Executable Object Modelling

- analysis $\Rightarrow$ use cases $\Rightarrow$ class diagrams
- analysis $\Rightarrow$ use cases $\Rightarrow$ (message) sequence diagrams
- $\Rightarrow$ Object-model diagrams
- $\Rightarrow$ Statecharts $\Rightarrow$ sequence diagrams $\Rightarrow$ *test* use cases

Executable Object Modelling with Statecharts

- OO development: intuitive/graphical *and* rigourous
- fully executable models (simulation)
- code synthesis

Executable Object Modelling with Statecharts

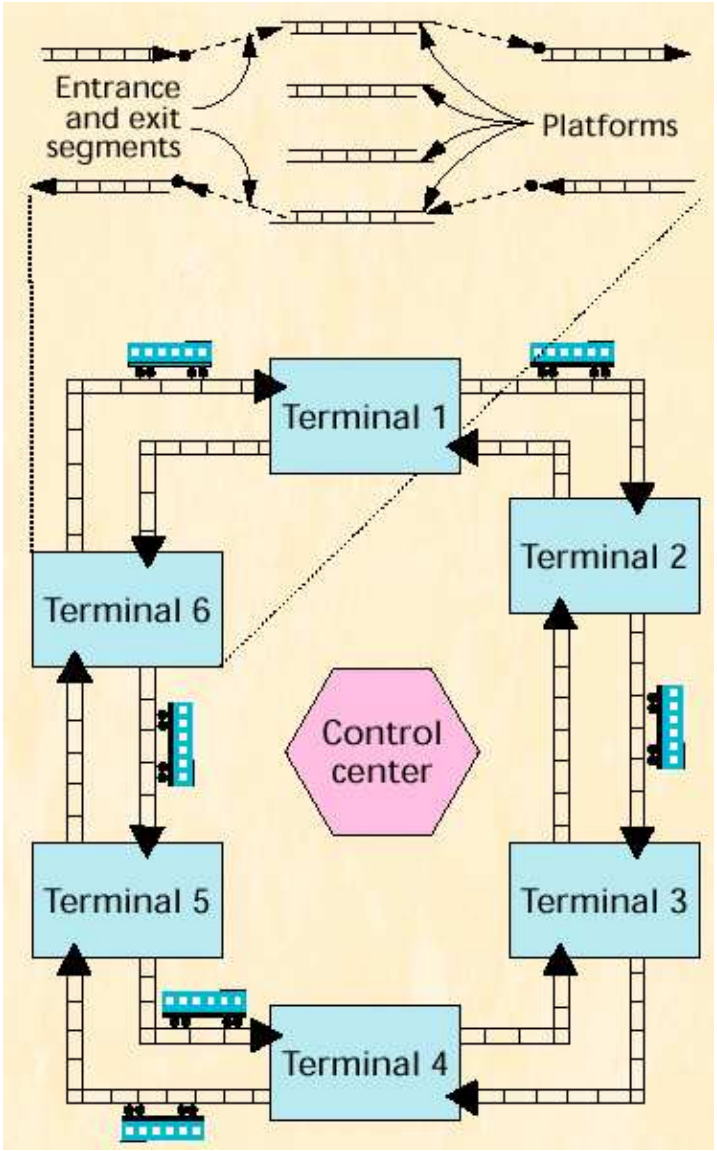
- Structure (classes, multiplicities, relationships)

  $\Rightarrow$ Object-model diagrams (higraph version of ER-diagrams)

- Behaviour

  $\Rightarrow$ StateCharts

Automated Railcar System: Physical View


Entrance
and exit
segments
Platforms
Terminal 1
Terminal 2
Terminal 6
Control
center
Terminal 5
Terminal 3
Terminal 4

Scenarios (Use Cases)

1. Car approaches terminal
2. Car departs from terminal
3. Passenger in terminal

Use Case:
Car approaches terminal

When the car is 100 yards from the terminal, the system allocates it a platform and an entrance segment, which connects it to the incoming track.

If the car is to pass through without stopping, the system also allocates it an exit segment.

If the allocation is not completed within 80 yards from the terminal, the system delays the car untill all is ready.

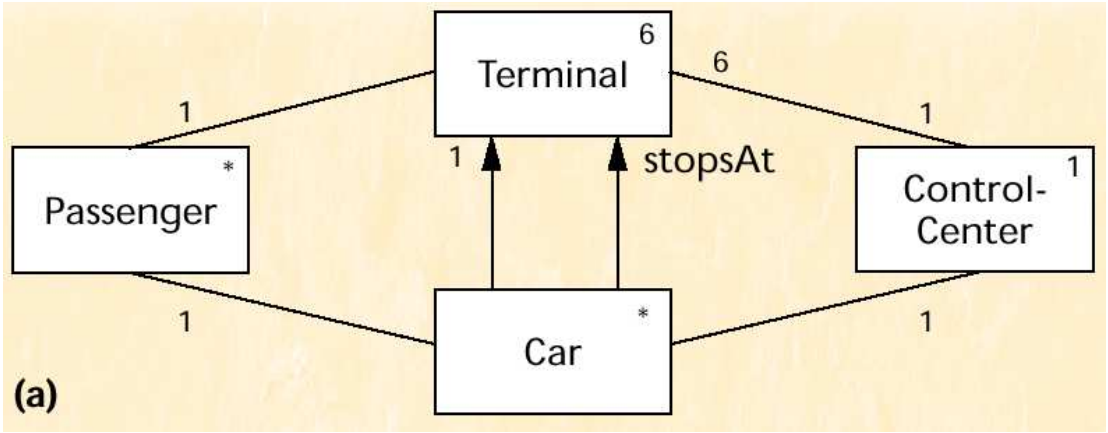
Use Case:
Car departs from terminal

A car departs the terminal after being parked fro 90 seconds. The system connects the platform to the outgoing track via the exit segment, engages the car's engine, and turns off the destination indicators on the terminal destination board. The car can then depart unlesss it is within 100 yards of another car; if so, the system delays departure.

Use Case:
Passenger in terminal

A passenger in a terminal wishes to travel to some destination terminal, and there is no available car in the terminal travelling in the right direction. The passenger pushes the destination button and waits until a car arrives. If the terminal contains an idle car, the system will assign it to that destination. If not, the system will send a car in from some other terminal. The system indicates that a car is available with a flashing sign on the destination board.

Toplevel object-model diagram

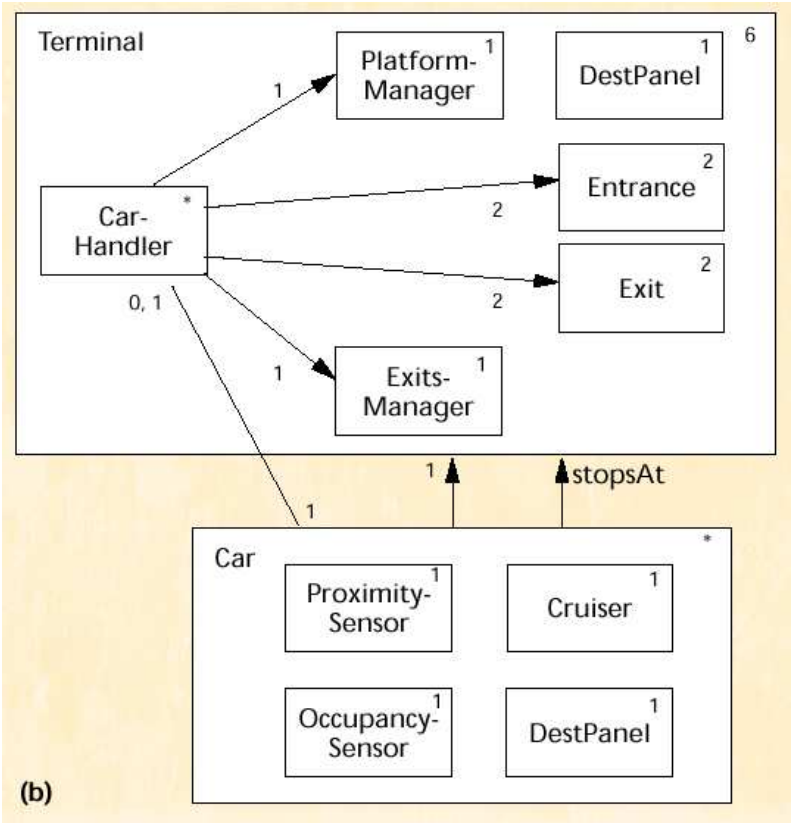


- object classes
- object multiplicities
- structural relationships (including navigatability and arity)

Object Navigation, Creation/Initialization

- navigatability
  - no relation name $\Rightarrow$ `its`
  - `Passenger->itsCar->stopsAt`
  - toplevel: `System->itsTerminal[1:6]`
- Code synthesis: creation/initialization $+$ dynamics over time
- Object multiplicity
- Associations
  1. unambiguous: multiplicities match
  2. ambiguous but bounded: any subset
  3. unworkable: canonical mappings or user defined (scripts)

Zoom out: aggregation



Dynamics of Object Communication and Collaboration

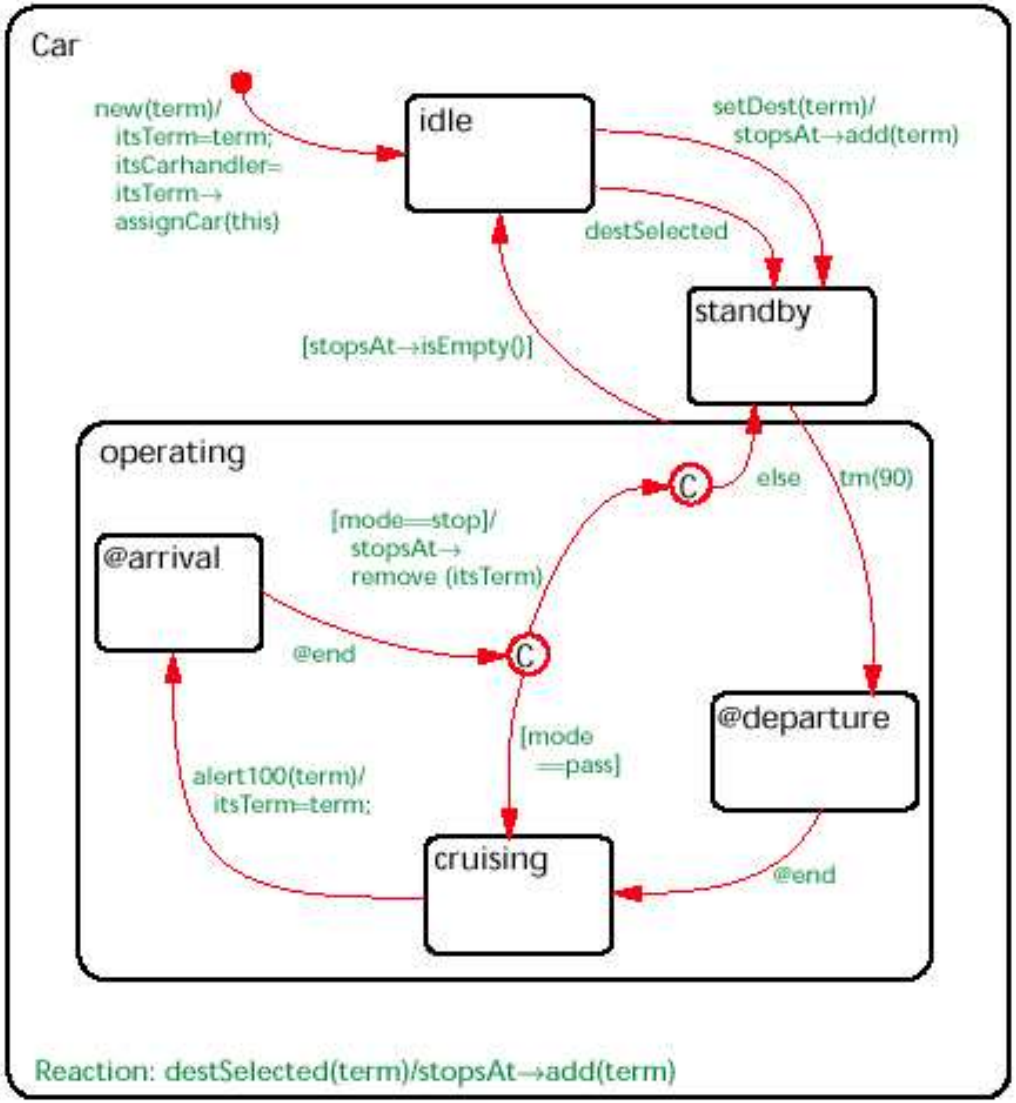
1. Objects generate events which are queued

   `serverObject->gen(event(<params>))`

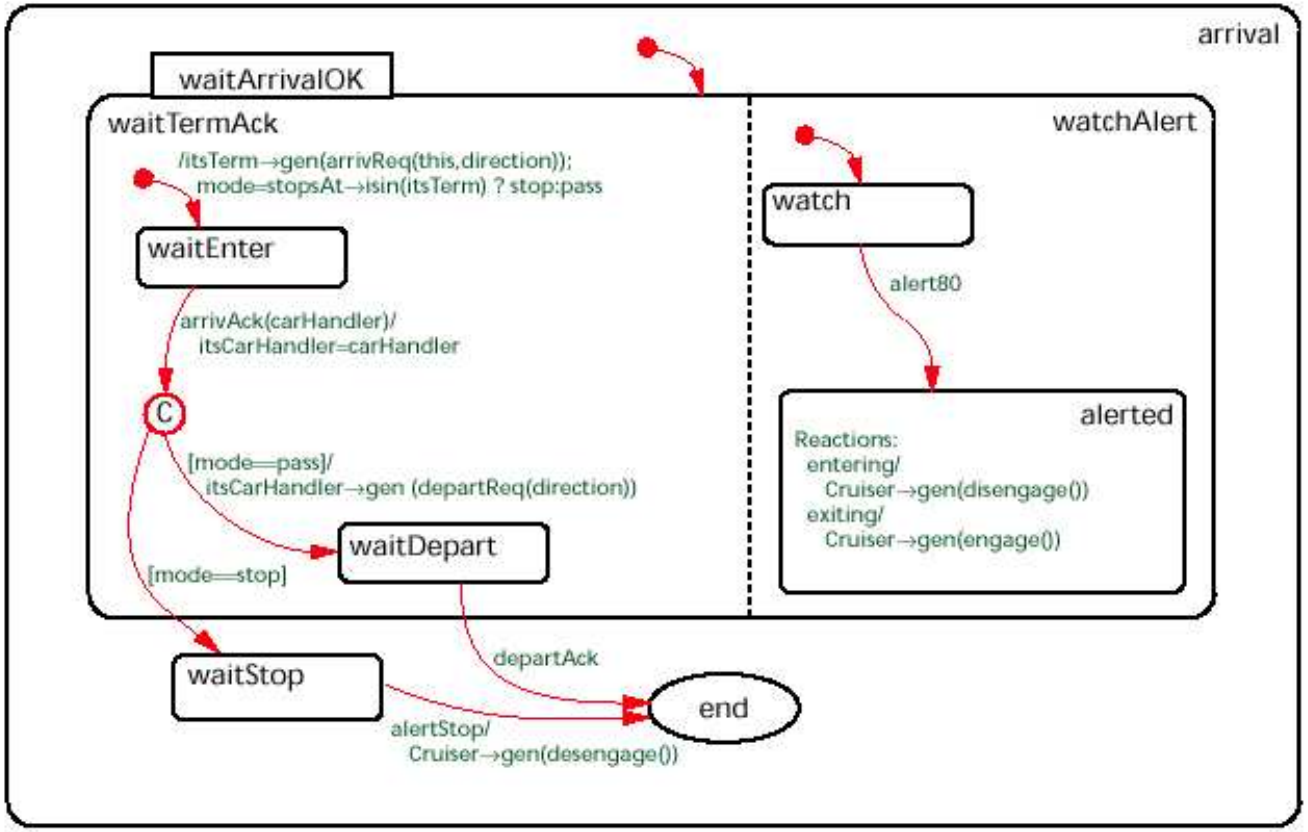
2. Objects can directly invoke an operation/method

   `serverObject->method(<params>)`

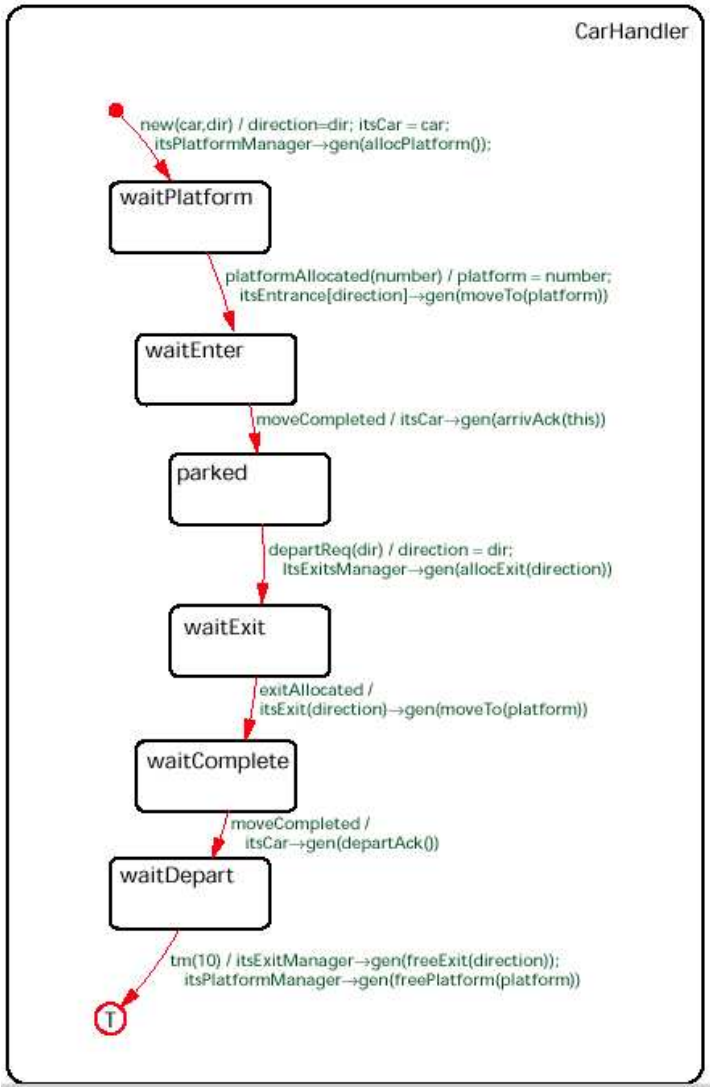
Car dynamics


Car
new(term)/
itsTerm=term;
itsCarhandler=
itsTerm→
assignCar(this)
idle
setDest(term)/
stopsAt→add(term)
destSelected
standby
[stopsAt→isEmpty()]
operating
C
else
tm(90)
[mode==stop]/
stopsAt→
remove (itsTerm)
@arrival
@end
C
@departure
[mode
==pass]
alert100(term)/
itsTerm=term;
cruising
@end
Reaction: destSelected(term)/stopsAt→add(term)

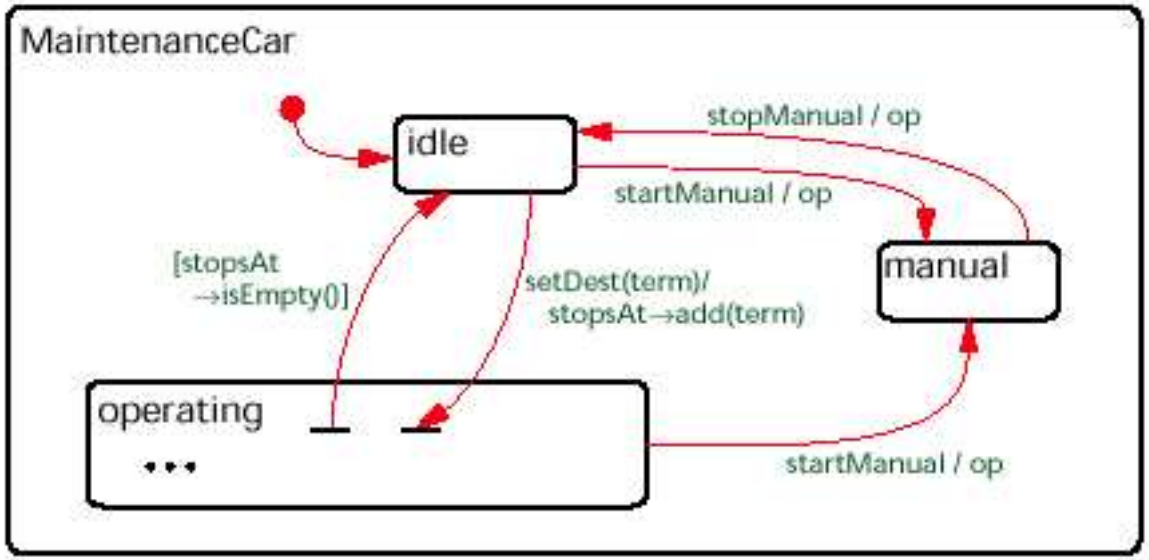
Arrival dynamics



CarHandler lifecycle



Zooming


MaintenanceCar
idle
stopManual / op
startManual / op
manual
[stopsAt
→isEmpty()]
setDest(term)/
stopsAt→add(term)
operating
startManual / op

Inheritance

- structural or behavioural conformity
- interface subtyping (plug in)
- Modify states
  - Decompose state in OR or AND components
  - Add sub-states to OR state
  - Add orthogonal components to any state