Dynamic Modelling *with State Machines*

Prof. T.H. Tse
Department of Computer Science
Email: thtse@cs.hku.hk
Web: hku.hk/thtse.

State Machines

Motivation

Two types of operations in real life

(1) *Continuous Activities or Inactivities*

(2) *Actions* ...

State Machines

Motivation

(1) *Continuous Activities* — ***Alarm ringing***

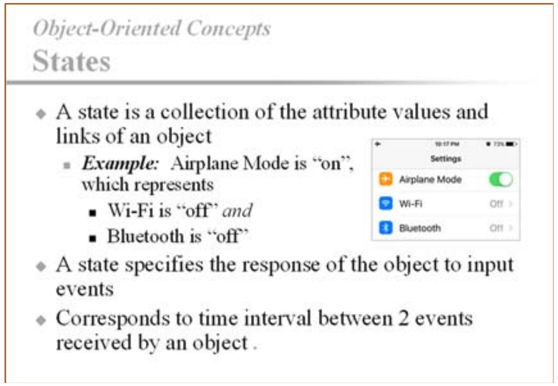
or Inactivities — ***Alarm being off***

- An activity or inactivity takes time
- Persists until it is
  - ***either*** completed — ***15 min has passed***
  - ***or*** interrupted by an ***action*** — ***Press "Stop" button***

State Machines

Motivation

- An activity or inactivity is modelled by a ***state*** in state machines:



Object-Oriented Concepts

States

- A state is a collection of the attribute values and links of an object
  - ***Example:*** Airplane Mode is "on", which represents
    - Wi-Fi is "off" *and*
    - Bluetooth is "off"
- A state specifies the response of the object to input events
- Corresponds to time interval between 2 events received by an object.

Motivation

(1) *Continuous Activities or Inactivities*

(2) *Actions*

- Everything takes time in real life
- The time actually taken is
  - ***either*** negligible
  - ***or*** of no interest to the user

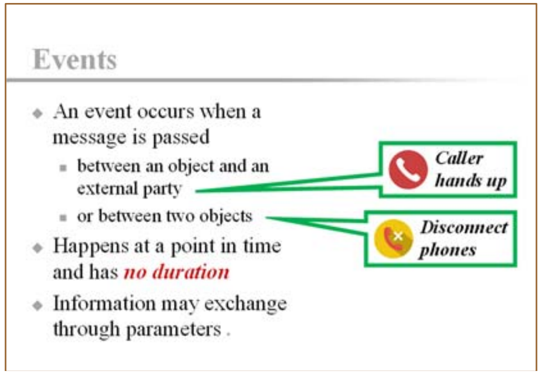
click ⟨ok⟩ button

wipe the slider

plane takes off.

Motivation

- An ***action*** is modelled by an ***event*** in state machines:


Events

- An event occurs when a message is passed
  - between an object and an external party
  - or between two objects
- Happens at a point in time and has ***no duration***
- Information may exchange through parameters.

Caller hands up

Disconnect phones

State Machines

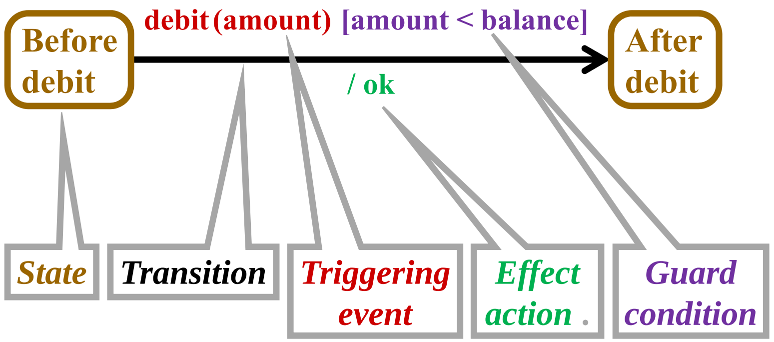
- Formerly known as state-transition diagrams

A ***transition*** is

- a change of ***state***
- caused by an ***triggering event***
- possibly with a ***guard condition***
- may result in an ***effect action***

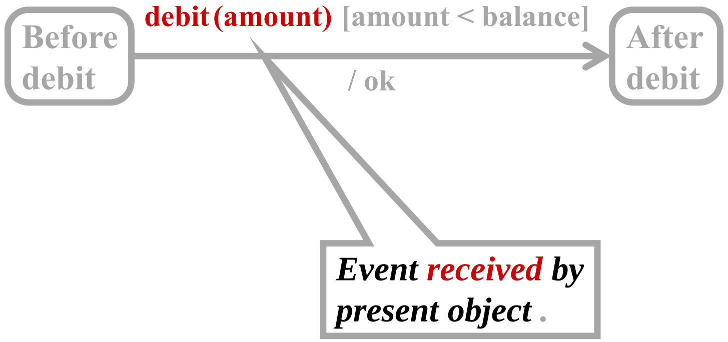
I will use 5 different colours.

Example 1



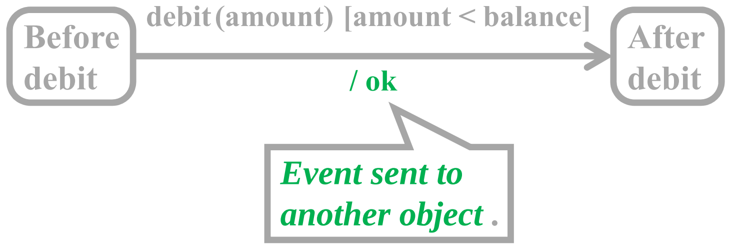
Example 1 (Continued)

Triggering Events



Example 1 (Continued)

Effect Action



State Machines

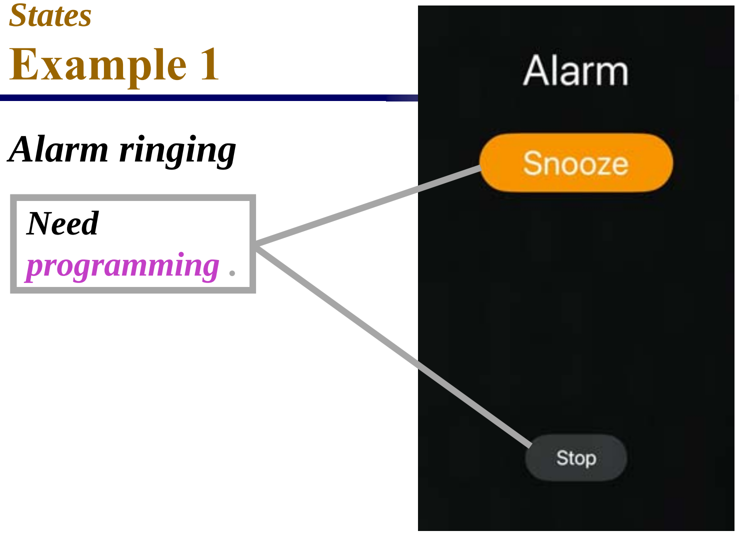
Guard Conditions

- A condition is a Boolean expression relating attributes or values
  - *Example:* debit (amount) [amount < balance]
- Conditions are used as guards on transitions
- A transition fires
  - *when* the triggering event occurs, ***and***
  - *if* the guard condition is satisfied .

States

Example 1

Alarm ringing



States

Example 1 (Continued)

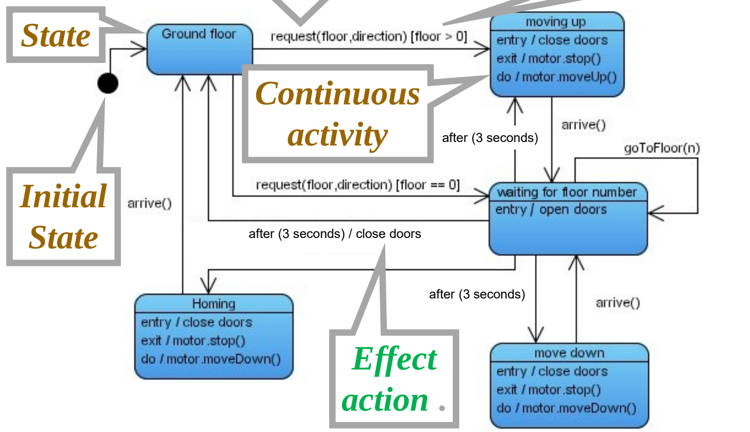
***Difficult** to connect with other events and states.*

Traditional Documentation

- ***State**.* Alarm ringing
- ***Previous State**.* Alarm being turned on
- ***Triggering Event**.* Reach target time
- ***Guard Condition**.* Sound being turned on
- ***Subsequent Events Accepted**:*
  - Press ⟨snooze⟩ button
  - Press ⟨ok⟩ button

State Machines | UML Syntax

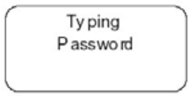
Overview



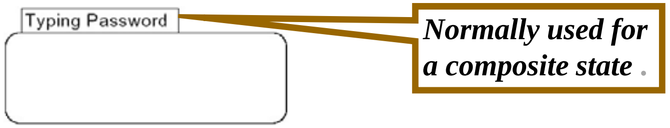
State Machines | UML Syntax

State

- A state is generally shown as a rectangle with rounded corners, with the state name inside:

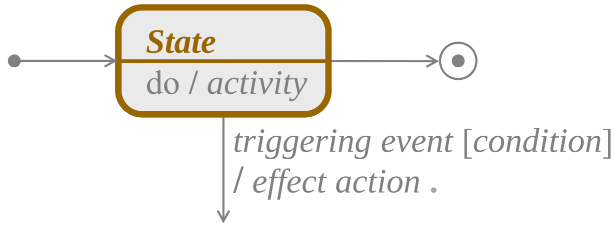


- Optionally, it may have an attached name tab:



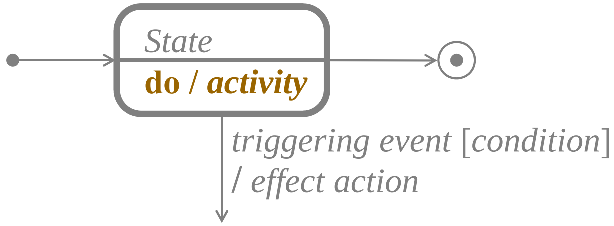
State Machines | UML Syntax

- A state machine describes the dynamic behaviour of ***one*** object
- A node represents a state:



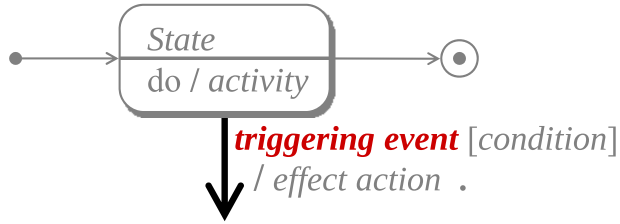
State Machines | UML Syntax

- "do / activity" represents the continuous activity during that state:



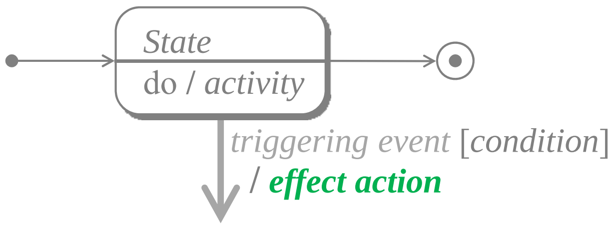
State Machines | UML Syntax

- An arrow represents a ***transition***, causing a change of state
- It is due to a triggering event (an ***incoming event***):



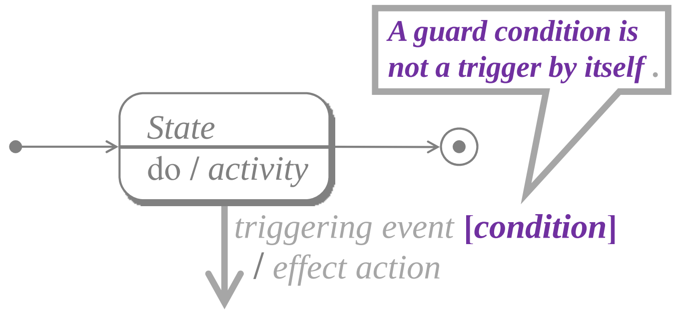
State Machines | UML Syntax

- The transition may result in a effect action (an ***outgoing event***):



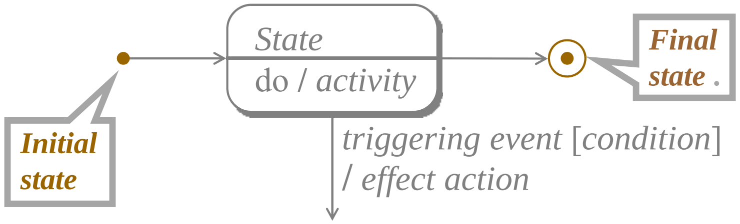
State Machines | UML Syntax

- Square brackets represent a guard condition:



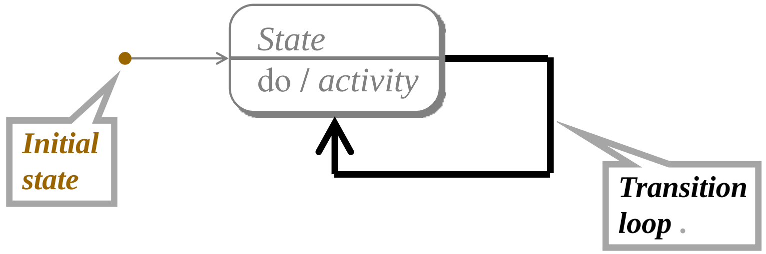
State Machines | UML Syntax

- State machines can be
  - ***One-shot*** life cycles, with ***initial*** and ***final states***:



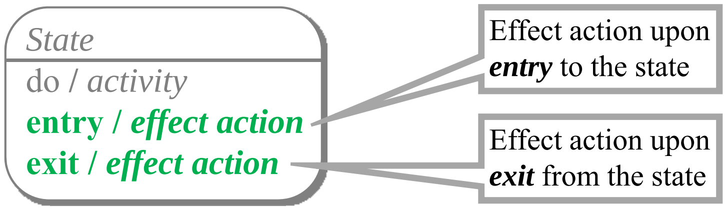
State Machines | UML Syntax

- State machines can be
  - ***One-shot*** life cycles, with ***initial*** and ***final states***
  - Or ***transition loops***:



State Machines | UML Syntax

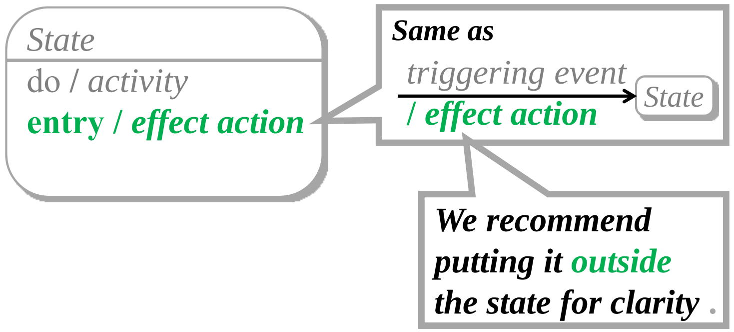
Other Activity Labels



- Example:

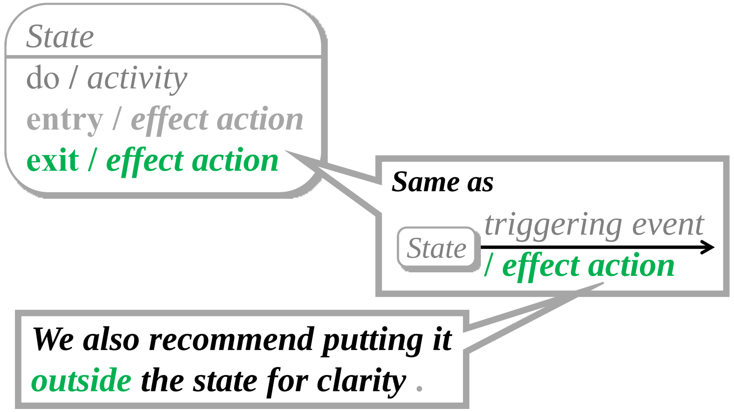
State Machines | UML Syntax

Other Activity Labels (Continued)



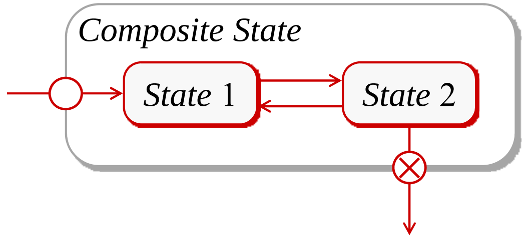
State Machines | *UML Syntax*

Other Activity Labels (Continued)



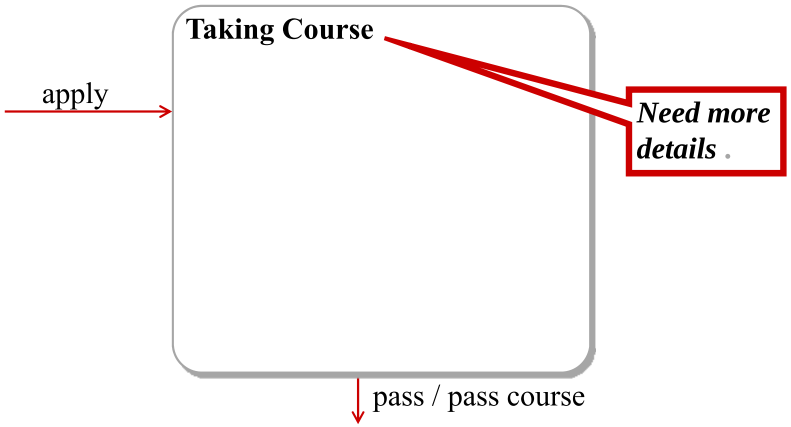
State Machines | *UML Syntax*

Composite States



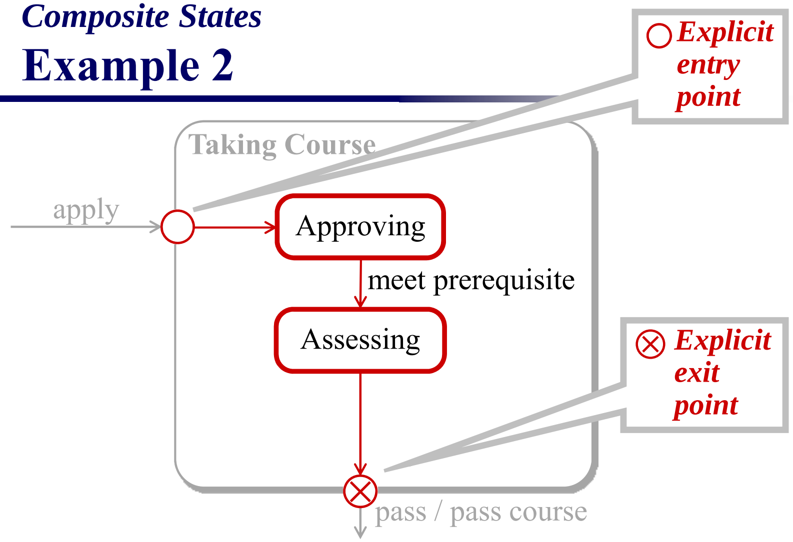
Composite States

Example 1



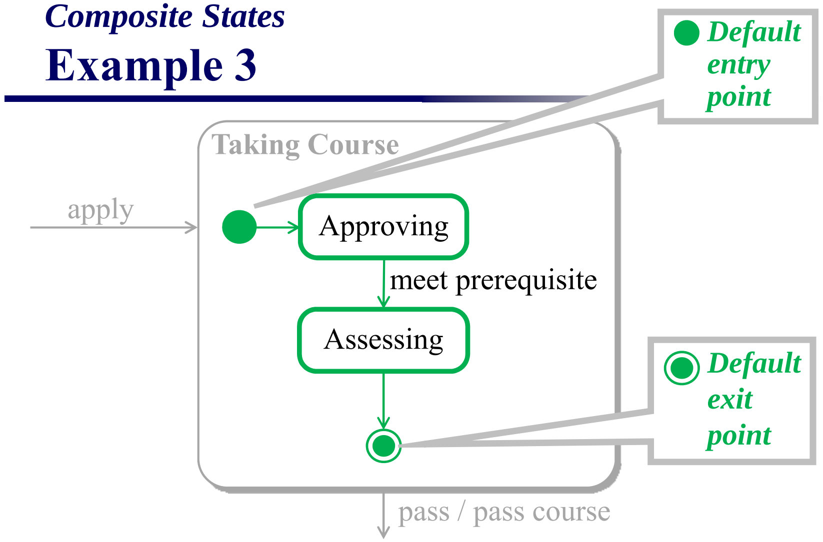
Composite States

Example 2



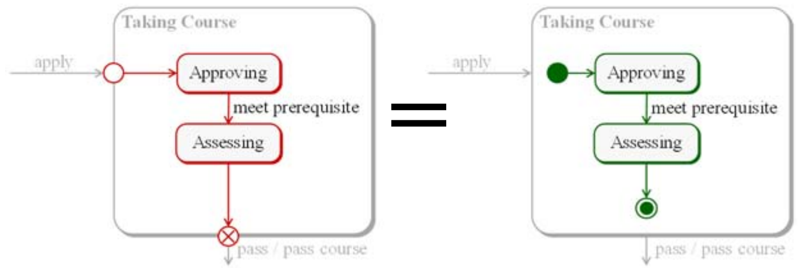
Composite States

Example 3



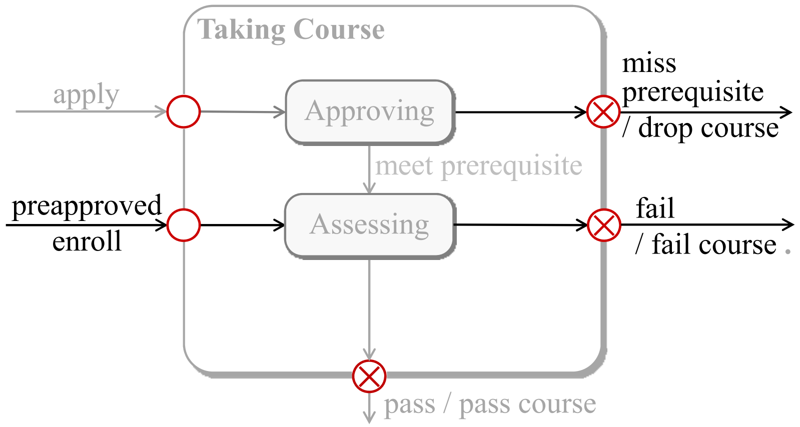
Composite States

Example 2 versus Example 3



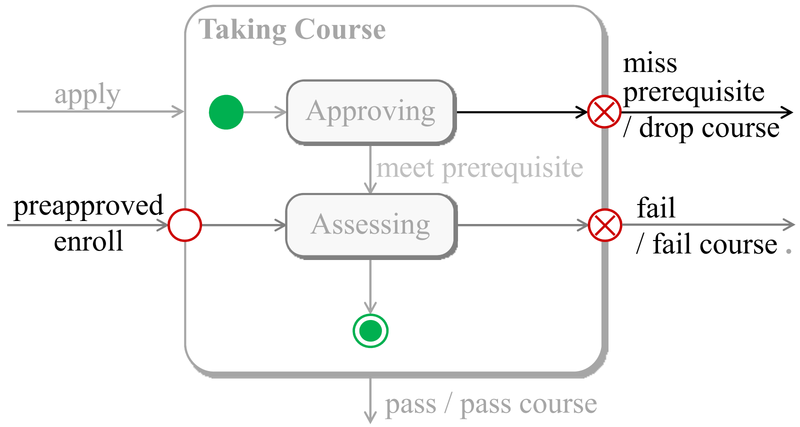
Composite States

Example 4



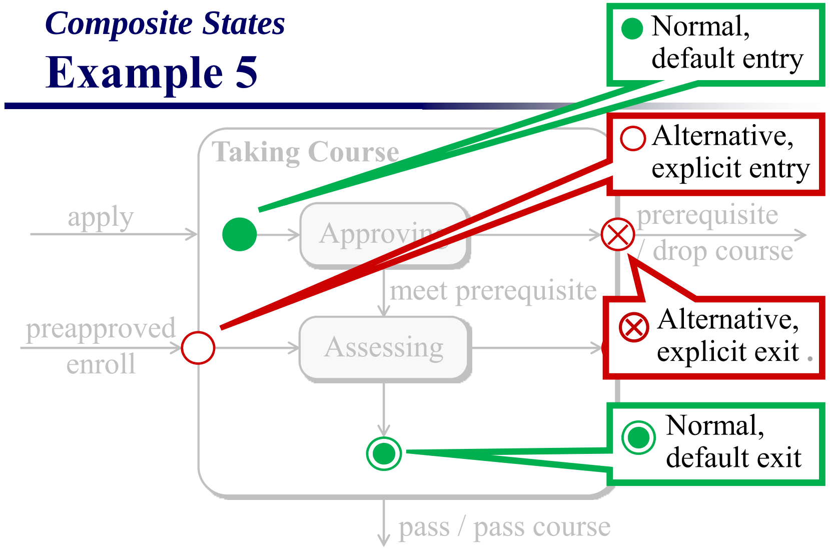
Composite States

Example 5



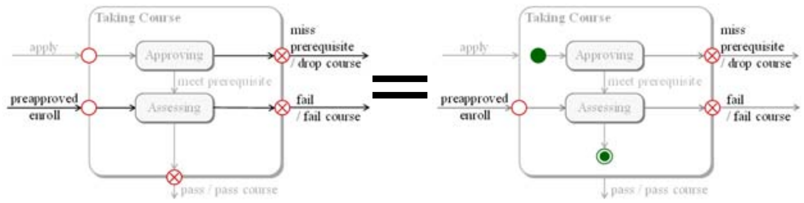
Composite States

Example 5



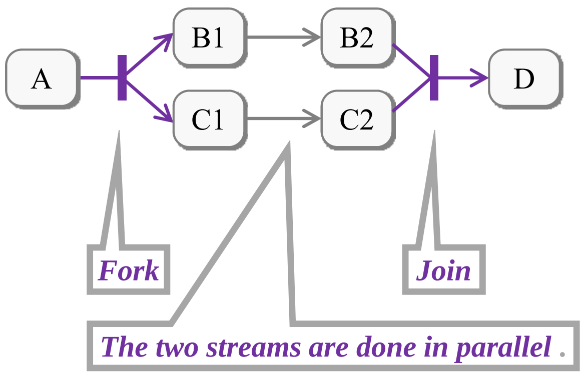
Composite States

Example 4 versus Example 5



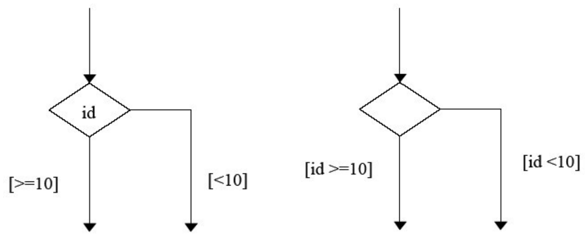
State Machines | UML Syntax

Fork and Join



State Machines | UML Syntax

Choice Pseudostate



- Recall that a guard condition is only part of a transition
- Very confusing to beginners
- ***Avoid at all costs*** .

Warning

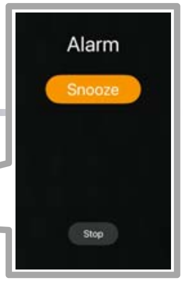
- *Semantics* (meaning) is more important than syntax
- *Process* is also more important than syntax .

Dynamic Modelling

Philosophy

- Specify the time-dependent behaviour of the system and its objects
- Draw ***sequence diagrams*** and ***state machines***
- Important for interactive systems .

Dynamic Modelling



Process

- Prepare interface formats
- Identify scenarios from ***use cases***
- Identify event interactions among objects according to ***class diagram***
- Prepare ***sequence diagrams*** for ***scenarios***
- Prepare a ***state machine*** for ***each class***
- Match events among objects to verify consistency .

Preparing Scenarios

- Prepare ***normal*** scenario
- Consider ***alternative*** scenarios .

Identifying Events

- Examine the scenarios to identify all events ***external to an object***
- Events may include
  - user inputs
  - user choices such as ⟨ok⟩ and ⟨cancel⟩
  - inputs from external devices
  - signals from other objects
  - interrupts
  - ***effect actions*** .

Class Diagram *versus* Sequence Diagram

- Relationships in class diagrams are persistent
- They show potential information flows
- Paths in sequence diagrams are transient
- They show interactions

How many class diagram(s)?

How sequence diagram(s)? .

Building State Machines

Prepare a state machine for each class with dynamic behaviour

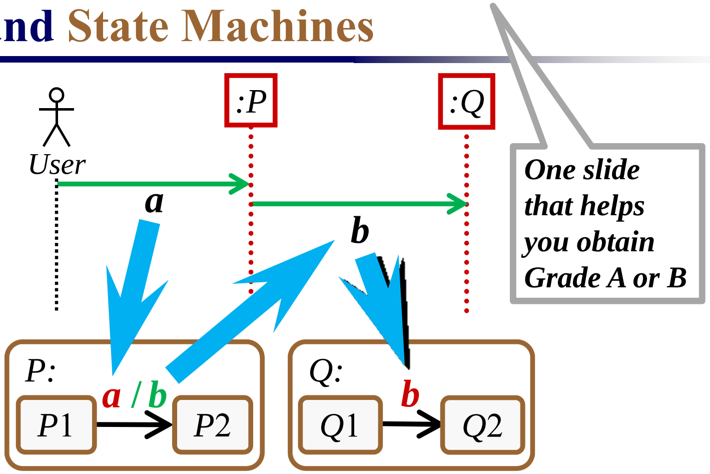
- Pick a ***scenario*** from sequence diagram
- Arrange the events into a ***path*** in state machine, with ***transitions*** labelled by the input/output events along a ***lifeline***
- Replace repeating sequences of events with loops ...

Building State Machines

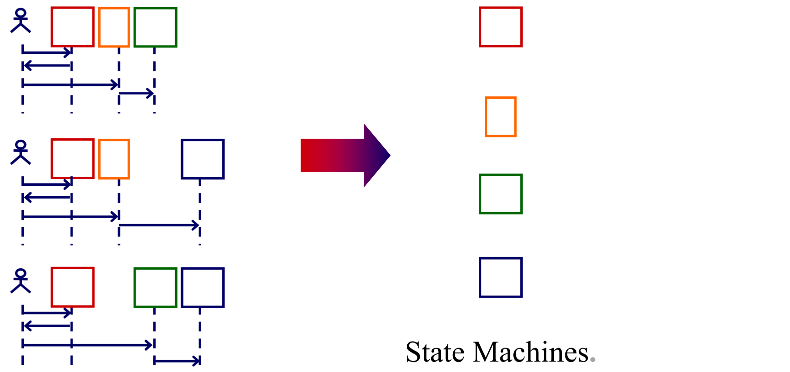
Prepare a state machine for each class with dynamic behaviour (continued)

- Iteratively add other ***paths*** to the state machine
  - Find a point in the ***scenario*** where it diverges
  - Attach the new event sequence to the existing state as an alternative ***path***
- Add any other possible events, such as boundary (interface) cases and exceptional cases .

Relating Sequence Diagram and State Machines

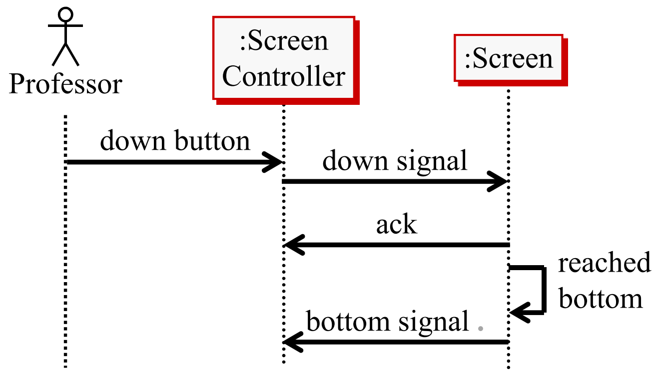


How Many State Machines?



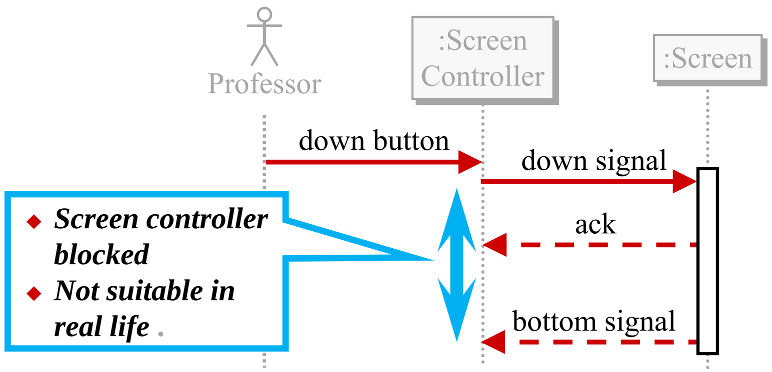
Example 1: Screen Control System

Sequence Diagram (1st Draft)

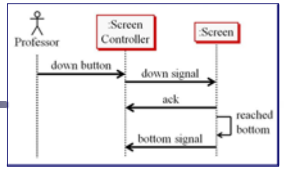


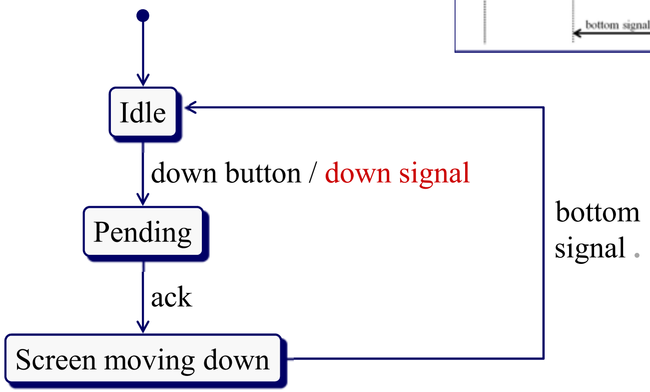
Example 1: Screen Control System

(For Students Who Like Synchronous Version)

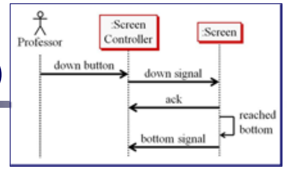


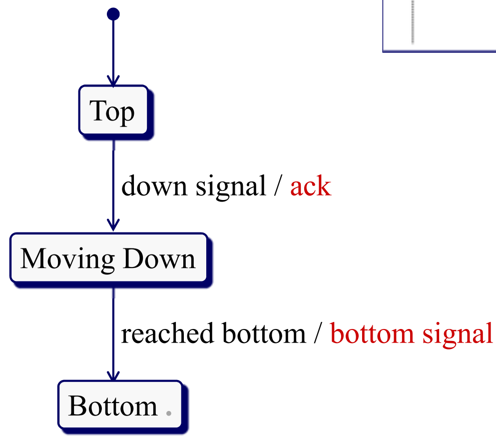
Screen Controller
State Machine (1st Draft)



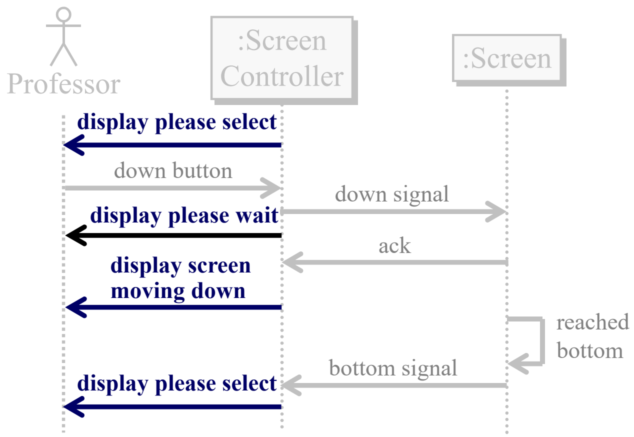


Screen
State Machine (1st Draft)

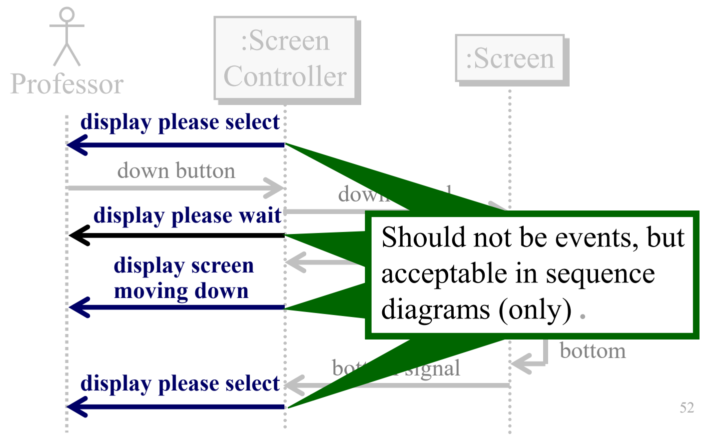




Screen Control System
Sequence Diagram (2nd Draft)

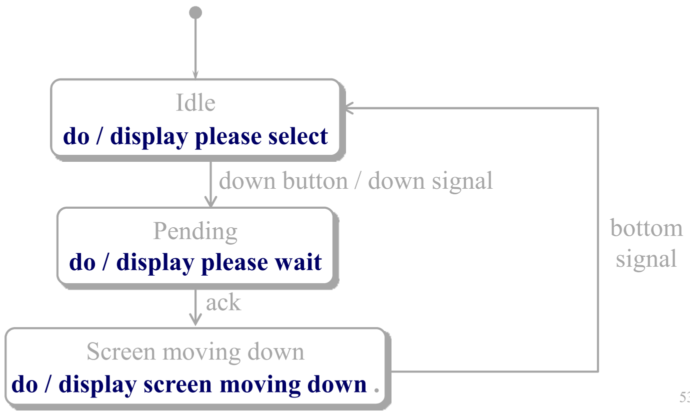


Screen Control System
Sequence Diagram (2nd Draft)



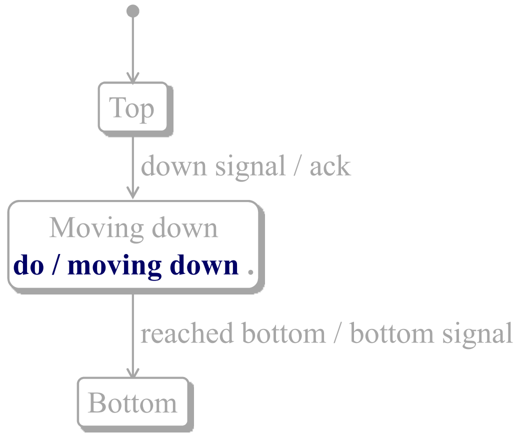
Screen Controller

State Machine (2nd Draft)



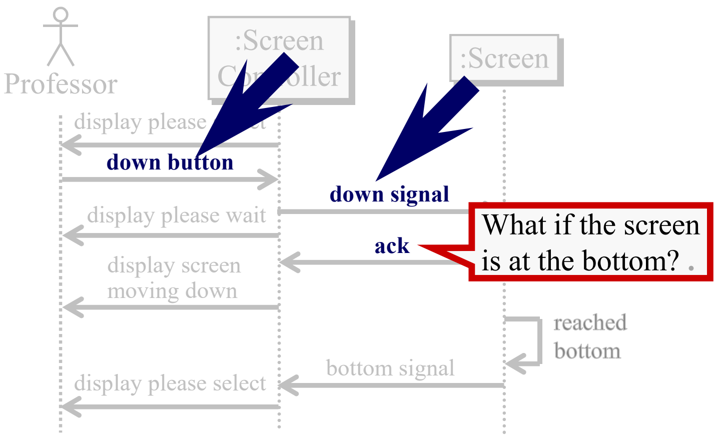
Screen

State Machine (2nd Draft)



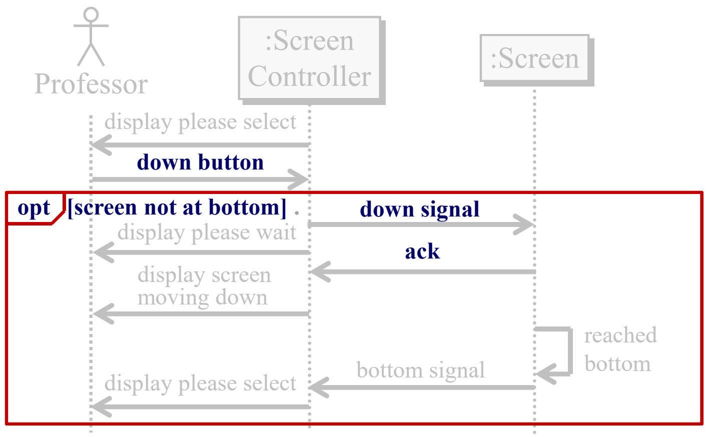
Screen Control System

Sequence Diagram (2nd Draft)

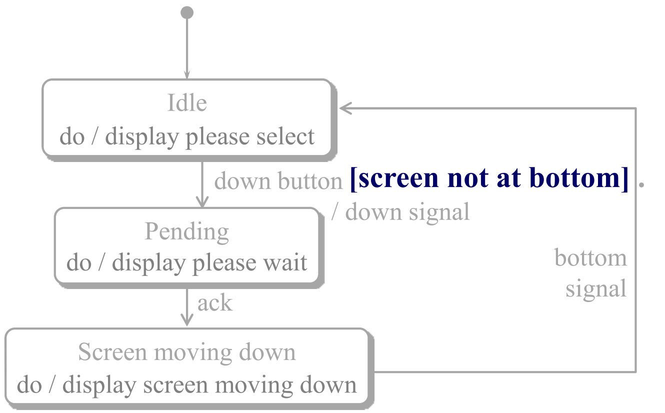


Screen Control System

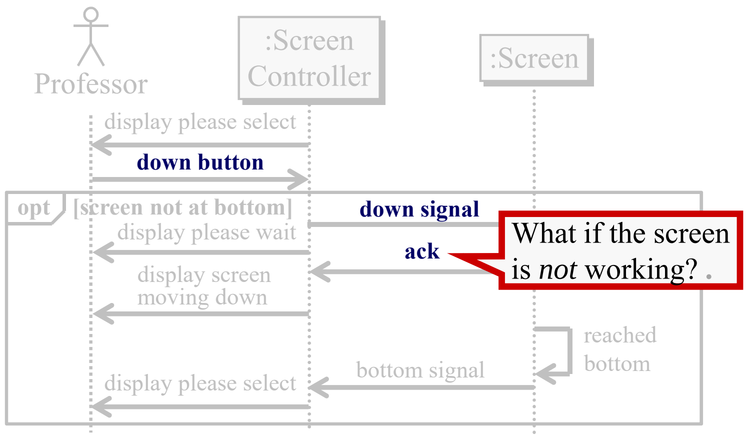
Sequence Diagram (3rd Draft)



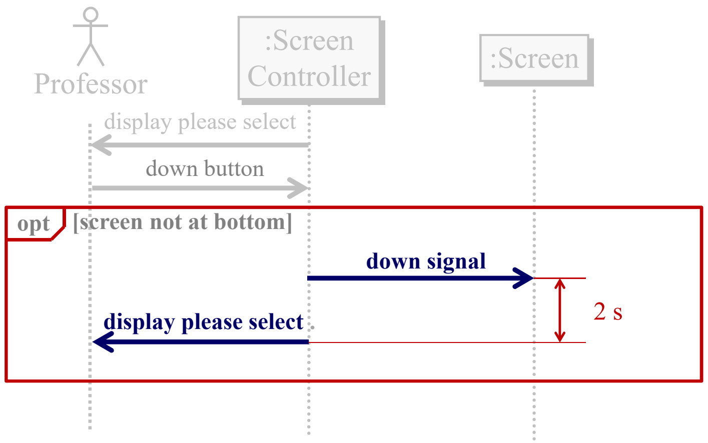
Screen Controller
State Machine (3rd Draft)



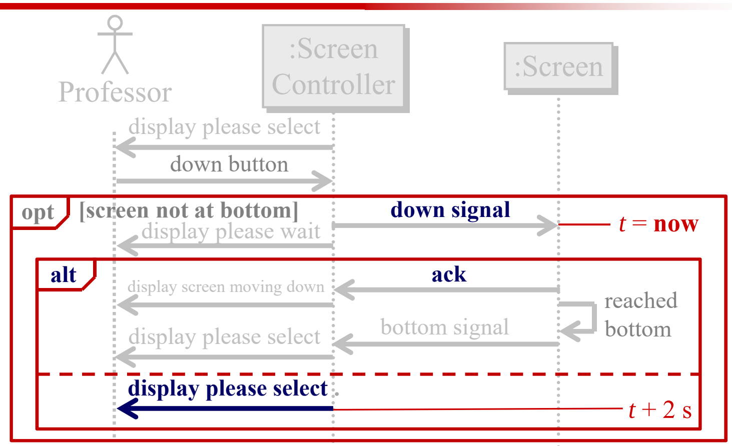
Screen Control System
Sequence Diagram (3rd Draft)



Sequence Diagram (4th Draft: New Scenario)

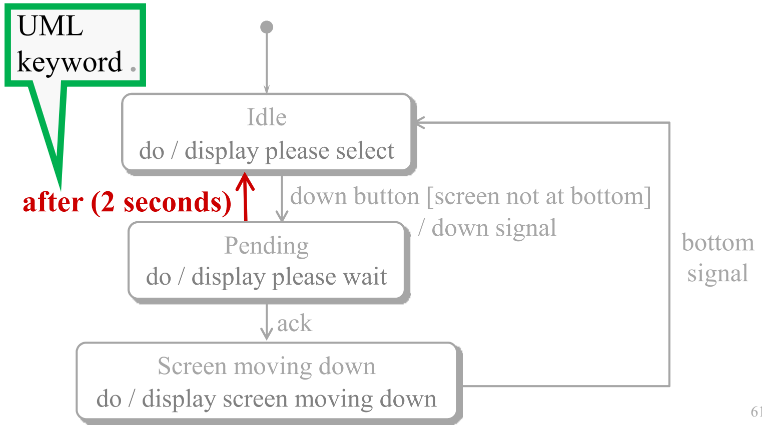


Sequence Diagram
(For Students Who Like to Combine Scenarios)



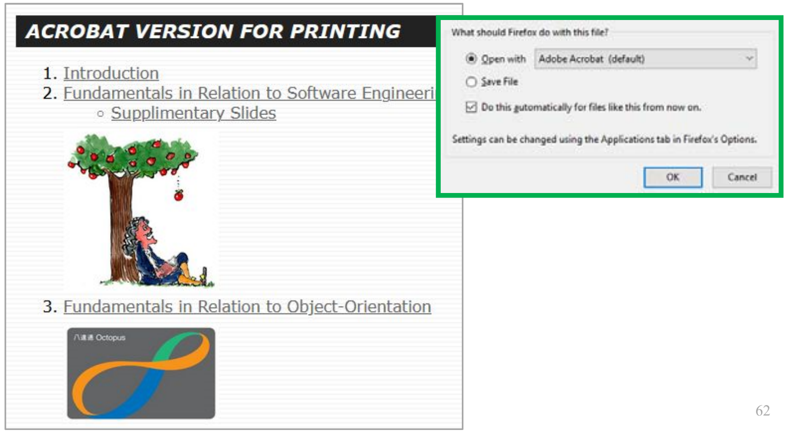
Screen Controller

State Machine (4th Draft)

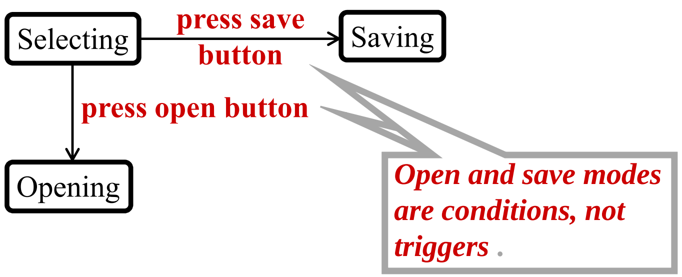


Example 2

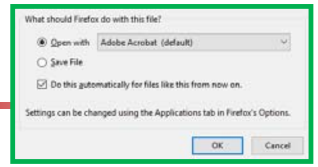
Lecture Materials

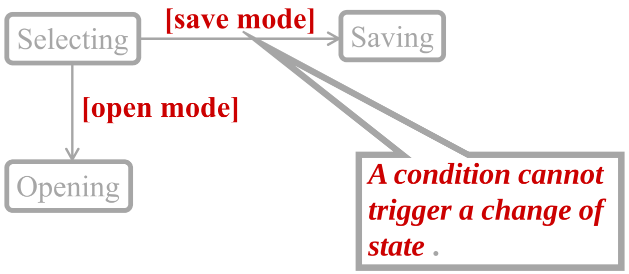


We Learn from Mistakes

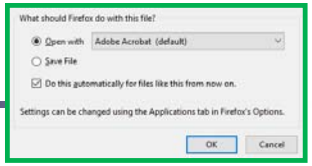


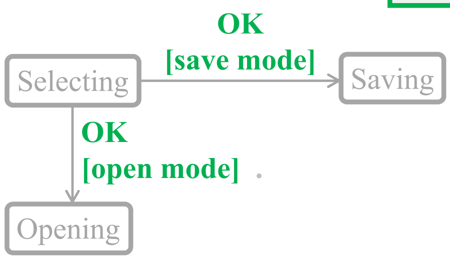
We Learn from Mistakes





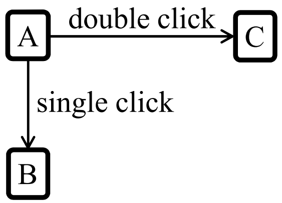
Recommended Solution





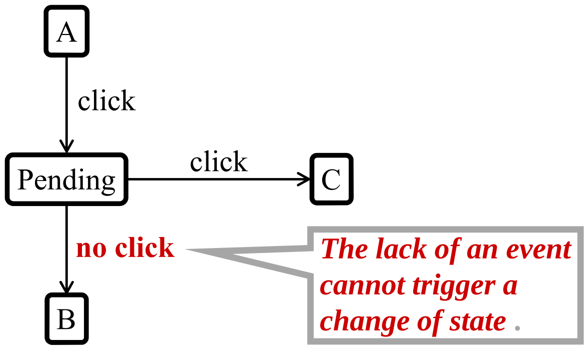
Example 3

Single and Double Clicks

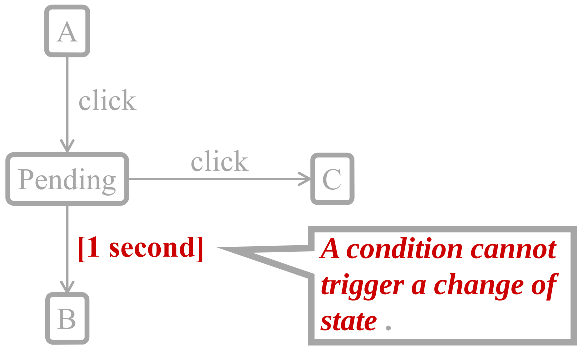


- Legitimate (high level) model
- What if we would like to define "single click" and "double click" via the state machine?

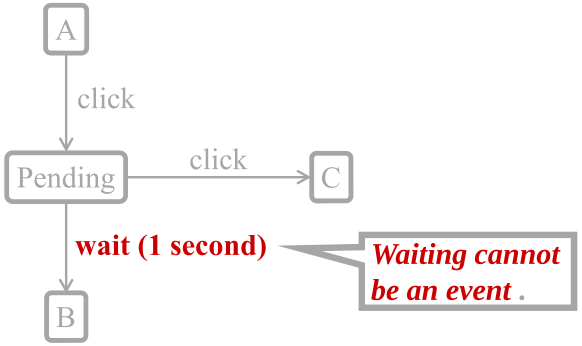
We Learn from Mistakes



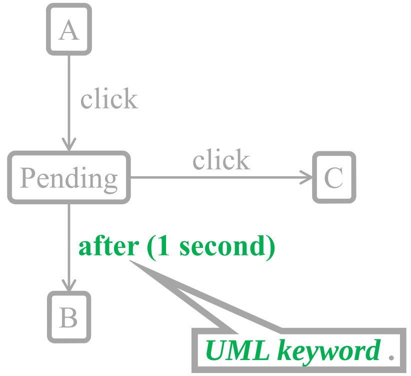
We Learn from Mistakes



We Learn from Mistakes

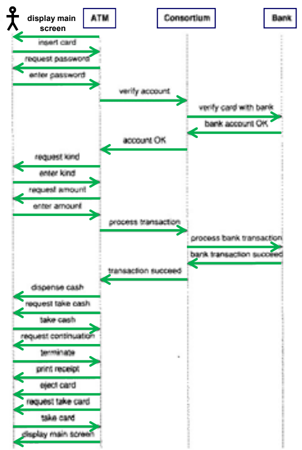


Recommended Solution



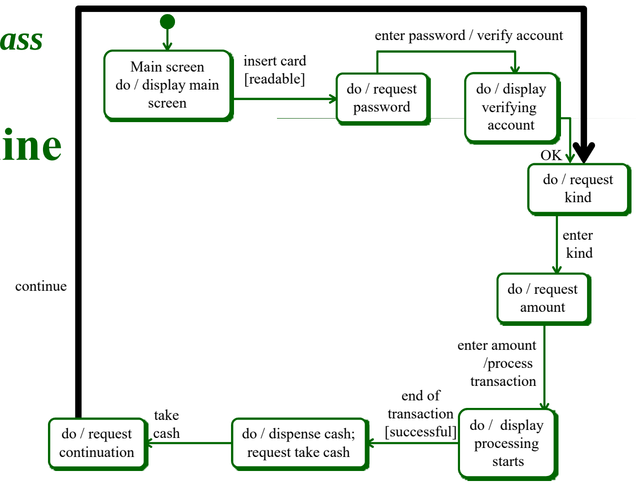
Example 4: ATM System

Sequence Diagram

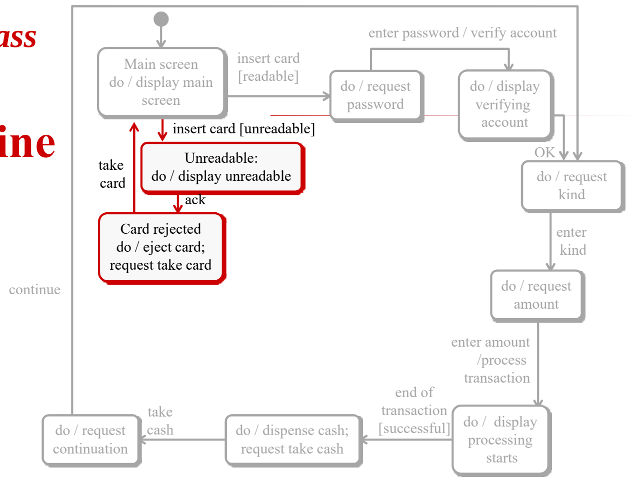


ATM Class

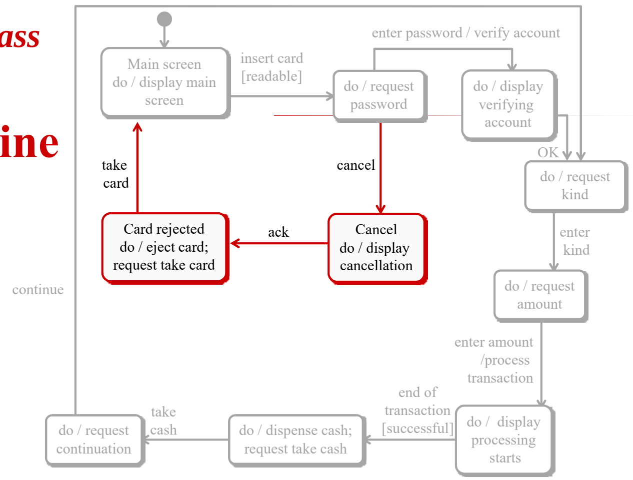
State Machine



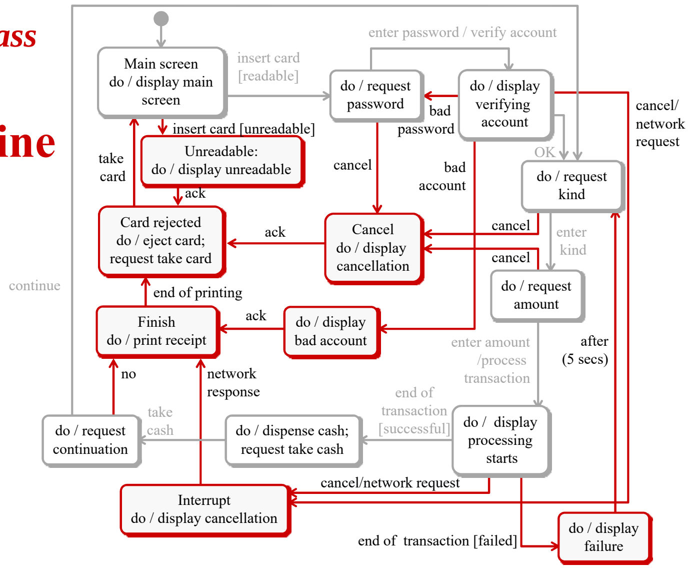
ATM Class
State Machine



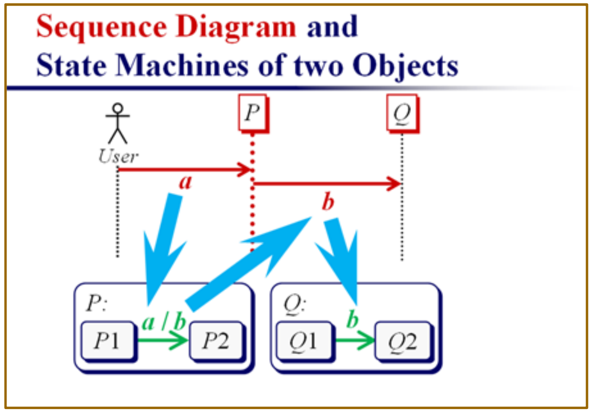
ATM Class
State Machine



ATM Class
State Machine

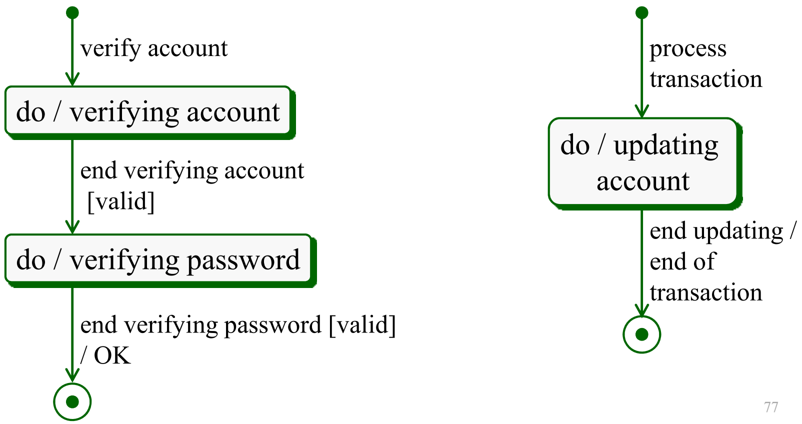


Recall



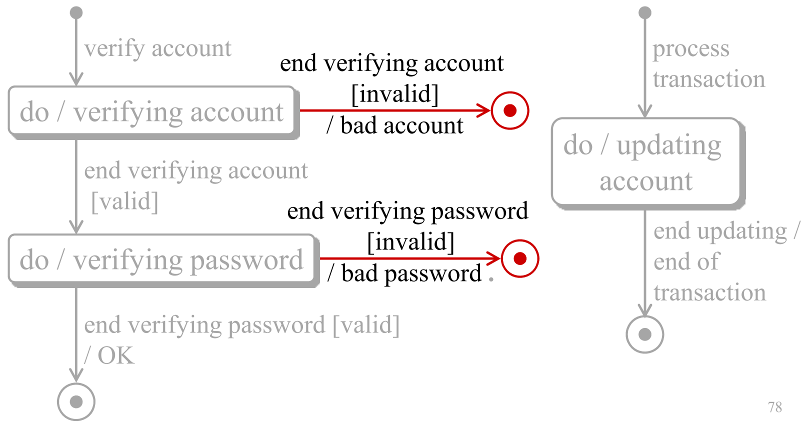
Bank Class

State Machine



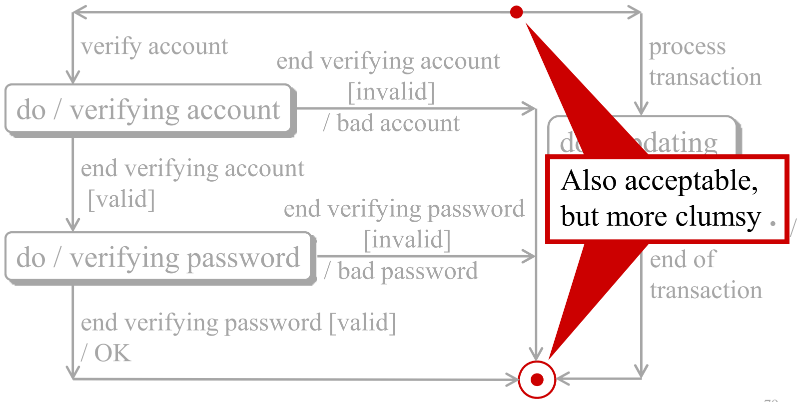
Bank Class

State Machine



Bank Class

State Machine



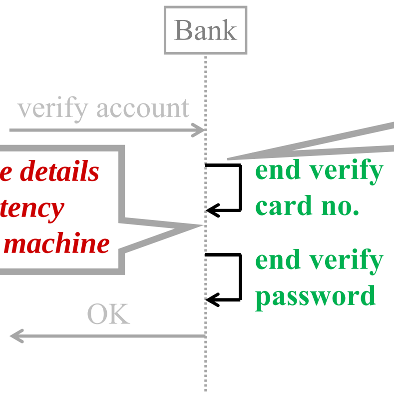
Matching Events Between Objects

Check completeness and consistency

- A sender and a receiver for each event
- Predecessors or successors for every state
- Consistencies of corresponding events between ***sequence diagrams*** and ***state machines***
- Consistencies of corresponding events between different ***state machines***
- Potential synchronization errors, especially when an input occurs at an awkward time.

Matching Events Between Objects

Example 5



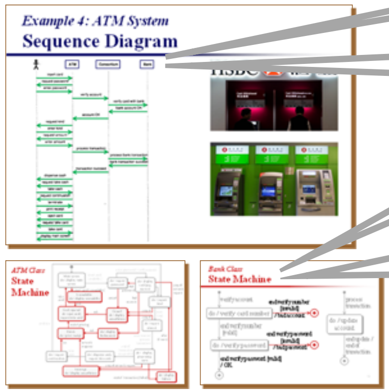
Need more details for consistency with state machine

This syntax is acceptable

But ***not*** .

Matching Events Between Objects

Example 6



How many objects?

Hence, how many state machines?

How many state machines here?

State machine for Consortium class is missing .