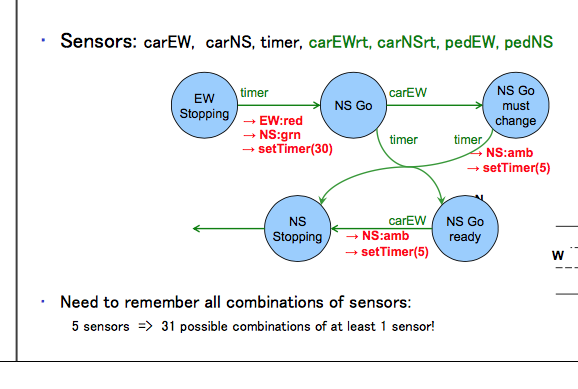
• actions that the system will perform

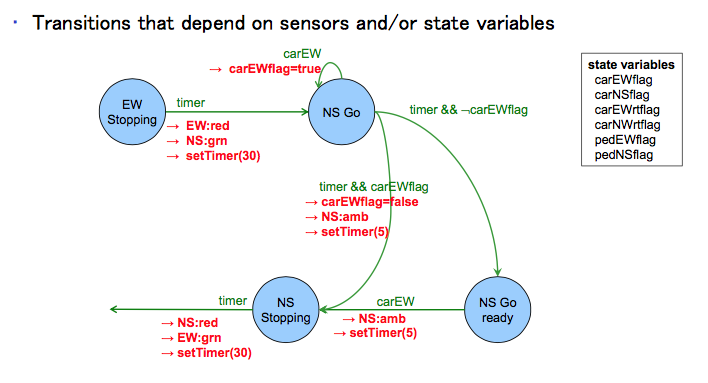
a controller for a lift

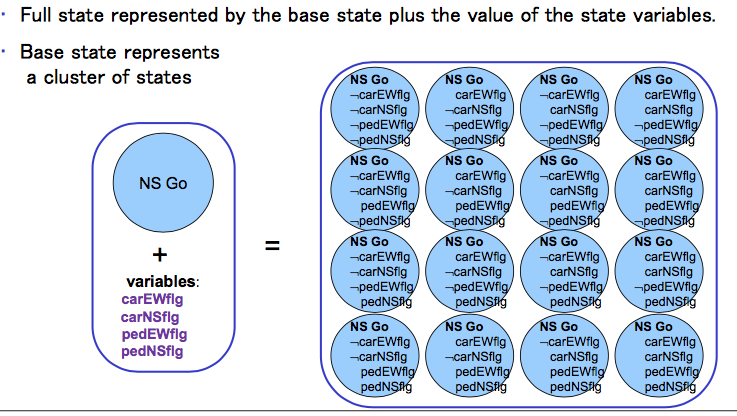


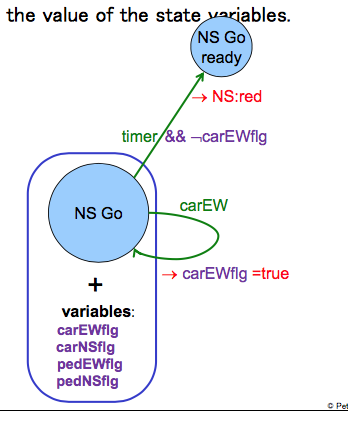












* can be used to specify the states exhibited by an object or the system
* determine the responses (state transitions) to outside stimuli (events)
* are concerned with when operations execute, rather than what operations do, or how they are implemented

Reactive Behaviour

• Events

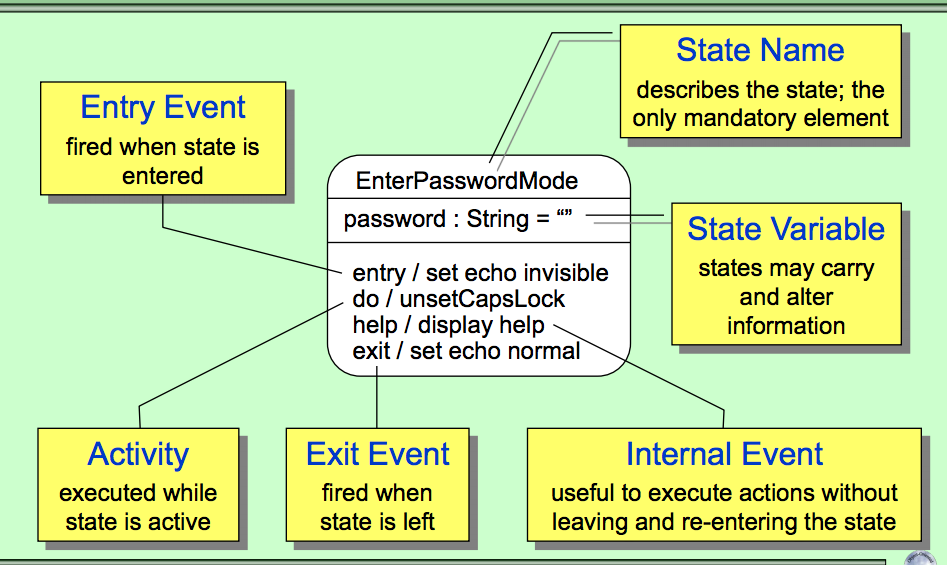
» make object/system transition between states

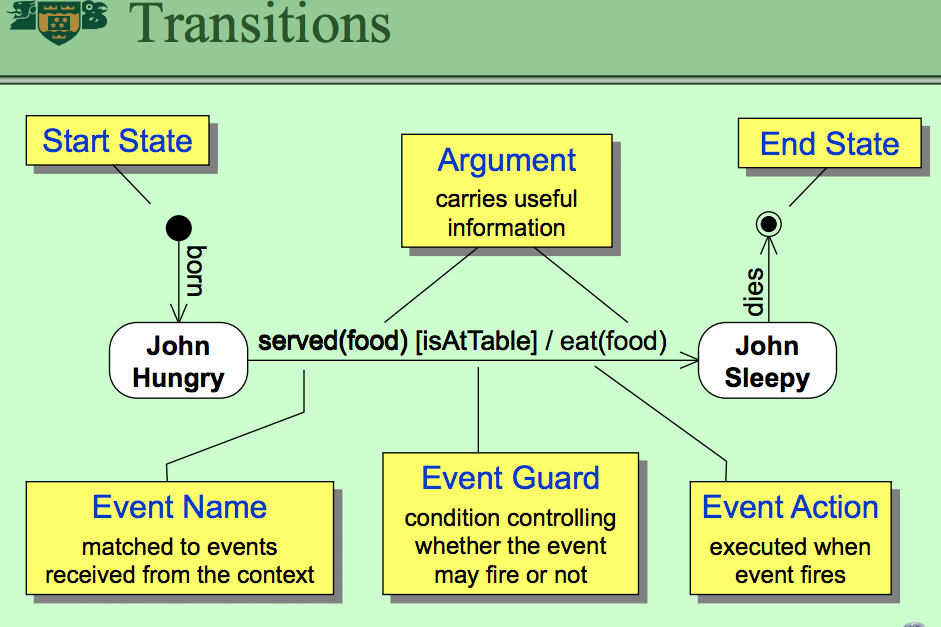
» are enabled/disabled, depending on state

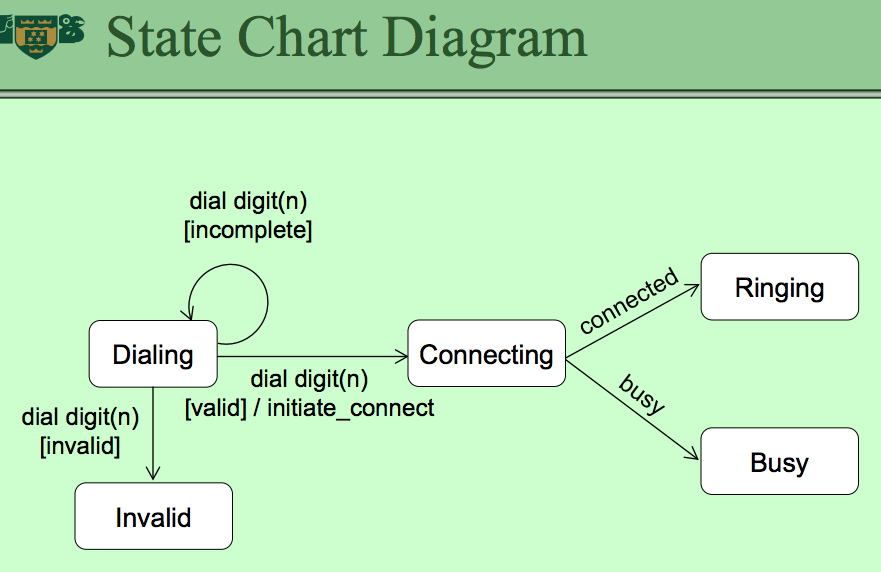
» yield different actions/transition, depending on state

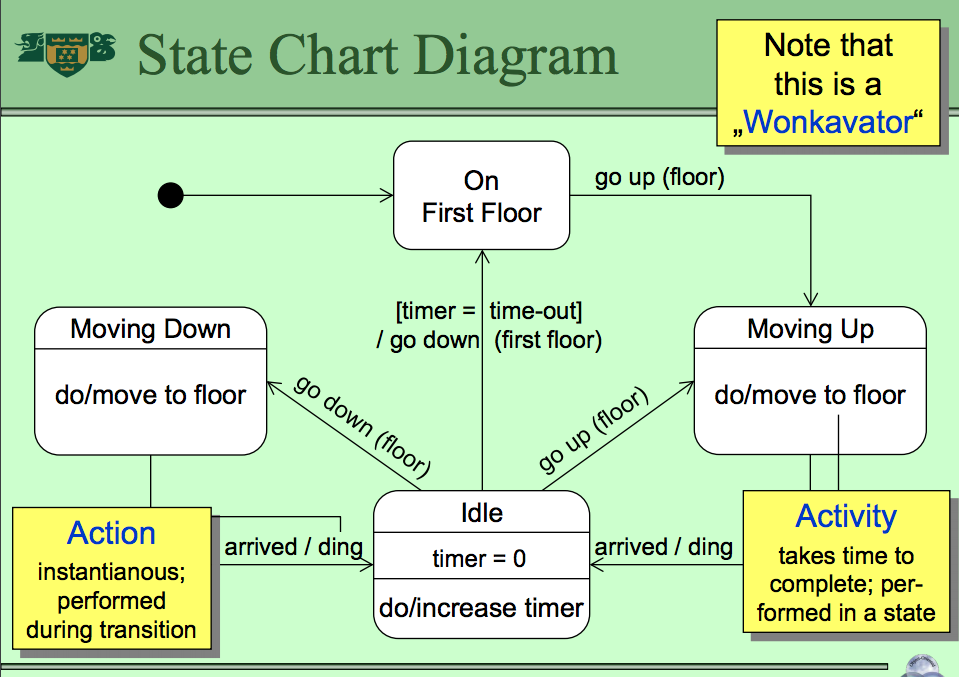
Areas

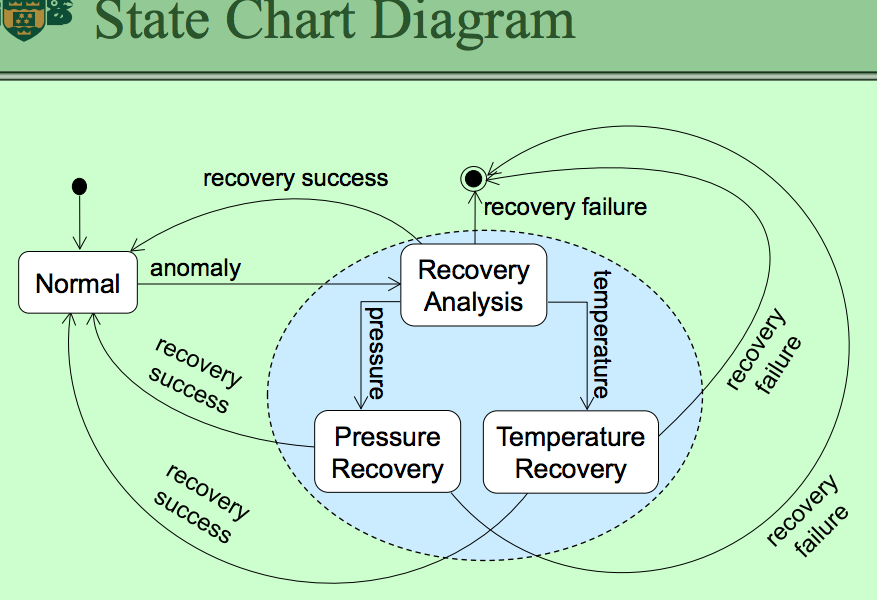
• Broad Range of Applications

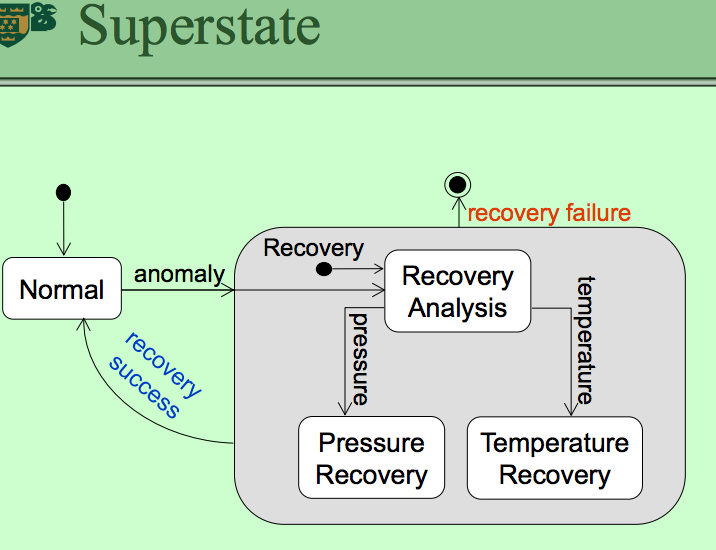
» computer programs, business processes, protocols, web page navigation,

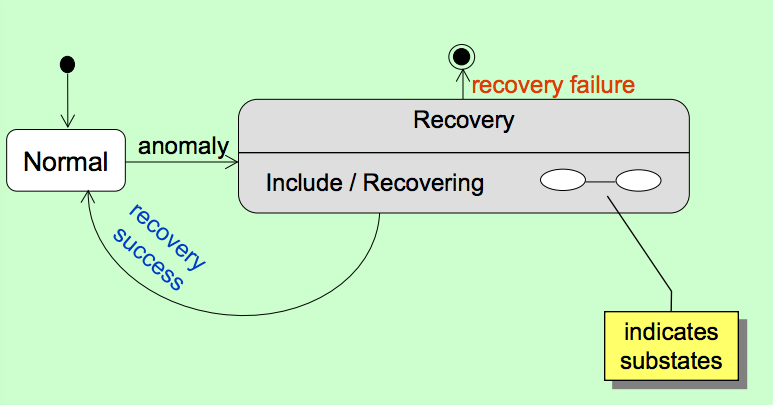












Events

•SignalEvent

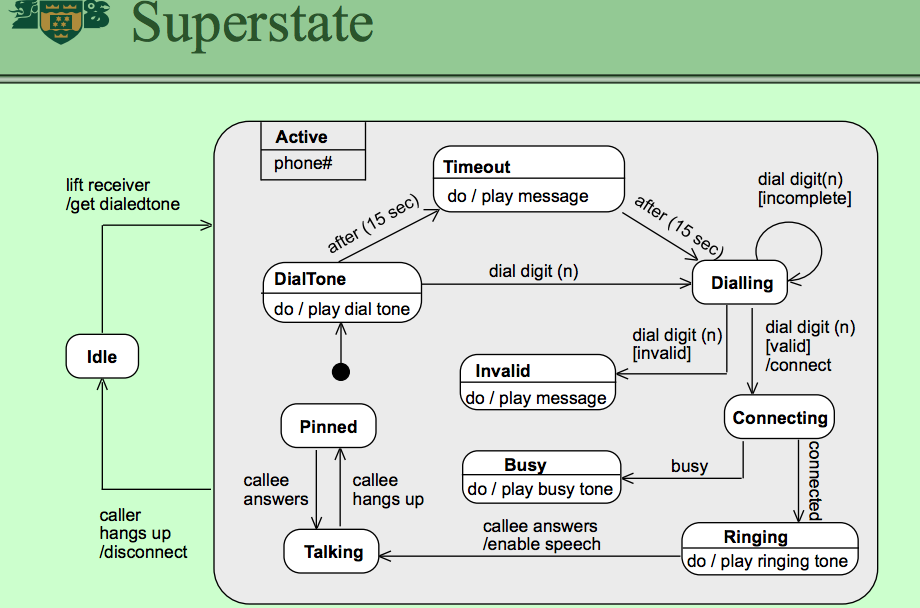
» represents the receipt of an asynchronous event and is queued by the receiver until it's ready to handle it

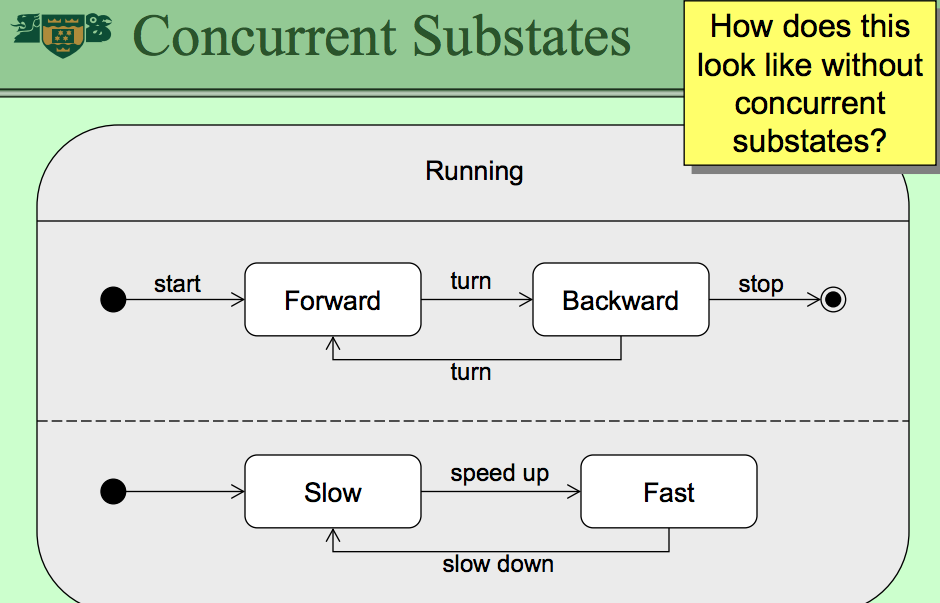
• CallEvent

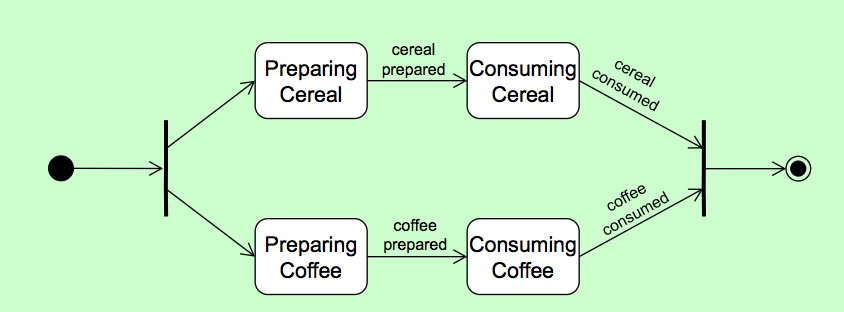
» models the synchronous receipt of a message by an object, invoking a call of an operation

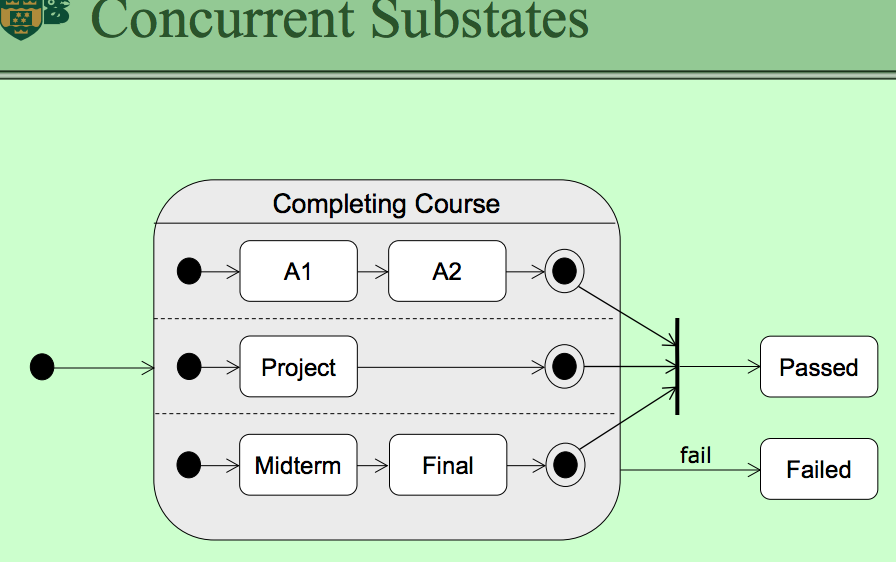
• TimeEvent

» after the specified time, the event occurs. The keyword after is often used in conjunction with time events









**Statecharts**

Multiple States

• Classic State Diagrams are XOR diagrams

» machine can only be in one of several states

• Statecharts are OR diagrams

» machine can be in multiple states

» extreme form are Petri Nets

» the possible combinations constitute new global states themselves

− different interpretation of a single substate

− way to concisely describe large finite state machines