# State Diagram

Applicability

•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

Applications

**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, ...

Foundation

**Finite State Machines**

•States and Events

» limited expressiveness

» can “recognise” regular languages

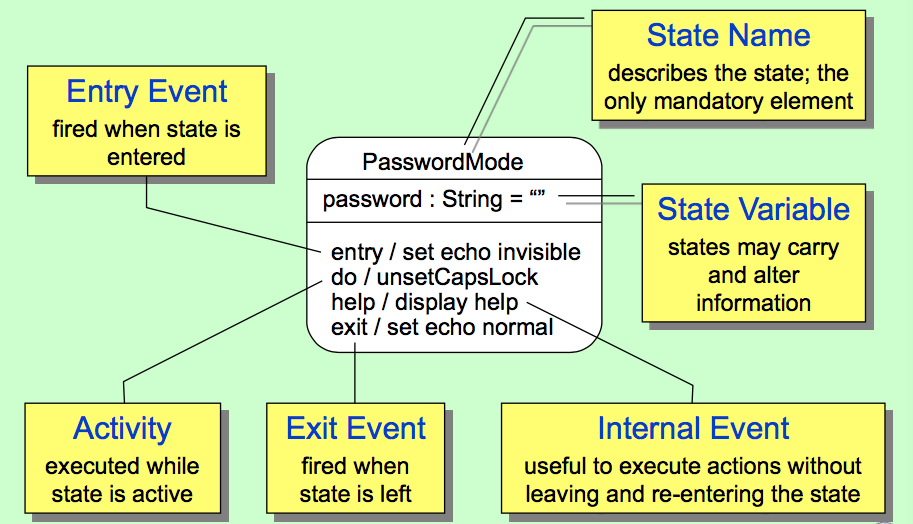
» can also be represented as state transition tables

**Harel Statecharts**

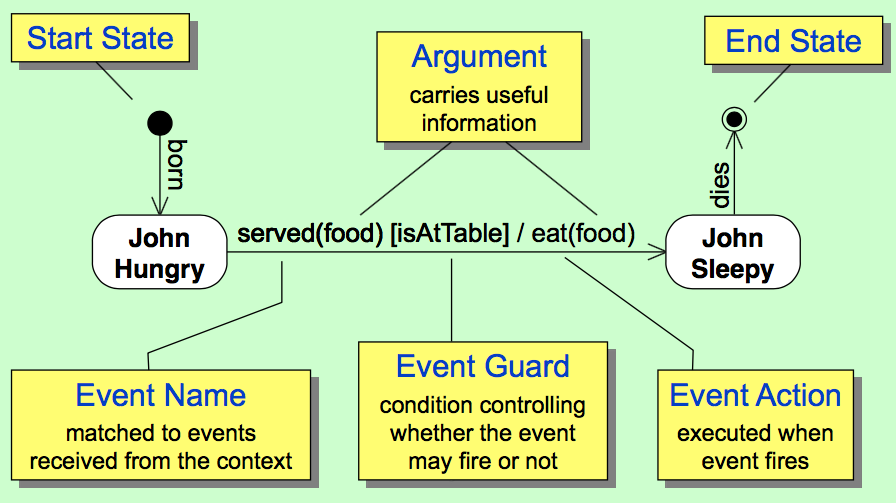
•Multiple States

» extremely useful to structure state diagrams

States



Transitions



Events

(Some) Kinds of Events

•SignalEvent

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

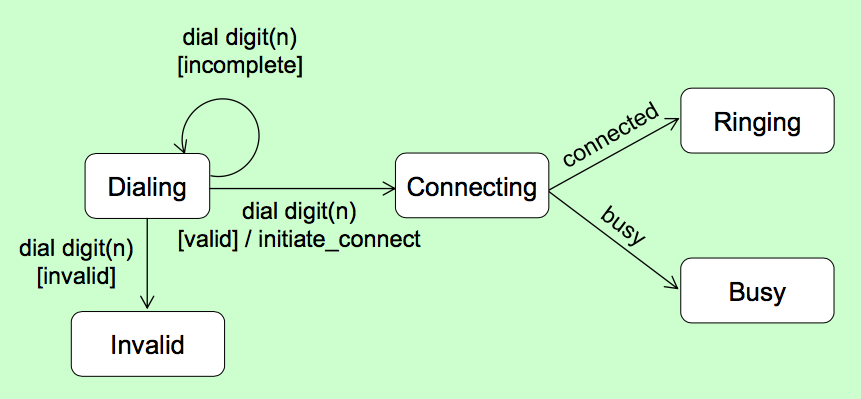
•CallEvent

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

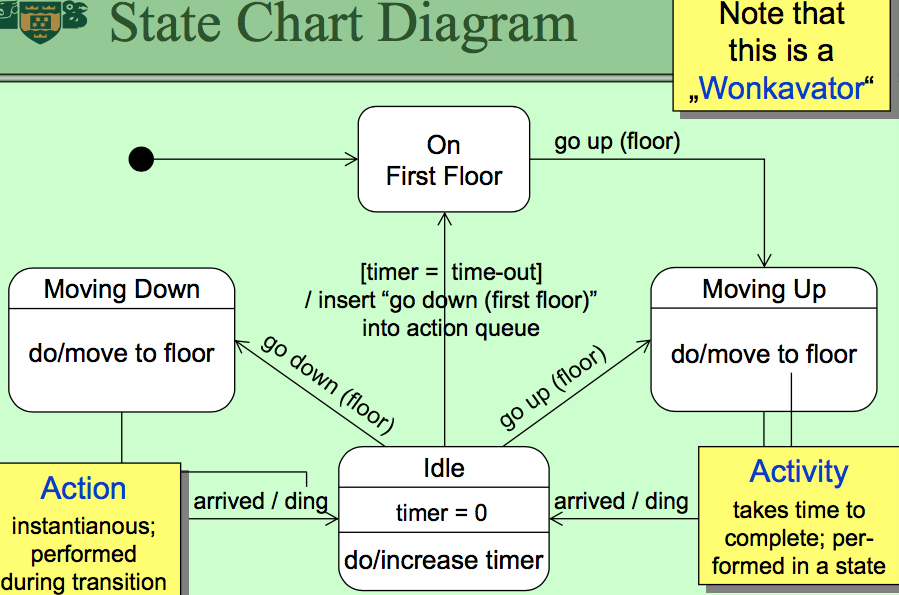
•TimeEvent

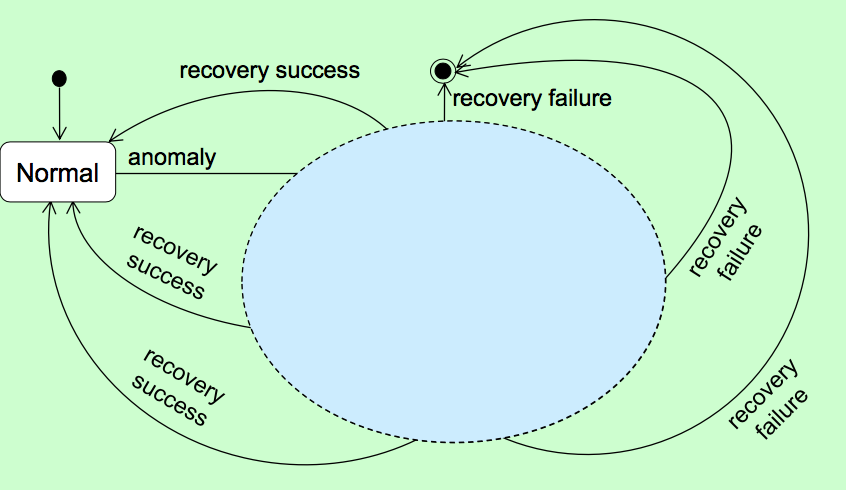
» after the specified time, the event occurs. The keyword after is often used in conjunction with time events

State Char Diagram

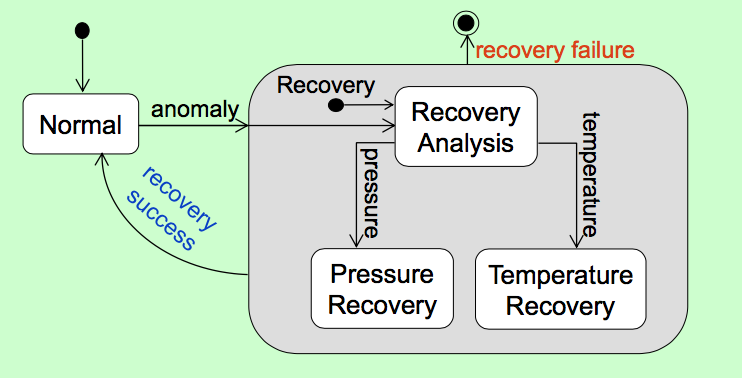


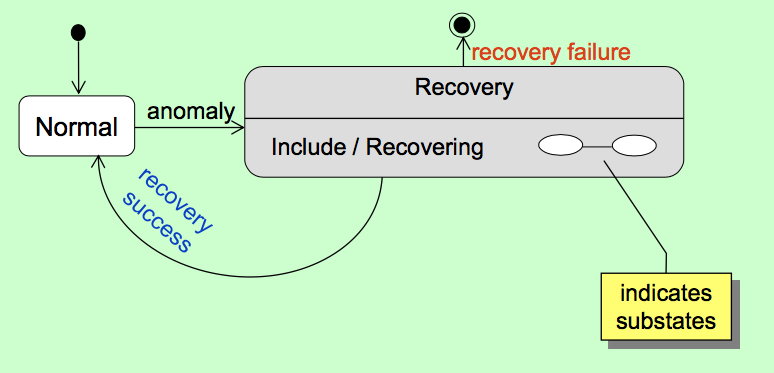
State Char Diagram

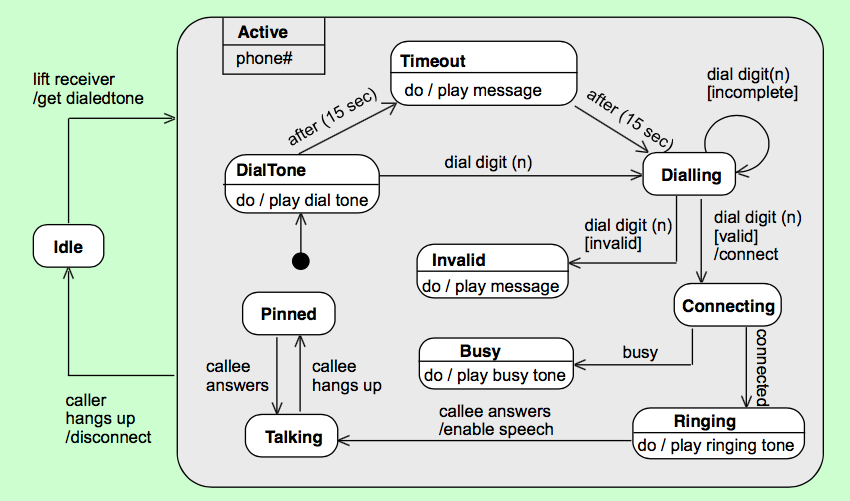




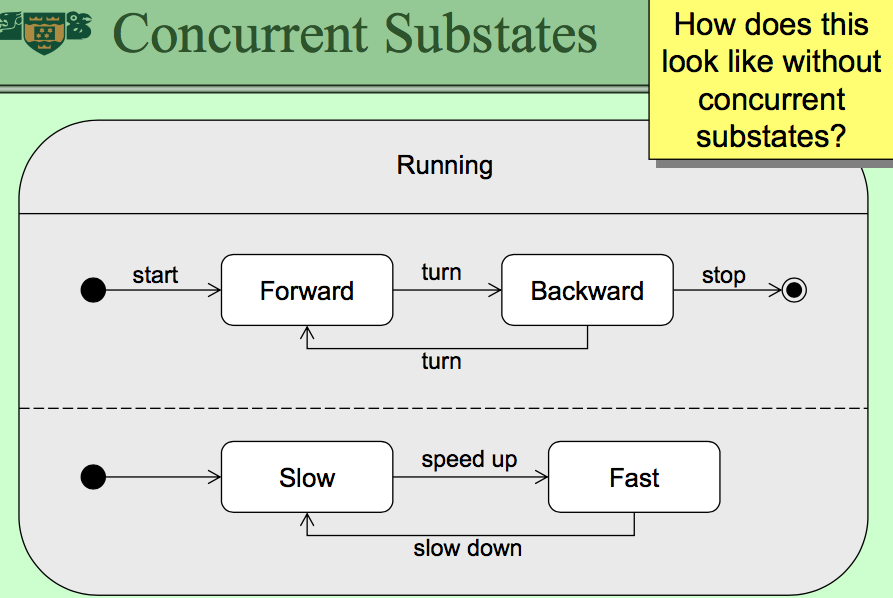
Superstate



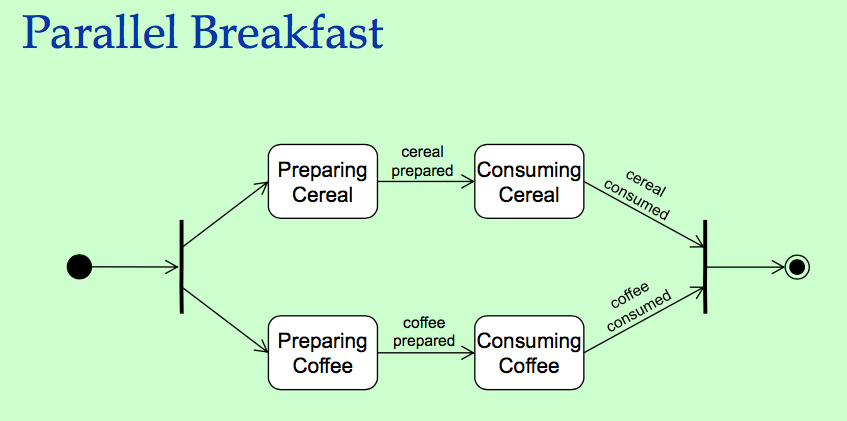


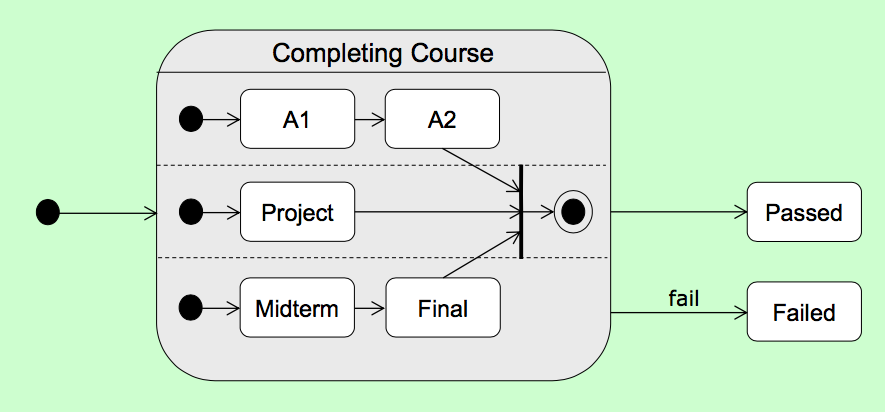


Concurrent Substates



Fork and Join





Statecharts

**Multiple States**

1. Classic State Diagrams are XOR diagrams

» machine can only be in one of several states

1. 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

Complexity Reduction Through...

1. Superstates

» combination of several substates into one superstate

» reduces complexity by hiding

aggregated states and multiplied transitions

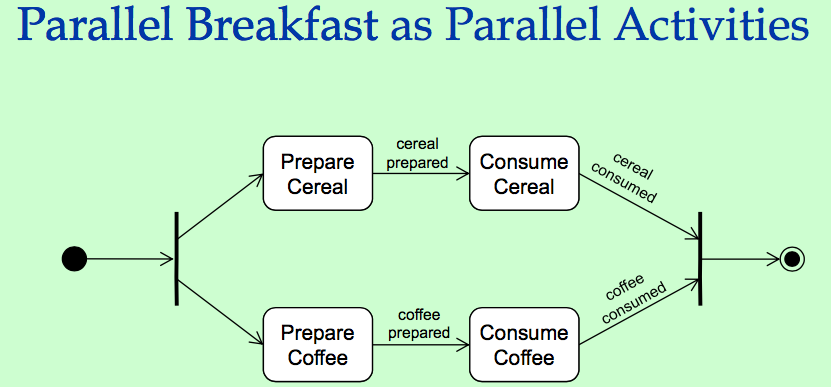
1. Concurrent Machines

» parallel execution

» reduces complexity by factorising

-multiplied states and corresponding transitions

Activity Diagrams Revisited



**Activity Diagrams & State Diagrams**

•A Correspondence

» Activities correspond to States » Control flow lines correspond to (termination) Events

•Same Semantic Foundation

» token flow -+ Petri Nets

•Different interpretation and applications

» context with events vs automatic/implicit “next” events » reactive behaviour vs control flow

Imprecise Natural Language

**Informal "Specifications"**

•“I'm going to teach you how to solve the Rubik's cube in about 30min.”

» does not appear to be a fast method, right?

•“No eating or drinking from cups without lids.”

» so I can eat from a plate?