CS4125
SYSTEMS ANALYSIS
SPRING SEMESTER 2010-2011

J.J. Collins
Dept of CSIS
University of Limerick

Lecture L
W7L1 – W8L1)

Analysis: Behaviour Modelling – State Charts Part 1

Introduction

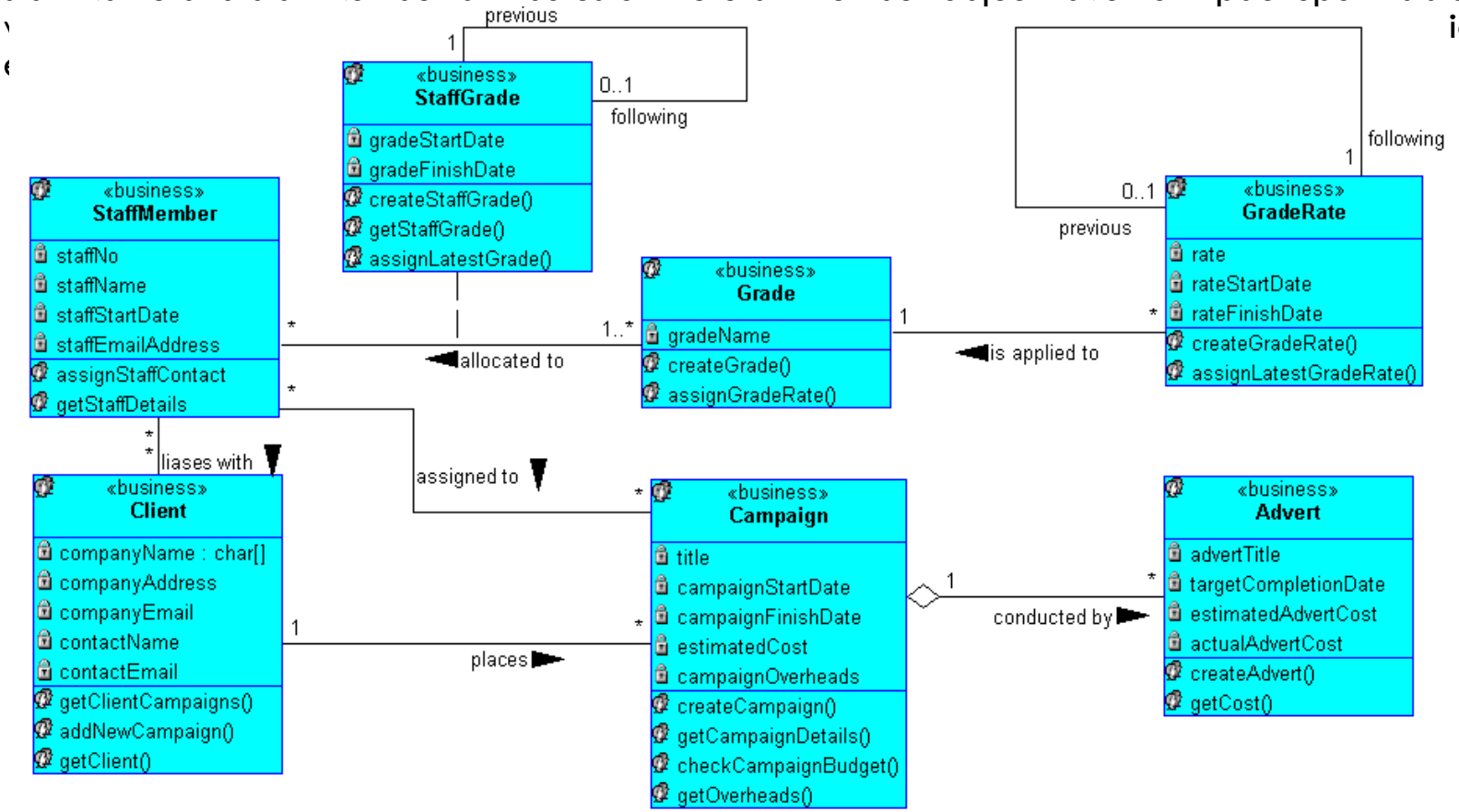
- An object has state, behaviour and unique identity.
- Important to model state dependent variations in behaviour since they represent constraints on the way the system should behave.
- e.g. vending machine: variation in behaviour determined by state.
- UML uses statecharts.
- The notation used in UML is based on the work of Harel (1987), and was adopted by OMT and in the second version of the Booch approach (Booch, 1994).
- A model of state in a statechart captures all the possible responses of a single object to all the use cases in which it is involved.
- By contrast, a sequence diagram captures the responses of all objects that are involved in a single use case.

Introduction

- A statechart can be viewed as a description of all possible life cycles that an object of a class may follow.
- Some CASE tools handle statecharts as a child diagram to a class or class diagram.
- SSADM uses entity life histories to model state.
- In the OO approach, statecharts can be used for other purposes other than modelling life cycles i.e. to build models of human-computer dialogues.

States and Events

- The current state of an object is a result of its attributes and the links that it has with other objects.
- Some attributes and links are significant for determining state, while others are not.
- staffName and staffNumber attributes of the StaffMember object have no impact upon its state,



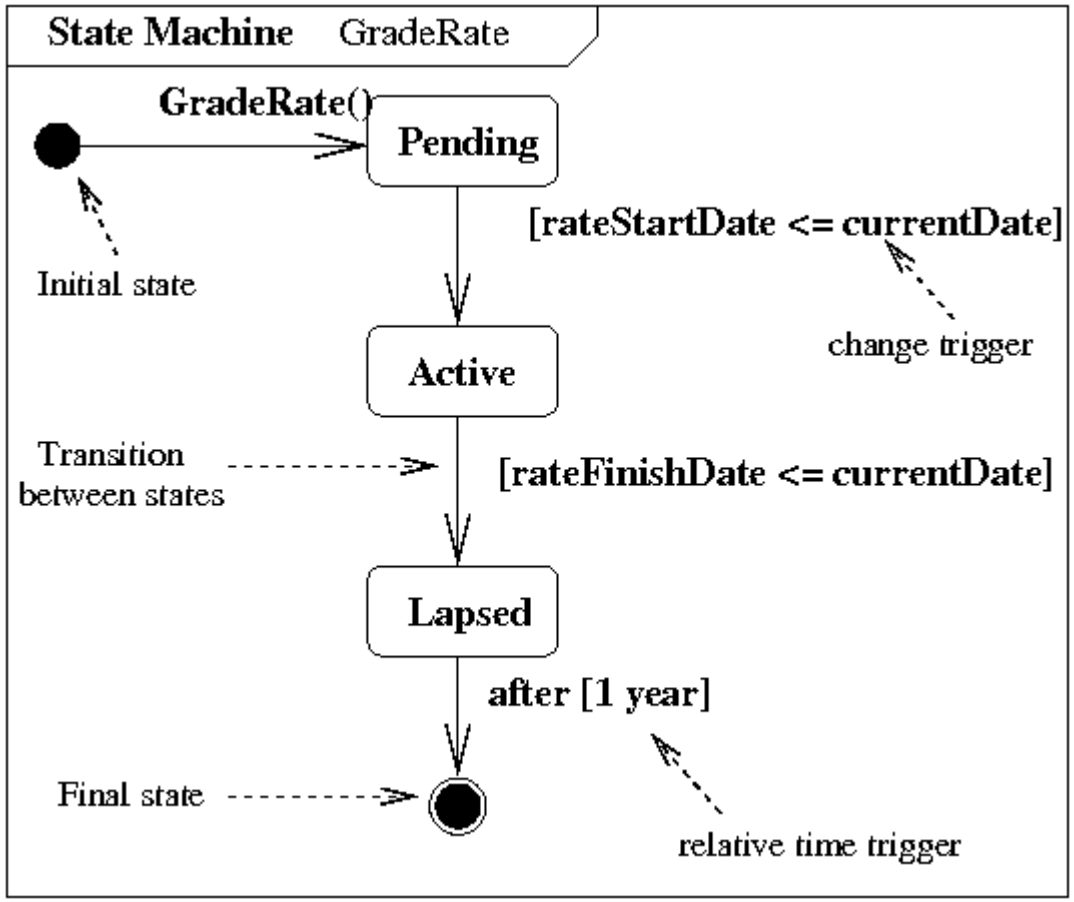
Fragment of class diagram from Agate case study in Bennett et al.

States and Events

- The StaffMember object is in the Probationary state for the first six months of employment.
- The possible states that an object can occupy are limited by its class.
- Objects of some classes may have only one possible state.
- e.g. in the **Agate case study**, a Grade object either exits or it does not. Either it is Available for use or it is not.

States and Events

- Objects have more than one possible state
- e.g. an object of the class GradeRate may be Pending, Active or Lapsed, as shown in figure 2.
- Movement from one state to another is called a transition, and is triggered by an event.
- When its triggering event occurs a transition is said to fire.
- Transition shown as a solid arrow from the source state to the target state.
- An event is an instance of an event type.
- It is usually event types that are modelled, but are referred to simply as events.



Statechart for the class *GradeRate*.

Triggers

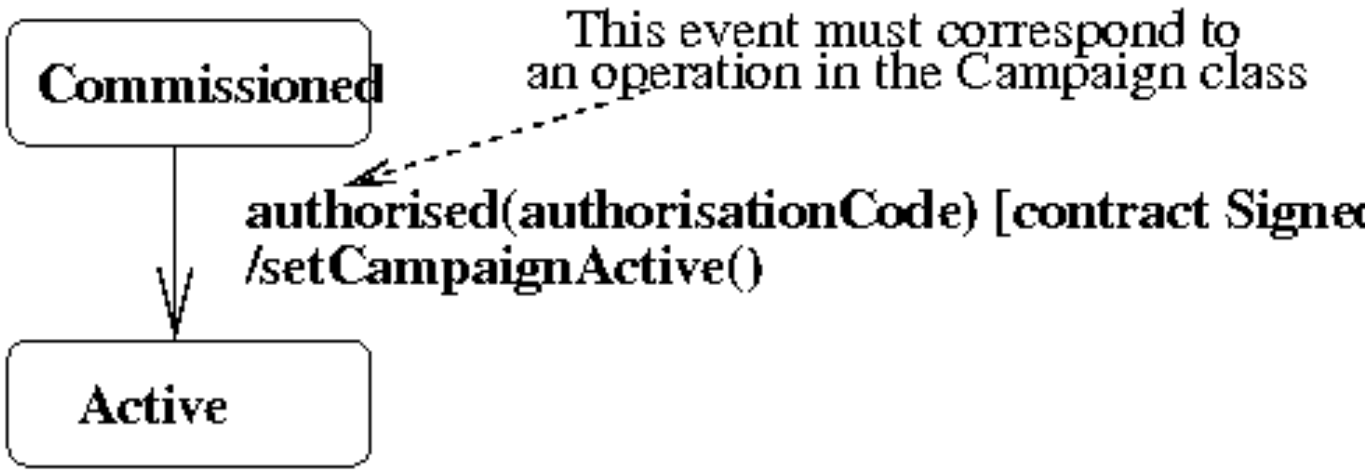
- Change Trigger:
  - Occurs when a condition (usually Boolean) becomes true.
  - Annotated by the keyword **when** in UML 1.x, followed by the Boolean expression in parenthesis.
  - This form of conditional event is different from a guard condition that is only evaluated at the moment that it's associated event fires.
- Call Trigger:
  - Occurs when an object receives a call for one of its operations. Annotated by the signature of the operation as the trigger for the transition.
- Signal Trigger:
  - Occurs when an object receives a signal.
  - As with call events, annotated with the signature of the operation invoked.
  - No syntactic difference between call and signal events.
- Relative Time Trigger:
  - Caused by the passage of a designated period of time after a specified event.
  - Where the time event reflects the passage of time after entry to the current state, the event is annotated by the keyword **after** followed by the amount of time in parenthesis.

Notation

- The basic syntax for a call or signal event is:
  trigger-signature ‘[‘ constraint ‘]’ ‘/’ activity-expression
- Where the trigger signature is:
  event-name ‘(‘ parameter-list ‘)’
- Where the parameters in the parameter list are of the form
  parameter-name `:' type-expression
- An activity-expression corresponds to invocation of an operation during firing of the transition
- A small solid filled circle indicates initial state of a life cycle - notational convenience..
- i.e. the GradeRate object enters the Pending state immediately.
- Transition from the initial state can be optionally labelled with the event that creates the object.
- The end point of a life cycle is shown by a bull's-eye symbol.

Notation

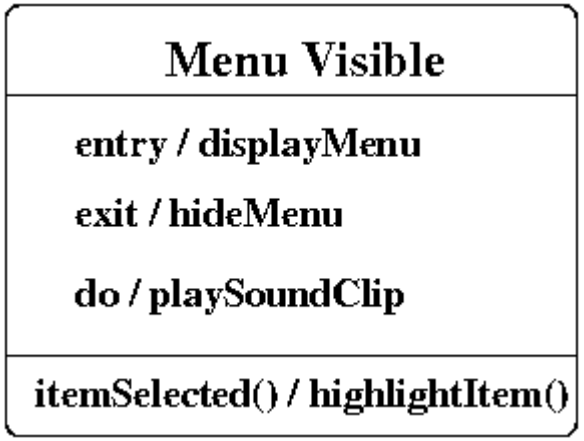
- All other states are shown as a rectangle with rounded corners.
- Figure 3 illustrates the basic notation for a statechart with two states for the class *Campaign*.
- A transition should be labelled with a transition string.



Fragment of statechart for the class *Campaign*.

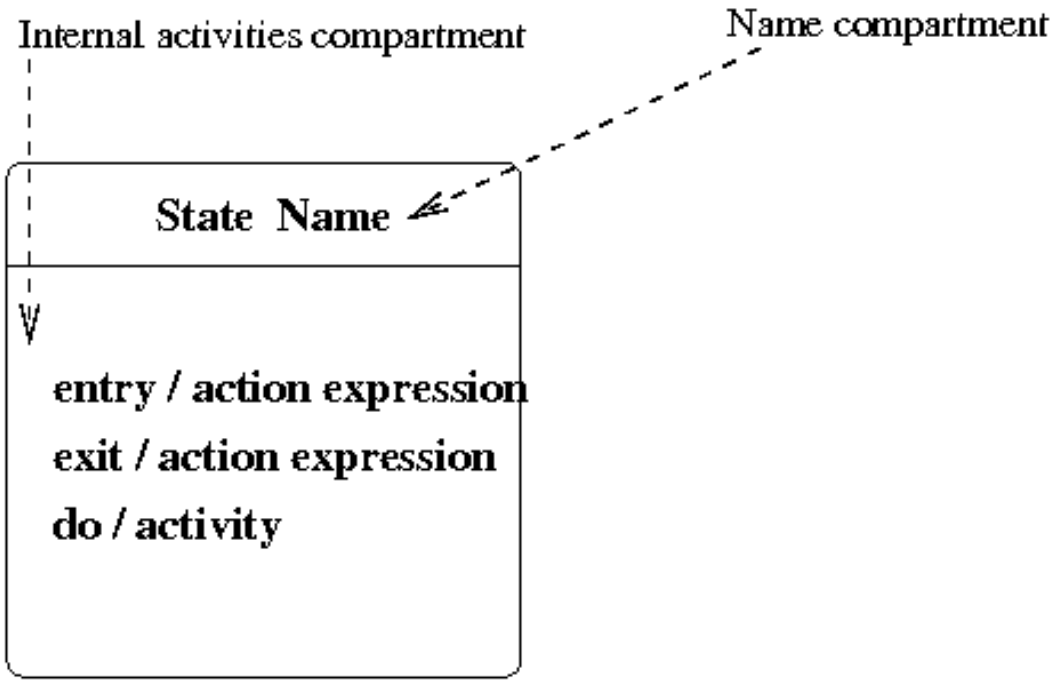
Events That Do Not Change State

- An activity may persist for the duration of the state unlike actions that are transitory.
- An internal transition occurs in response to an event, results in an action, but does not cause a change in state.
- Shown in internal activities compartment
- Two kinds of internal events have a special notation:
  - Entry event.
  - Exit event.
- These cannot have a guard condition, and are invoked implicitly on entry or exit from the state.
- State activities preceded by the keyword **do**.



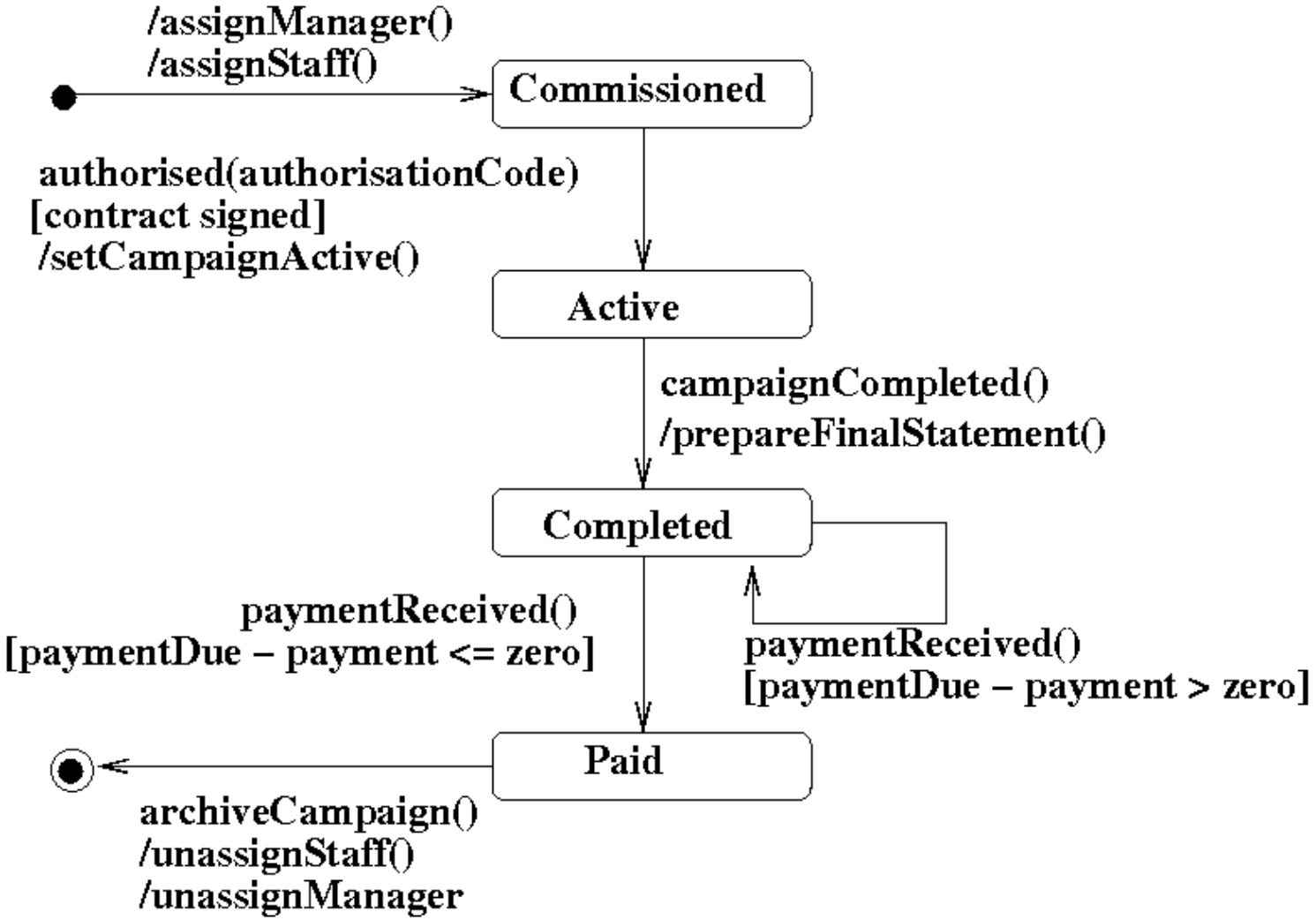
Menu Visible state for a DropDownMenu object.

Events That Do Not Change State



Internal actions and activities for a state.

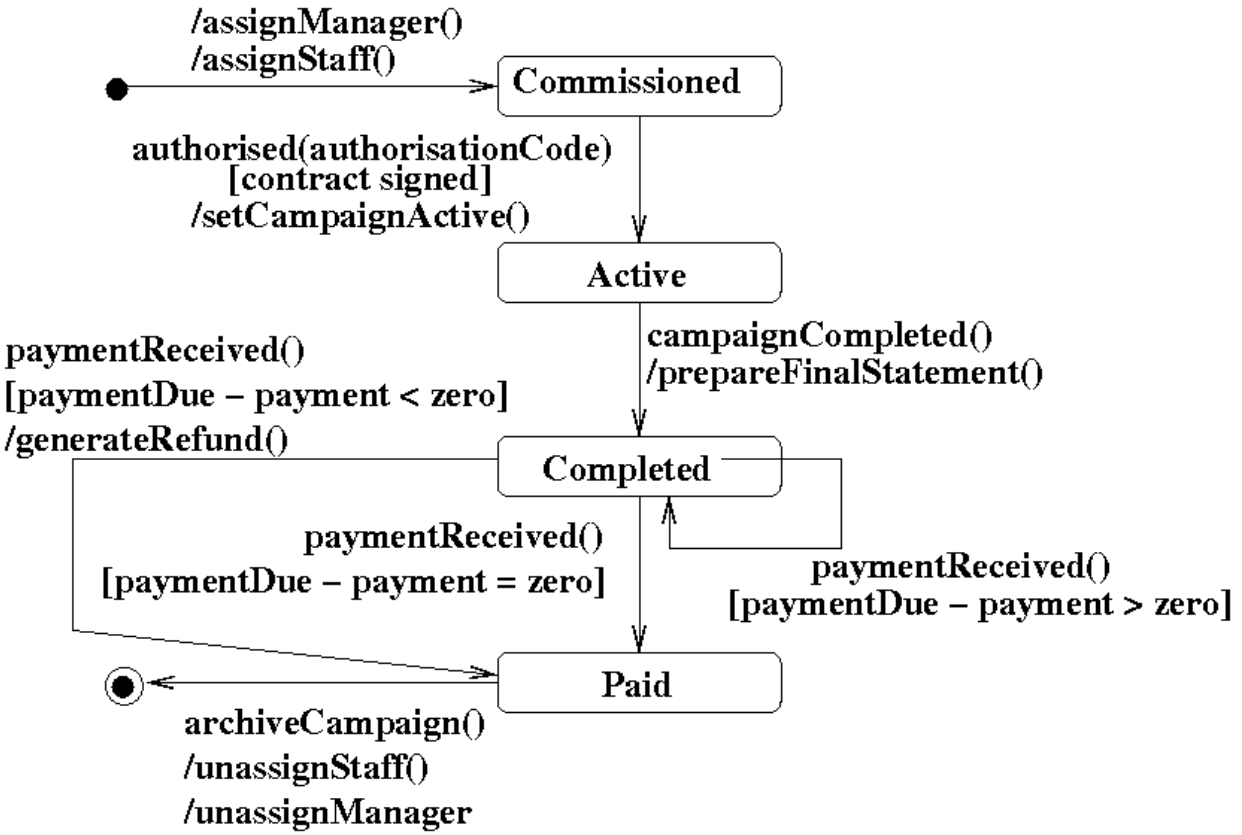
Guard Conditions



Statechart for the class Campaign

Guard Conditions

- Life cycle is unambiguous when all the transitions from each state are mutually exclusive.
- Figure: requirements changed so that an overpayment now results in the automatic generation of a refund.



A revised statechart for the class Campaign.

Reading

- Chapter11 in Bennett et al. or
- Chapters11 and 12 in Stevens and Pooley