Specifying Control

Based on Chapter 9
Bennett, McRobb and Farmer

Object Oriented Systems Analysis and Design Using UML

4th Edition, McGraw Hill, 2010

McGraw Hill Education

© 2010 Bennett, McRobb and Farmer

In This Lecture You Will Learn:

- how to identify requirements for control in an application;
- how to model object life cycles using state machines;
- how to develop state machine diagrams from interaction diagrams;
- how to model concurrent behaviour in an object;
- how to ensure consistency with other UML models.

Mc Graw Hill Education

State

- The current state of an object is determined by the current value of the object's attributes and the links that it has with other objects.
- For example the class `StaffMember` has an attribute `startDate` which determines whether a `StaffMember` object is in the probationary state.

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

State

- A state describes a particular condition that a modelled element (e.g. object) may occupy for a period of time while it awaits some event or *trigger*.
- The possible states that an object can occupy are limited by its class.
- Objects of some classes have only one possible state.
- Conceptually, an object remains in a state for an interval of time.

© 2010 Bennett, McRobb and Farmer

state machine

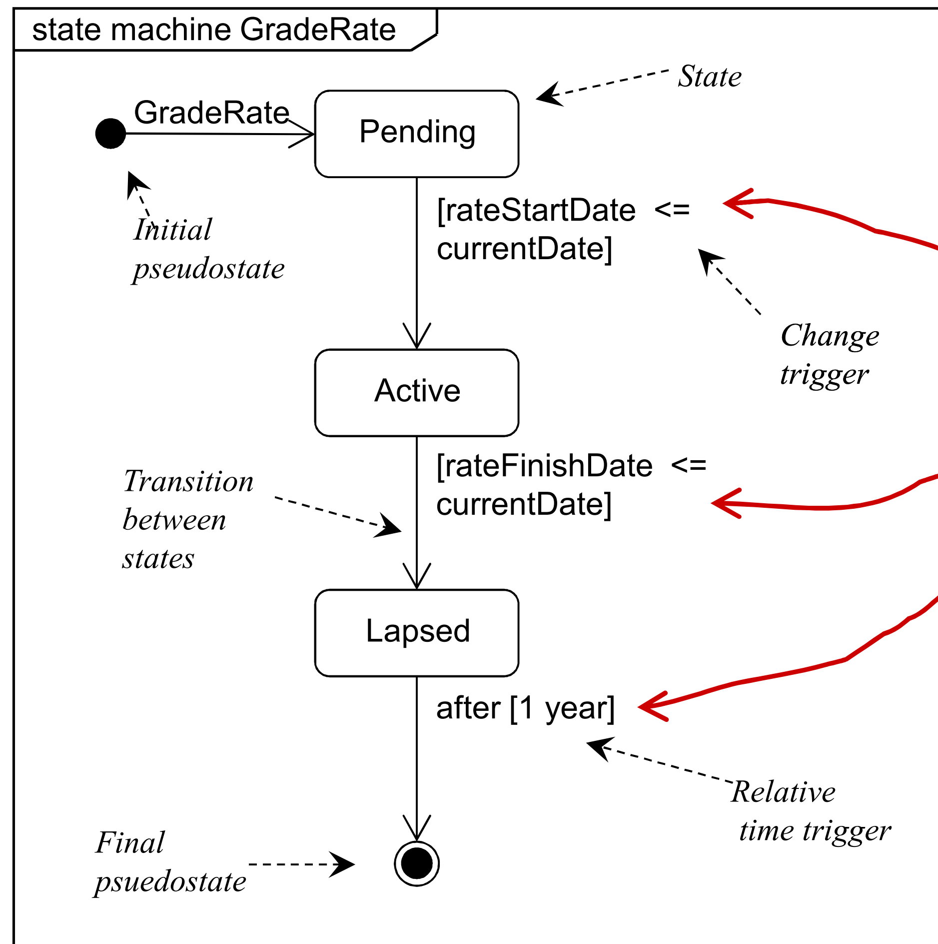
- The current state of a `GradeRate` object can be determined by the two attributes `rateStartDate` and `rateFinishDate`.
- An enumerated state variable may be used to hold the object state, possible values would be Pending, Active or Lapsed.

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

state machine

state machine for the class `GradeRate`.



Movement from one state to another is dependent upon events that occur with the passage of time.

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

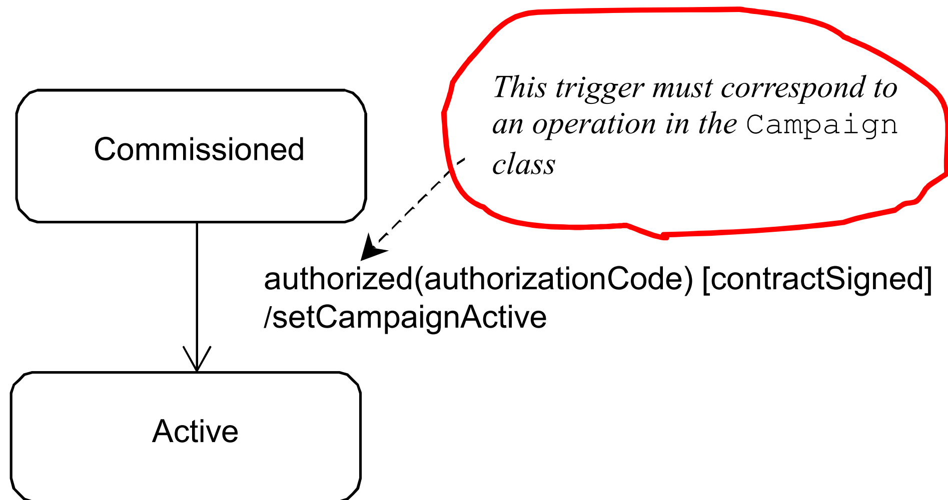
Types of Event

- A *change trigger* occurs when a condition becomes true.
- A *call trigger* occurs when an object receives a call fo one of its operations either from another object or from itself.
- A *signal trigger* occurs when an object receives a signal (an asynchronous communication).
- An *relative-time trigger* is caused by the passage of a designated period of time after a specified event (frequently the entry to the current state).

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

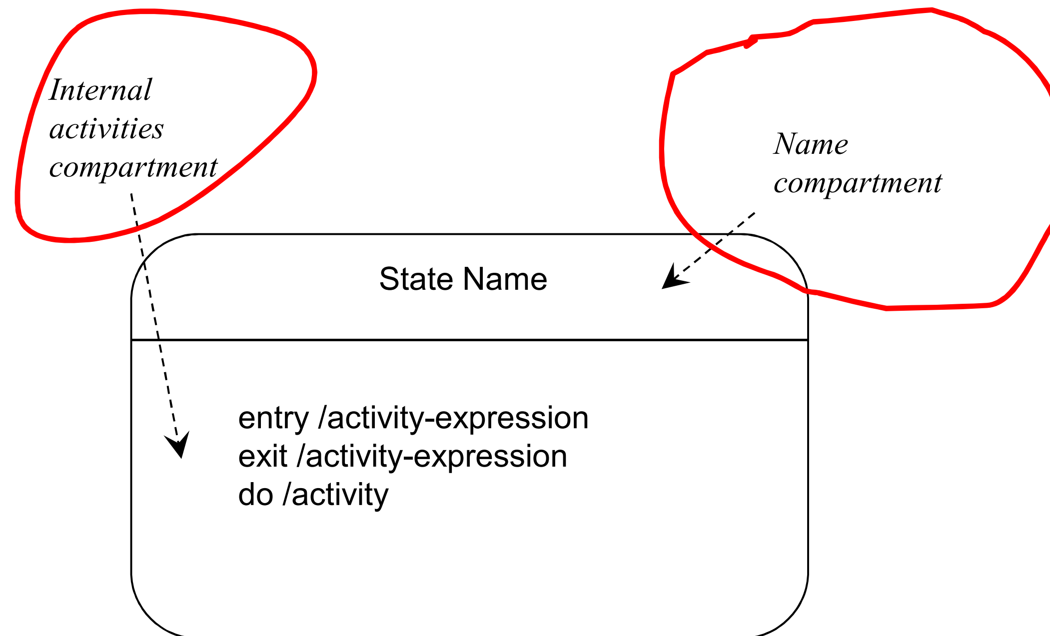
Events



© 2010 Bennett, McRobb and Farmer

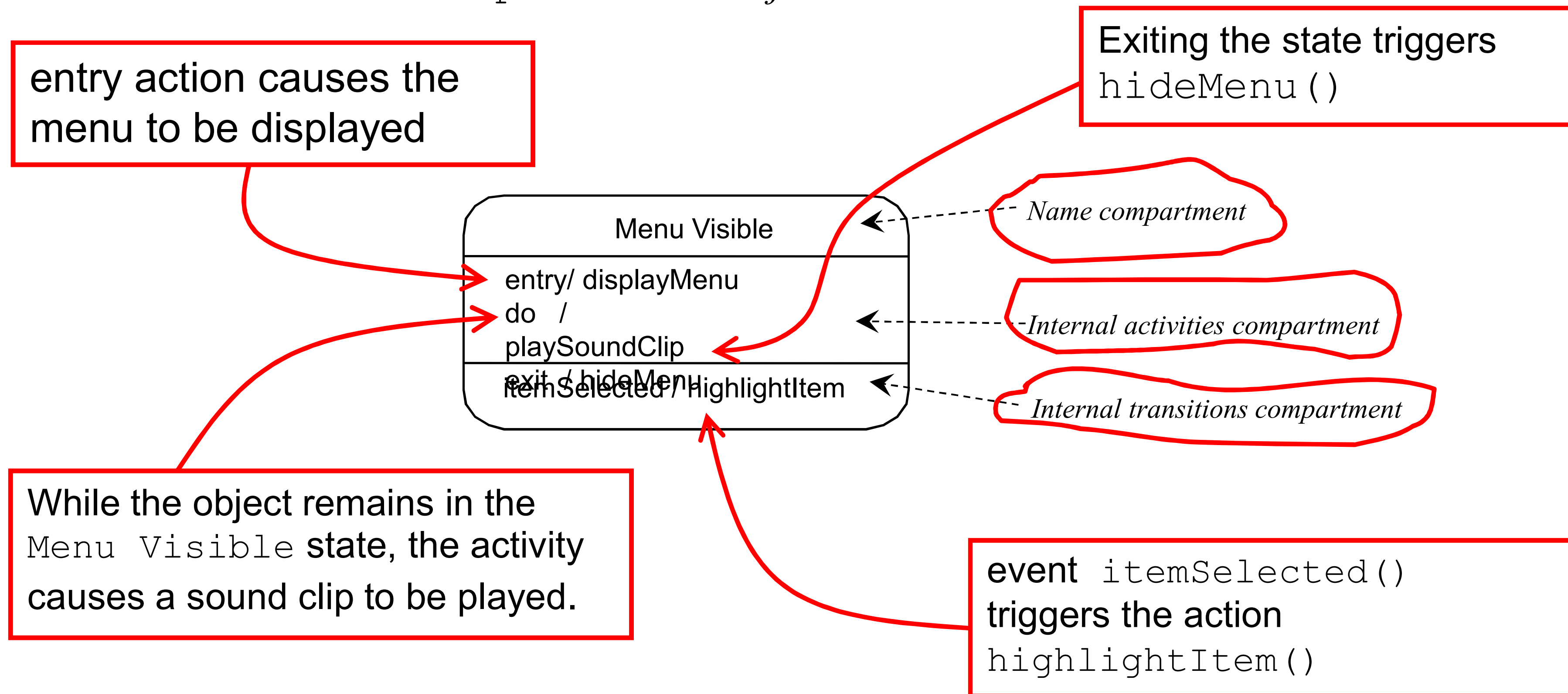
Mc Graw Hill Education

Internal Activities



Mc Graw Hill Education

‘Menu Visible’ State



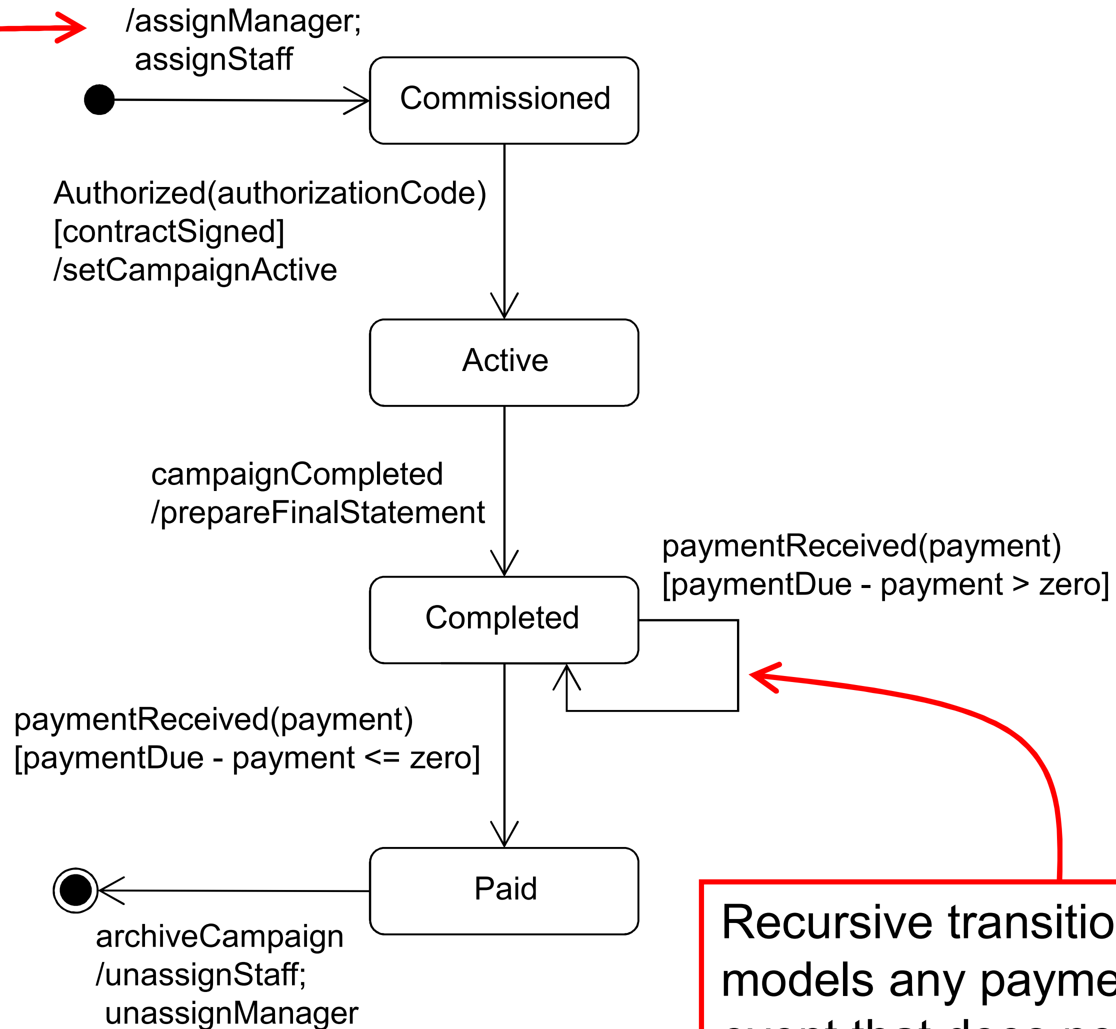
© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

Action-expression assigning manager and staff on object creation

state machine for the class **`Campaign`**.

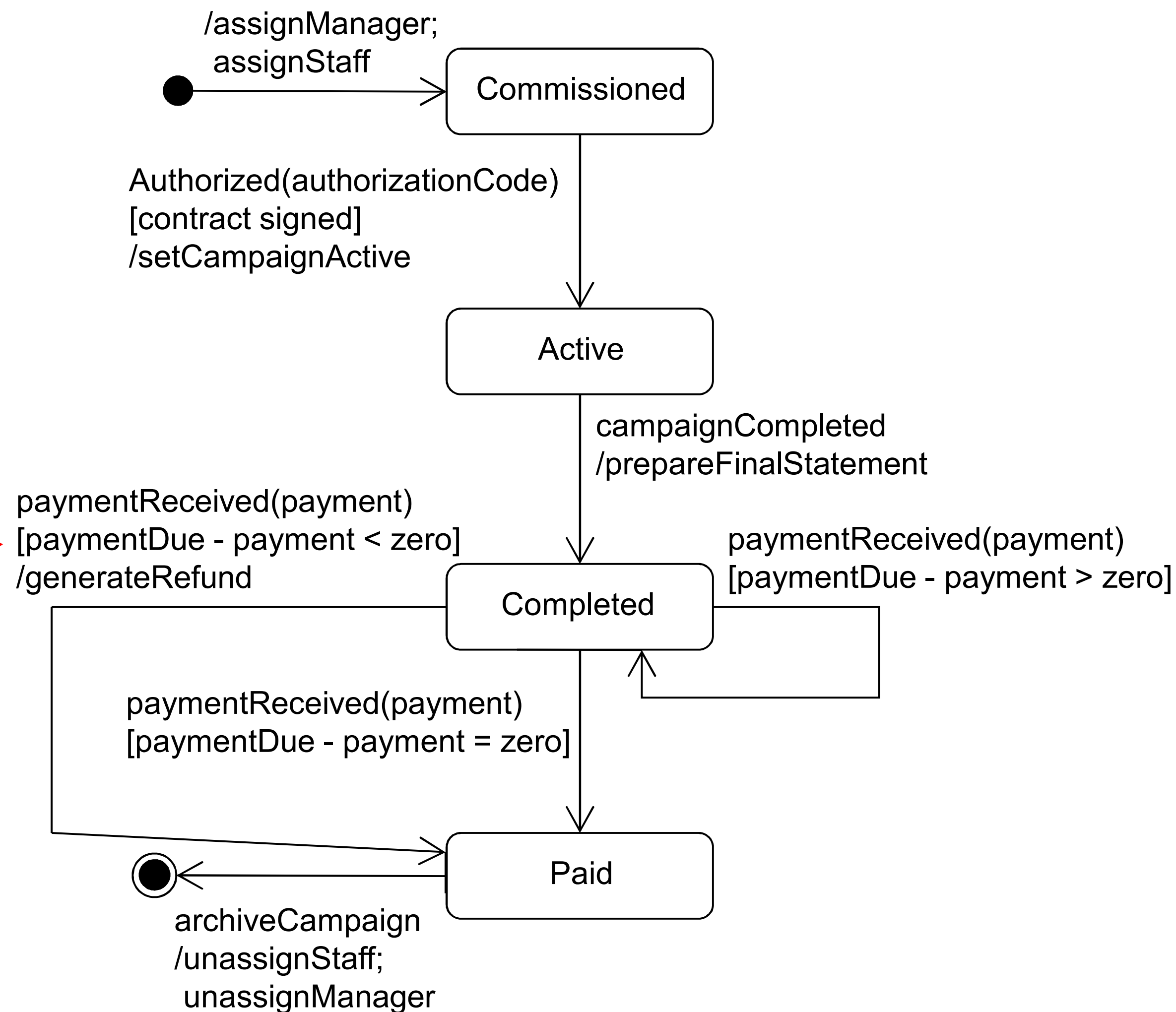
Guard condition ensuring complete payment before entering `Paid`



Recursive transition models any payment event that does not reduce the amount due to zero or beyond.

© 2010 Bennett, McRobb and Farmer

A revised state machine for the class `Campaign`



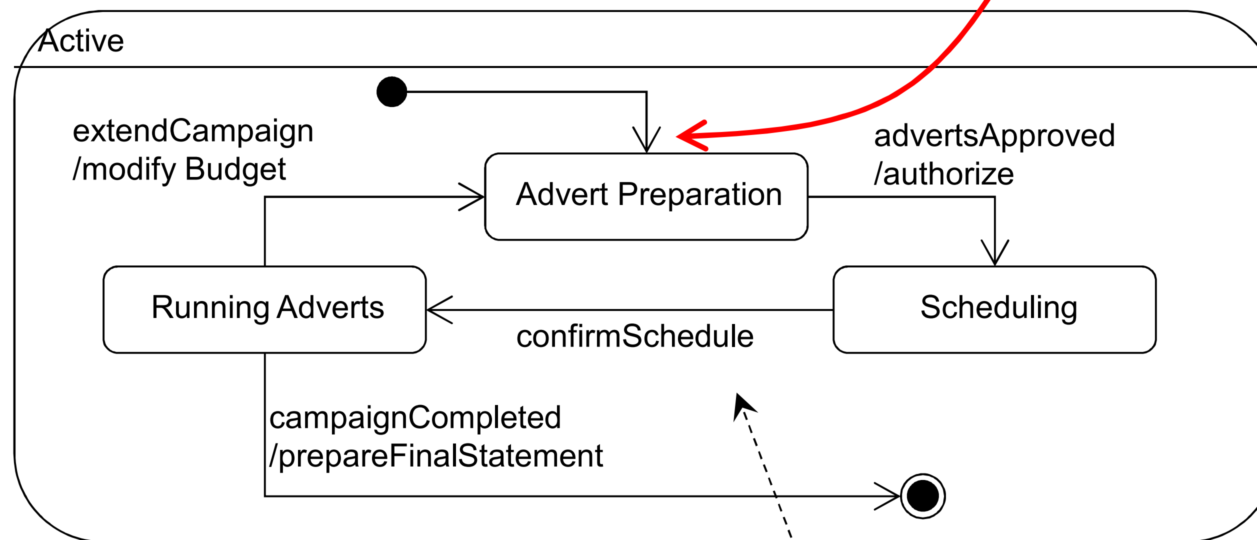
If the user requirements were to change, so that an overpayment is now to result in the automatic generation of a refund, a new transition is added.

© 2010 Bennett, McRobb and Farmer

Nested Substates

The `Active` *state of* `Campaign` *showing nested substates.*

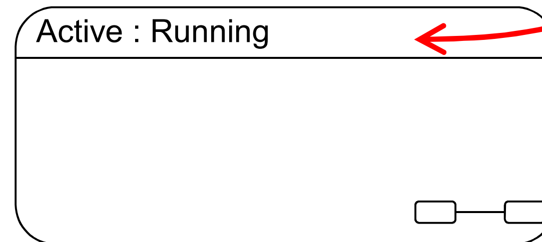
The transition from the initial pseudostate symbol should not be labelled with an event but may be labelled with an action, though it is not required in this example



Decomposition compartment

Nested States

The Active *state of* Campaign *with the detail hidden.*

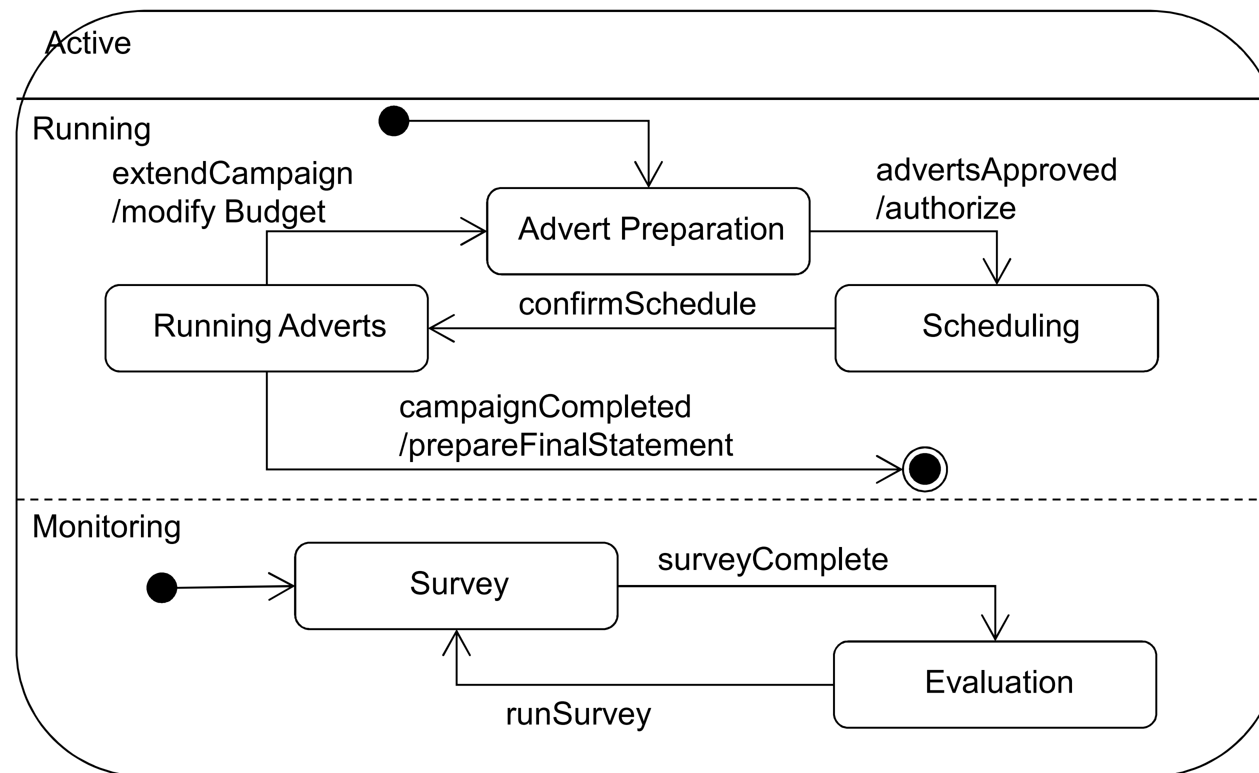


The submachine Running is referenced using the include statement.

Hidden decomposition indicator icon

Mc Graw Hill Education

The Active state with concurrent substates.



© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

Concurrent States

- A transition to a complex state is equivalent to a simultaneous transition to the initial states of each concurrent state machine.
- An initial state must be specified in both nested state machines in order to avoid ambiguity about which substate should first be entered in each concurrent region.
- A transition to the `Active` state means that the `Campaign` object simultaneously enters the `Advert Preparation` and `Survey` states.

© 2010 Bennett, McRobb and Farmer

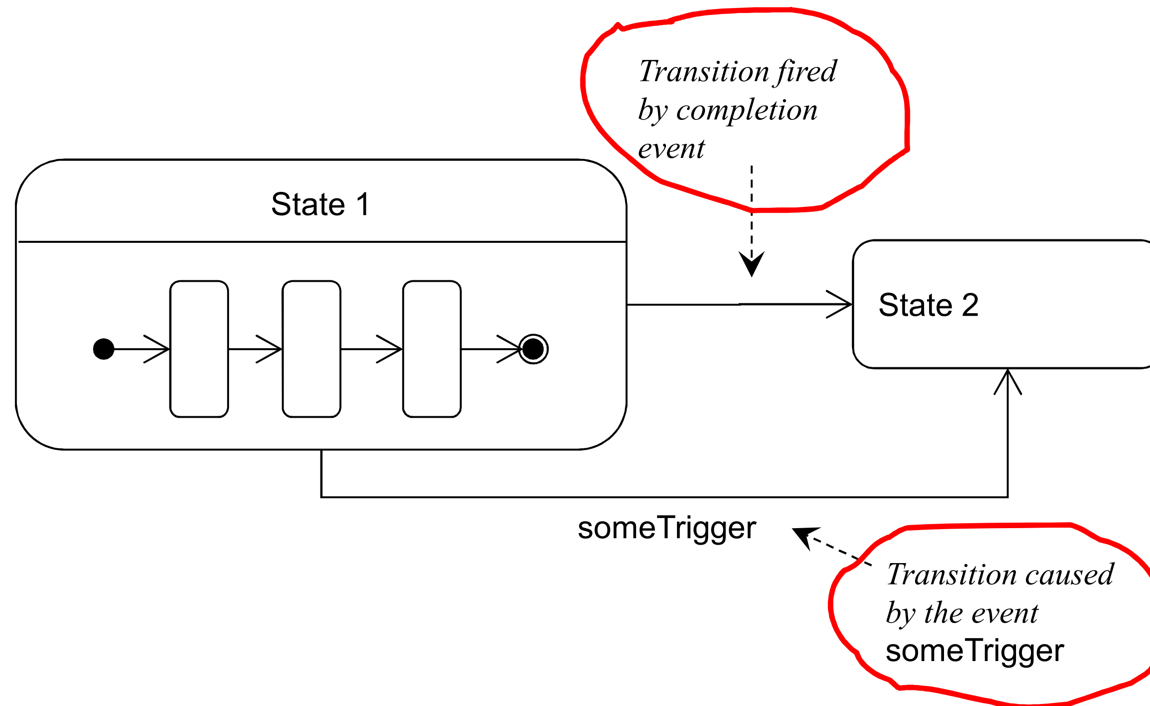
Mc Graw Hill Education

Concurrent States

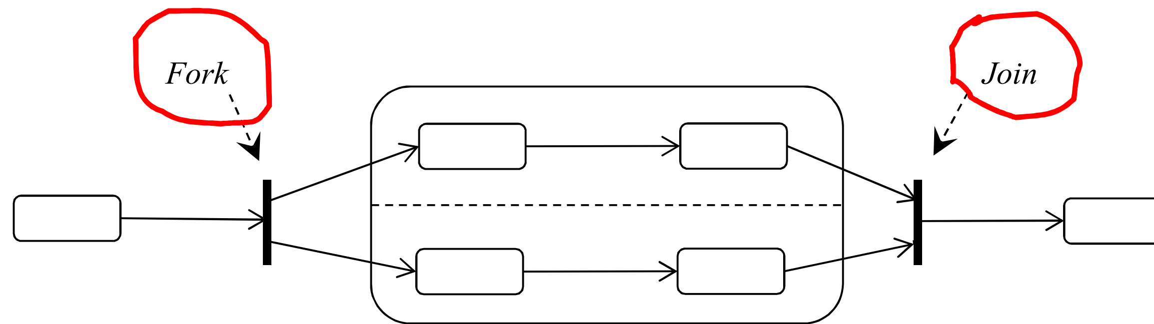
- Once the composite state is entered a transition may occur within either concurrent region without having any effect on the state in the other concurrent region.
- A transition out of the `Active` state applies to all its substates (no matter how deeply nested).

McGraw Hill Education

Completion Event



Synchronized Concurrent Threads.



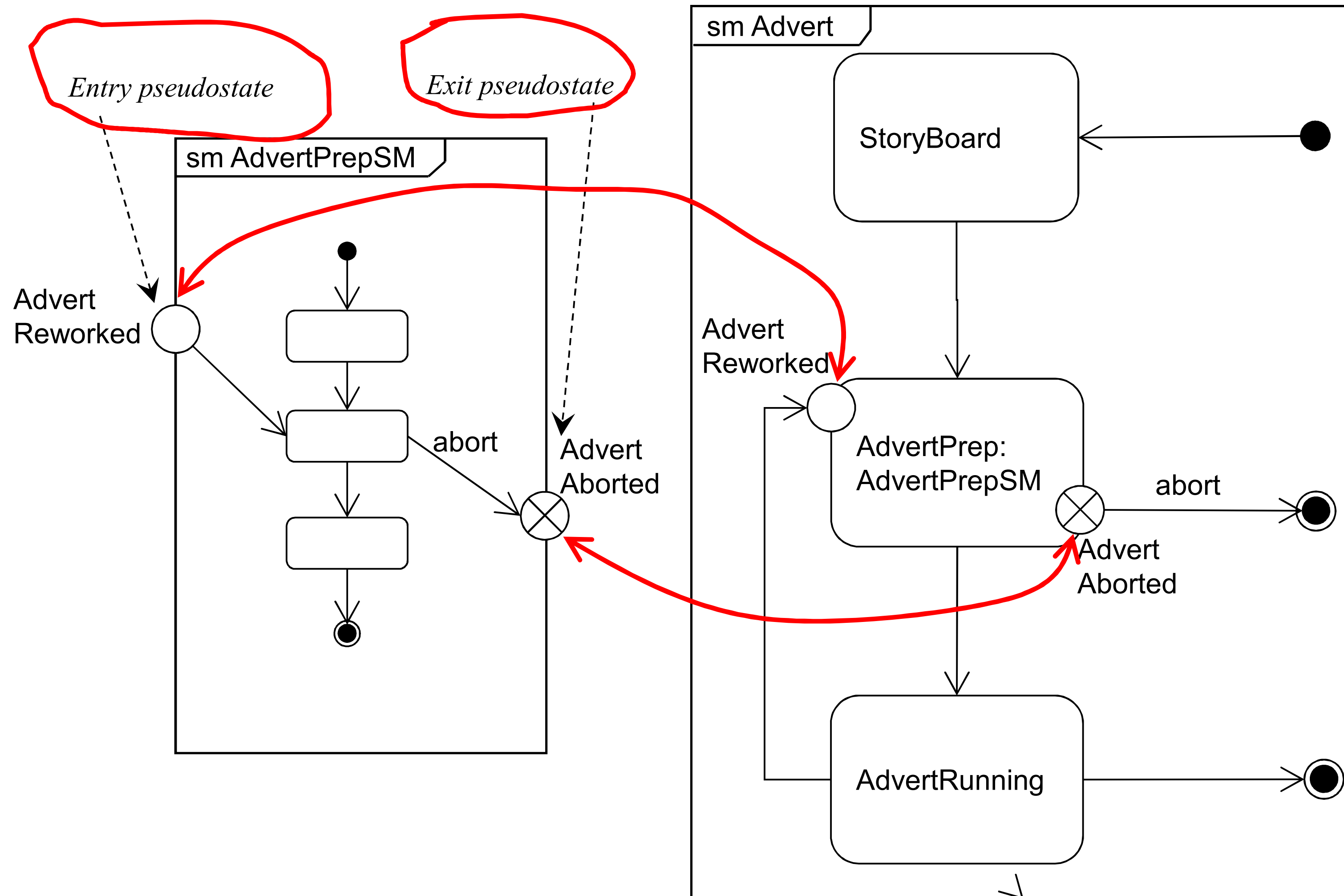
•Explicitly showing how an event triggering a transition to a state with nested concurrent states causes specific concurrent substates to be entered.

•Shows that the composite state is not exited until both concurrent nested state machines are exited.

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

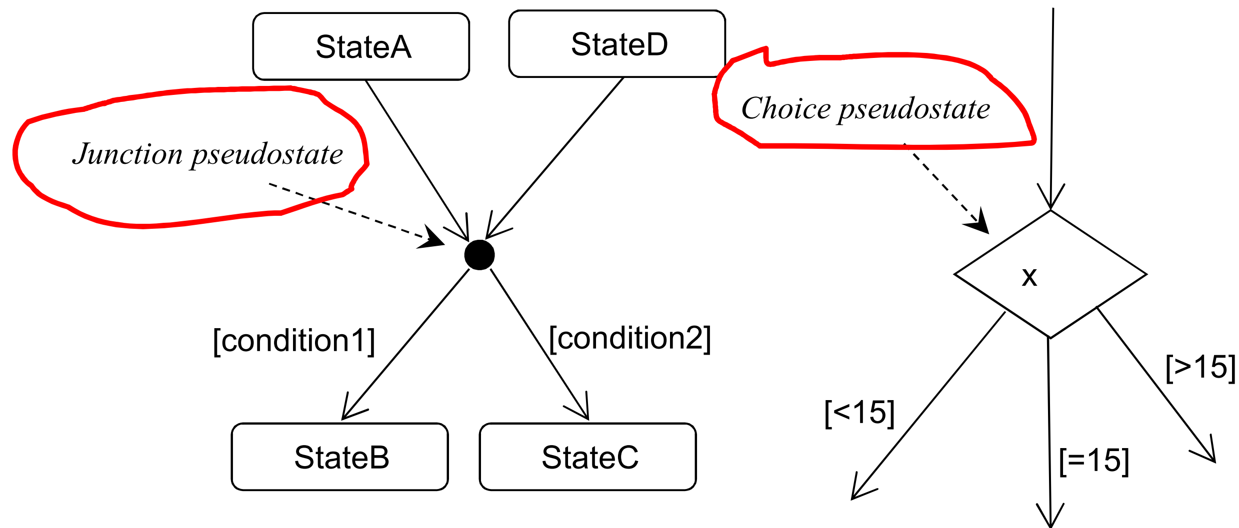
Entry & Exit Pseudostates



© 2010 Bennett, McRobb and Farmer

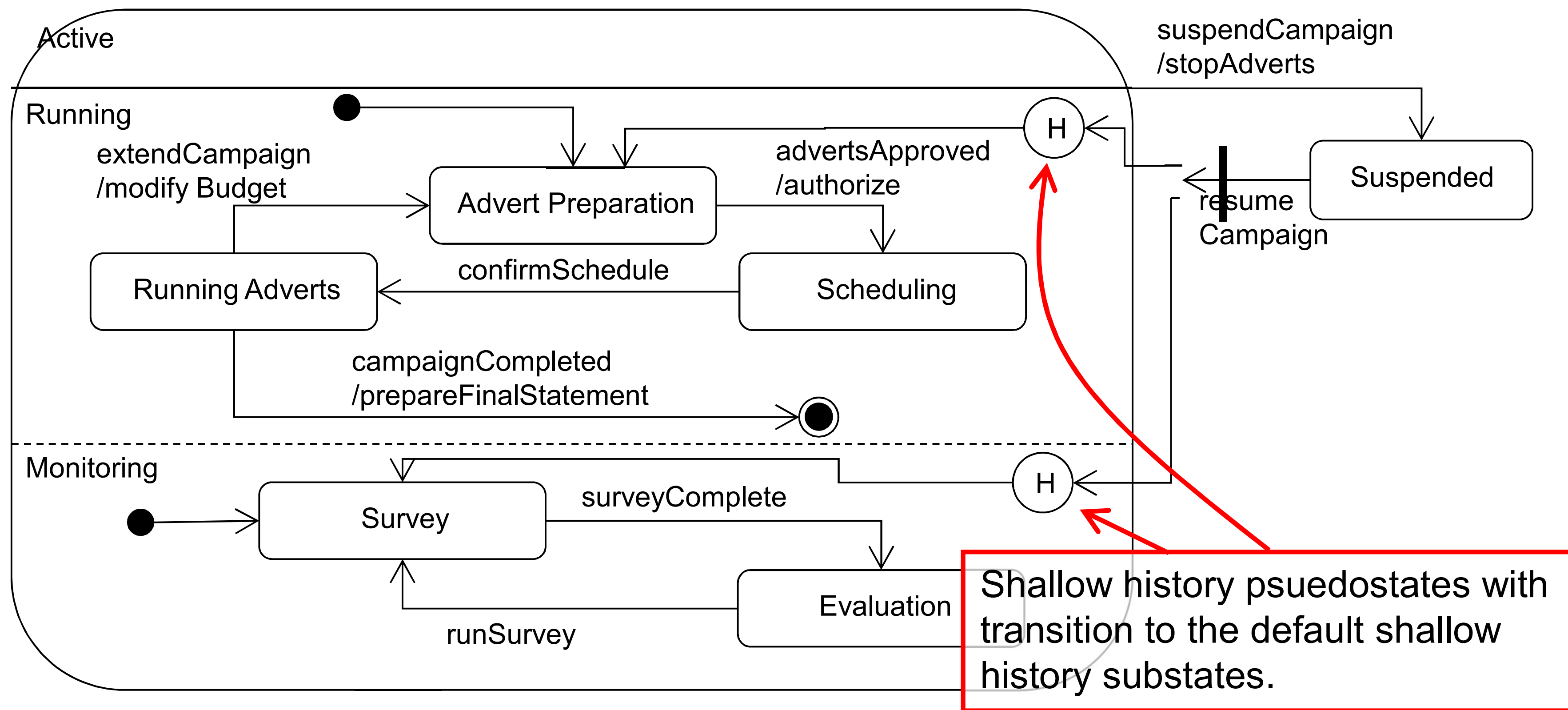
Mc Graw Hill Education

Junction & Choice Pseudostates



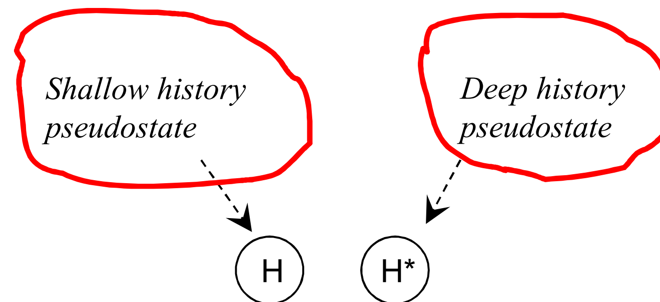
Mc Graw Hill Education

History Pseudostates



Mc Graw Hill Education

History Pseudostates



© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

Preparing state machines

- Two approaches may be used:
  - Behavioural approach
  - Life cycle approach

Allen and Frost (1998)

Mc Graw Hill Education

Behavioural Approach

1. Examine all interaction diagrams that involve each class that has heavy messaging.
2. Identify the incoming messages on each interaction diagram that may correspond to events. Also identify the possible resulting states.
3. Document these events and states on a state machine.
4. Elaborate the state machine as necessary to cater for additional interactions as these become evident, and add any exceptions.

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

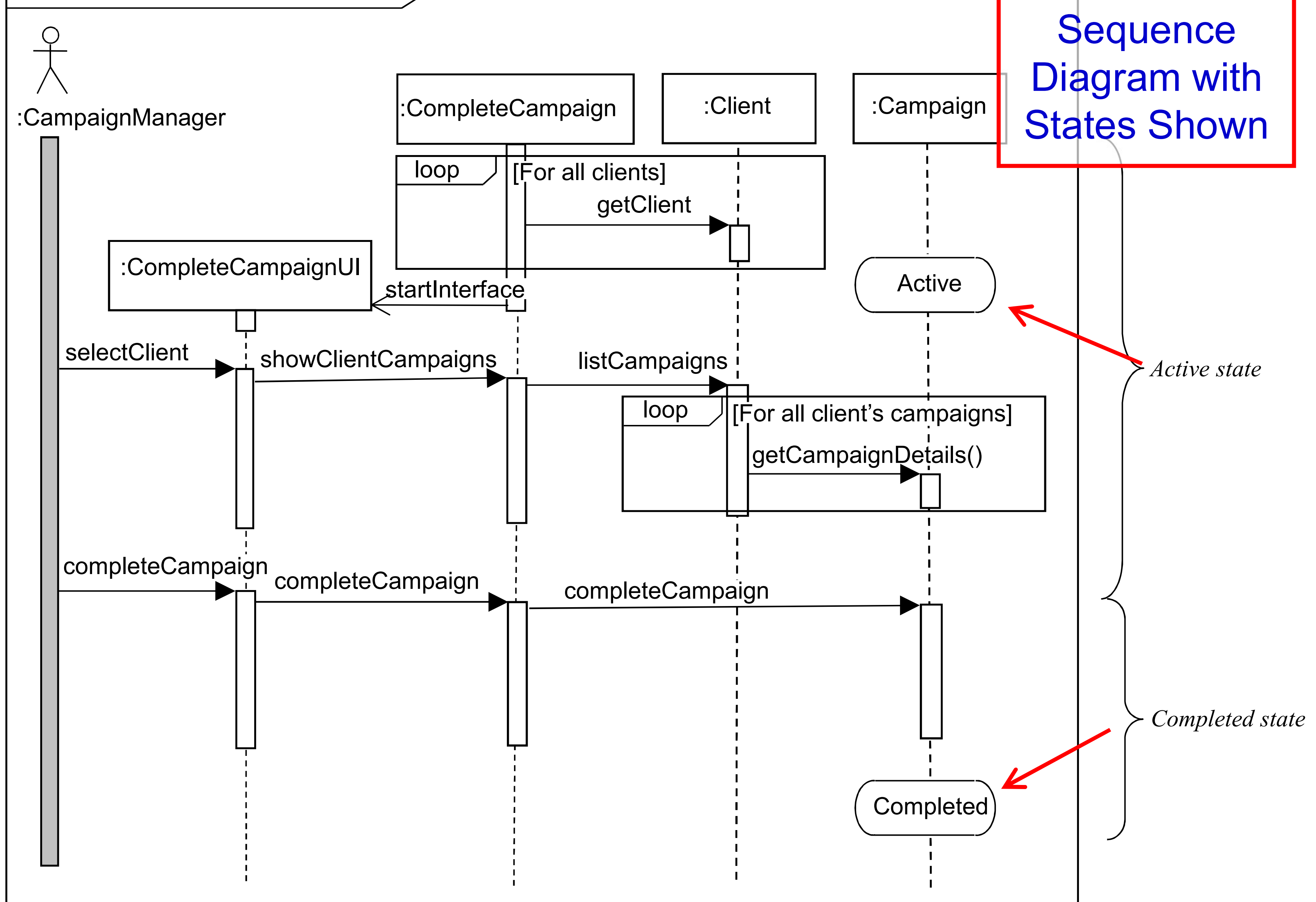
Behavioural Approach

5. Develop any nested state machines (unless this has already been done in an earlier step).
6. Review the state machine to ensure consistency with use cases. In particular, check that any constraints that are implied by the state machine are appropriate.

© 2010 Bennett, McRobb and Farmer

Behavioural Approach

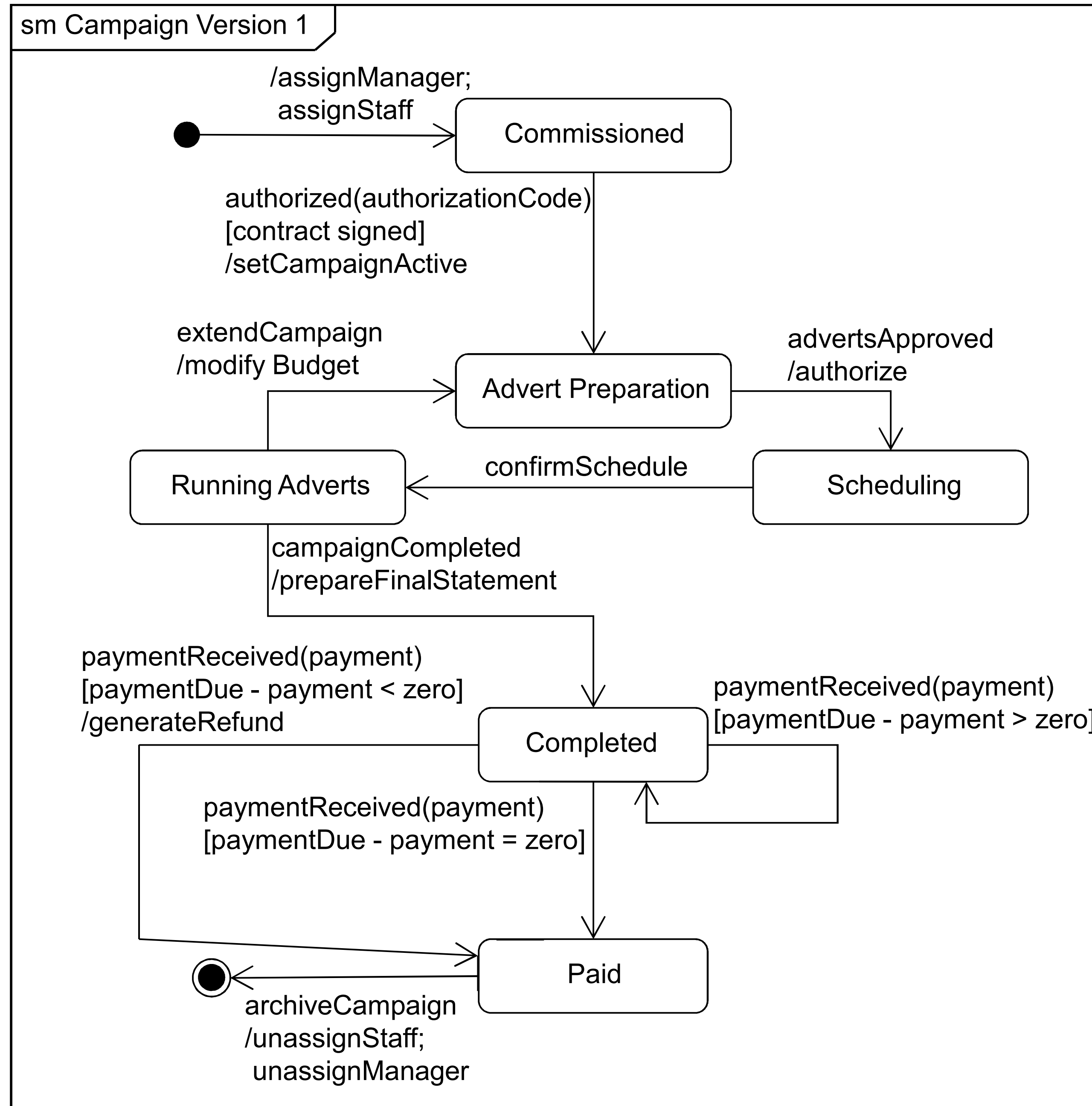
7. Iterate steps 4, 5 and 6 until the state machine captures the necessary level of detail.
8. Check the consistency of the state machine with the class diagram, with interaction diagrams and with any other state machines and models.



© 2010 Bennett, McRobb and Farmer

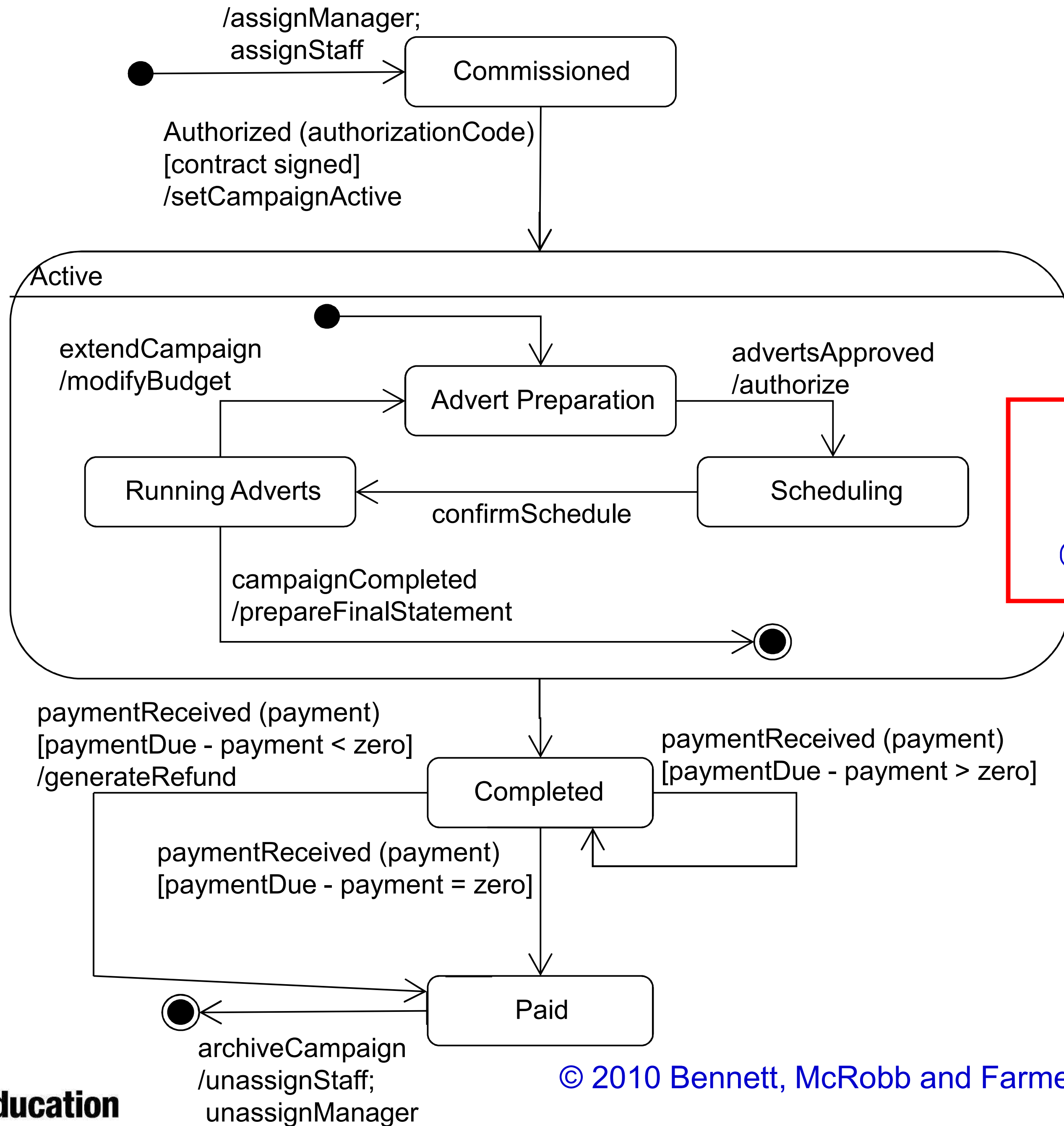
Mc Graw Hill Education

Initial state machine for the Campaign class—a behavioral approach.



© 2010 Bennett, McRobb and Farmer

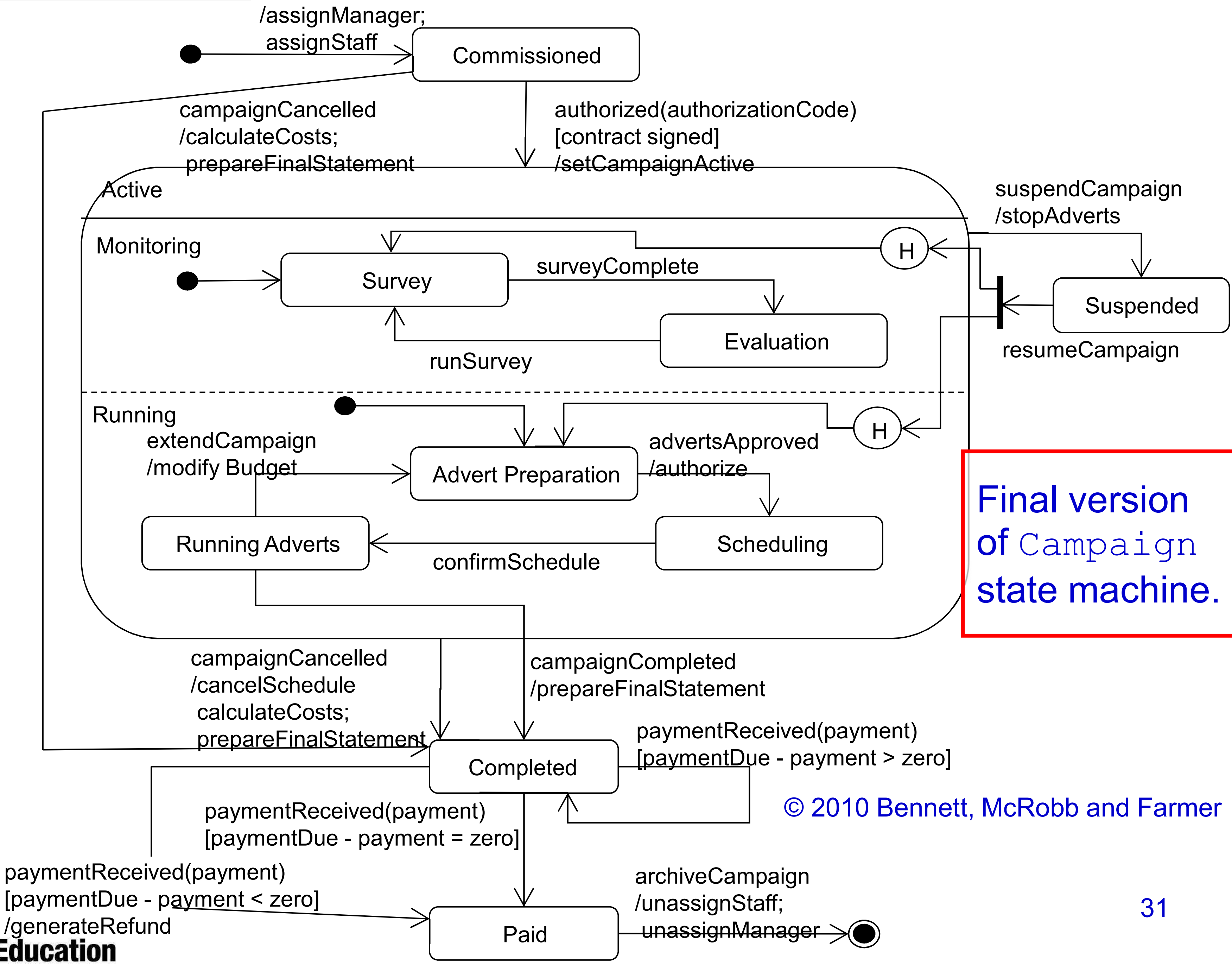
Mc Graw Hill Education



Revised state machine for the `Campaign` class.

Mc Graw Hill Education

© 2010 Bennett, McRobb and Farmer



Final version of `Campaign` state machine.

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

Life Cycle Approach

- Consider the life cycles for objects of each class.
- Events and states are identified directly from use cases and from any other requirements documentation that happens to be available.
- First, the main system events are listed.
- Each event is then examined in order to determine which objects are likely to have a state dependent response to it.

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

Life Cycle Approach Steps

1. Identify major system events.
2. Identify each class that is likely to have a state dependent response to these events.
3. For each of these classes produce a first-cut state machine by considering the typical life cycle of an instance of the class.
4. Examine the state machine and elaborate to encompass more detailed event behaviour.

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

Life Cycle Approach Steps

5. Enhance the state machine to include alternative scenarios.
6. Review the state machine to ensure that is consistent with the use cases. In particular, check that the constraints that the state machine implies are appropriate.
7. Iterate through steps 4, 5 and 6 until the state machine captures the necessary level of detail.
8. Ensure consistency with class diagram and interaction diagrams and other state machines.

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

Life Cycle Approach

- Less formal than the behavioural approach in its initial identification of events and relevant classes.
- Often helpful to use a combination of the two, since each provides checks on the other.

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

Protocol State Machines

- UML 2.0 introduced a distinction between protocol and behavioural state machines.
- All the state machines so far have been behavioural.
- Protocol state machines differ in that they only show all the legal transitions with their pre- and post-conditions.

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

Protocol State Machines

- The states of a protocol state machine cannot have
  - entry, exit or do activity sections
  - deep or shallow history states
- All transitions must be protocol transitions

Protocol State Machines

- The syntax for a protocol transition label is as follows.

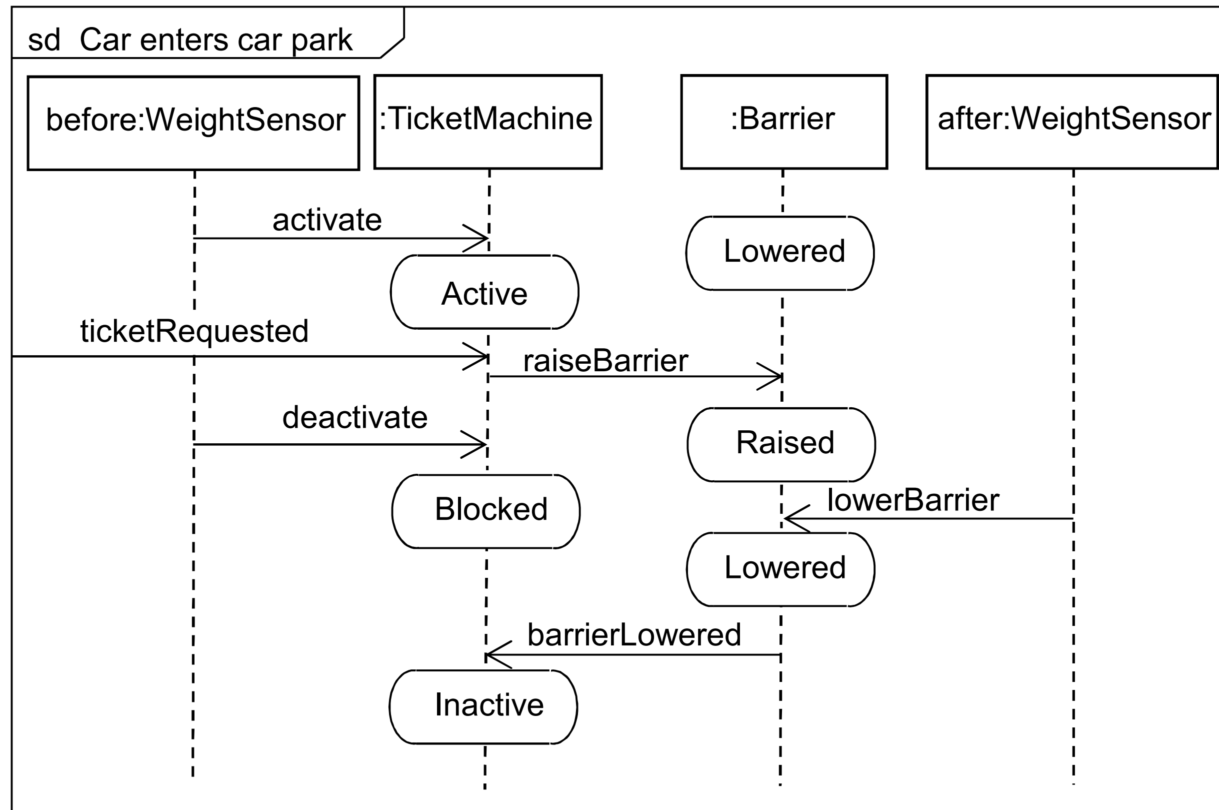
```
'[' pre-condition ']' trigger '/'
'[' post-condition ']'
```

- Unlike behavioural transitions protocol transitions do not have activity expressions.

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

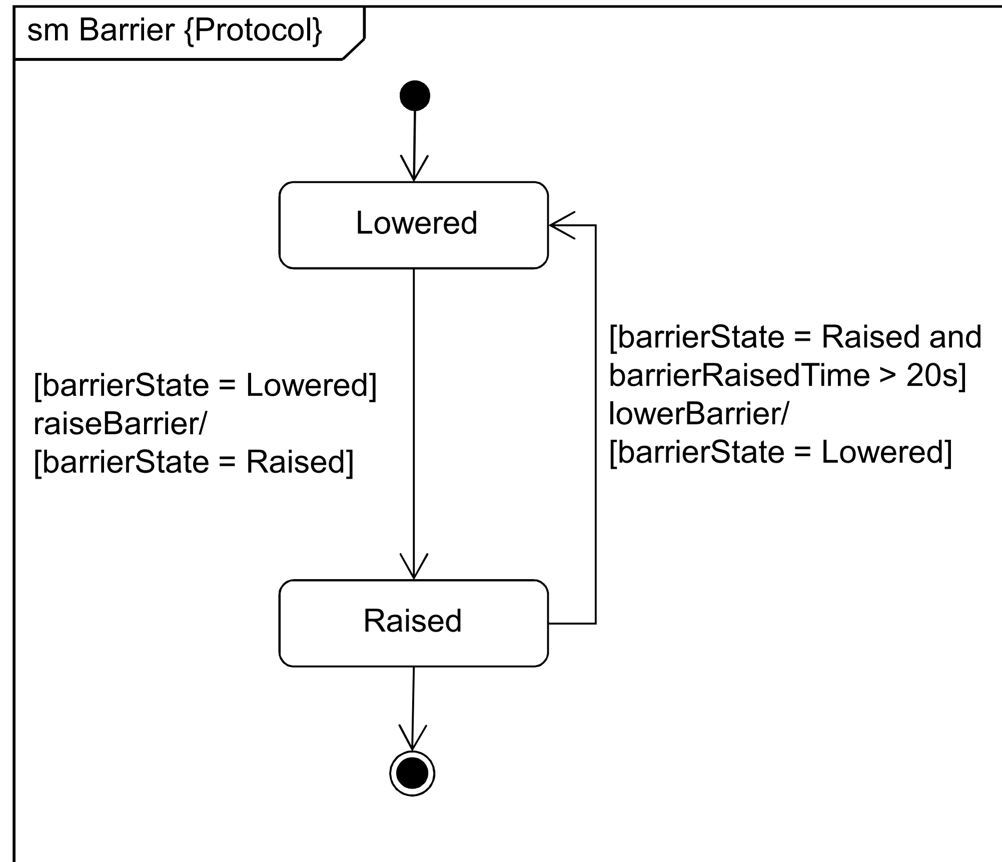
Sequence Diagram for Protocol State Machine Example



© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

Protocol State Machine



© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

Consistency Checking

- Every event should appear as an incoming message for the appropriate object on an interaction diagram(s).
- Every action should correspond to the execution of an operation on the appropriate class, and perhaps also to the dispatch of a message to another object.
- Every event should correspond to an operation on the appropriate class (but note that not all operations correspond to events).
- Every outgoing message sent from a state machine must correspond to an operation on another class.

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

Consistency Checking

- Consistency checks are an important task in the preparation of a complete set of models.
- Highlights omissions and errors, and encourages the clarification of any ambiguity or incompleteness in the requirements.

Summary

In this lecture you have learned about:

- how to identify requirements for control in an application;
- how to model object life cycles using state machines;
- how to develop state machine diagrams from interaction diagrams;
- how to model concurrent behaviour in an object;
- how to ensure consistency with other UML models.

© 2010 Bennett, McRobb and Farmer

Mc Graw Hill Education

References

- UML 2.2 Superstructure Specification (OMG, 2009)
- Douglass (2004)

(For full bibliographic details, see Bennett, McRobb and Farmer)

© 2010 Bennett, McRobb and Farmer

McGraw Hill Education