CS4125
SYSTEMS ANALYSIS
SPRING SEMESTER 2010-2011

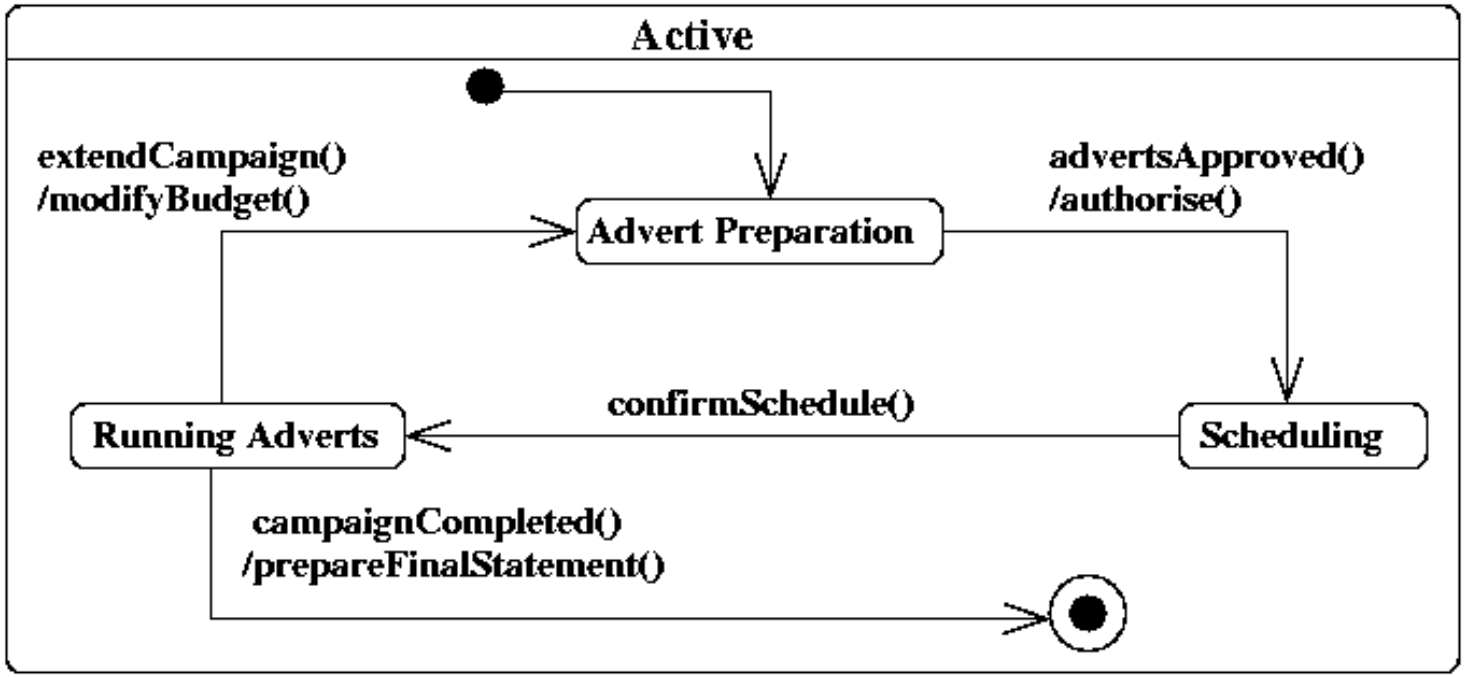
J.J. Collins
Dept of CSIS
University of Limerick

Lecture M - W8L2

Analysis: Specifying Control – State Charts Part 2

State Diagram: Composite States

- When the state behaviour for an object or interaction is complex, it may be necessary to represent it at different levels of detail.
- In the statechart for Campaign, the Active state is comprised of three substates: Advert Preparation, Scheduling and Running Adverts.
- Diagram shows a single state that contains a nested state diagram.

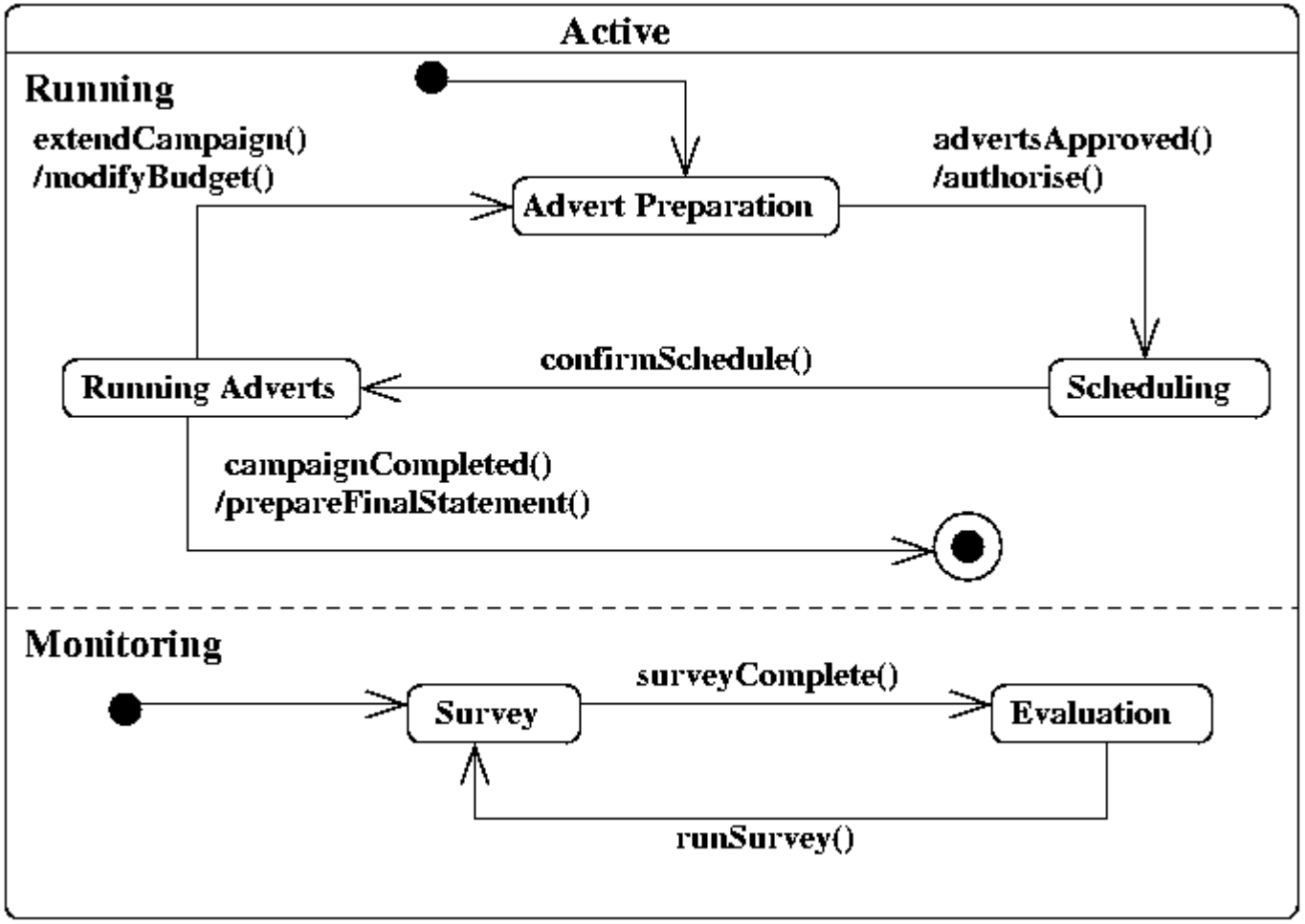


The Active state of Campaign showing nested substates.

Concurrent States

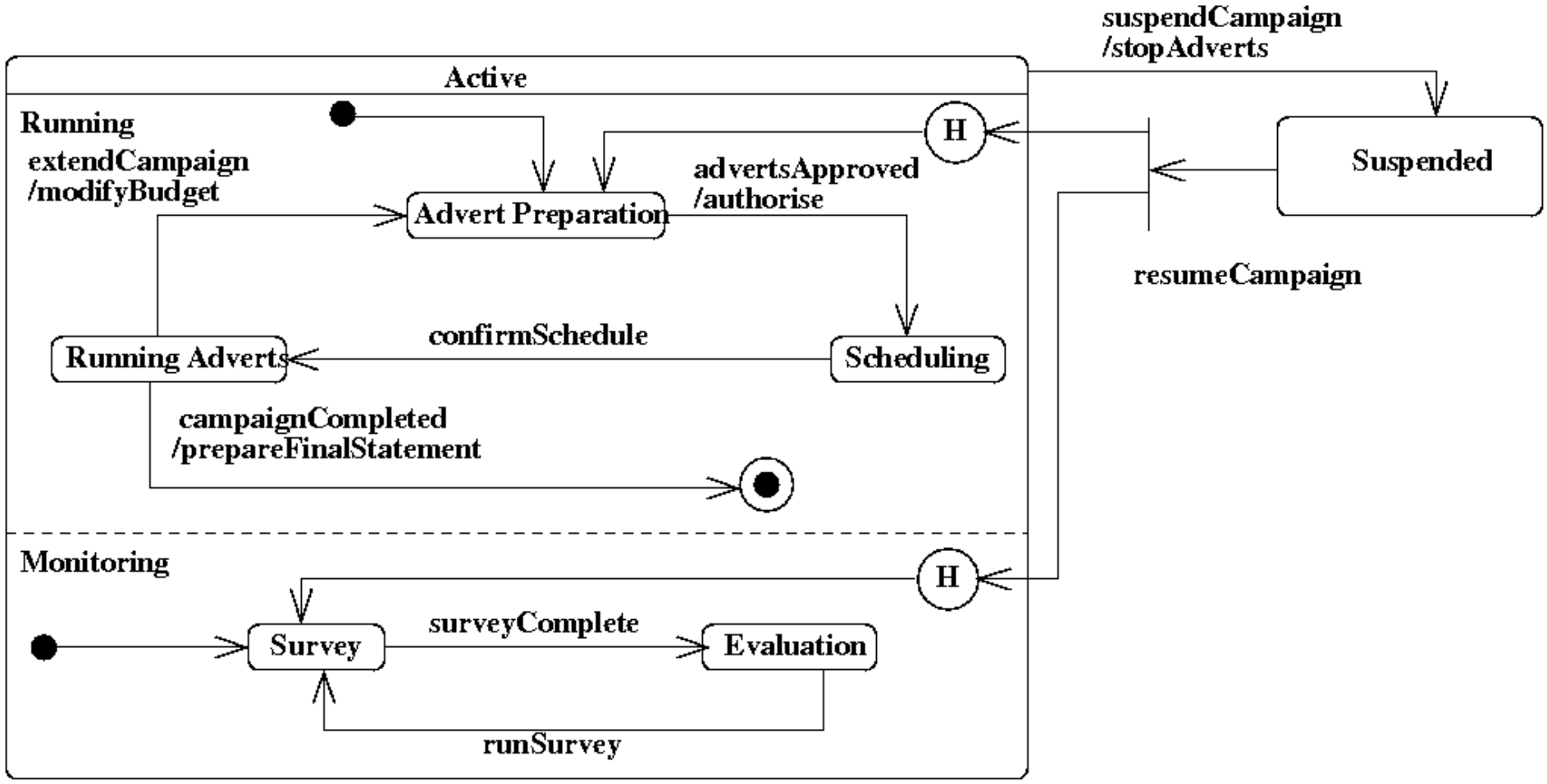
- Objects can have concurrent states.
- The behaviour of the object can be regarded as a product of two distinct sets of substates.
- Each state of which can be entered and exited independently of substates in the other set.
- Suppose that a campaign is surveyed and evaluated while it is also Active.
- Model this by splitting the Active state into two concurrent nested statecharts, Running and Monitoring.

Concurrent States



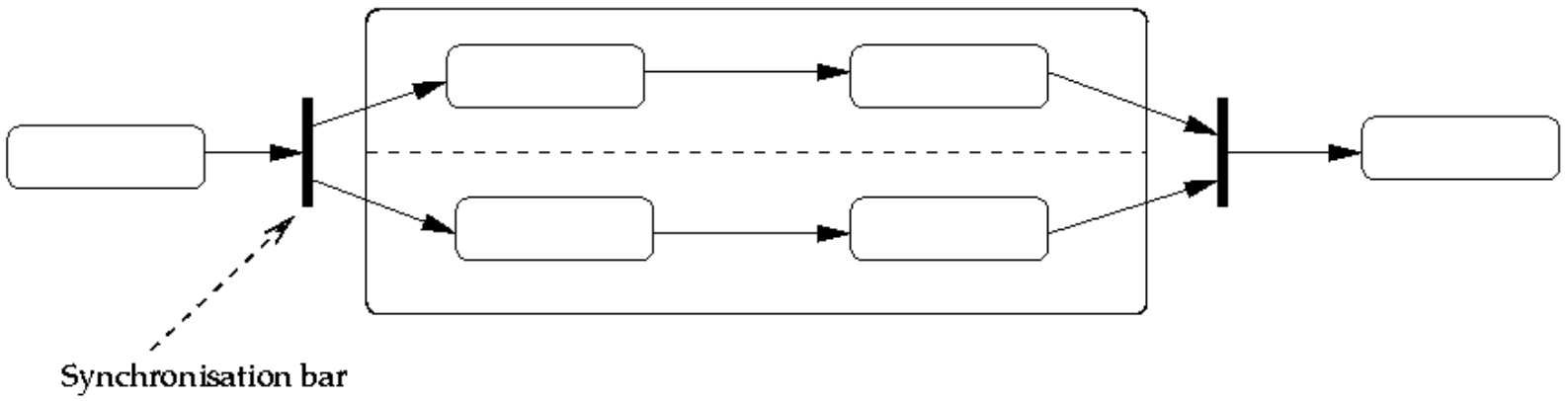
The Active state with concurrent substates.

UML: State Diagrams - Using History Pseudostates



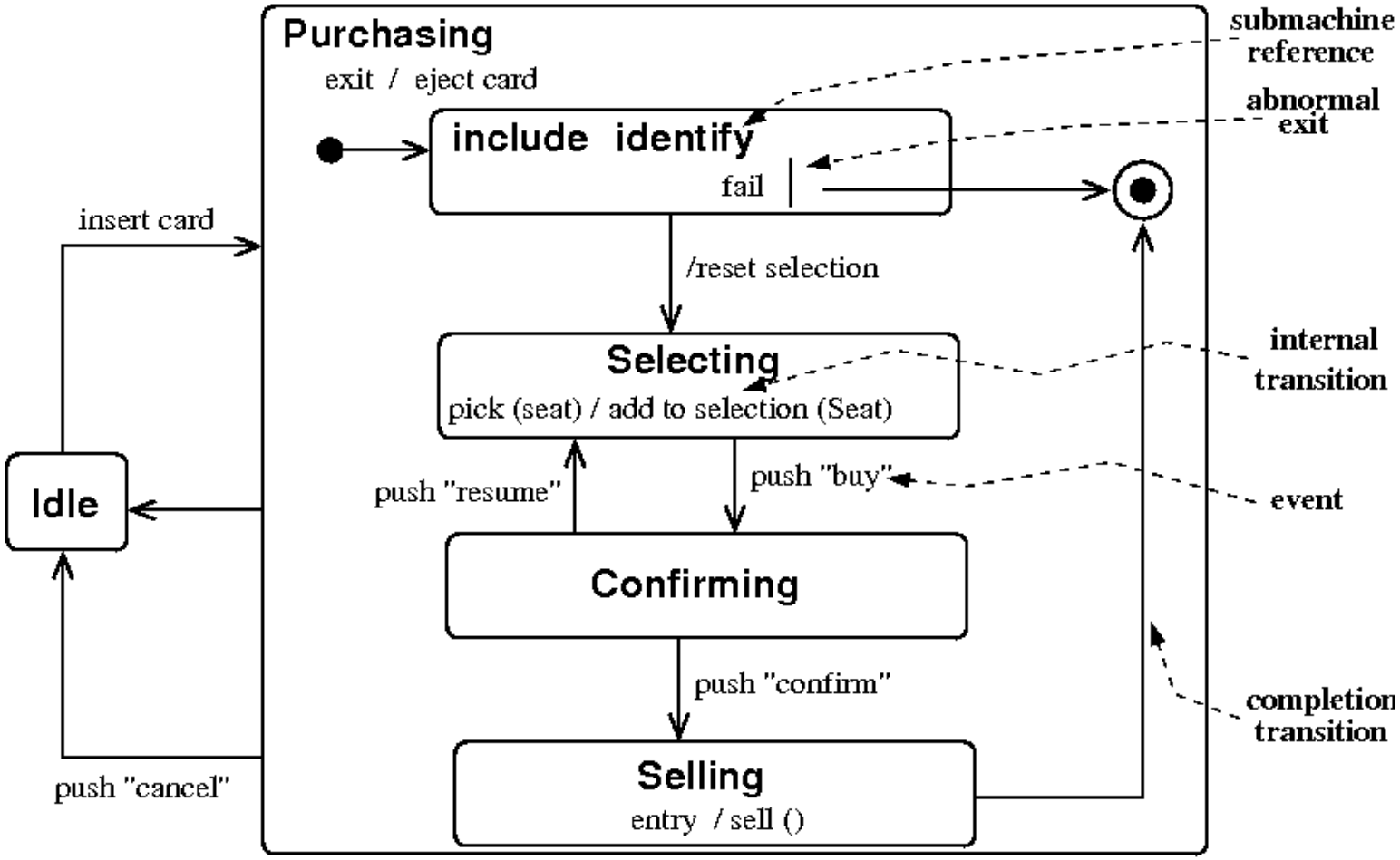
Concurrent States

- Figure shows the use of synchronisation bars to show explicitly the transition to and from state of nested concurrent substates.
- In this case, the super state is not exited until both concurrent nested statecharts are exited.



Synchronised concurrent threads.

Statechart



An example of a state machine

Preparing Statecharts

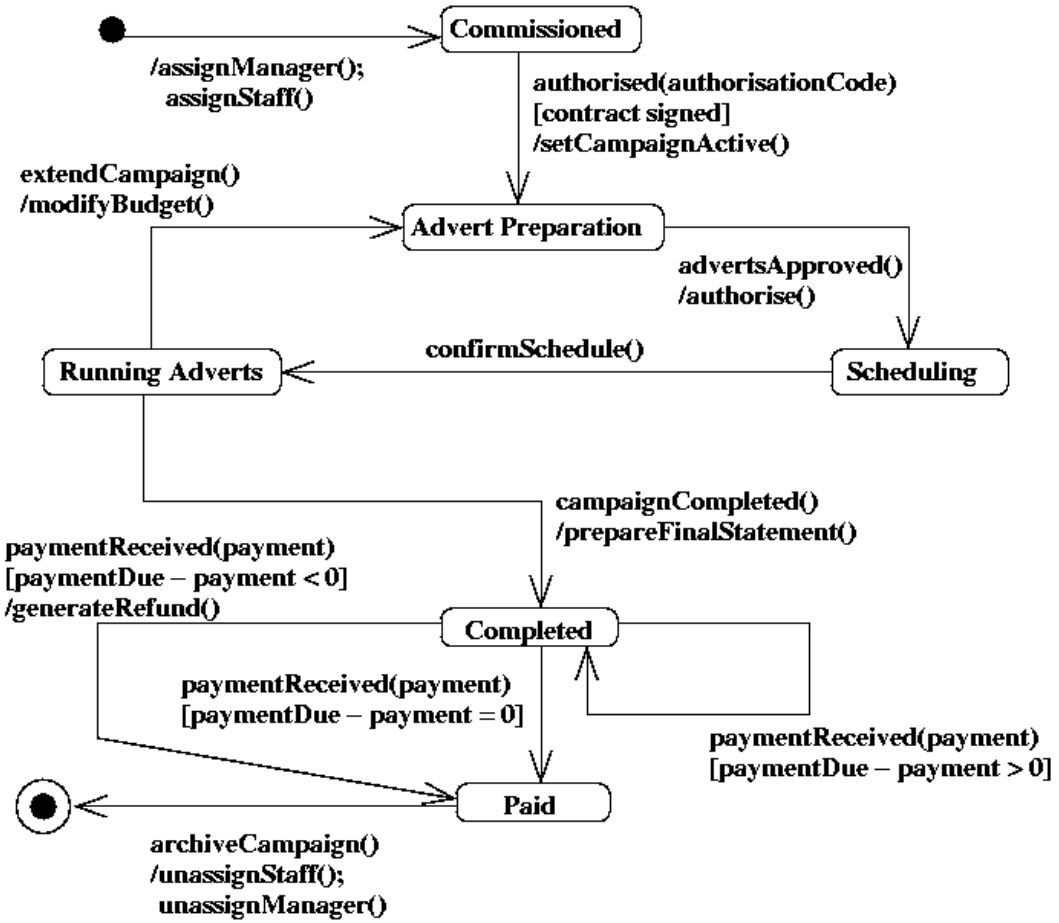
- Allen and Frost (1998) describe the use of interaction diagrams to develop a statechart as a behavioural approach.
- Interaction diagrams show the messages that an object receives during the execution of a use case.
- Messages are events that may cause a transition to fire.
- Query (e.g. listAdverts()) and Get (e.g. getTitle()) messages generally do not cause transition to fire.
- Even if a message changes an attribute value, state may remain unchanged e.g. a message receivePayment() to a Campaign object will only cause a change of state to Paid if full payment received.

Sequence of Steps: 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. Identify possible resulting states.
3. Document these on a statechart.
4. Elaborate the statechart to cater for additional interactions and add any exceptions.
5. Develop nested statecharts.
6. Review statechart to ensure consistency with use cases. Ensure that any constraints implied by the statechart are appropriate.
7. Iterate through steps 4, 5 and 6.
8. Check the statechart for consistency with class and interaction diagrams.

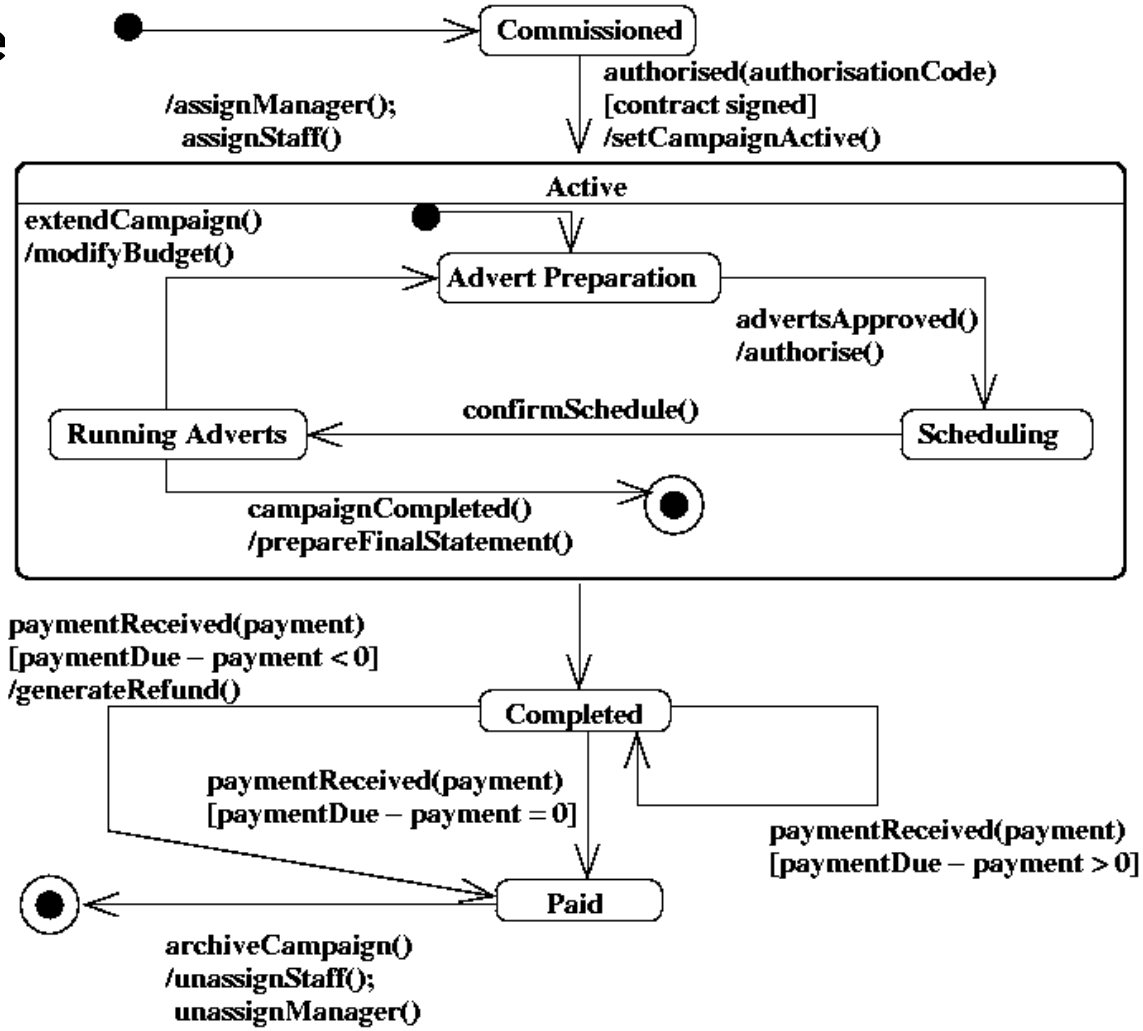
Example

- Initial statechart for the Campaign class - a behavioural approach.



Example

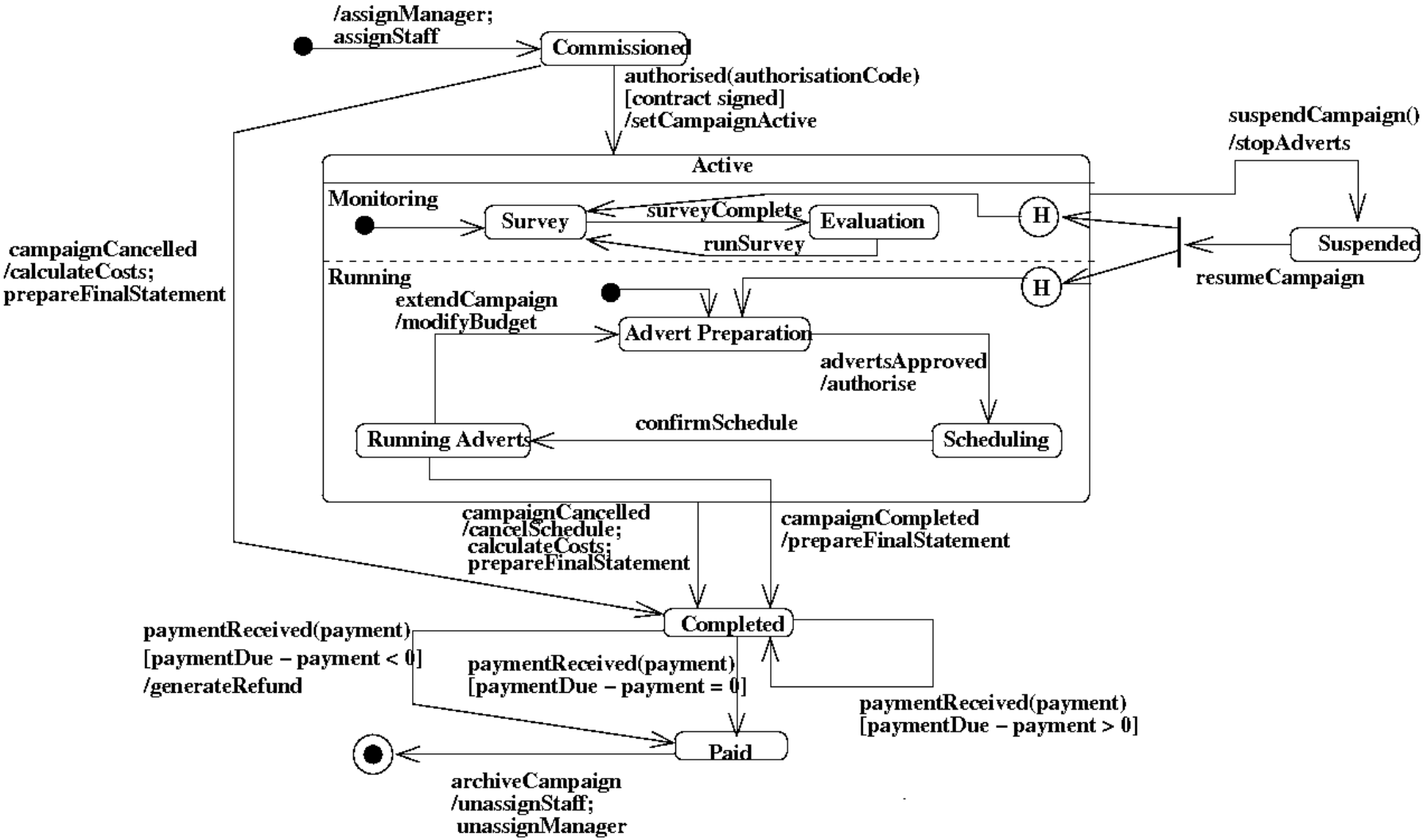
- Revise



Checking Consistency in Statecharts

- Every event should appear as an incoming message for the appropriate object on an interaction diagram.
- Every event should correspond to an operation on the appropriate class.
- Every action should correspond to the passing of a message from the appropriate object on an interaction diagram.
- Every outgoing message (send-clause) sent from a statechart must correspond to an operation on another class.

Example


/assignManager;
assignStaff
Commissioned
authorised(authorisationCode)
[contract signed]
/setCampaignActive
suspendCampaign()
/stopAdverts
Active
Monitoring
Survey
surveyComplete
Evaluation
runSurvey
H
Suspended
resumeCampaign
campaignCancelled
/calculateCosts;
prepareFinalStatement
Running
extendCampaign
/modifyBudget
Advert Preparation
advertsApproved
/authorise
confirmSchedule
Running Adverts
Scheduling
campaignCancelled
/cancelSchedule;
calculateCosts;
prepareFinalStatement
campaignCompleted
/prepareFinalStatement
Completed
paymentReceived(payment)
[paymentDue – payment < 0]
/generateRefund
paymentReceived(payment)
[paymentDue – payment = 0]
paymentReceived(payment)
[paymentDue – payment > 0]
Paid
archiveCampaign
/unassignStaff;
unassignManager

Notes:

- In UML 2.0 a distinction has been made between behavioural state machines and protocol state machines.
- Protocol state machines used to specify usage protocols for classes, interfaces, and ports.
- A submachine refers to either a composite state, or the nested state chart enclosed by a state – depends on context.
- Did not discuss:
  - Entry and exit pseudostates
  - Junction and choice pseudostates

Reading

- Chapter 11 in Bennett et al. or
- Chapters11 and 12 in Stevens and Pooley