Modeling requirements

Nagwovuma Margaret

1

TOOLS FOR MODELLING REQ'TS

- UML

2

Unified Modeling Language (UML)

- Use case diagrams
- Class diagrams
- Sequential diagrams
- State Diagrams

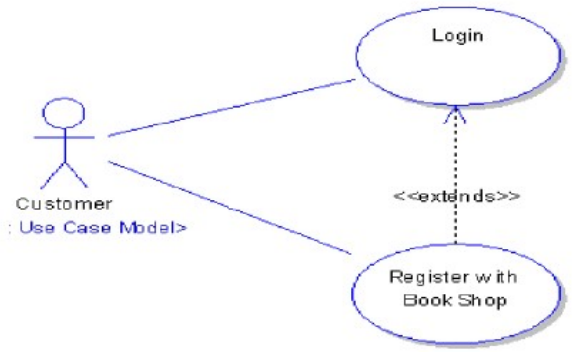
3

Use Case Model

- The Use Case Model describes the proposed functionality of the new system
- A Use Case represents a discrete unit of interaction between a user (human or machine) and the system
- A Use Case is a single unit of meaningful work;

4

Use Case Model



- For example login to system, register with system and create order are all Use Cases
- **A Use Case may 'include' another Use Case's functionality or 'extend' another Use Case with its own behaviour.**

5

Use Case Model

- A Use Case description will generally include:
  - General comments and notes describing the use case
  - Requirements - Things that the use case must allow the user to do, such as <ability to update order>, <ability to modify order> & etc

6

What the Use Case Model may include

- Constraints- Rules about what can and can't be done. Includes
  - i) pre-conditions that must be true before the use case is run -e.g. <create order> must precede <modify order>;
  - ii) post-conditions that must be true once the use case is run e.g. <order is modified and consistent>;
  - iii) invariants: these are always true - e.g. an order must always have a customer number

7

What the Use Case Model may include

- Scenarios - Sequential descriptions of the steps taken to carry out the use case. May include multiple scenarios, to cater for exceptional circumstances and alternate processing paths;
- Scenario diagrams -Sequence diagrams to depict the workflow – as above but graphically portrayed.
- Additional attributes such as implementation phase, version number, complexity rating, stereotype and status

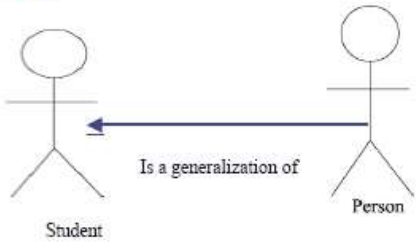
8

An Actor

- An Actor is a user of the system
- This includes both human users and other computer systems
- An Actor uses a Use Case to perform some piece of work which is of value to the business
- The set of Use Cases an actor has access to defines their overall role in the system and the scope of their action

9

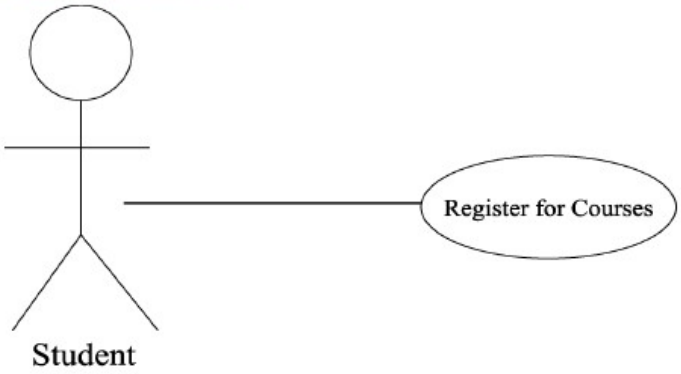
An Actor



- Actors can participate in a generalization relation with other actors

10

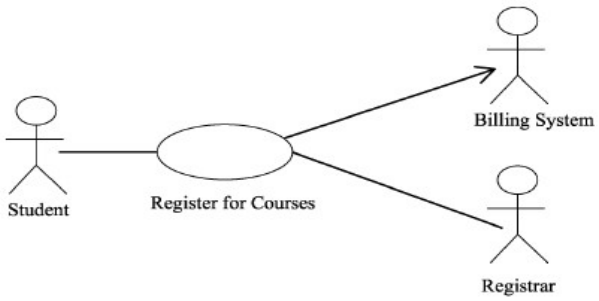
Actor cont.....



- Actors may be connected to use cases only by associations

11

Use Case Model Example



- Here we have a *Student* interacting with the *Registrar* and the *Billing System* via a *"Register for Courses"* use case

12

Constrains, Requirements & Scenarios

- The formal specification of a Use Case includes:
- Requirements.
  - These are the formal functional requirements that a Use Case must provide to the end user. They correspond to the functional specifications found in structured methodologies. A requirement is a contract that the Use Case will perform some action or provide some value to the system

13

Formal Specifications for a use case

- Constraints.
  - These are the formal rules and limitations that a Use Case operates under, and includes pre- post-and invariant conditions. A pre-condition specifies what must have already occurred or be in place before the Use Case may start. A post-condition documents what will be true once the Use Case is complete. An invariant specifies what will be true throughout the time the Use Case operates

14

Formal Specifications for a use case

- Scenarios.
  - These are formal descriptions of the flow of events that occurs during a Use Case instance.
  - Usually described in text and correspond to a textual representation of the Sequence Diagram
- **Includes and Extends relationships between Use Cases**
  - May include the functionality of another as part of its normal processing
  - it is assumed that the included Use Case will be called every time the basic path is run
  - example <list orders> Use Case may be included every time the <modify order> Use Case is run.
  - A Use Case may be included by one or more Use Cases. This helps to reduce duplication of functionality by factoring out common behavior into Use Cases that are re-used many times

15

Sequence Diagrams

- A *sequence diagram* is an interaction diagram that emphasizes the time ordering of messages
- It shows a set of objects and the messages sent and received by those objects
- Sequence diagrams can be used to document use case scenarios
- Captures required objects early in analysis and verify object usage later in design
- Shows the flow of messages from one object to another, and as such correspond to the methods and events supported by a class/object

16

Elements of a Sequence Diagrams

- Graphically, a sequence diagram is a table that shows objects arranged along the X axis and messages, ordered in increasing time, along the Y axis

17

Incorrect PIN Scenario

- 1. The Customer inserts a bank card, the Card Input sends the card's information to the Card Transaction Handler, which detects that the card is valid (not invalid; if no message is returned, the card is assumed valid).
- 2. The Card Transaction Handler instructs the Graphical User Interface (GUI) to display a message requesting the customer's Personal Identification Number (PIN).
- 3. The GUI requests the PIN and the customer enters his or her PIN, which is then passed to the Card Transaction Handler.
- 4. The Card Transaction Handler checks if the PIN is correct. In this scenario it is not, and the GUI is instructed to inform the customer that the PIN is invalid.

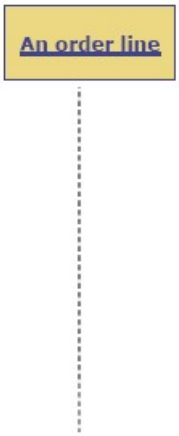
18

SOLUTION

- **SOLUTION!!!!**

19

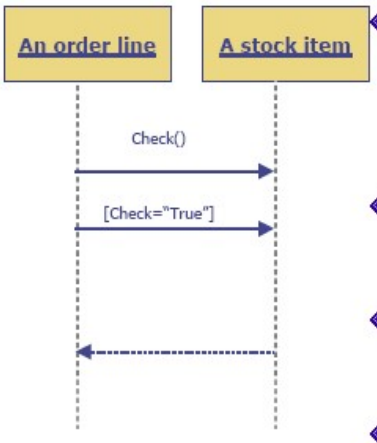
Sequence Diagrams –Object symbols



- An object in a sequence diagram is rendered as a box with a dashed line descending from it.
- The line is called the *object lifeline,* and it
- represents the existence of an object over a period of time

20

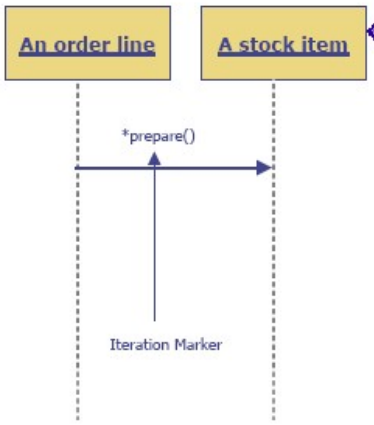
Sequence Diagrams –Message indicators



- Messages are rendered as horizontal {check()}arrows being passed from object to object as time advances down the object lifelines.
- Conditions (such as[check = "true"]) indicate when a message gets passed.
- Notice that the bottom arrow is not solid, and there is no accompanying message
- This arrows indicates return from a previous message not a new message

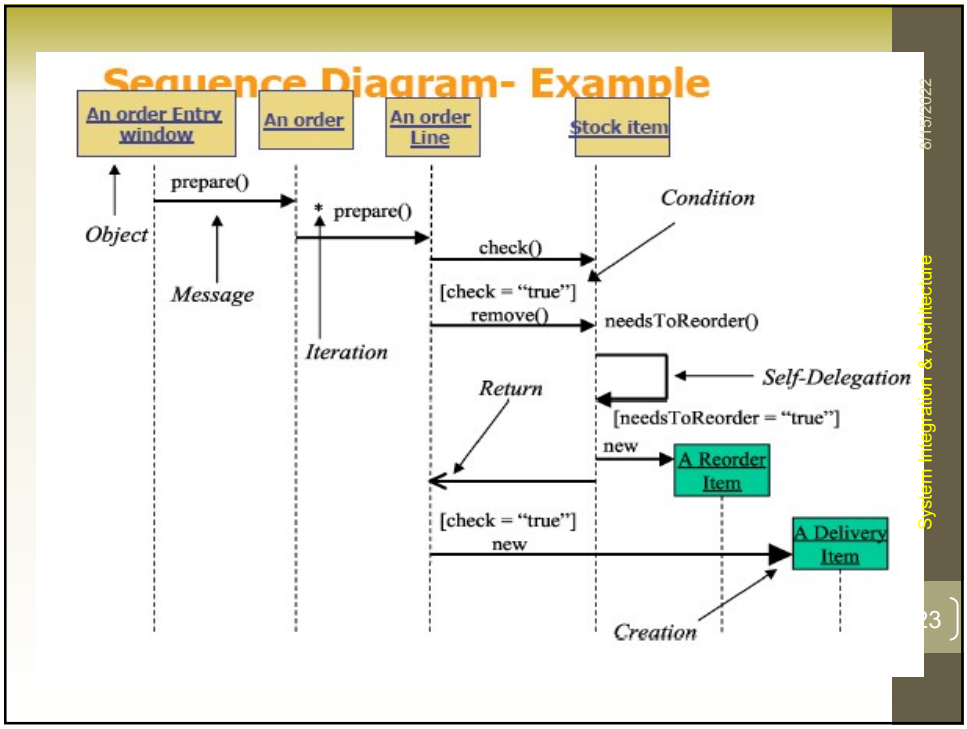
21

Sequence Diagrams – Iteration Marker

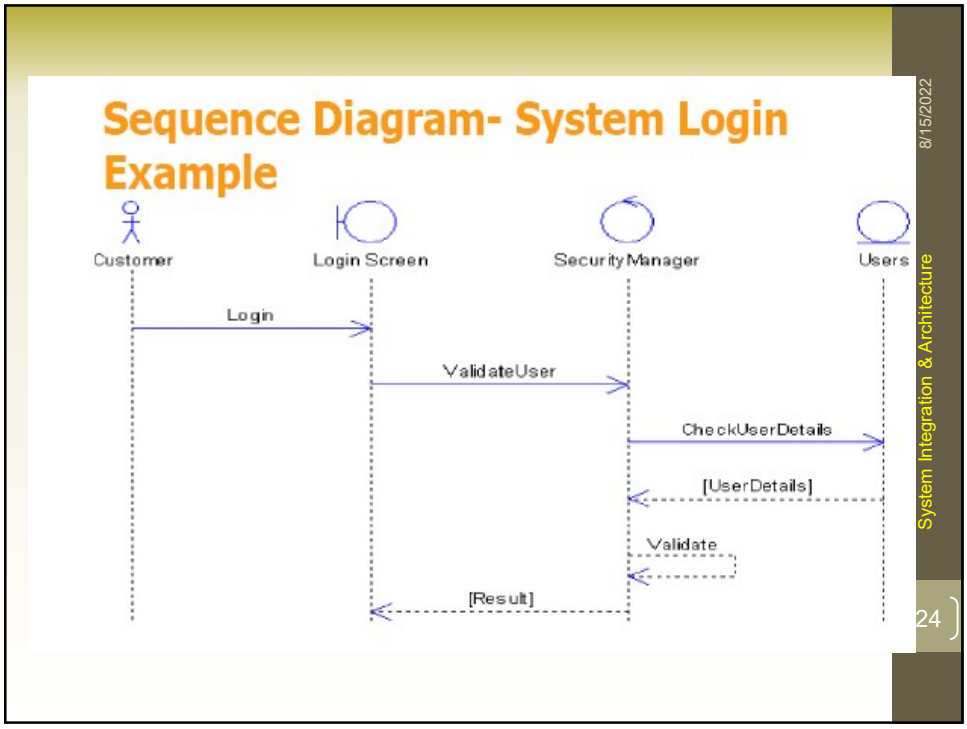


- An iteration marker, such as * (as shown) or *[I=1…n], indicates that a message will be repeated as indicated

22


Sequence Diagram- Example
An order Entry window
An order
An order Line
Stock item
prepare()
* prepare()
Condition
Object
check()
[check = "true"]
remove()
needsToReorder()
Message
Iteration
Self-Delegation
Return
[needsToReorder = "true"]
new
A Reorder Item
[check = "true"]
new
A Delivery Item
Creation
System Integration & Architecture
8/15/2022
23

23


Sequence Diagram- System Login Example
Customer
Login Screen
Security Manager
Users
Login
ValidateUser
CheckUserDetails
[UserDetails]
Validate
[Result]
System Integration & Architecture
8/15/2022
24

24

Activity Diagram

- The activity diagram is to model the procedural flow of actions that are part of a larger activity
- Activity diagrams can also be used to model system-level functions,
- The activity diagram focuses on the action sequence of execution and the conditions that trigger or guard those actions.

25

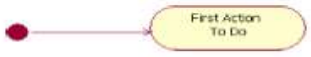
Activity Diagram

- The activity diagram's notation is very similar to that of a state machine diagram

- An action is indicated on the activity diagram by a "capsule" shape -- a rectangular object with semicircular left and right ends
- The text inside it indicates the action (e.g., Customer Calls Ticket Office or Registration Office Opens).

26

Activity Diagram



- The initial state is drawn as a solid circle with a transition line (arrow) that connects it to the first action in the activity's sequence of actions.
- It is important to note that there can *be only one* initial state on an activity diagram and *only one* transition line connecting the initial state to an action.

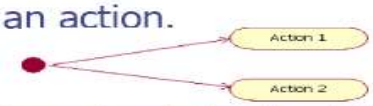
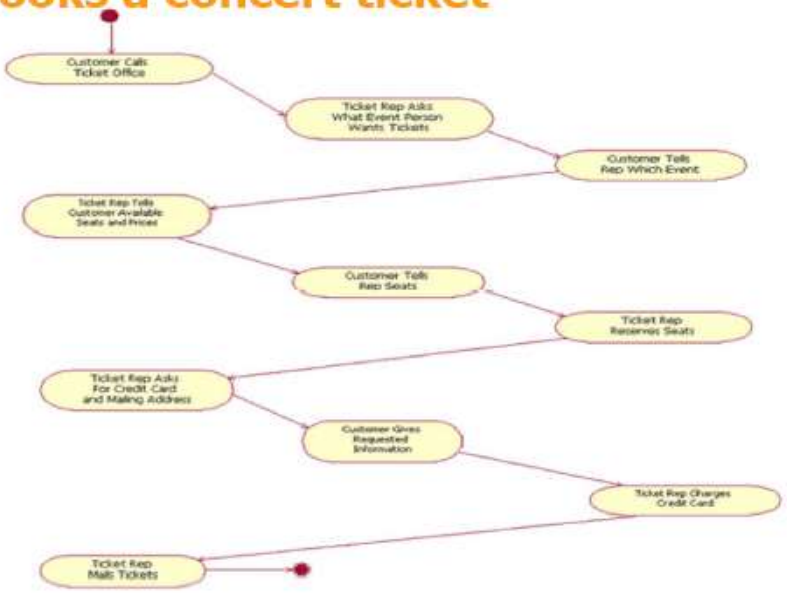


Figure indicate Incorrect rendering of an initial state within an activity diagram.

Activity Diagram

- With arrows indicating direction, the transition lines on an activity diagram show the sequential flow of actions in the modeled activity
- The arrow will always point to the next action in the activity's sequence.

Activity Diagram – How a customer books a concert ticket



29

Book a concert- activity diagram

- The sample activity diagram in previous slide documents the activity "Booking a Concert Ticket," with actions in the following order
  1. Customer calls ticket office.
  2. Ticket rep asks what event person wants tickets for.
  3. Customer tells rep event choice.
  4. Ticket rep tells customer available seats and prices.
  5. Customer tells rep seating choice.
  6. Ticket rep reserves seats.
  7. Ticket rep asks for credit card and billing address.
  8. Customer gives requested information.
  9. Ticket rep charges credit card.
  10. Ticket rep mails tickets

30

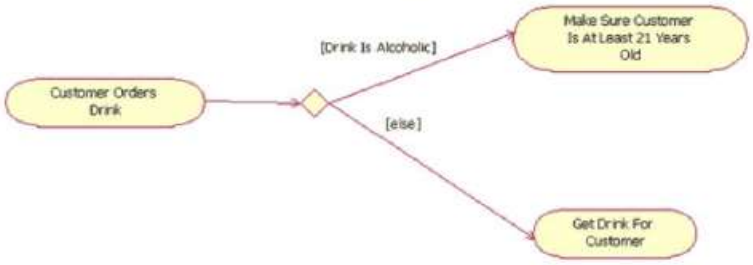
Activity Diagram

- The action order is clear from the diagram, because the diagram shows an initial state (starting point), and from that point one can follow the transition lines as they connect the activity's actions
- It is possible for an activity diagram to show multiple final states. Unlike initial state symbols, of which there can be only one on an activity

31

Activity Diagram- Decision points

- Typically, decisions need to be made throughout an activity, depending on the outcome of a specific prior action



32

Activity Diagram- Decision points

- Each transition line involved in a decision point must be labeled with text above it to indicate "guard conditions," commonly abbreviated as *guards.*
- *Guard condition text is always placed in brackets -- for example, [guard condition text]*
- *A guard condition explicitly tells when to follow a transition line to the next action*

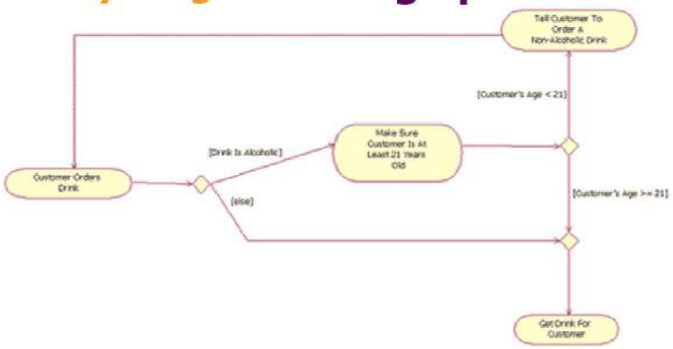
33

Activity Diagram- Merge points

- Sometimes the procedural flow from one decision path may connect back to another decision path
- In these cases, we connect two or more action paths together using the same diamond icon with multiple paths pointing to it, but with only one transition line coming out of it
- This indicate a *merge* decision point.

34

Activity Diagram- Merge points



A partial activity diagram, showing two decision points ("Drink is alcoholic" and "Customer's age < 21 ") and one merge ("else" and "Customer's age >= 21 ")

35

Activity Diagram-Synch states

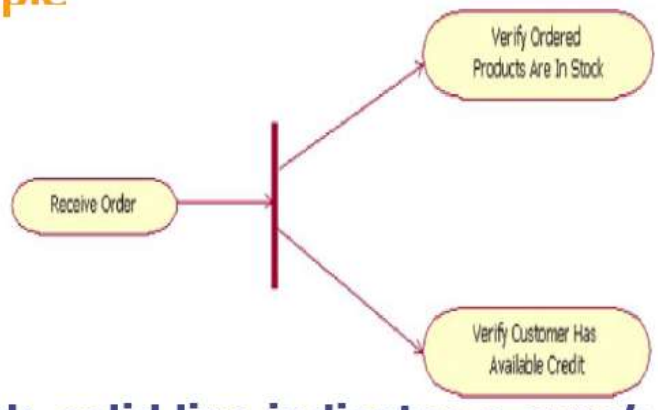
- When modeling activities, you sometimes need to show that certain action sequences can be done in parallel, or asynchronously
- *Synch states,* as they are officially named allows you to show parallel action sequences
- Synch states allow the forking and joining of execution threads
- A synch state that forks actions into two or more threads represents a de-synchronizing of the flow (asynchronous actions)
- A synch state that joins actions back together represents a return to synchronized flow

36

Activity Diagram-Synch states

- A synch state is drawn as a thick, solid line with transition lines coming into it from the left (usually) and out of it on the right (usually).
- To draw a synch state that forks the action sequence into multiple threads, first connect a transition line from the action preceding the parallel sequence to the synch state. Then draw two transition lines coming out of the synch state, each connecting to its own action

Activity Diagram-Synch states example



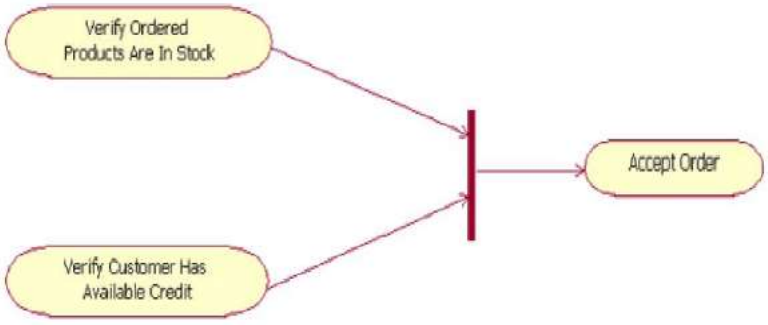
A thick, solid line indicates a *synch state allowing two or more action sequences to proceed in parallel.*

Activity Diagram-Synch states example

- In the previous slides after the action "Receive Order" is completed, two threads are kicked off in parallel
- This allows the system to process both the "Verify Ordered Products Are In Stock" and
- the "Verify Customer Has Available Credit" actions at the same time
- When you fork execution into multiple threads, typically you have to rejoin them at some point for later processing
- The synch state element is also used to denote multiple threads joining back together into a single thread.

39

Activity Diagram-Synch states example



When parallel action sequences terminate, a synch state(thick line) is used to indicate that the multiple threads are joined back into a single thread

40

Activity Diagram-Synch states

- In the previous slide, the "Verify Ordered Products Are In Stock" and the "Verify Customer Has Available Credit" actions have completed processing,[4] and the "Accept Order" action is processed
- Note that the single transition line coming out of the synch state means there is now only one thread of execution
- Next slide shows an activity diagram fragment that uses a synch state as a synchronization point.

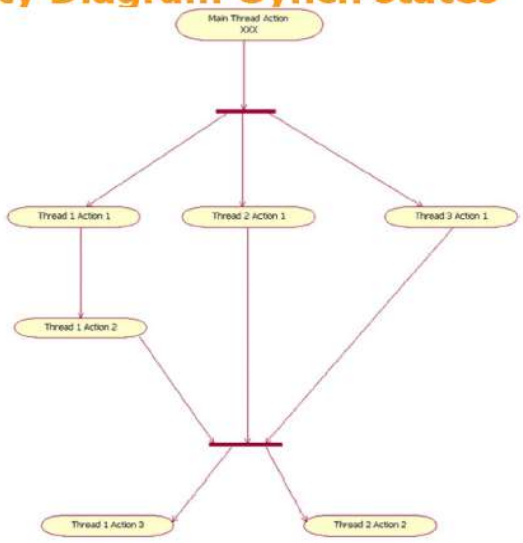
41

Activity Diagram-Synch states

- In the previous, The activity starts with all the action in one thread, "Main Thread Action XXX" and
- When the action is done, the activity breaks into three threads executing in parallel.
- In the first thread, "Thread 1 Action 1" is executed,
- then "Thread 1 Action 2" is executed.
- At the same time this first thread is executing, the second and third thread actions are being executed --"Thread 2 Action 1" and "Thread 3 Action 1," respectively.

42

Activity Diagram-Synch states



43

Activity Diagram-Synch states

- Remember that when multiple threads are executing, the actions in one thread must not impact the actions executing in a parallel thread.
- In the Figure, the action "Thread 1 Action 1" may be done quickly, and "Thread 1 Action 2" could begin processing before "Thread 2 Action 1" is complete.
- The only thing that will cause threads to wait for another parallel thread is a synch state, placed as shown in the Figure.

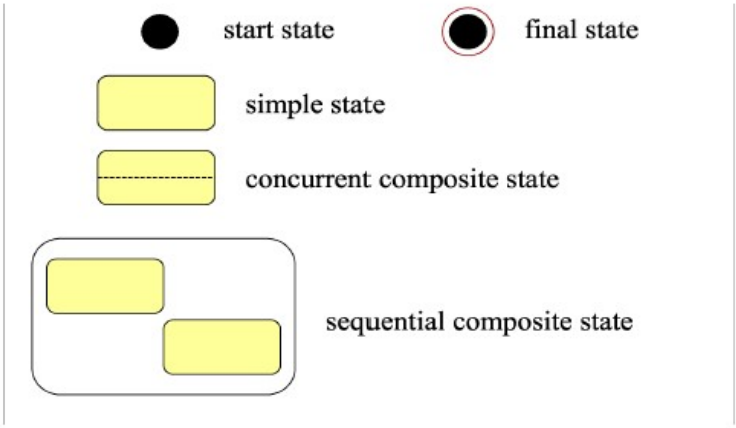
44

State Machine Chart

- The state machine view describes the dynamic behavior of objects over time by modeling the lifecycles of objects of each class.
- Each object is treated as an isolated entity that communicates with the rest of the world by detecting events and responding to them.
- Events represent the kinds of changes that objects can detect.
- Anything that can affect an object can be characterized as an event.

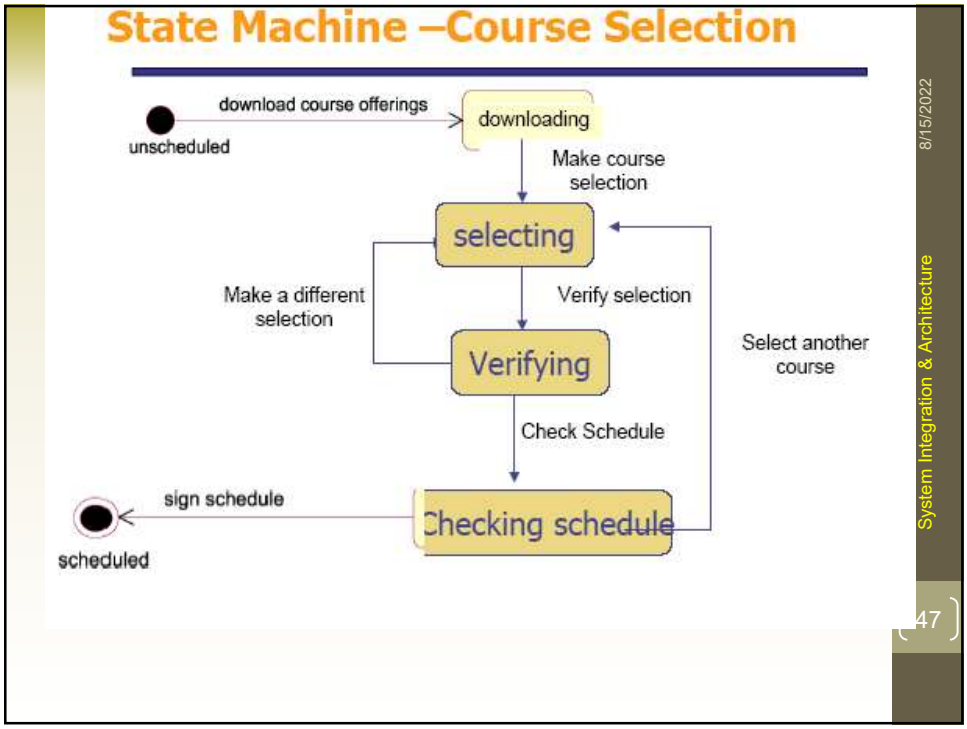
45

State Machine Chart Notations



- There can be only one start state in a state diagram, but there maybe many intermediate and final states.

46


State Machine –Course Selection
download course offerings
unscheduled
downloading
Make course selection
selecting
Make a different selection
Verify selection
Verifying
Select another course
Check Schedule
sign schedule
scheduled
Checking schedule
8/15/2022
System Integration & Architecture
47

47