

STATE TRANSITION TESTING

**Reference: Software Testing Course
@ Lonsdale System**

Outline

- ❑ Generating State Transition Diagrams
- ❑ State-Transition Tables
- ❑ Creating Test Cases
- ❑ Other examples

Define States

☐ Purpose

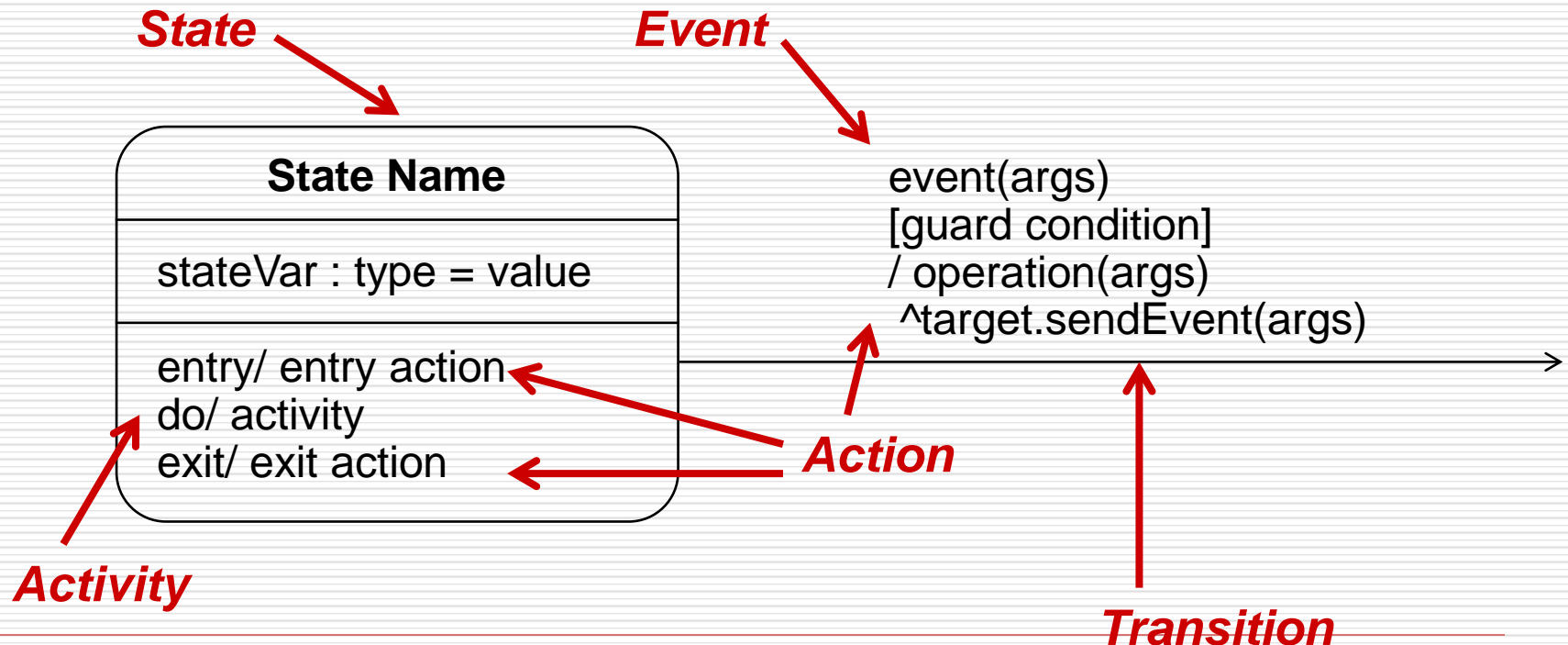
- Design how an object's state affects its behavior
- Develop statecharts to model this behavior

☐ Things to consider :

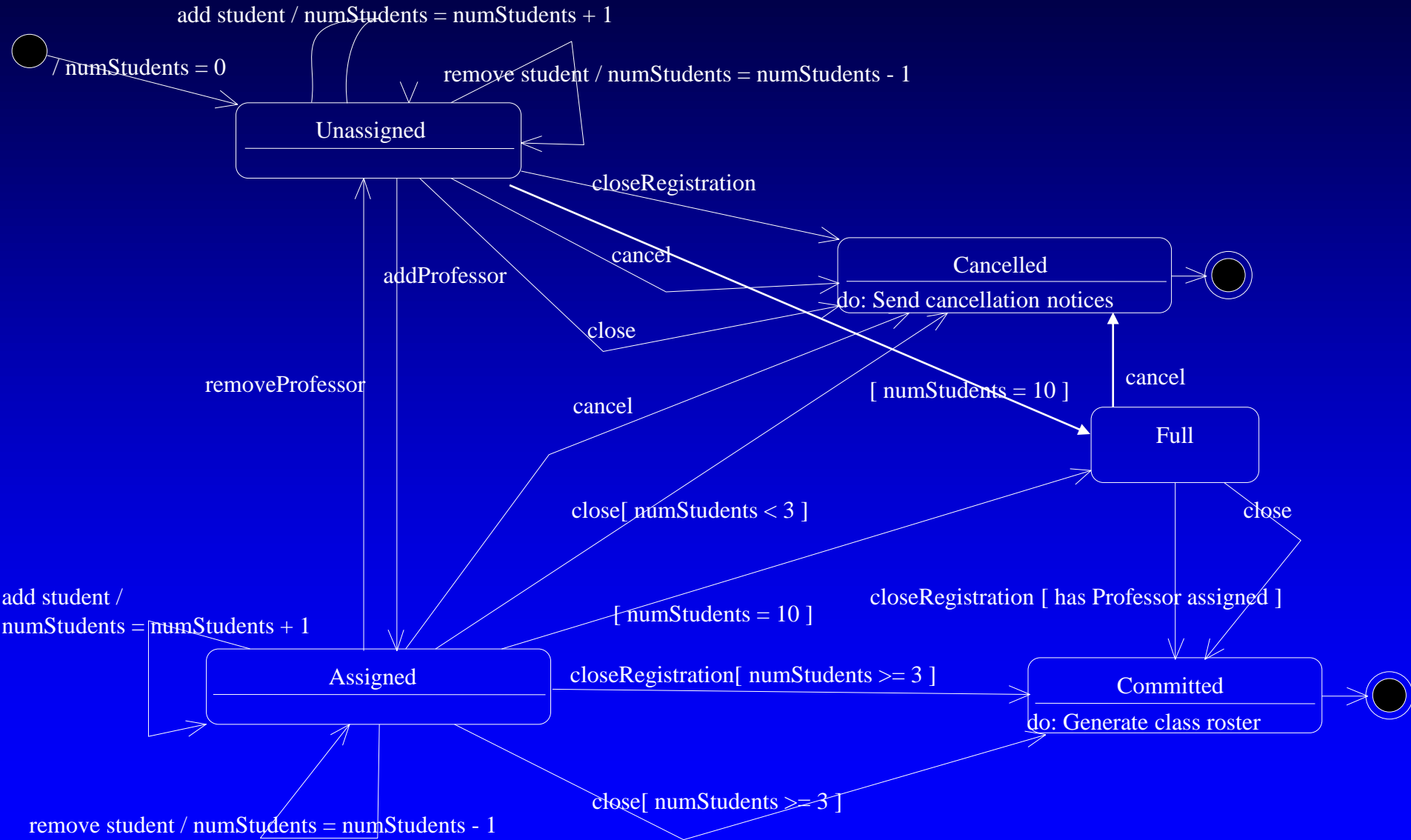
- Which objects have significant state?
 - How to determine an object's possible states?
 - How do statecharts map to the rest of the model?
-

What is a Statechart?

- ❑ A directed graph of states (nodes) connected by transitions(directed arcs)
- ❑ Describes the life history of a reactive object



Statechart for Course Offering



State-Transition Testing Example

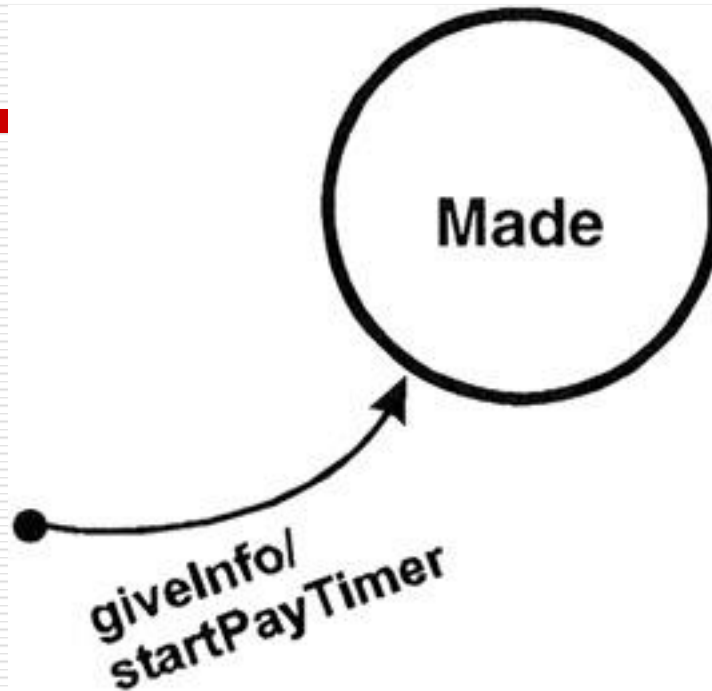
- ❑ State-Transition diagrams are an excellent tool to capture certain types of system requirements and to document internal system design.
- ❑ These diagrams document the events that come into and are processed by a system as well as the system's responses.

Reservation System Example

1- Make a reservation

- ❑ Provide information including departure and destination cities, dates, and times.
- ❑ A reservation agent uses that information to make a reservation.
- ❑ At that point, the **Reservation** is in the **Made** state.
The system creates and starts a timer.
- ❑ If this **timer expires** before the **reservation is paid** for, the reservation **is cancelled** by the system.

Reservation System Example

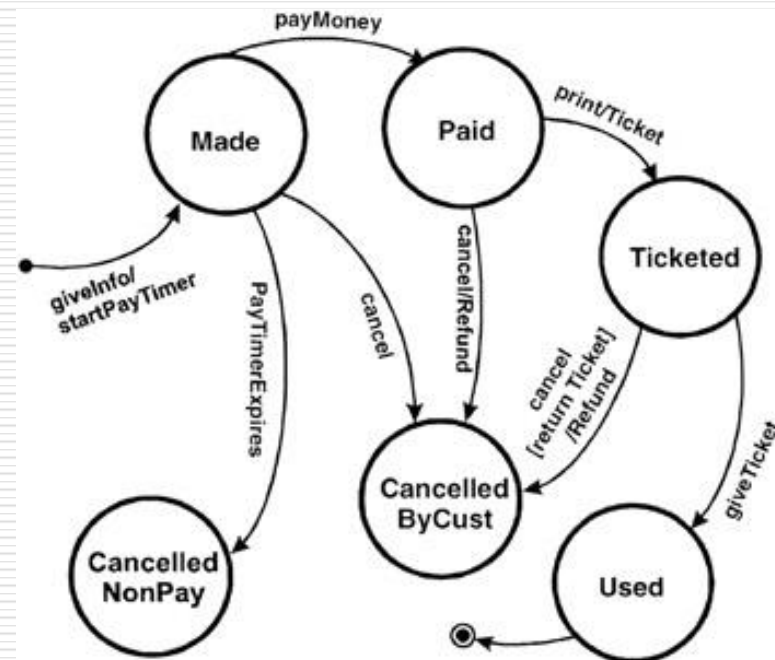


giveInfo, is an **event** that comes into the system from the outside world.

The command after the "/" denotes **an action** of the system; in this case **startPayTimer**.

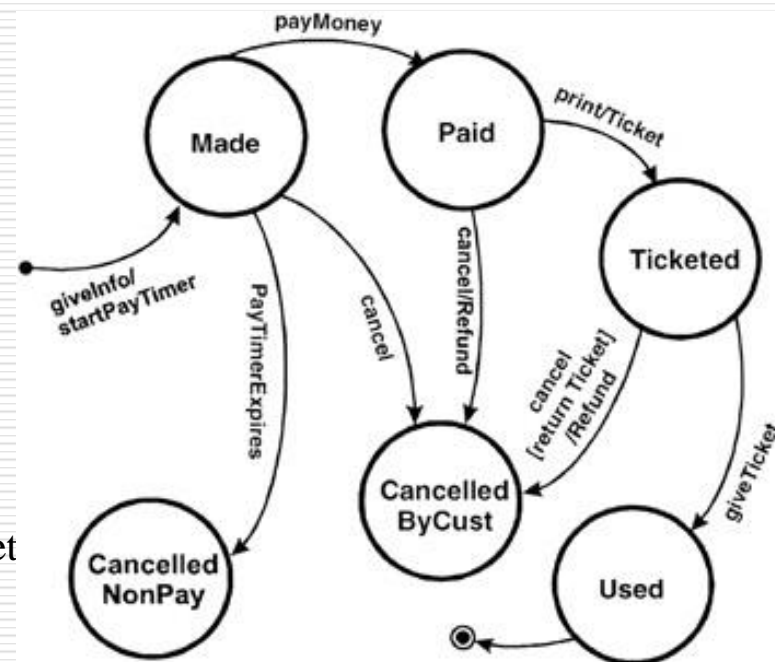
Reservation System Example

- When money is **paid**, through initiation of the **PayMoney** action, the system goes into the “Paid” state.
 - Events may have parameters associated with them. For example, **Pay Money** may indicate **Cash, Check, Debit Card, or Credit Card**
- When **the ticket is printed**, the system goes into the **Ticketed State**
- Upon boarding the plane, the customer gives the **boarding pass along with the ticket**, which signals that the ticket has been **Used**.
 - Anything else after this state?

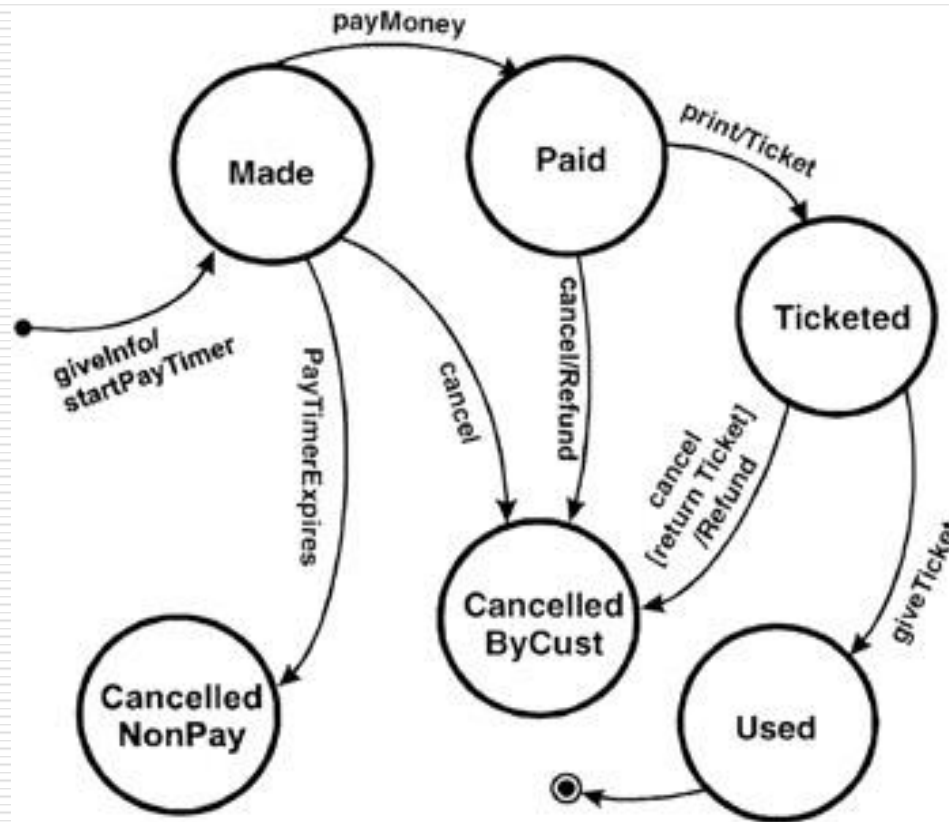


Reservation System Example

- If the **Reservation** is not paid on time, the **PayTimer expires** and the Reservation is **cancelled** for non-payment.
- A reservation may become cancelled if the customer wishes to do so.
 - This can happen prior to payment or after payment.
 - If after payment, a refund needs to be generated.
 - If the customer had the tickets with him, a refund cannot be given unless the printed ticket itself is returned to the agent.
 - This introduces one new notational element—square brackets [] that contain a conditional that can be evaluated either **True** or **False**.
 - This conditional acts as a guard allowing the transition only if the condition is true.



State Transition Diagram of the Reservation System



State-Transition Tables

- ❑ State-transition tables **may be easier to use** in a **complete and systematic manner**.
- ❑ State-transition tables consist of **four columns—Current State, Event, Action, and Next State**.
- ❑ Build the state-transition table as follows
 - For each state of the system
 - For each event/trigger of the system
 - Find the corresponding (Action, Next State) [if any]
 - Document (state, event, action, next state)

State-Transition table for Reservation.

Current State	Event	Action	Next State
null	giveInfo	startPayTimer	Made
null	payMoney	--	null
null	print	--	null
null	giveTicket	--	null
null	cancel	--	null
null	PayTimerExpires	--	null

Made	<u>giveInfo</u>	--	Made
Made	<u>payMoney</u>	--	Paid
Made	print	--	Made
Made	<u>giveTicket</u>	--	Made
Made	cancel	--	Can- <u>Cust</u>
Made	<u>PayTimerExpires</u>	--	Can- <u>NonPay</u>
Paid	<u>giveInfo</u>	--	Paid
Paid	<u>payMoney</u>	--	Paid
Paid	print	Ticket	Ticketed
Paid	<u>giveTicket</u>	--	Paid
Paid	cancel	Refund	Can- <u>Cust</u>
Paid	<u>PayTimerExpires</u>	--	Paid

Ticketed	<u>giveInfo</u>	--	Ticketed
Ticketed	<u>payMoney</u>	--	Ticketed
Ticketed	print	--	Ticketed
Ticketed	<u>giveTicket</u>	--	Used
Ticketed	cancel	Refund	Can-Cust <u>om</u>
Ticketed	<u>PayTimerExpires</u>	--	Ticketed
Used	<u>giveInfo</u>	--	Used
Used	<u>payMoney</u>	--	Used
Used	print	--	Used
Used	<u>giveTicket</u>	--	Used
Used	cancel	--	Used
Used	<u>PayTimerExpires</u>	--	Used

Can- <u>NonPay</u>	<u>giveInfo</u>	--	Can- <u>NonPay</u>
Can- <u>NonPay</u>	<u>payMoney</u>	--	Can- <u>NonPay</u>
Can- <u>NonPay</u>	print	--	Can- <u>NonPay</u>
Can- <u>NonPay</u>	<u>giveTicket</u>	--	Can- <u>NonPay</u>
Can- <u>NonPay</u>	cancel	--	Can- <u>NonPay</u>
Can- <u>NonPay</u>	<u>PayTimerExpires</u>	--	Can- <u>NonPay</u>
Can- <u>Cust</u>	<u>giveInfo</u>	--	Can- <u>Cust</u>
Can- <u>Cust</u>	<u>payMoney</u>	--	Can- <u>Cust</u>
Can- <u>Cust</u>	print	--	Can- <u>Cust</u>
Can- <u>Cust</u>	<u>giveTicket</u>	--	Can- <u>Cust</u>
Can- <u>Cust</u>	cancel	--	Can- <u>Cust</u>
Can- <u>Cust</u>	<u>PayTimerExpires</u>	--	Can- <u>Cust</u>

State-Transition Tables (Cont.)

- ❑ The **advantage** of a state-transition table is that it **lists all possible state-transition combinations**, not just the valid ones.
- ❑ When testing critical, **high-risk systems** such as avionics or medical devices, **testing every state-transition pair may be required**, including those **that are not valid**.
- ❑ Creating a state-transition table **often unearths combinations that were not identified**, documented, or dealt with in the requirements. It is **highly beneficial to discover these defects before coding begins**.

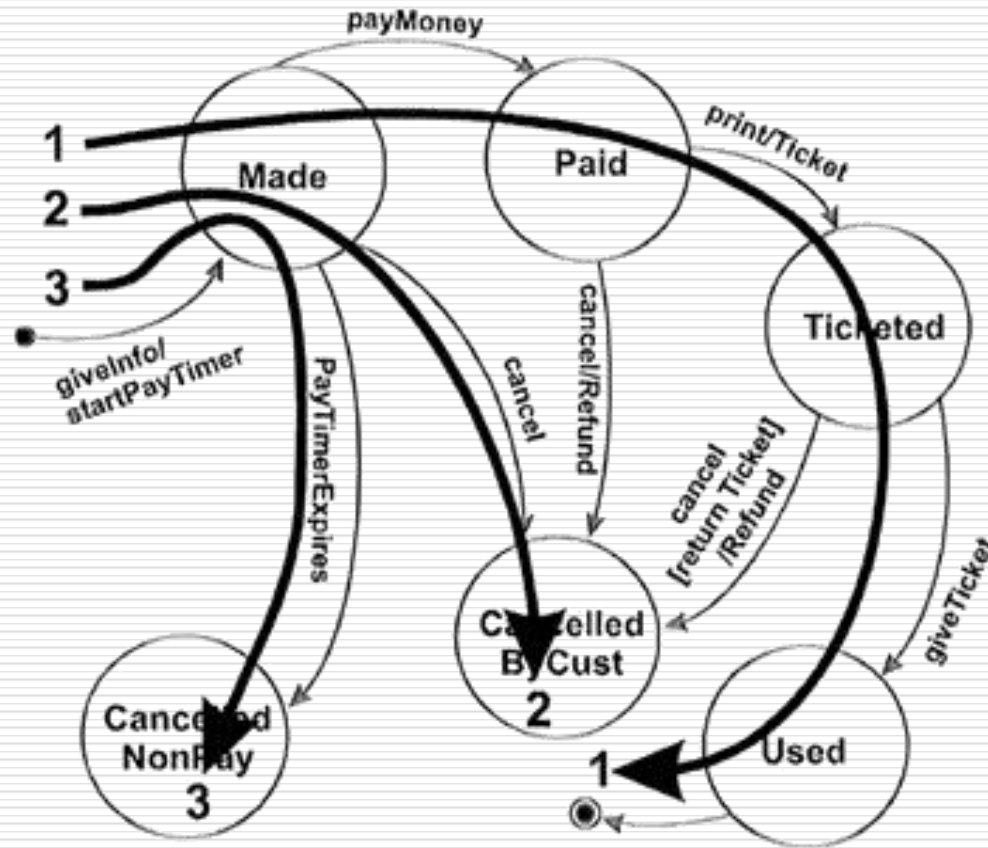
State-Transition Tables (Cont.)

- ❑ Using a state-transition table can help detect defects in implementation that **enable invalid paths from one state to another**.
- ❑ The **disadvantage** of such tables is that they become **very large very quickly as the number of states and events increases**.
- ❑ In addition, the tables are **generally sparse**; that is, **most of the cells are empty**.

Creating Test Cases

- 1. Create a set of test cases such that **all states are "visited" at least once under test.**** The set of three test cases shown below meets this requirement. Generally this is a weak level of test coverage.

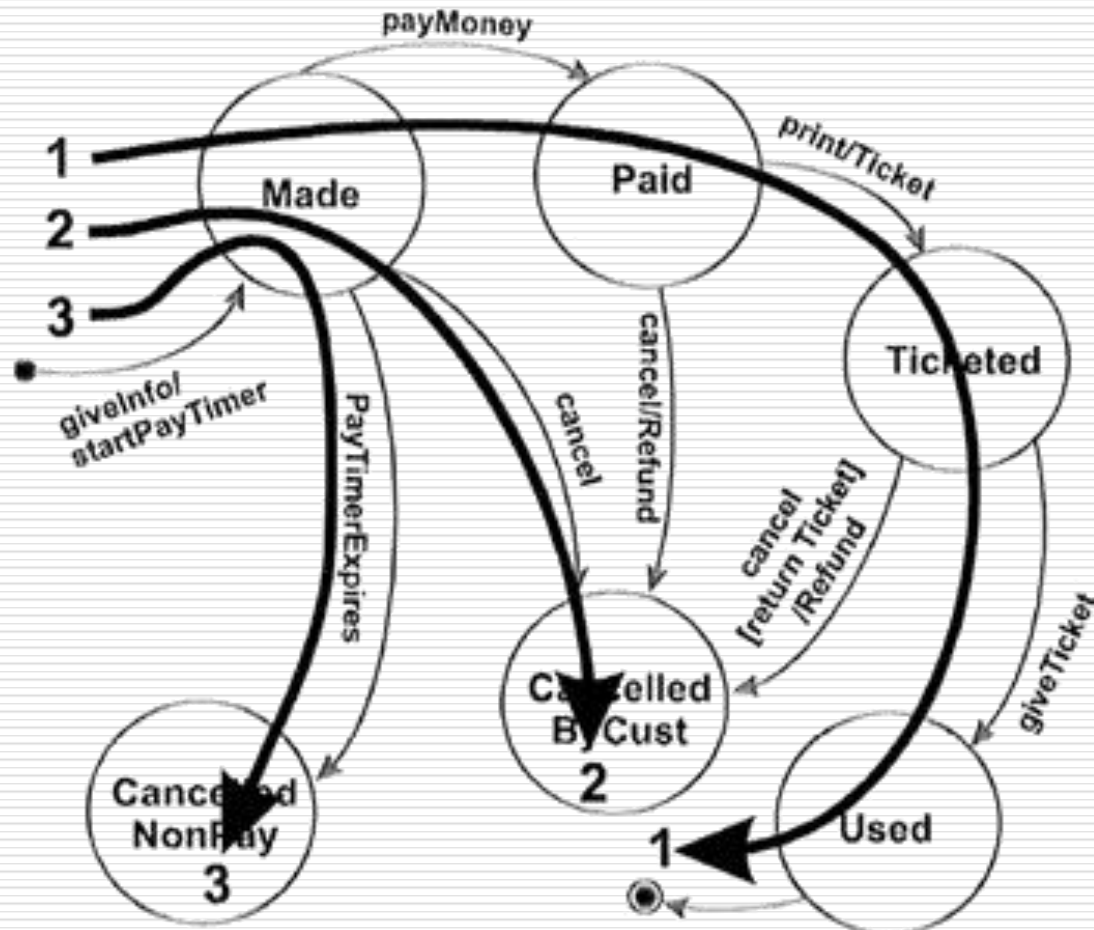
A set of test cases that "visit" each state



Creating Test Cases (Cont.)

2. Create a set of test cases **such that all events are triggered at least once under test.** Note that the test cases that cover each event can be the same as those that cover each state. Again, this is a weak level of coverage.

A set of test cases that trigger all events at least once



Creating Test Cases (Cont.)

3. Create a set of test cases such **that all paths are executed at least once under test.**

- ☐ While this level is the most preferred because of its level of coverage, it may not be feasible.
- ☐ If the **state-transition diagram has loops**, then the number of possible paths may be infinite.

For example, given a system with two states, A and B, where A transitions to B and B transitions to A. A few of the possible paths are:

- ☐ A→B
- ☐ A→B→A
- ☐ A→B→A→B→A→B
- ☐ A→B→A→B→A→B→A
- ☐ ...
- ☐ and so on forever.

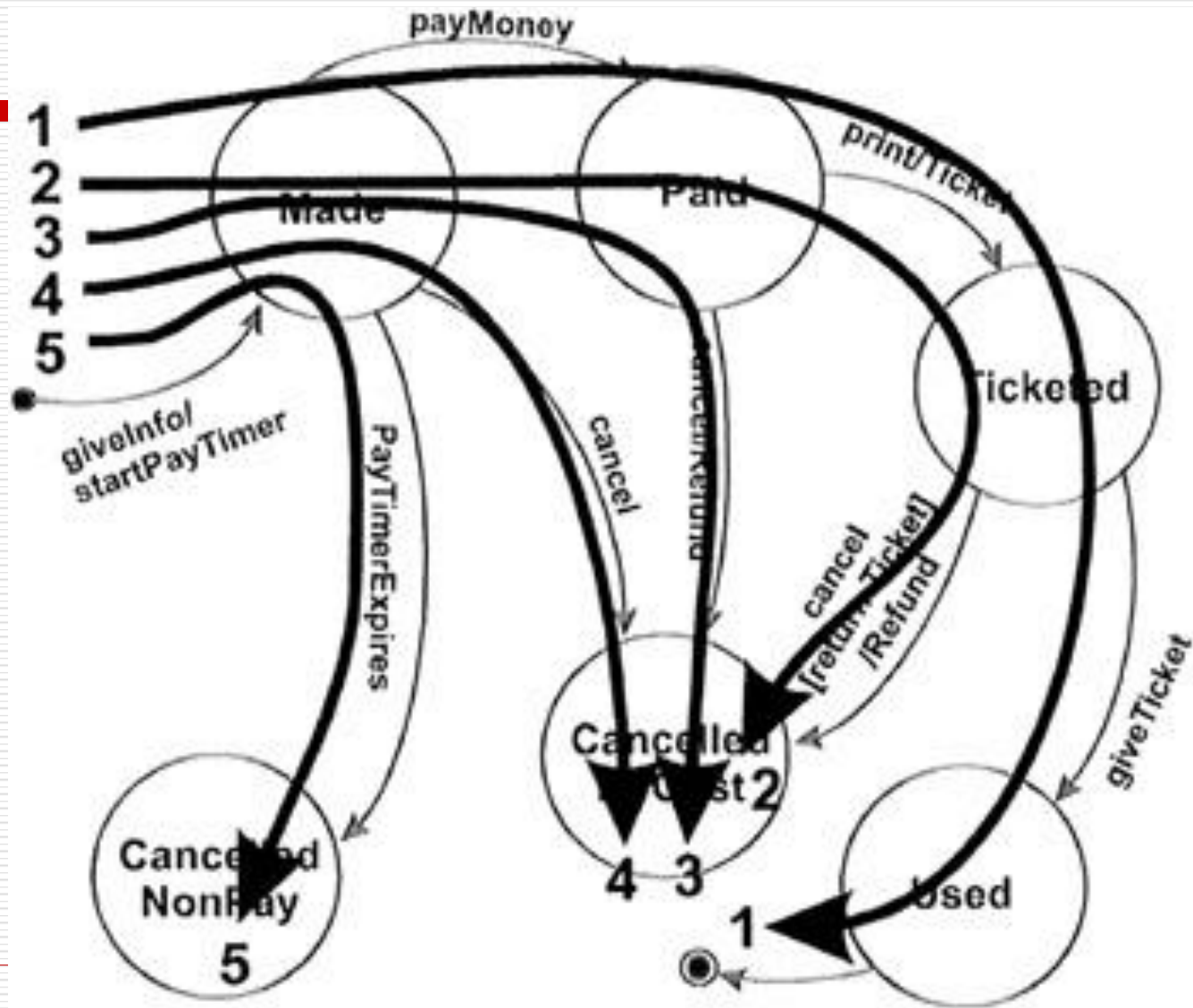
3. Create a set of test cases such that all paths are executed at least once under test (Cont.)

- Testing of loops such as this can be important if they may result in accumulating computational errors or resource loss (locks without corresponding releases, memory leaks, etc.).

Creating Test Cases (Cont.)

4. Create a set of test cases such that all transitions are exercised at least once under test. This level of testing provides a good level of coverage without generating large numbers of tests. This level is generally the one recommended.

A set of test cases that trigger all transitions at least once



4. A set of test cases that trigger all transitions at least once (Cont.)

- ❑ Test cases can also be read directly from the state-transition table. The gray rows in the following table show all the valid transitions.

Testing all valid transitions from a State-transition table.

Current State	Event	Action	Next State
null	giveInfo	startPayTimer	Made
null	payMoney	--	null
null	print	--	null
null	giveTicket	--	null
null	cancel	--	null
null	PayTimerExpires	--	null

Made	<u>giveInfo</u>	--	Made
Made	<u>payMoney</u>	--	Paid
Made	print	--	Made
Made	<u>giveTicket</u>	--	Made
Made	cancel	--	Can- <u>Cust</u>
Made	<u>PayTimerExpires</u>	--	Can- <u>NonPay</u>
Paid	<u>giveInfo</u>	--	Paid
Paid	<u>payMoney</u>	--	Paid
Paid	print	Ticket	Ticketed
Paid	<u>giveTicket</u>	--	Paid
Paid	cancel	Refund	Can- <u>Cust</u>
Paid	<u>PayTimerExpires</u>	--	Paid

Ticketed	<u>giveInfo</u>	--	Ticketed
Ticketed	<u>payMoney</u>	--	Ticketed
Ticketed	print	--	Ticketed
Ticketed	<u>giveTicket</u>	--	Used
Ticketed	cancel	Refund	Can- <u>Cust</u>
Ticketed	<u>PayTimerExpires</u>	--	Ticketed
Used	<u>giveInfo</u>	--	Used
Used	<u>payMoney</u>	--	Used
Used	print	--	Used
Used	<u>giveTicket</u>	--	Used
Used	cancel	--	Used
Used	<u>PayTimerExpires</u>	--	Used

Can-NonPay	giveInfo	--	Can-NonPay
Can-NonPay	payMoney	--	Can-NonPay
Can-NonPay	print	--	Can-NonPay
Can-NonPay	giveTicket	--	Can-NonPay
Can-NonPay	cancel	--	Can-NonPay
Can-NonPay	PayTimerExpires	--	Can-NonPay
Can-Cust	giveInfo	--	Can-Cust
Can-Cust	payMoney	--	Can-Cust
Can-Cust	print	--	Can-Cust
Can-Cust	giveTicket	--	Can-Cust
Can-Cust	cancel	--	Can-Cust
Can-Cust	PayTimerExpires	--	Can-Cust

State-Transition Testing (Cont.)

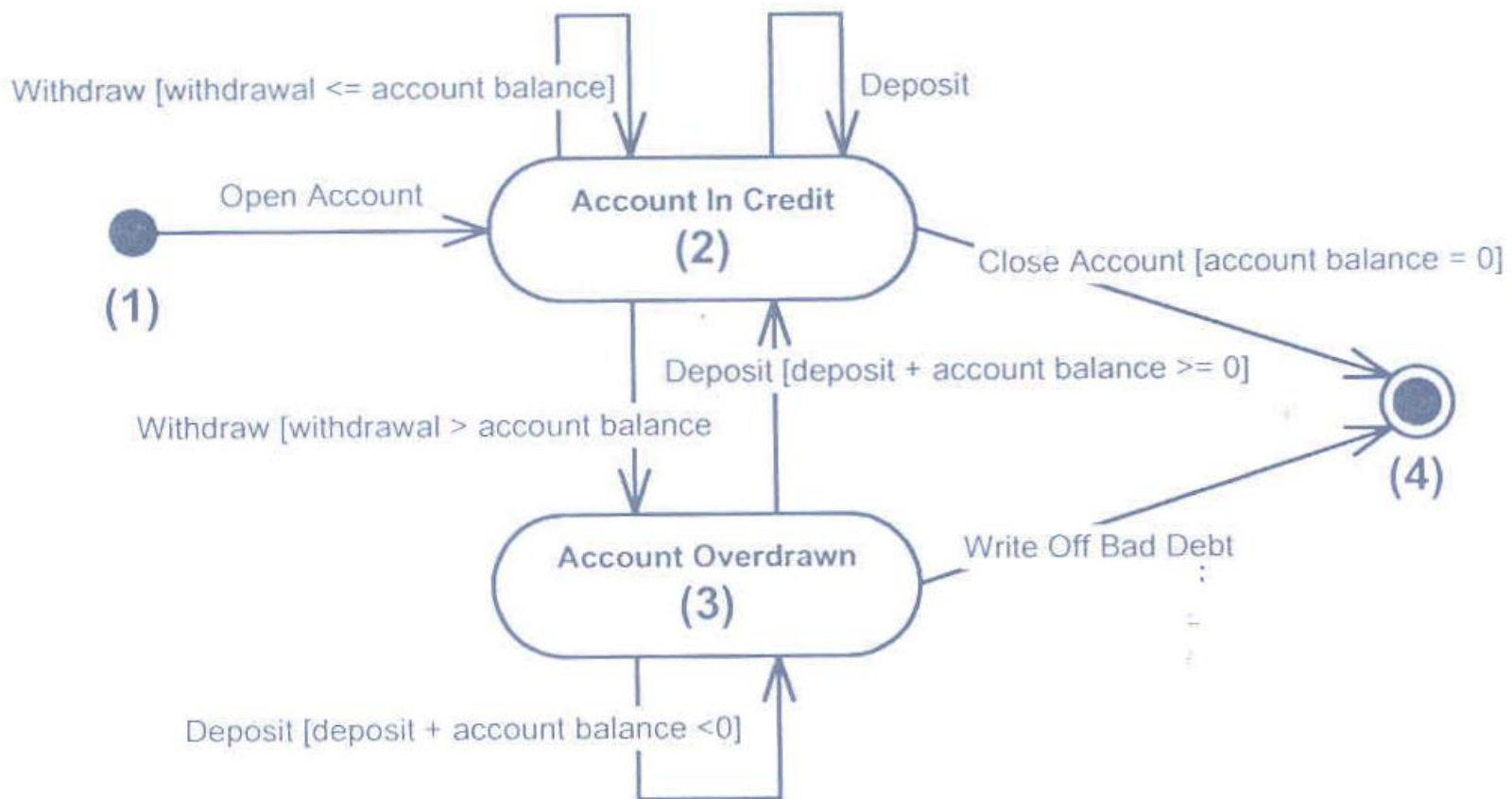
- ❑ In addition, depending on the system risk, you may want to create test cases for some or all of **the invalid state/event pairs to make sure the system has not implemented invalid paths.**
- ❑ ***Applicability and Limitations***
 - State-Transition diagrams **are excellent** tools to capture certain system requirements, namely those that **describe states and their associated transitions.** These diagrams then can be used to direct our testing efforts by identifying the states, events, and transitions that should be tested.
 - State-Transition diagrams **are not applicable** when the system **has no state or does not need to respond to real-time events** from outside of the system.

An example is a payroll program that reads an employee's time record, computes pay, subtracts deductions, saves the record, prints a paycheck, and repeats the process.

Key Points

- ❑ State-Transition diagrams **direct our testing efforts** by identifying the states, events, actions, and transitions that should be tested. **Together, these define how a system interacts with the outside world**, the events it processes, and the valid and invalid order of these events.
- ❑ A state-transition diagram is not the only way to document system behaviour. They may be easier to comprehend, but **state-transition tables may be easier to use in a** complete and systematic manner.
- ❑ The generally recommended level of testing using **state-transition diagrams is to create a set of test cases such that all transitions are exercised** at least once under test.
- ❑ In **high-risk systems, you may want to create even more test cases, approaching all paths if possible.**

State Transition for a Bank Account



Prior State	New State	Valid Transition	Comment
1	1	N	
1	2	Y	New account
1	3	N	Possible negative test case
1	4	N	
2	1	N	
2	2	Y	Deposit and withdraw [withdrawal \leq account balance]
2	3	Y	Withdraw [withdrawal $>$ account balance]
2	4	Y	Closed account [account balance=0]
3	1	N	
3	2	Y	Deposit [deposit + account balance ≥ 0]
3	3	Y	Deposit [deposit + account balance < 0]
3	4	Y	Write Off Bad Debt [account balance < 0]
4	1	N	
4	2	N	Possible negative test case
4	3	N	Possible negative test case
4	4	N	Possible negative test case

Table 1: State Table for a Bank Account

Ba

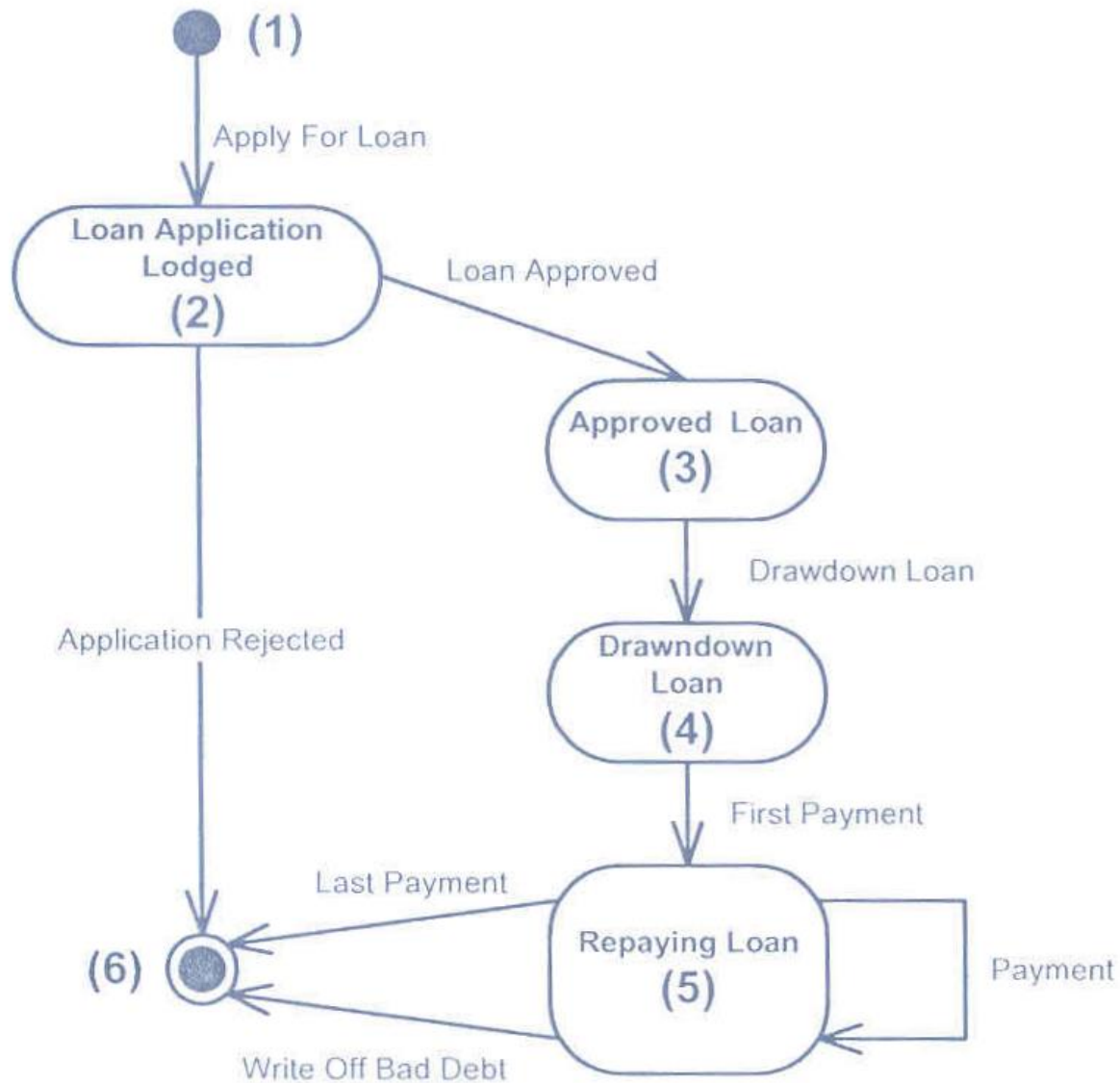


Figure 4: State transition diagram for a Bank Loan

Prior State	New State	Valid Transition	Comment
1	1	N	
1	2	Y	New loan application
1	3	N	
1	4	N	
1	5	N	
1	6	N	
2	1	N	
2	2	N	
2	3	Y	Loan approval
2	4	N	This tests that un-approved loans cannot be drawn down
2	5	?	Is there an application fee?
2	6	Y	Loan rejection
3	1	N	
3	2	?	Can approved loans be reconsidered?
3	3	N	
3	4	Y	Drawdown of loan
3	5	?	Can pre-payments be accepted?
3	6	N	
4	1	N	
4	2	?	Can a drawdown loan be reconsidered?
4	3	N	
4	4	?	Are multiple draw downs allowed?
4	5	Y	First Payment
4	6	?	Can a loan be written of before any payments have been made?
5	1	N	
5	2	N	
5	3	N	
5	4	?	Are multiple draw downs allowed?
5	5	Y	Payment during life of loan
5	6	Y	If account balance = 0, loan discharged If account balance < 0, loan default
6	1	N	
6	2	?	Can rejected loans be reconsidered
6	3	?	Can rejected loans be approved?
6	4	N	
6	5	?	Can written off loans be reinstated?
6	6	N	

Table 2: State Table for a Bank Loan

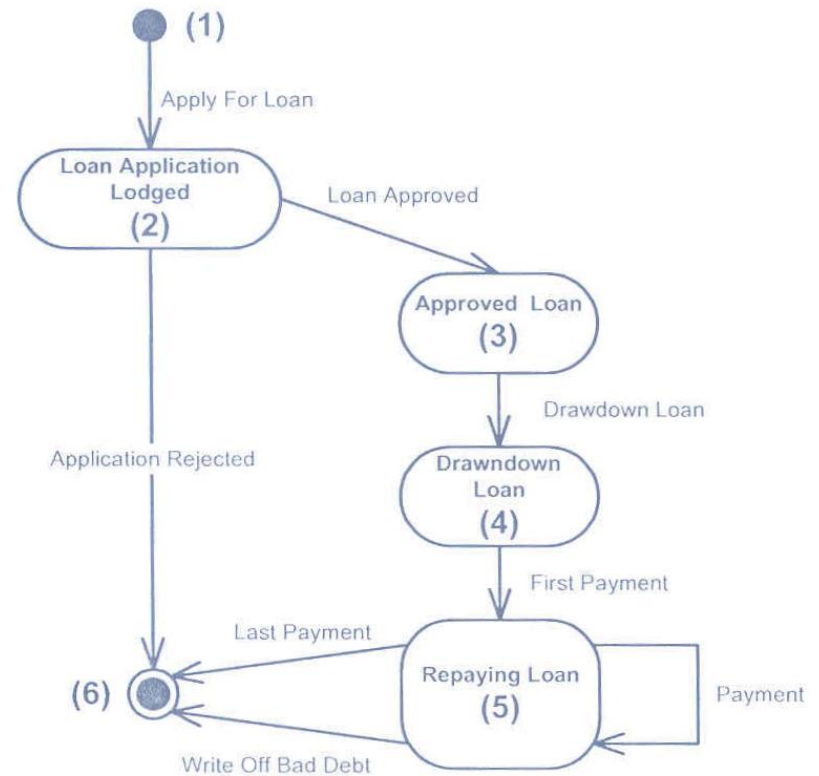
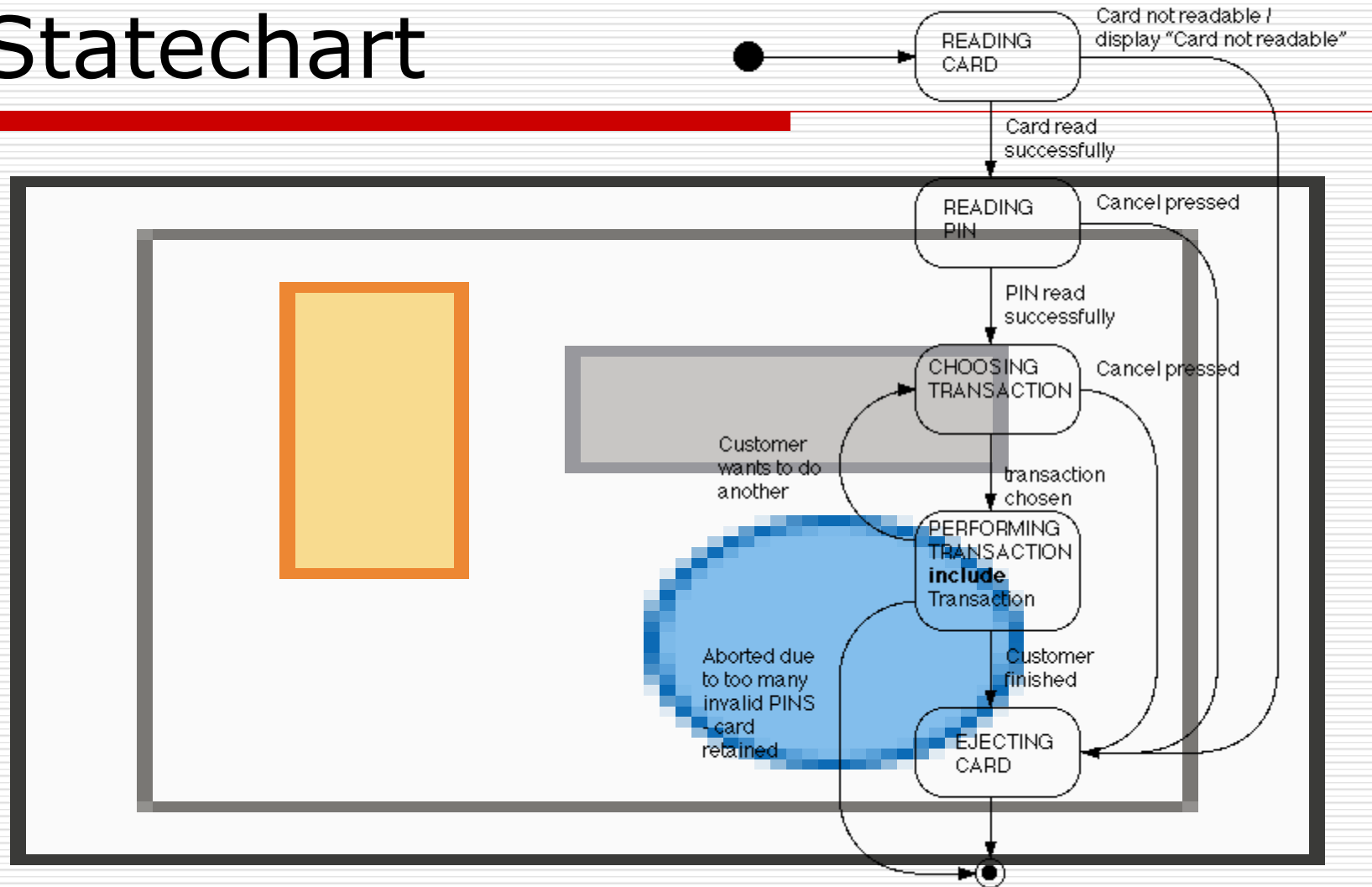
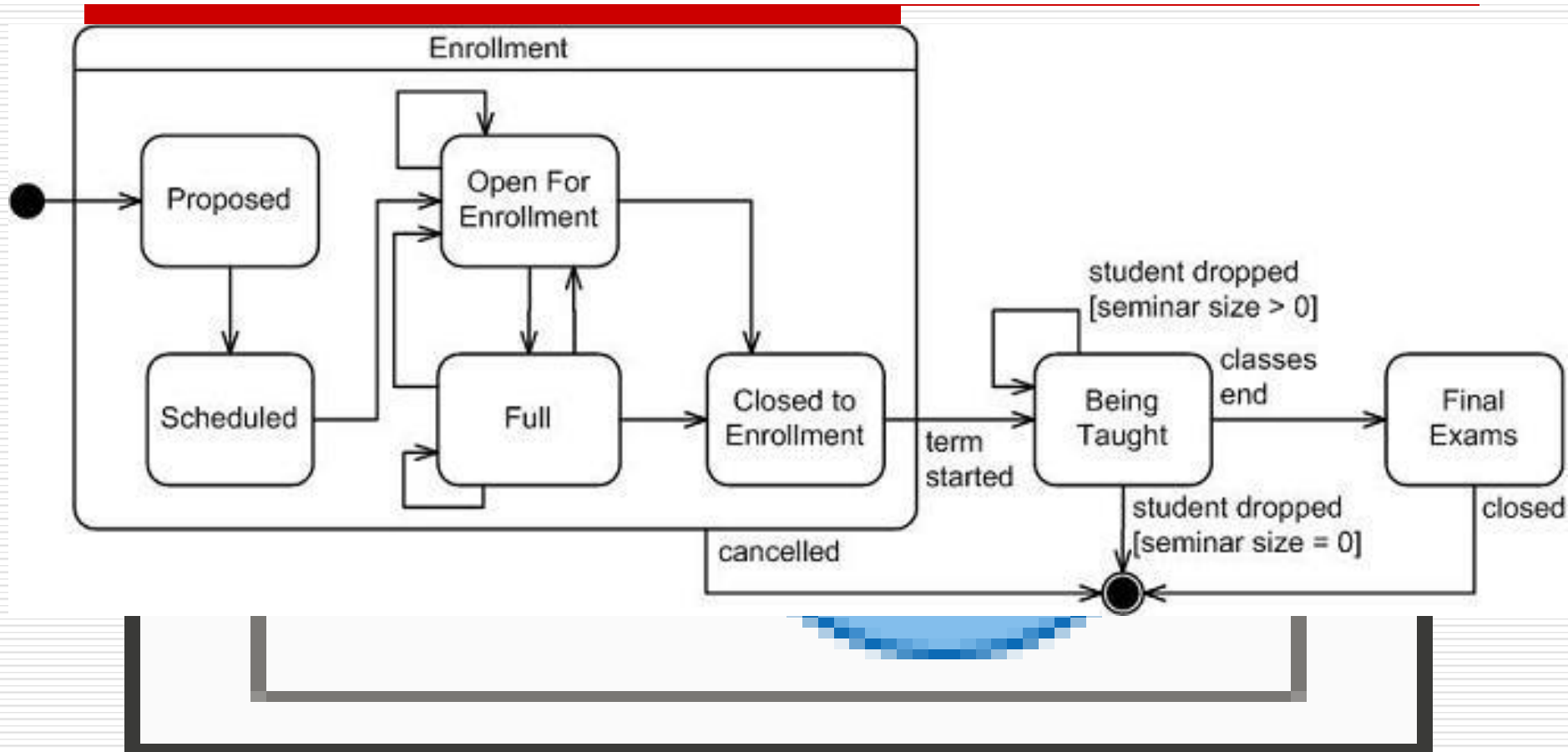


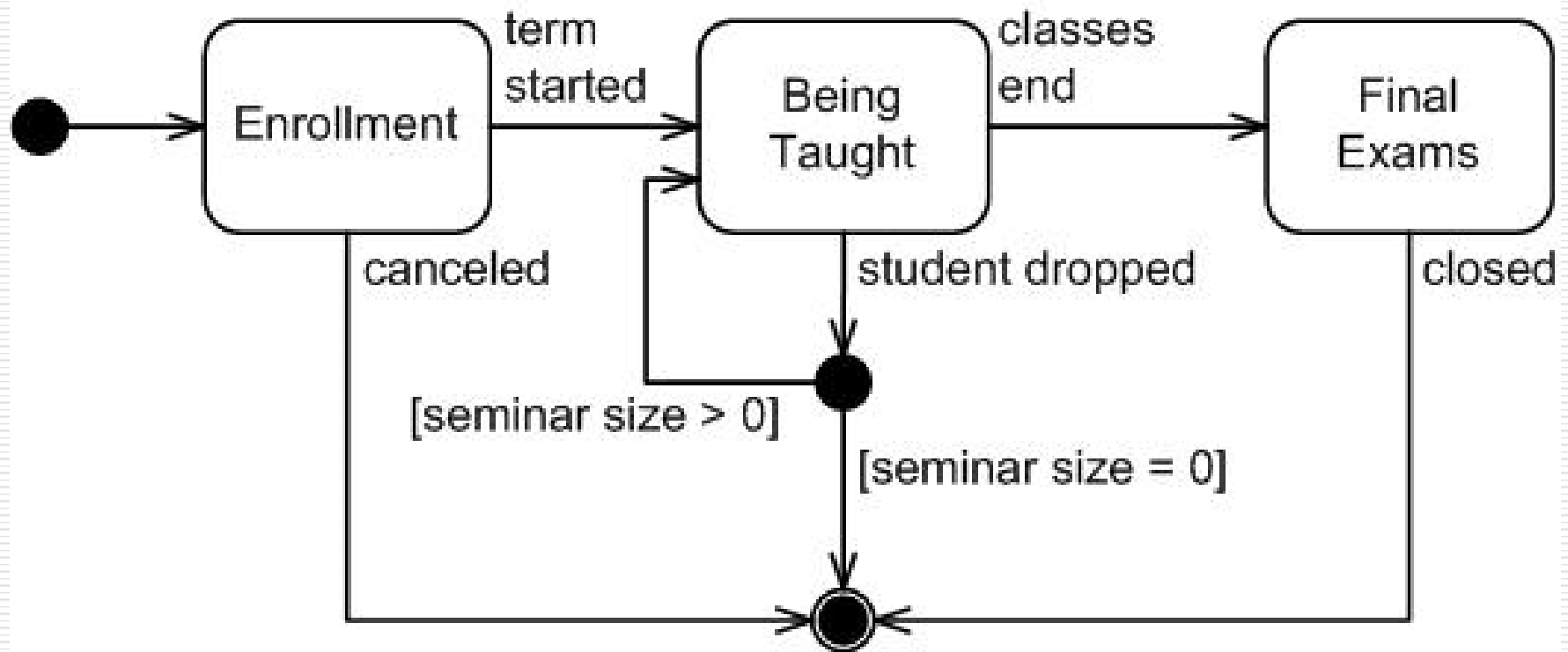
Figure 4: State transition diagram for a Bank Loan

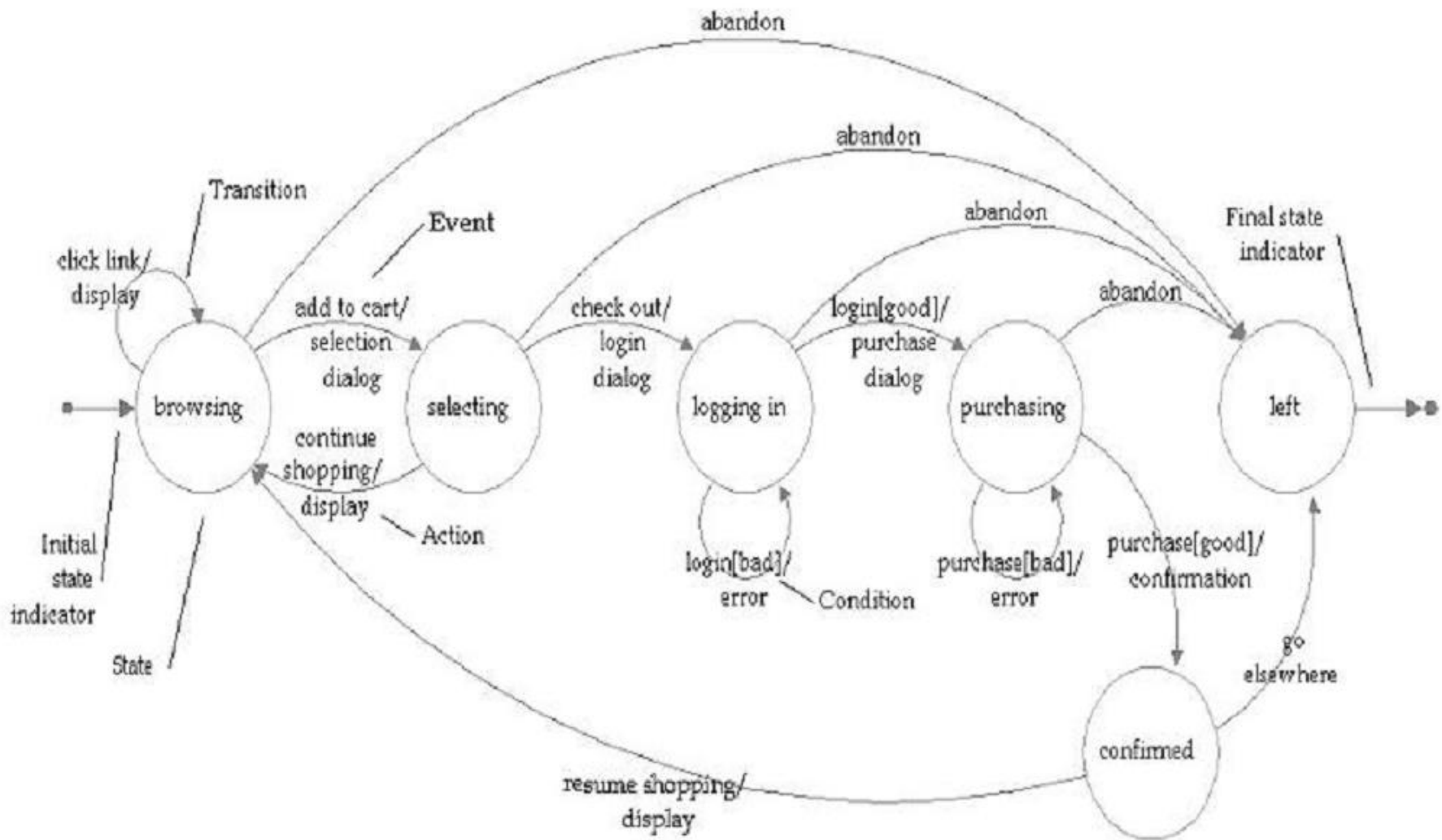
Statechart

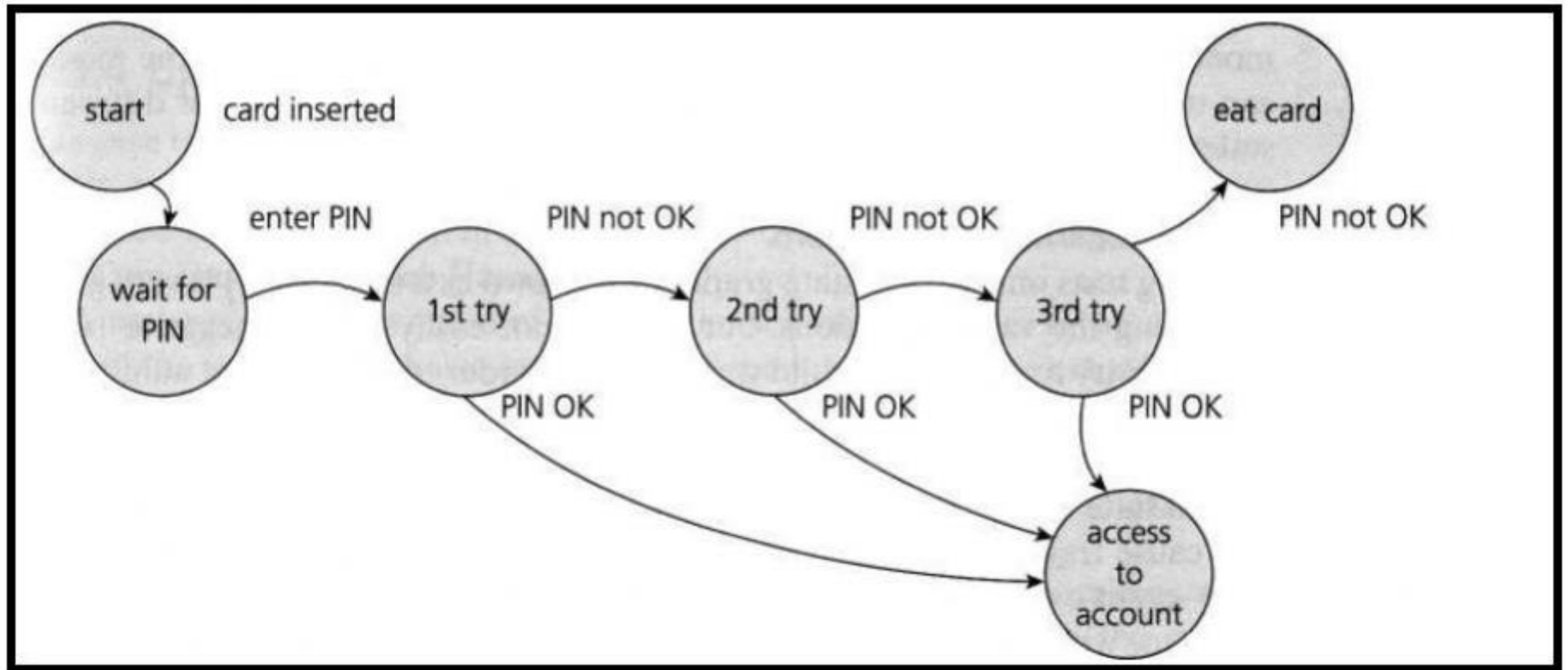
State-Chart for One Session

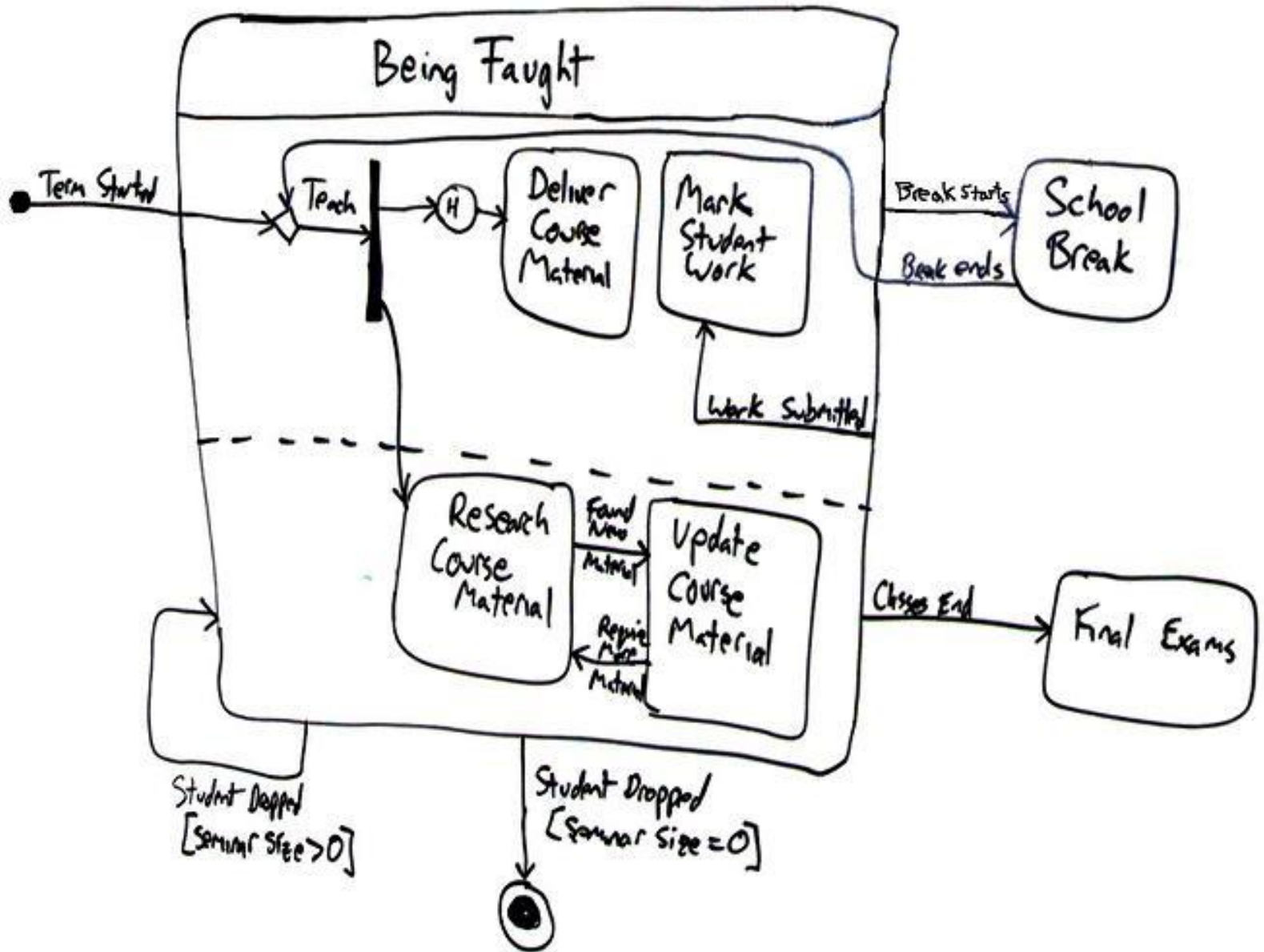















Exercise






















- ☐ Account mobifone
- ☐ ATM
- ☐ Moodle Assignment
- ☐ Student life cycle
- ☐ Bug life cycles

Assignment name*



You must supply a value here.

Description* ?

Arial 1 (8 pt) Lang **B** *I* U ~~S~~ x_2 x^2   

Path:

Grade ?

100

Available from

6 February 2015 10 15 ☐ Disable

Due date

13 February 2015 10 15 ☐ Disable

Prevent late submissions

Yes

Allow resubmitting ?

No

Email alerts to teachers ?

No

Maximum size

10MB

Group mode ?

No groups

Visible

Hide

ID number ?

Grade category

Uncategorised



Logged in as: administrator (administrator)

2015-02-06 04:19 CET



[My View](#) | [View Issues](#) | [Report Issue](#) | [Change Log](#) | [Roadmap](#) | [Summary](#) | [Manage](#) | [My Account](#) | [Logout](#)

Issue #

Jump

Recently Visited: 0000001

Assigned to Me (Unresolved) [^] (1 - 1 / 1)

0000001 Test bugs
[All Projects] General - 2015-02-06 04:18

Reported by Me [^] (1 - 1 / 1)

0000001 Test bugs
[All Projects] General - 2015-02-06 04:18

Recently Modified [^] (1 - 1 / 1)

0000001 Test bugs
[All Projects] General - 2015-02-06 04:18

Unassigned [^] (0 - 0 / 0)

Resolved [^] (0 - 0 / 0)

Monitored by Me [^] (0 - 0 / 0)

new

feedback

acknowledged

confirmed

assigned

resolved

closed


Copyright © 2000 - 2015 MantisBT Team
lamquangvu.shopping@gmail.com




Admin PIM **Leave** Time Recruitment My Info Performance Dashboard Directory Maintenance

Apply My Leave Entitlements ▾ Reports ▾ Configure ▾ **Leave List** Assign Leave

Leave List

From 

To 

Show Leave with Status All ☐ Rejected ☐ Cancelled ☐ Pending Approval ☒ Scheduled ☐ Taken ☐

Employee


Sub Unit ▾

Include Past Employees ☐

Search

Reset

Add Candidate

	<small>* First Name</small>	<small>Middle Name</small>	<small>* Last Name</small>
Full Name	<input type="text"/>	<input type="text"/>	<input type="text"/>
Email *	<input type="text"/>		
Contact No	<input type="text"/>		
Job Vacancy	<input type="text" value="-- Select --"/>		
Resume	<input type="button" value="Browse..."/> No file selected.		
	Accepts .docx, .doc, .odt, .pdf, .rtf, .txt up to 1MB		
Keywords	<input type="text" value="Enter comma separated words..."/>		
Comment	<input type="text"/>		
Date of Application	<input type="text" value="2022-03-01"/>		
Consent to keep data	<input type="checkbox"/>		

* Required field