ICS 3105
OBJECT ORIENTED SOFTWARE ENGINEERING

Chapter 5.5

State Diagrams

Learning Outcomes

- By the end of this chapter, the learner should be able to:
  - Define states, events and transition.
  - Define conditions and guard conditions.
  - Draw state diagrams with nested states and without.

Introduction

- A state diagram, also known as a state machine diagram, is another way of expressing dynamic information about a system.
- It is used to describe the externally visible behavior of a system or of an individual object.

State diagrams

- At any given point in time, the system or object is said to be in a certain state.
- It remains in this state until an event occurs that causes it to change state.
- Being in a state means that an object behaves in a specific way in response to any events that occur.

State diagrams

- Software engineers represent a state using a rounded rectangle that contains the name of the state.
- Several different types of event can cause the system to change from one state to another.

Transition and state

- In each state, the system behaves in a different way.
- A transition represents a change of state in response to an event, and is considered to occur instantaneously i.e. takes no time.

Transition and events

- A transition can be drawn using an arrow connecting two states.
- A label can also be shown on a transition; this represents the event that causes the change of state.

Other state diagram symbols

- There are two other special symbols that can appear on a state diagram:
  - A black circle represents the start state.
  - A black circle with a ring around it represents an end state.

Start state

- When the system or object starts running, it immediately takes a transition from the start state to a regular state.
- There should be only one start state in each top-level state diagram, and there should be only one unlabelled transition pointing out of the start state.

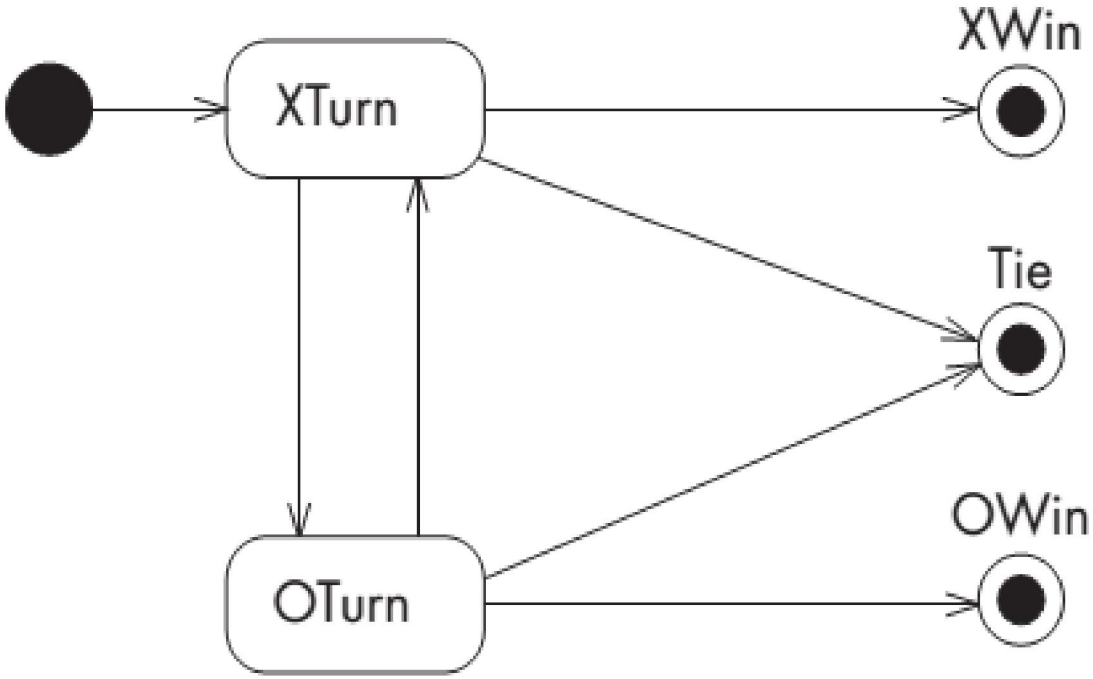
End state

- The system or object finishes its work when such a state is reached.
- There can be more than one end state in a state diagram.
- The symbol is supposed to resemble a target.

Example

- A game of tic-tac-toe, also known as noughts and crosses. Since player X always goes first, the initial transition from the start state points to the ‘X Turn’ state. From then on, the game alternates between ‘X Turn’ and ‘O Turn’ states, until the game ends. There can be three possible outcomes of the game, represented by the three end states: X can win, O can win or there can be a tie. The ‘Tie’ end state can be reached from both ‘X Turn’ state and ‘O Turn’ state.

State diagram for tic-tac-toe game



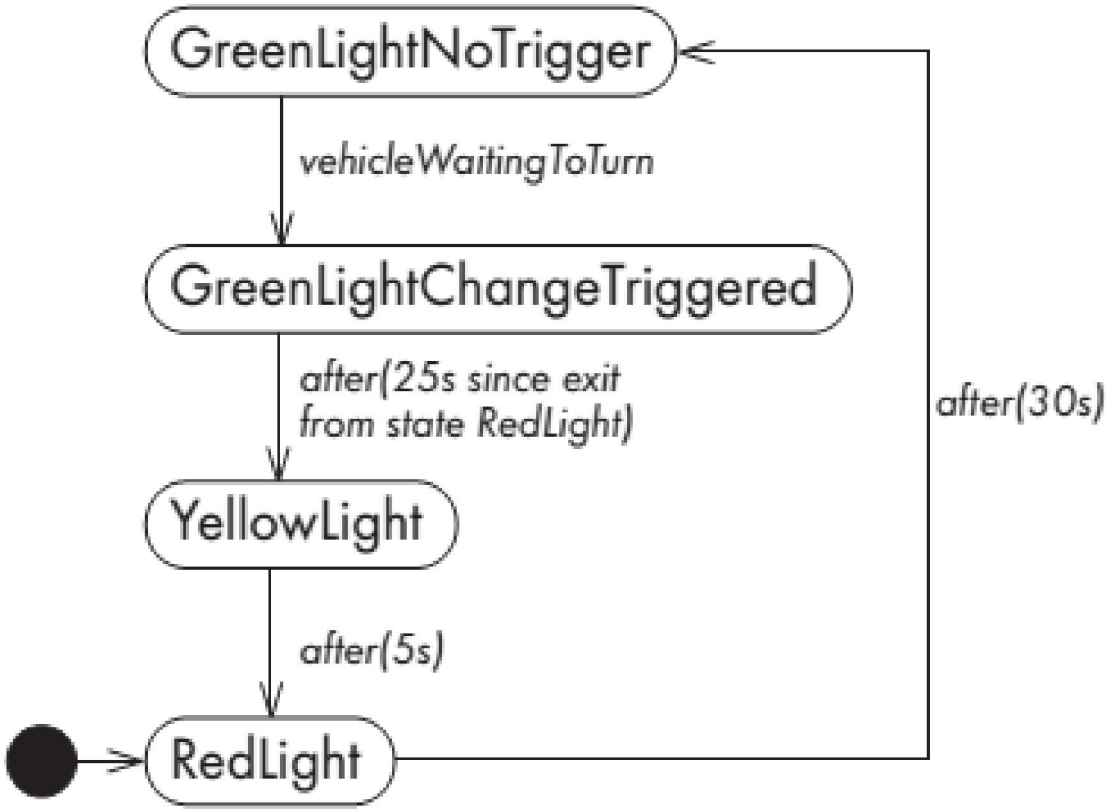
Other notation

- In addition, there are several other pieces of notation that can be placed inside states and on transitions to describe their behavior more precisely.
  - Elapsed time transitions.
  - Transitions triggered by a condition becoming true.
  - Nested substates and guard conditions.

Elapsed-time transitions

- The event that triggers a transition can be a certain amount of elapsed time.
- E.g. in traffic light, there are three main states, corresponding to the three colors of the traffic signal at a single. The initial state has a transition to 'RedLight' state to indicate what happens when the system starts up. After startup, the system indefinitely rotates among green, yellow and red, therefore there is no end state.

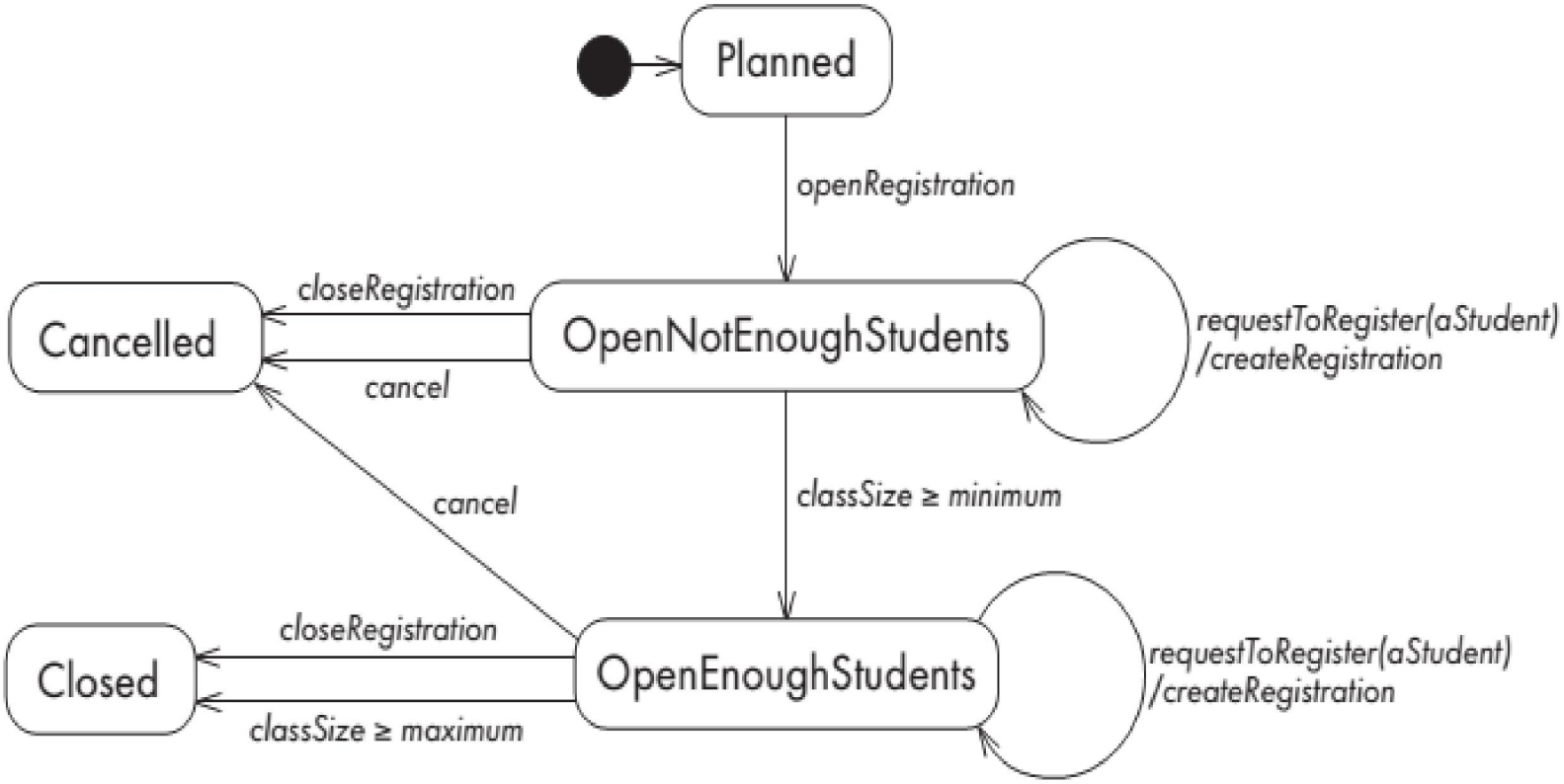
State diagram for traffic light



Transitions triggered by a condition becoming true

- A state diagram can contain transitions that are made whenever certain conditions become true.
- A condition can be distinguished from an event name since it contains a Boolean operator.

State diagram of a CourseSection class



State diagram of a CourseSection class

- The two conditions are classSize >= minimum, and classSize >= maximum.
- When it is first created, a CourseSection is in 'Planned' state and is not yet ready to receive students. When registration is opened, the system moves to 'OpenNotEnoughStudents' state.

State diagram of a CourseSection class

- In this state, the CourseSection can accept requests to register, but the course will not actually be taught until the class size reaches a certain minimum. The object evaluates the classSize >= minimum condition every time anything occurs that could make the condition true; as soon as it becomes true, a transition is taken to 'OpenEnoughStudents' state.

State diagram of a CourseSection class

- If requests to register continue while in 'OpenEnoughStudents' state, the object will eventually exceed a predefined maximum number of students, at which time it automatically moves to 'Closed' state. The course section can also be explicitly closed by a closeRegistration event, indicating that the registration deadline has passed. If there are not enough students, closing a course section has the same effect as canceling it.

State diagram of a CourseSection class

- There is no end state (the target symbol) because a permanent record of the course section is kept, whatever happens. The requestToRegister transitions shows that a transition can lead from a state back to the same state. Also, the /createRegistrationnotation designates an action;.

Nested substates and guard conditions

- A state diagram can be nested inside a state.
- The states of the inner diagram are called substates.

Nested substates and guard conditions

- A state diagram of an automatic transmission; at the top level this has three states: ‘Neutral’, ‘Reverse’ and a driving state, which is not explicitly named. The driving state is divided into substates corresponding to the three gears that the system automatically chooses.

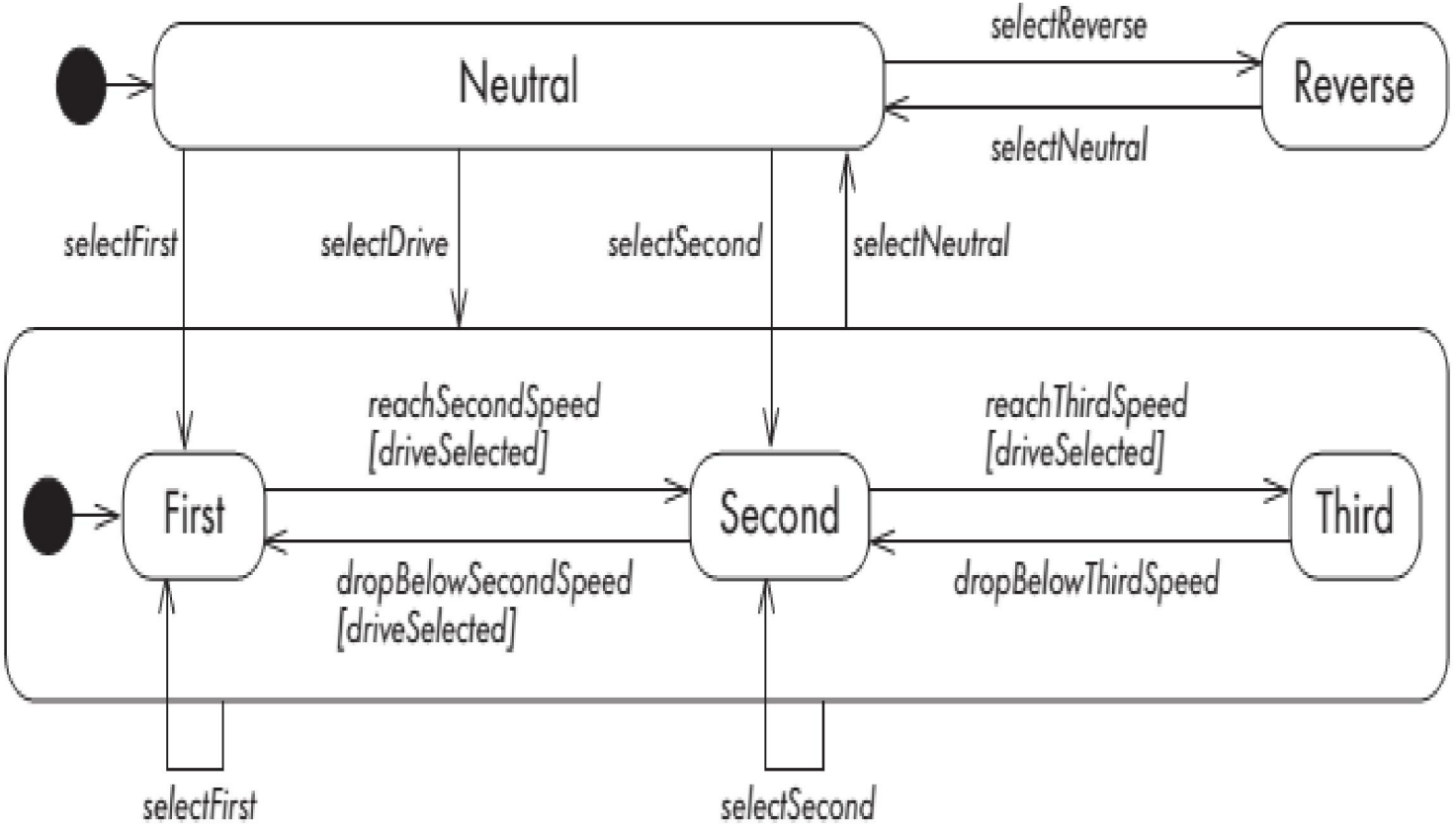
Nested substates and guard conditions

- The advantage of the nesting is that it shows compactly that the driving substates are all very similar to each other – in particular, that they can all transition to ‘Neutral’ at any time, upon the user’s command.

Nested substates and guard conditions

- The start symbol inside the driving state shows that it by default starts at the 'First' substate. However, the user can also manually select 'First' or 'Second' to force the transmission to move into, and stay in, these substates.

Nested substates and guard conditions



Guard conditions

- The notation reachSecondSpeed[driveSelected] illustrates the use of a guard condition.
- The system will only respond to the indicated event (reachSecondSpeed) if the condition in square brackets is true.

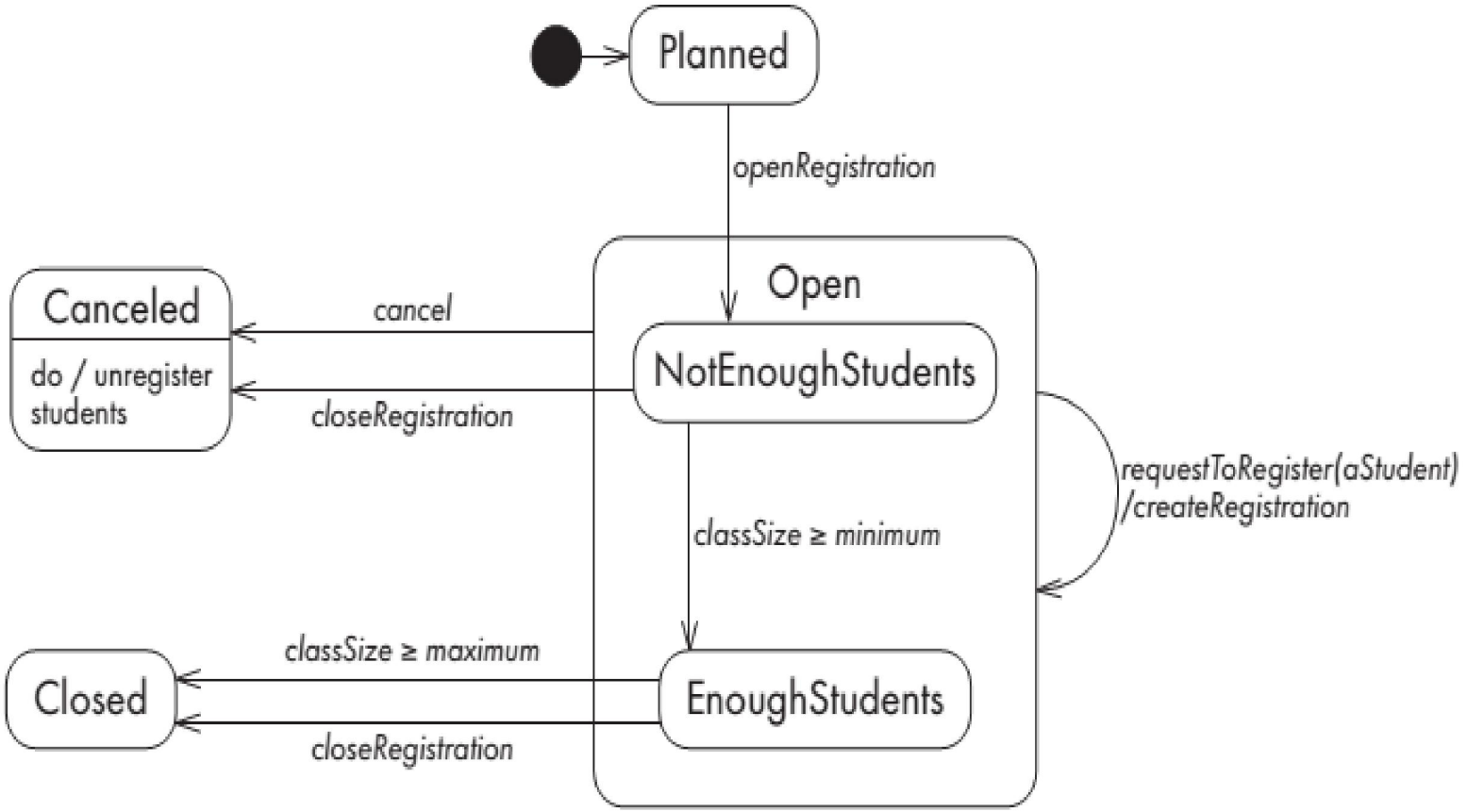
Guard conditions

- In the state diagram, guard condition is used to prevent the transmission from changing gear if the driver had manually selected first or second gear.

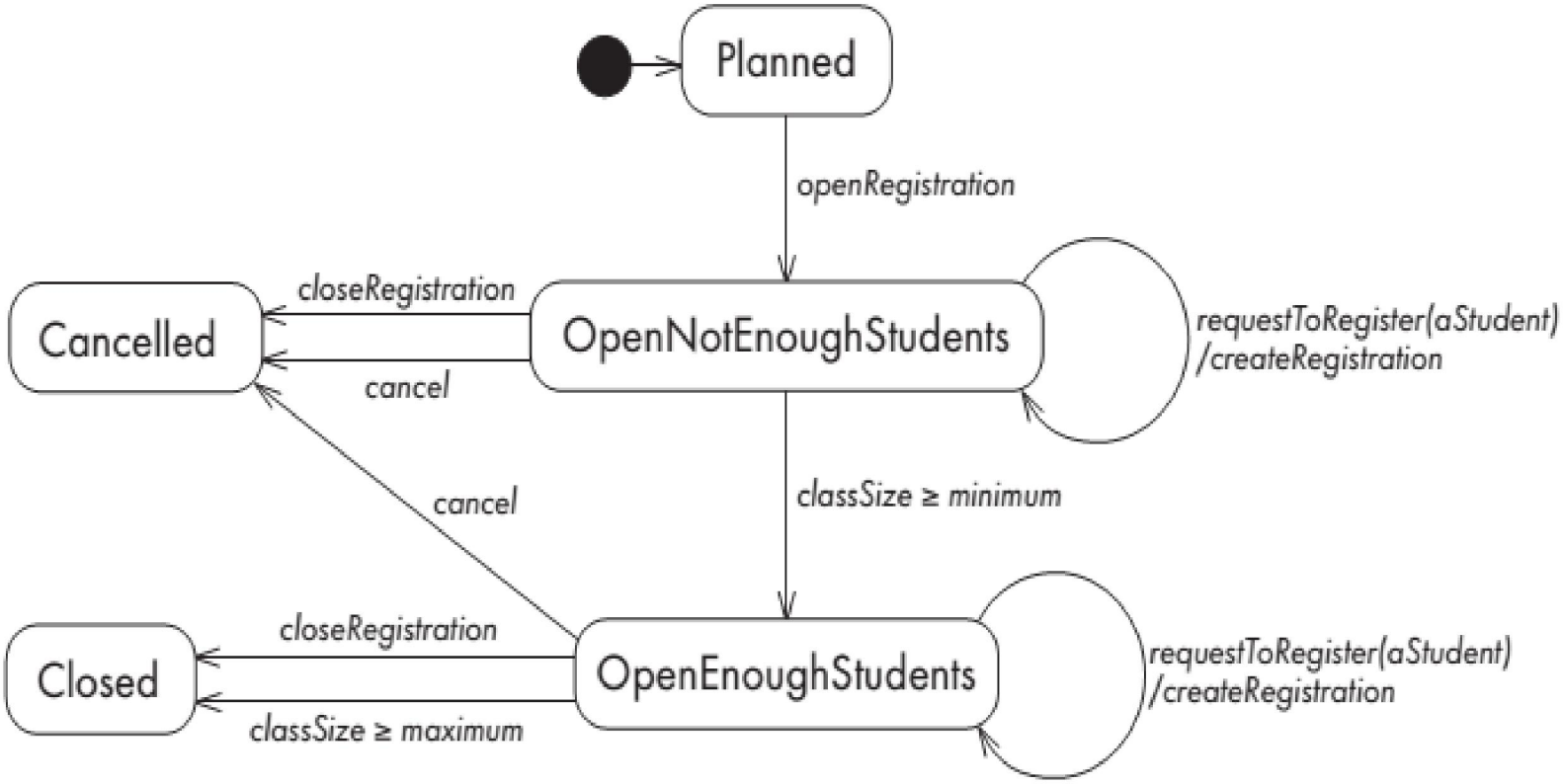
Guard conditions

- A guard condition differs from other type of conditions:
  - A guard condition is only evaluated when its associated event occurs.

Course section example showing the effect of nested states



Course section without nesting



End of chapter 5.5