Software Design

UNIT III

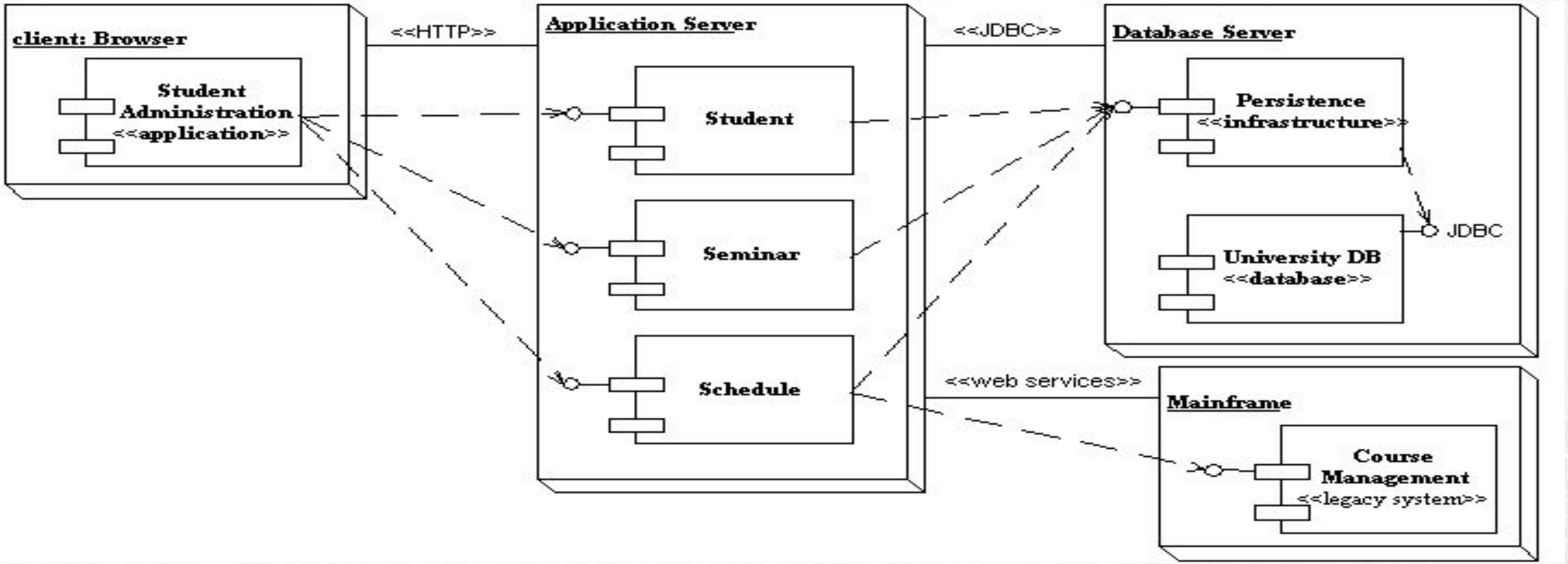
S.Y. Semester IV

Syllabus – Unit Iii

Dynamic Modeling

Use case diagram, Activity diagram- Interaction & Interaction overview diagram, sequence diagram, Timing diagram, Communication diagram, Advance state machine diagram

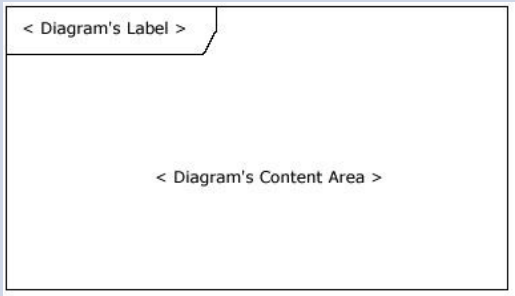
Sample Communication Links

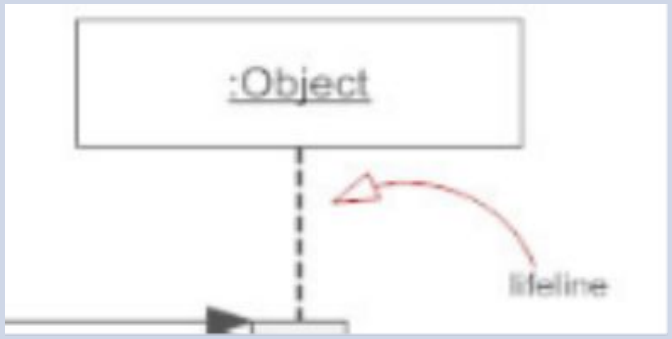
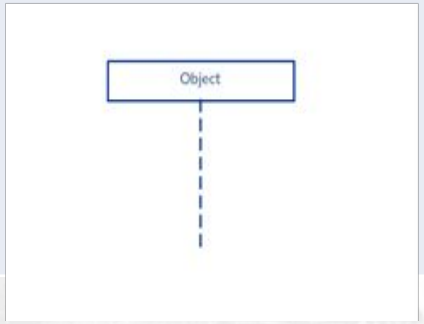


UML Sequence Diagrams

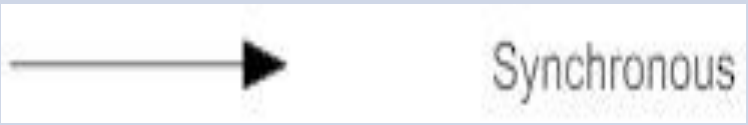
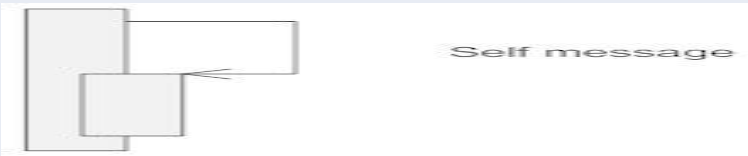
- Sequence diagrams model the dynamic aspects of a software system
- The emphasis is on the "sequence" of messages rather than relationship between objects
- Sequence diagrams provide more detail and show the messages exchanged among a set of objects over time.
- The main purpose of this diagram is to represent how different business object interacts

UML Sequence Diagrams

S.No	Name	Description	Notation
1	Class Roles or Participants	Class roles describe the way an object will behave in context	
2	Activation or Execution Occurrence/ Scope	Activation boxes represent the time an object needs to complete a task.	
3	Diagram Boundry		

S.No	Name	Description	Notation
3	Messages	Messages are arrows that represent communication between objects.	
4	Lifelines	Lifelines represent either roles or object instances that participate in the sequence being modeled.	

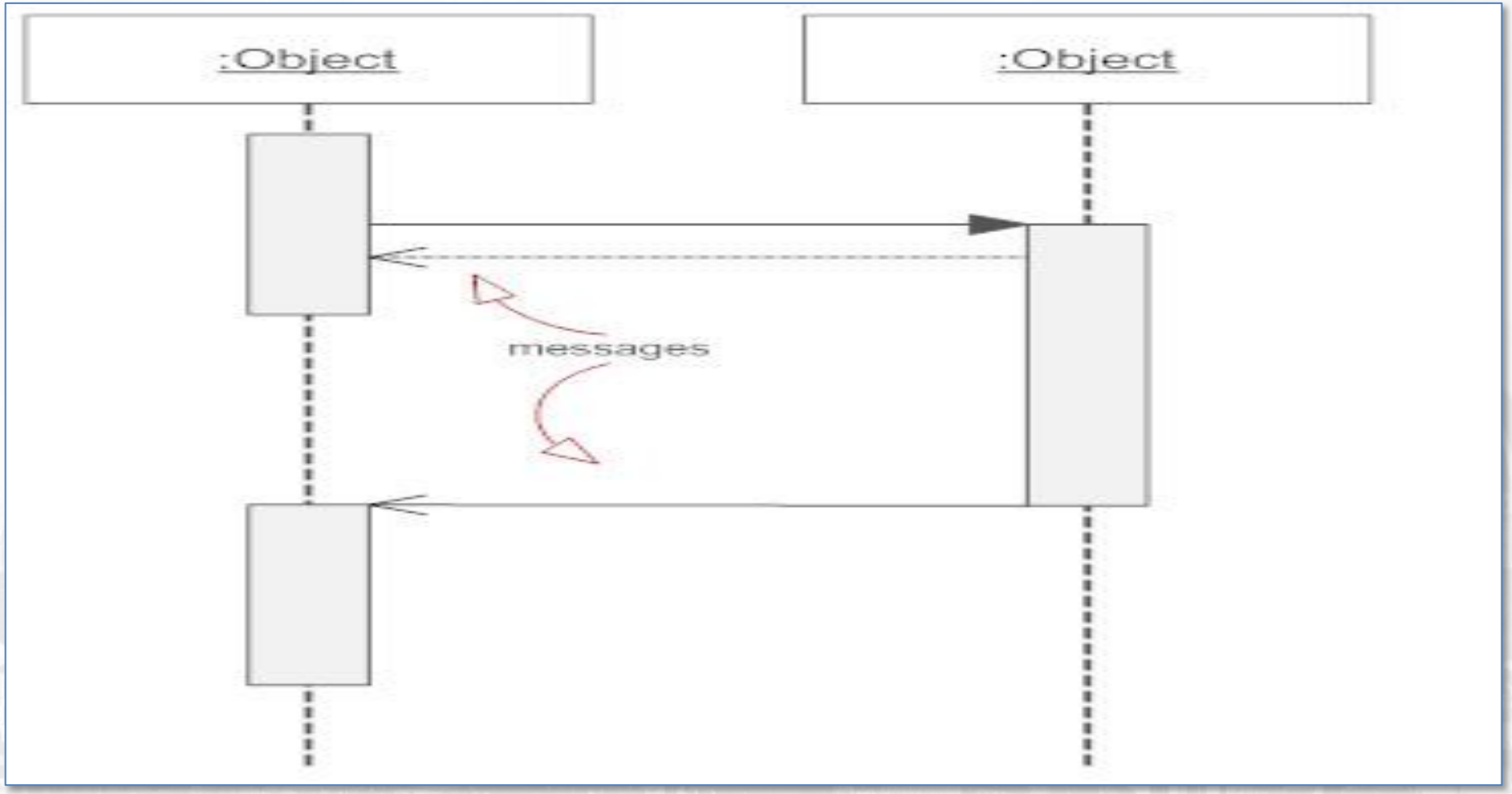
UML Sequence Diagrams- Types of Messages in Sequence Diagrams

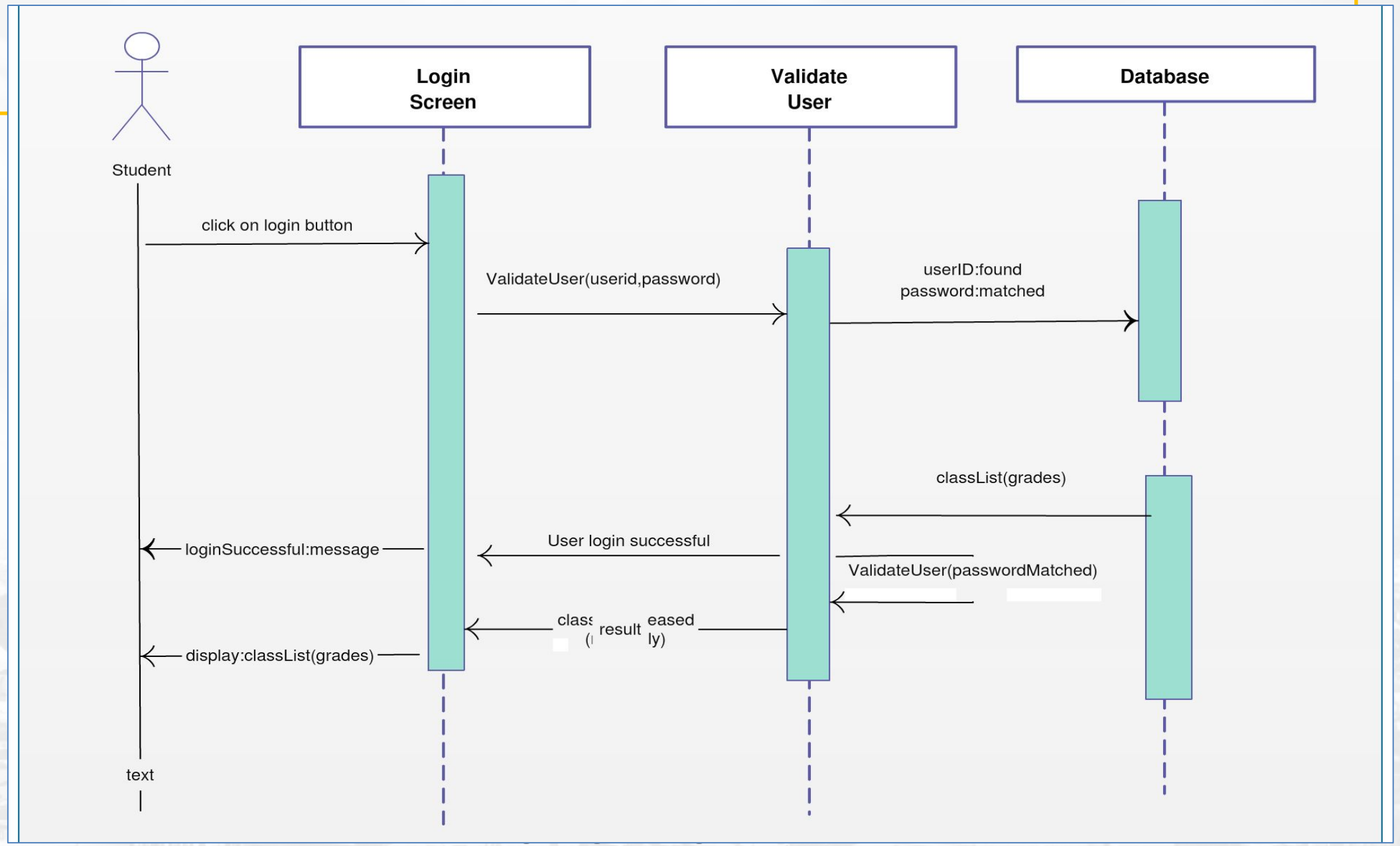
S.No	Name	Description	Notation
1	Synchronous Message	A synchronous message requires a response before the interaction can continue.	
2	Asynchronous Message	Asynchronous messages don't need a reply for interaction to continue.	
3	Reply or Return Message	A reply message is drawn with a dotted line and an open arrowhead pointing back to the original lifeline.	
4	Self Message	A message an object sends to itself, usually shown as a U shaped arrow pointing back to itself.	

UML Sequence Diagrams

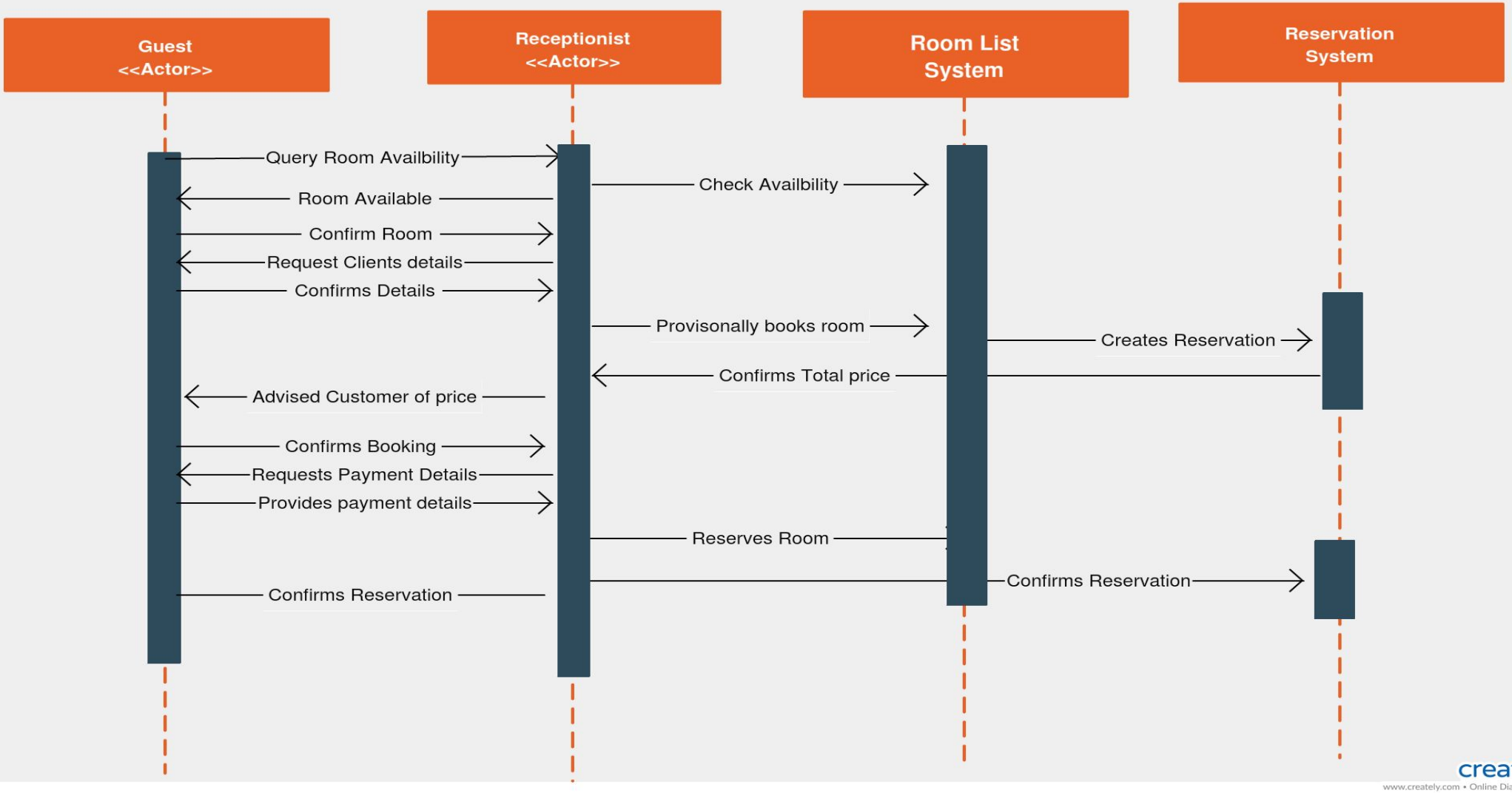
- Used during requirements analysis
  - To refine use case descriptions
  - to find additional objects (“participating objects”)
- Used during system design
  - to refine subsystem interfaces
- **Classes** are represented by columns
- **Messages** are represented by arrows
- **Activations** are represented by narrow rectangles
- **Lifelines** are represented by dashed lines

Drawing a Sequence Diagram




Student
Login Screen
Validate User
Database
click on login button
ValidateUser(userid,password)
userID:found
password:matched
classList(grades)
loginSuccessful:message
User login successful
ValidateUser(passwordMatched)
result
display:classList(grades)
text

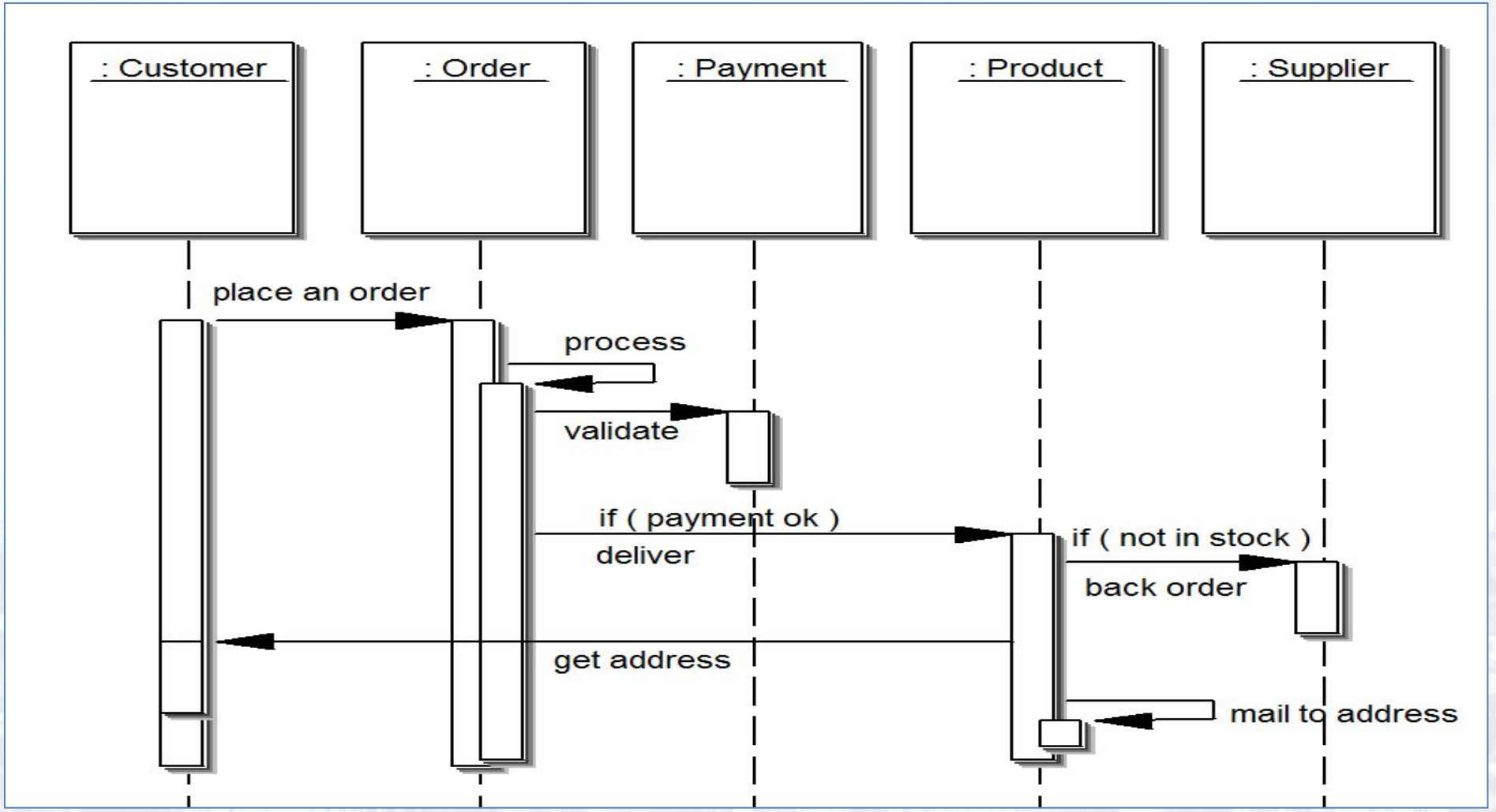
HOTEL RESERVATION SYSTEM



Sequence Diagram

- A sequence diagram is a good way to visualize and validate various runtime scenarios.
- These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modeling a new system.

Order System



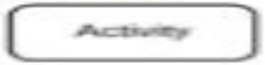
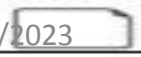
Activity Diagram

- It shows the structure of a process
- Supplements the use-case by providing a diagrammatic representation of procedural flow
- Shows flow of control from activity to activity
- Main Components of activity diagram
  - Action
  - Activity Node
  - Branching
  - Forking & Joining

Activity Diagram

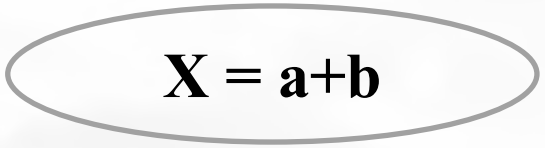
- Describes how activities are coordinated.
- Is particularly useful when you know that an operation has to achieve a number of different things, and you want to model what the essential dependencies between them are, before you decide in what order to do them.
- Records the dependencies between activities, such as which things can happen in parallel and what must be finished before something else can start.
- Represents the workflow of the process.

Activity diagram Notations

Initial Node 	A black circle is the standard notation for an initial state before an activity takes place. It can either stand alone or you can use a note to further elucidate the starting point.
Activity 	The activity symbols are the basic building blocks of an activity diagram and usually have a short description of the activity they represent.
Control Flow	Arrows represent the direction flow of the flow chart. The arrow points in the direction of progressing activities.
Branch 	A marker shaped like a diamond is the standard symbol for a decision. There are always at least two paths coming out of a decision and the condition text lets you know which options are mutually exclusive.
Fork 	A fork splits one activity flow into two concurrent activities
Join 	A join combines two concurrent activities back into a flow where only one activity is happening at a time.
	The final flow marker shows the ending point for a process in a flow. The difference between a final flow node and the end state node is that the latter represents the end of all flows in an activity.
 **Complete Activity Flow**	The black circle that looks like a selected radio button is the UML symbol for the end state of an activity. As shown in two examples above, notes can also be used to explain an end state.
Notes 	The shape used for notes.

Activity Diagram (Contd..)

- **Action:** Executable Computations

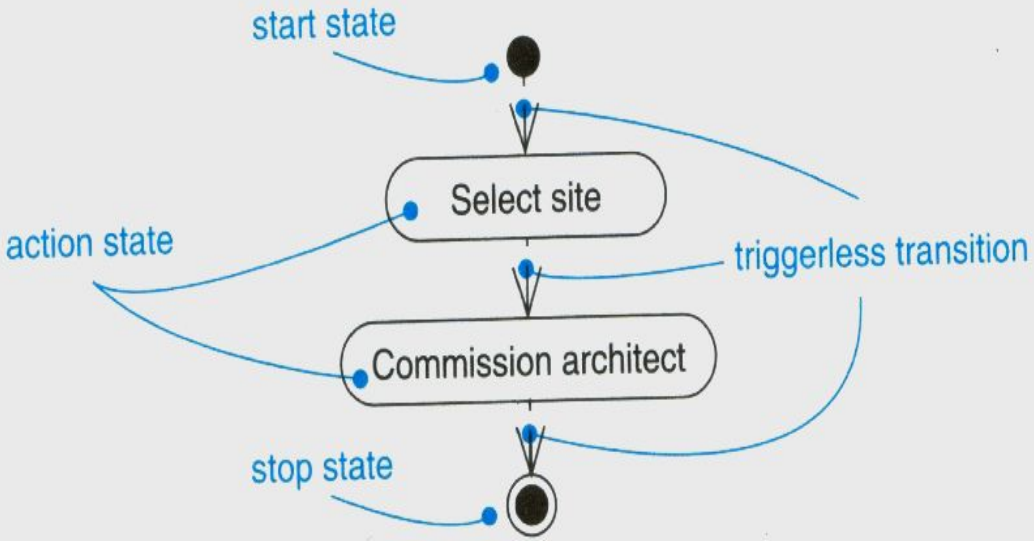


- **Activity Node:** Nested groupings of actions

- **Branching:** Specifies alternate paths taken based on some Boolean expression

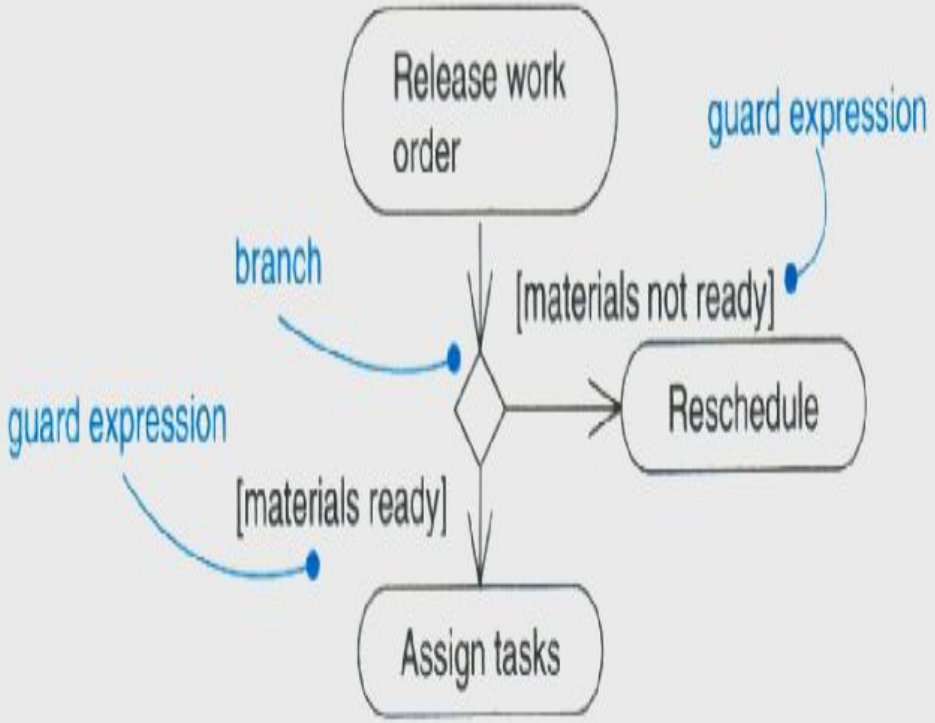
Transitions

- When the action or activity of a state completes, flow of control passes immediately to the next action or activity state
- A flow of control has to start and end someplace
  - initial state -- a solid ball
  - stop state -- a solid ball inside a circle



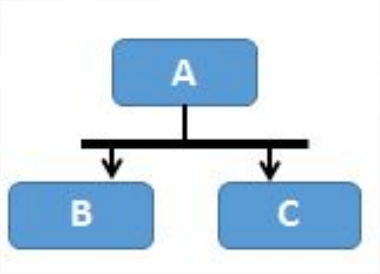
Branching

- A branch specifies alternate paths taken based on some Boolean expression
- A branch may have one incoming transition and two or more outgoing ones

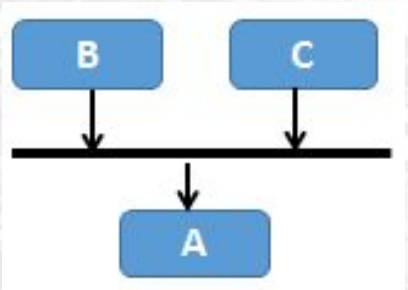


Activity Diagram (Contd..)

- **Forking & Joining:**
- **Fork:** Splitting of single flow of control into 2 or concurrent flow of control



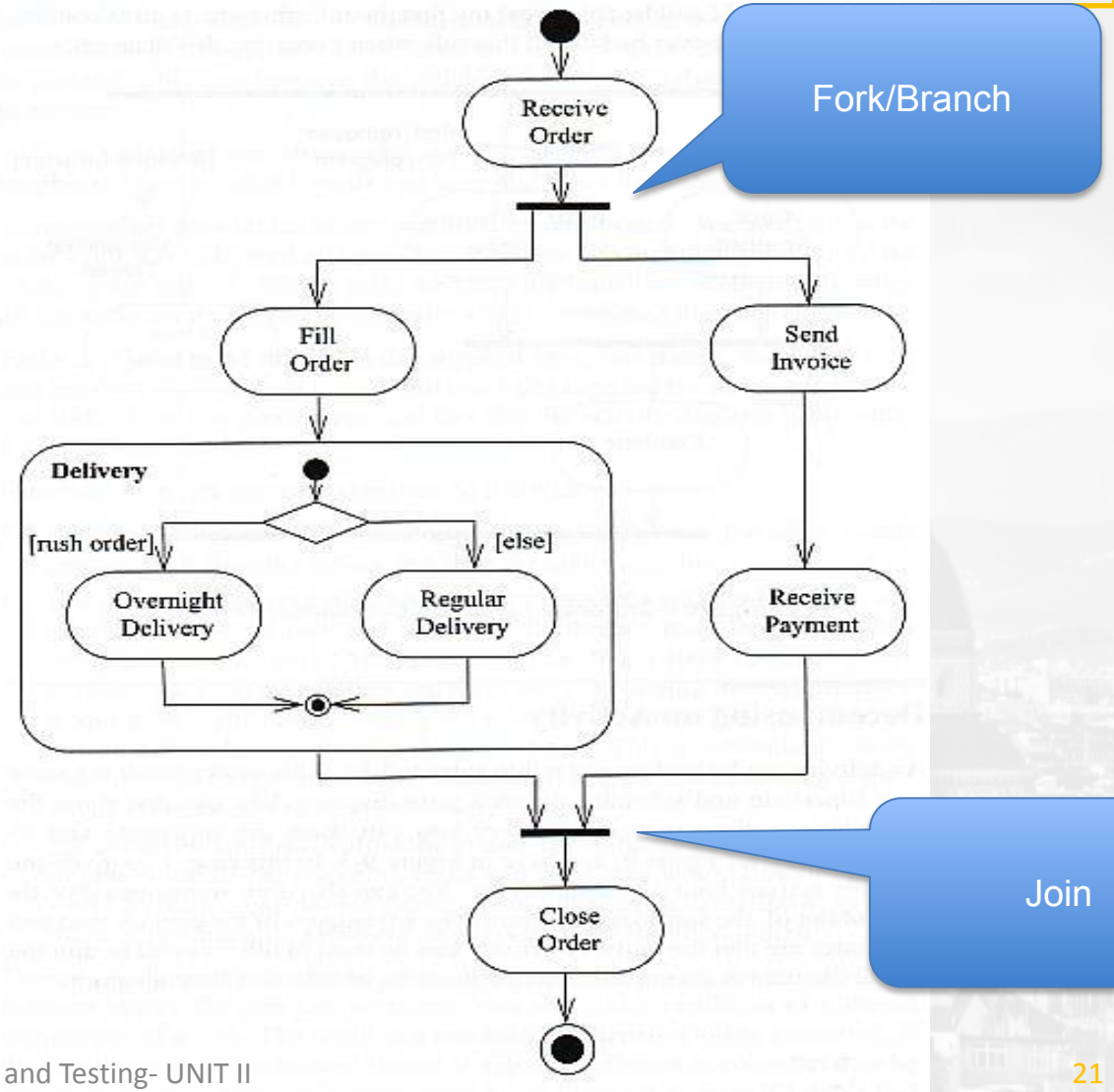
- **Join:** Synchronization of 2 or more concurrent flow of control

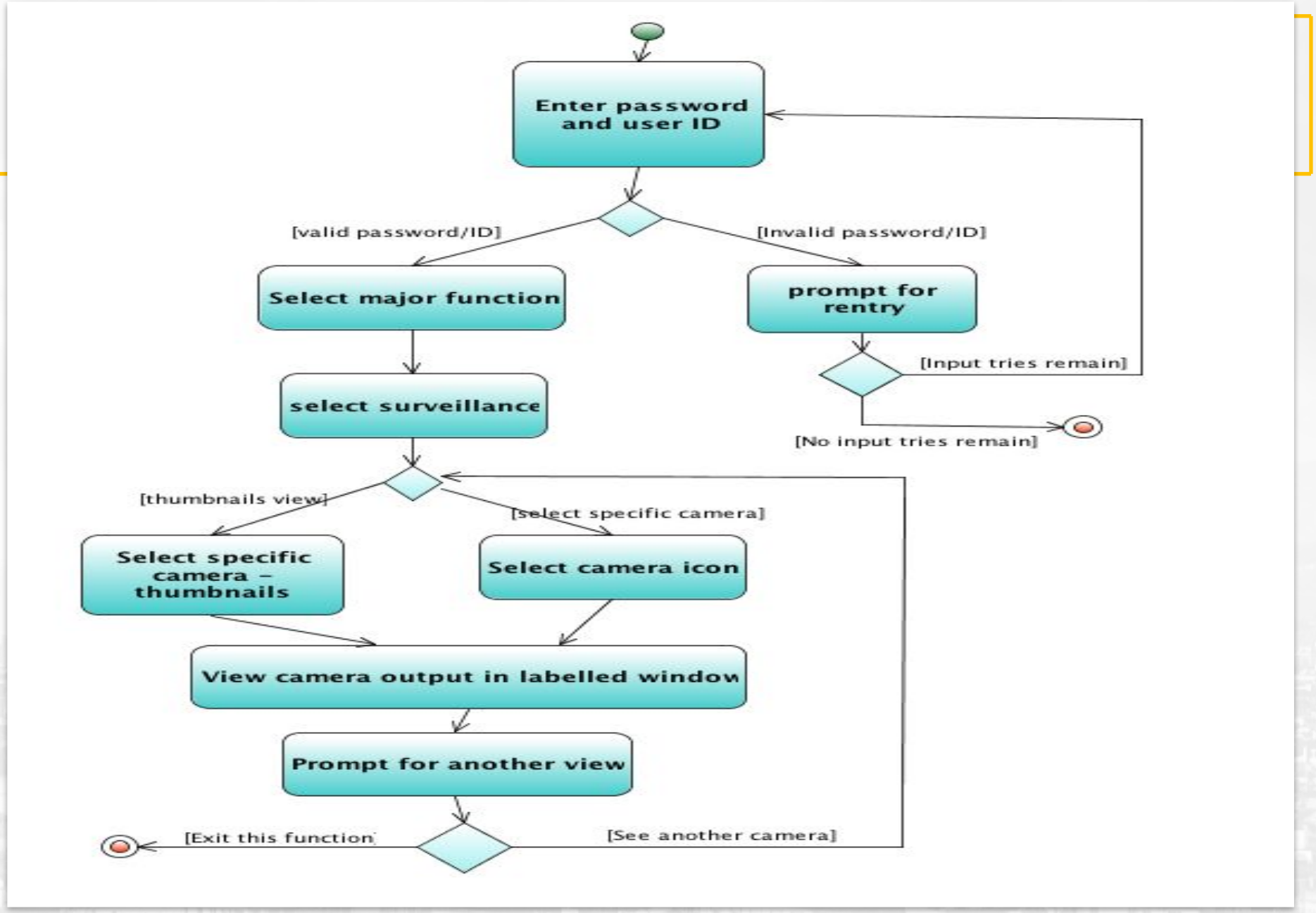


Activity Diagram Example

- To show concurrent activity, activity diagrams allow branches and joins.

- You can also reference or include other activity diagrams

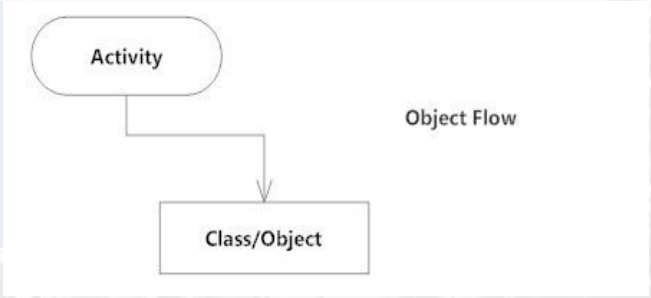



Enter password and user ID
[valid password/ID]
[Invalid password/ID]
Select major function
prompt for rentry
[Input tries remain]
select surveillance
[No input tries remain]
[thumbnails view]
[select specific camera]
Select specific camera – thumbnails
Select camera icon
View camera output in labelled window
Prompt for another view
[Exit this function]
[See another camera]

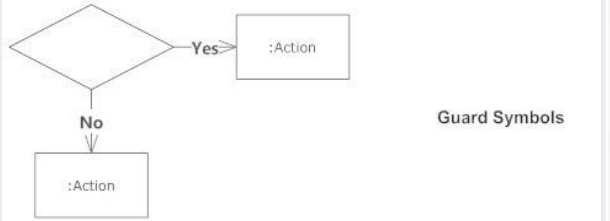
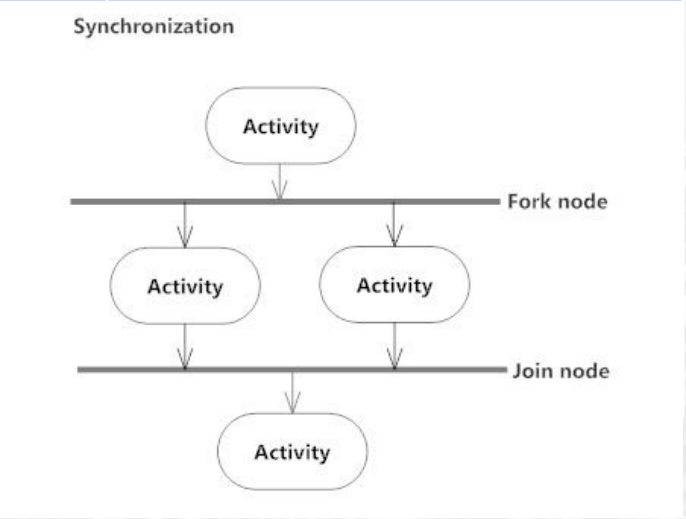
Activity Diagrams

- Normally employed in business process modelling.
- An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram.
- Activity diagrams are often used in business process modeling.
- They can also describe the steps in a use case diagram.
- Activities modeled can be sequential and concurrent.
- In both cases an activity diagram will have a beginning (an initial state) and an end (a final state).

Activity Diagrams

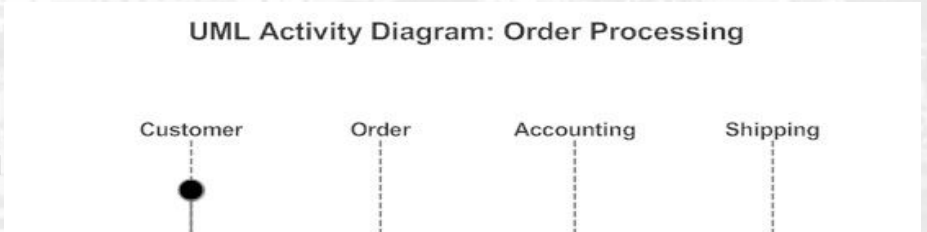
S.No	Name	Description	Notation
1	Initial State or Start Point	Represents the initial action state or the start point for any activity diagram	
2	Activity or Action State	An action state represents the non-interruptible action of objects.	
3	Action Flow	Action flows, also called edges and paths, illustrate the transitions from one action state to another	
4	Object Flow	Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object.	

Activity Diagrams

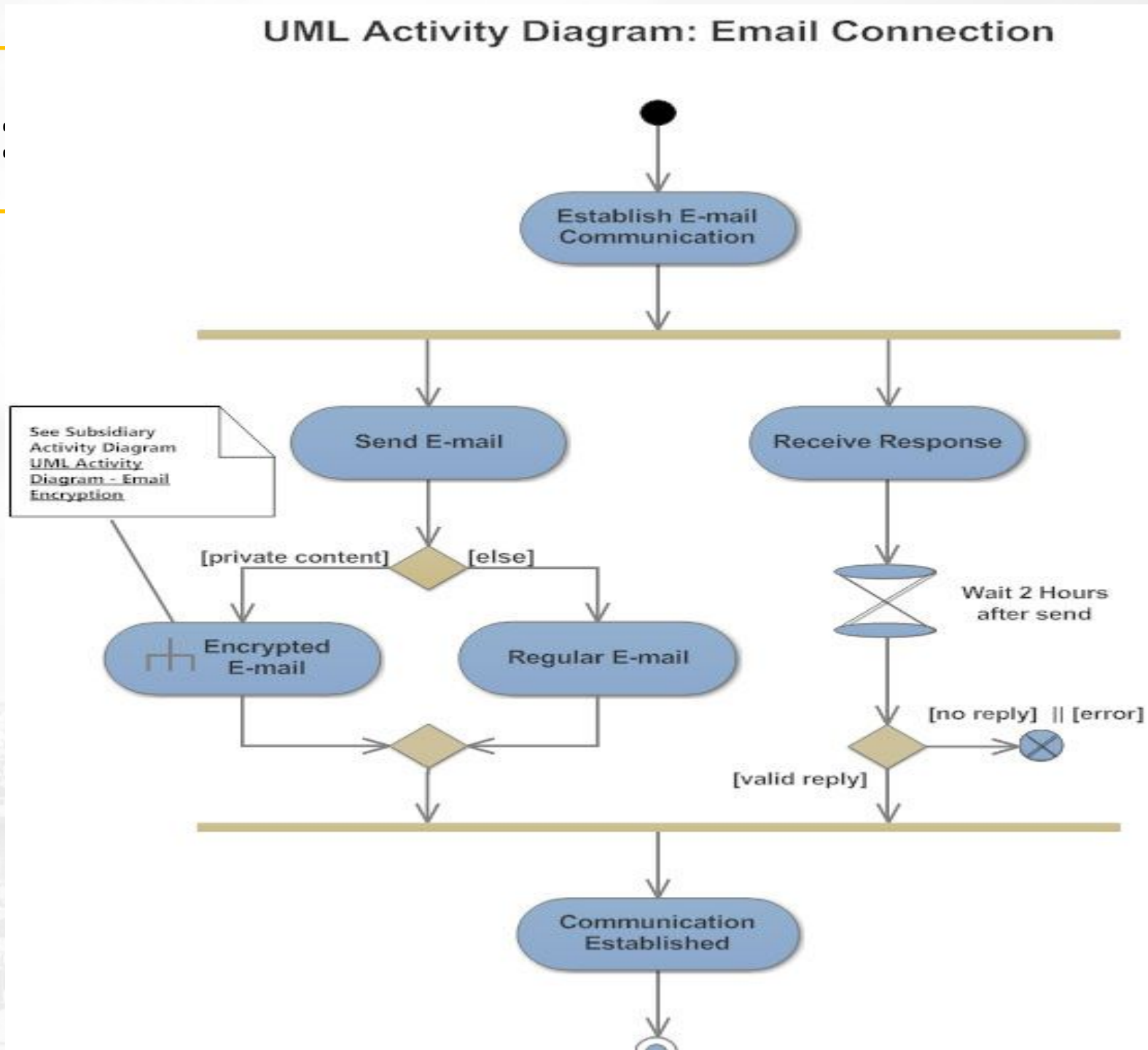
S.No	Name	Description	Notation
5	Decisions and Branching	When an activity requires a decision prior to moving on to the next activity, add a diamond between the two activities.	
6	Guards	In UML, guards are a statement written next to a decision diamond that must be true before moving next to the next activity.	
7	Synchronization	A fork node is used to split a single incoming flow into multiple concurrent flows. A join node joins multiple concurrent flows back into a single outgoing flow.	

Activity Diagrams

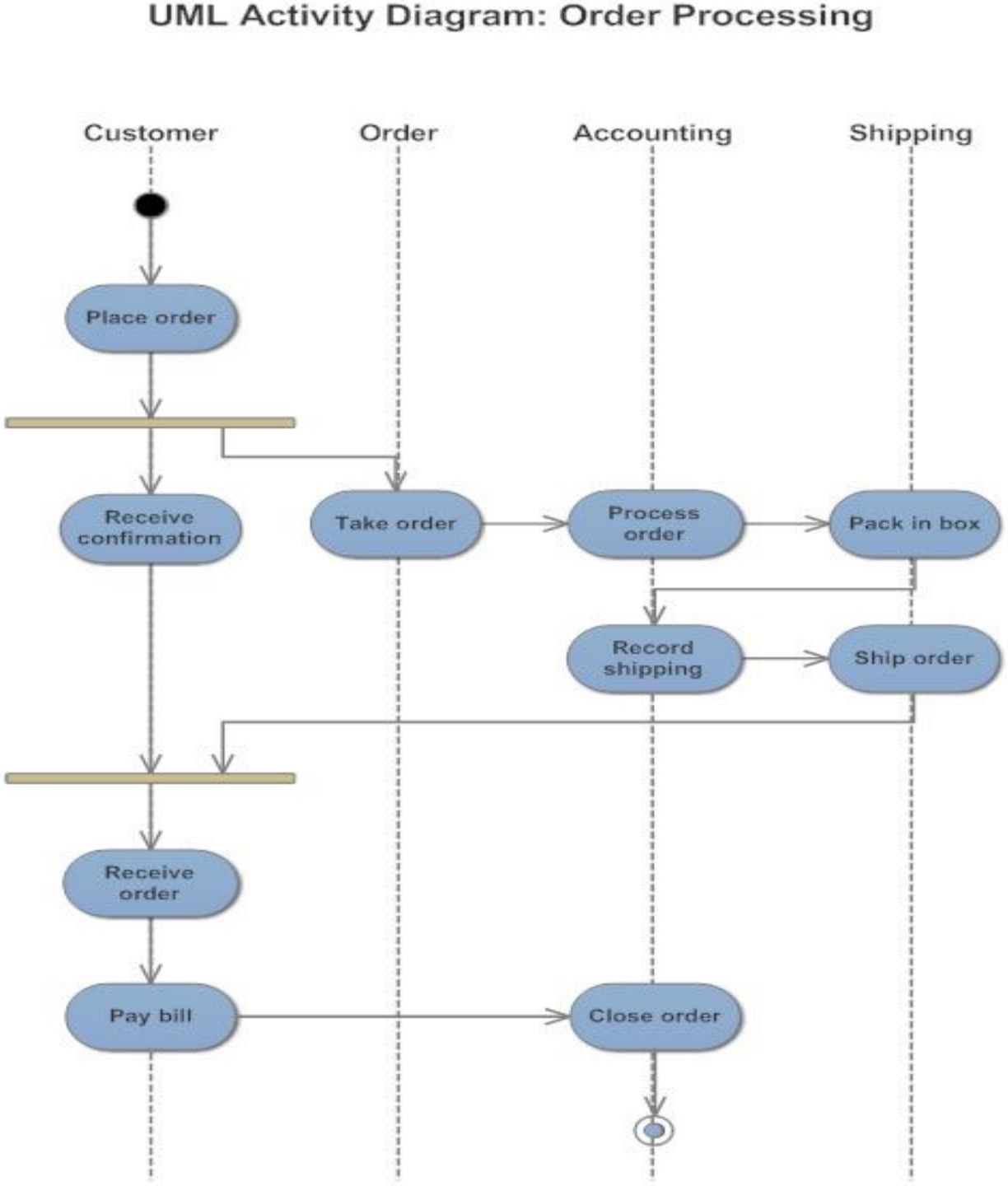
S.No	Name	Description	Notation
8	Merge Event	A merge event brings together multiple flows that are not concurrent.	
9	Final State or End Point	An arrow pointing to a filled circle nested inside another circle represents the final action state	
10	Swimlanes	Swimlanes group related activities into one column.	



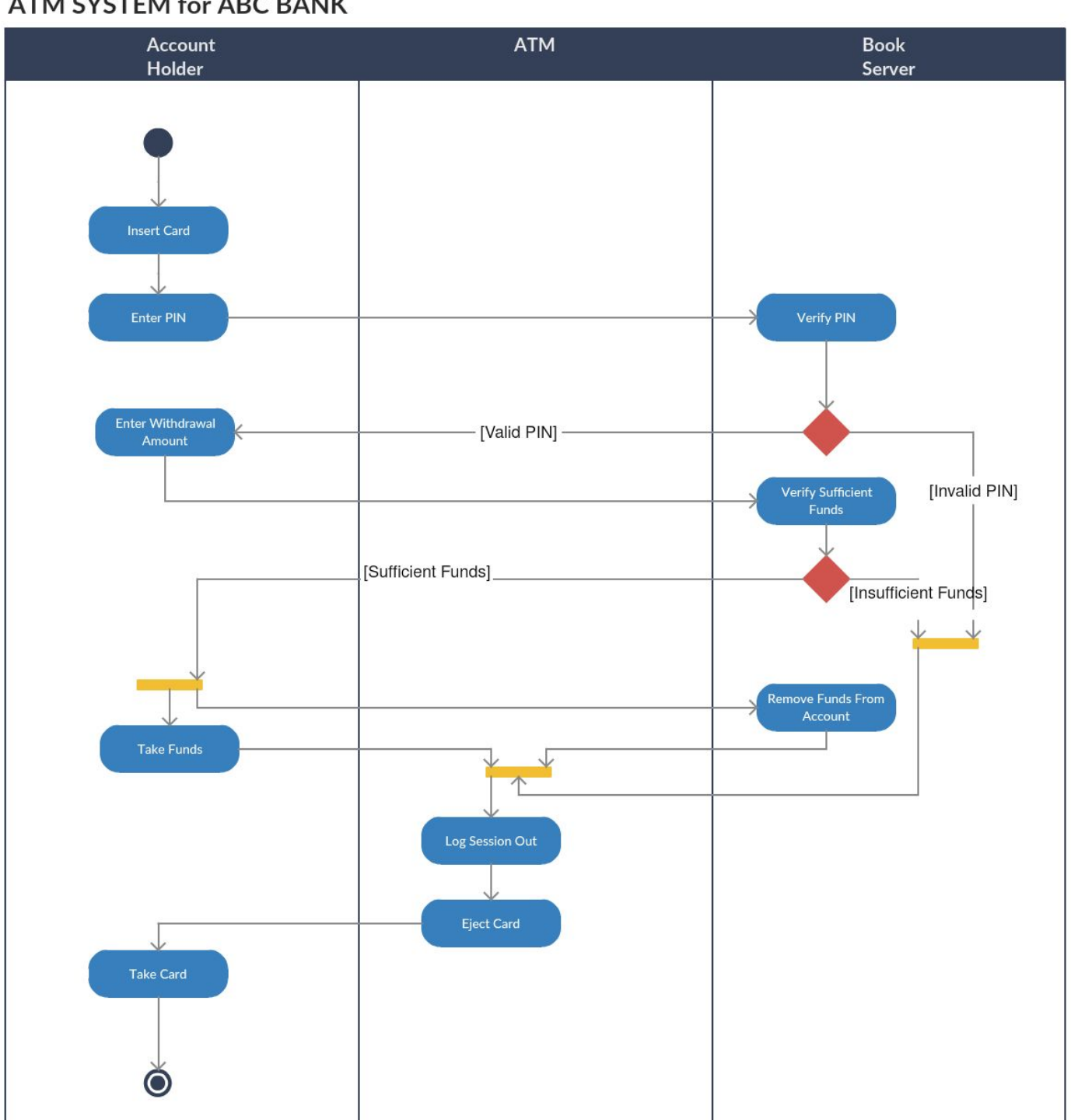
Example 1 :


UML Activity Diagram: Email Connection
Establish E-mail Communication
Send E-mail
Receive Response
See Subsidiary Activity Diagram UML Activity Diagram - Email Encryption
[private content]
[else]
Encrypted E-mail
Regular E-mail
Wait 2 Hours after send
[no reply] || [error]
[valid reply]
Communication Established

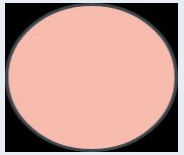
Example 2 : Food Ordering Processing with Swimlanes



Example 3: ABC Bank with swimlanes



Revision : Symbols of CFD

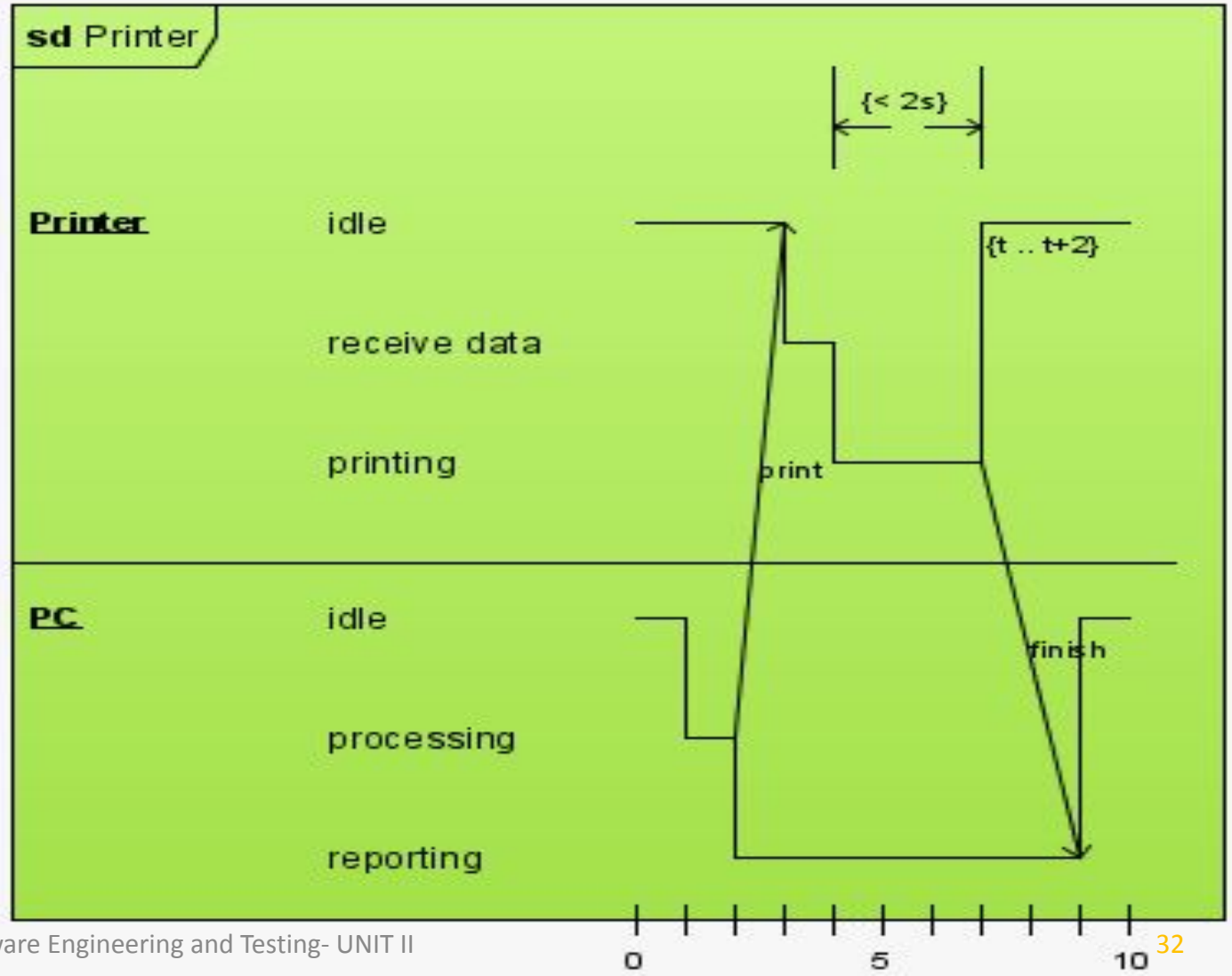
S.No	Name	Description	Notation Yourdon and Coad
1	**External Entity**	An outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.	
2	**Process**	Any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules	
3	**Data store**	Files or repositories that hold information for later use, such as a database table or a membership form.	
4	**Data flow**	The route that data takes between the external entities, processes and data stores.	

Timing diagrams

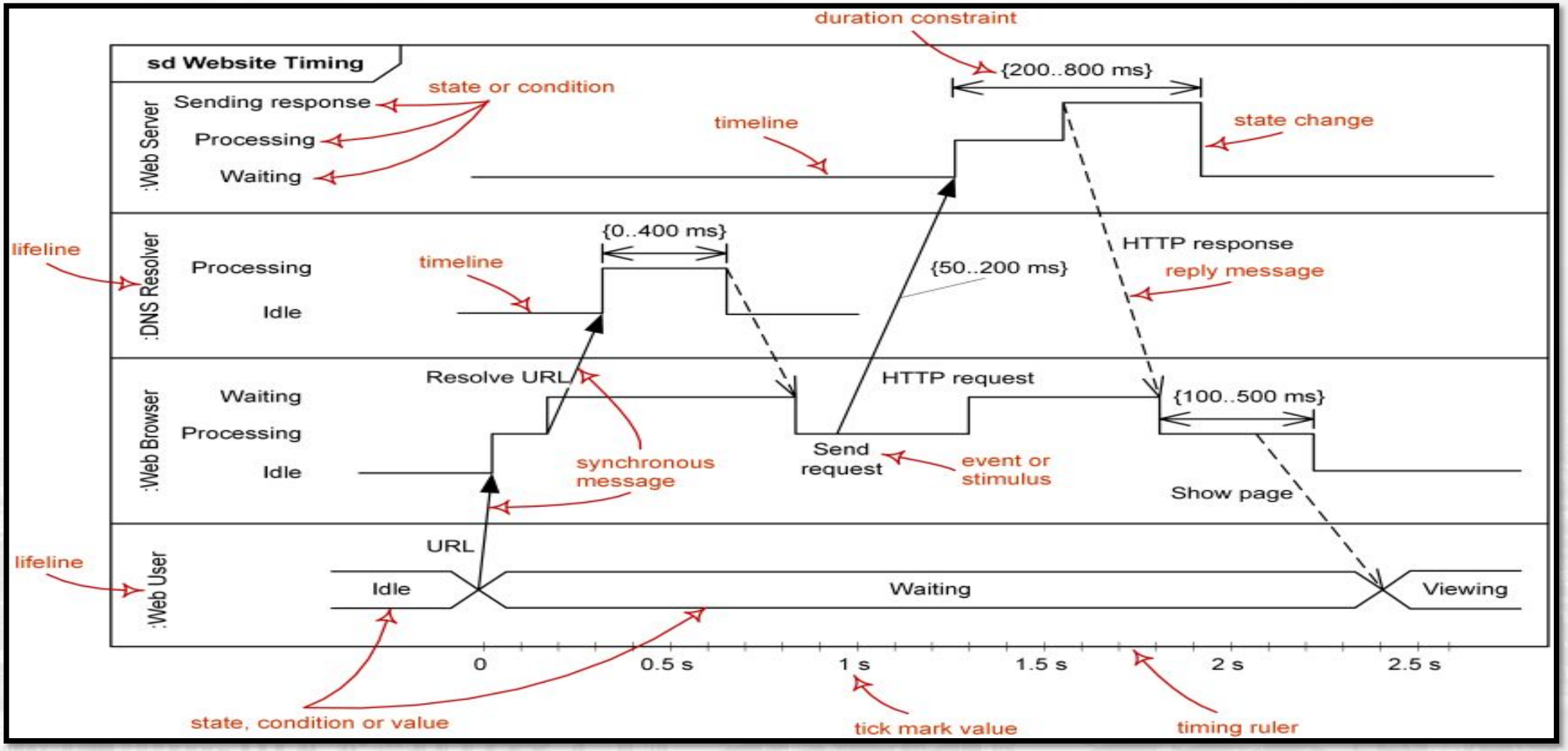
- UML 2 Timing diagrams shows the behavior of the objects in a given period of time. Timing diagram is a special form of a sequence diagram.
- The differences between timing diagram and sequence diagram are that the axes are reversed so that the time are increase from left to right and the lifelines are shown in separate compartments arranged vertically.
- Timing diagrams focus on conditions changing within and among lifelines along a linear time axis.

Timing diagram

- The timing diagram focusing attention on time of occurrence of events causing changes in the modeled conditions of the Lifelines.



Timing diagrams

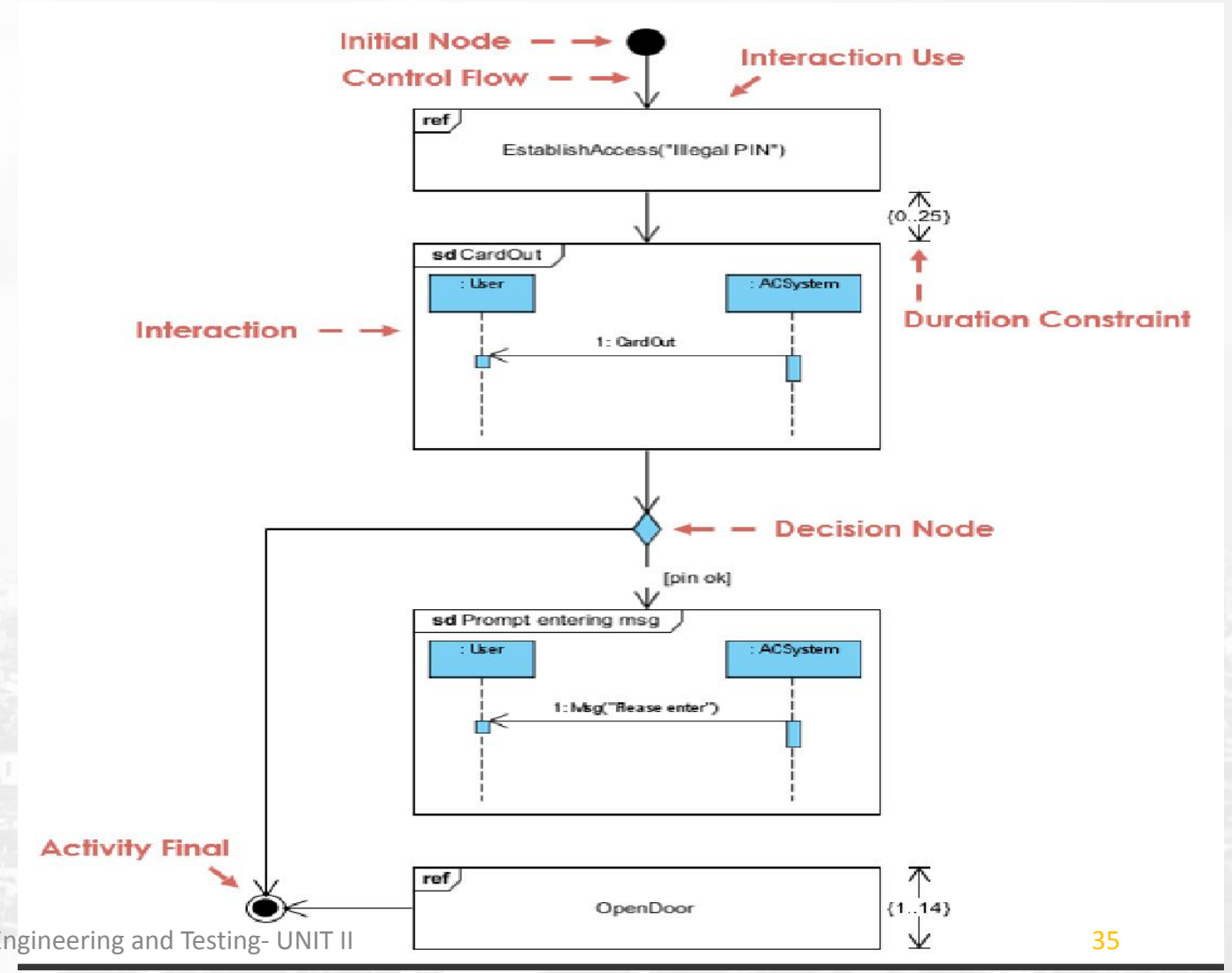


Interaction overview diagram

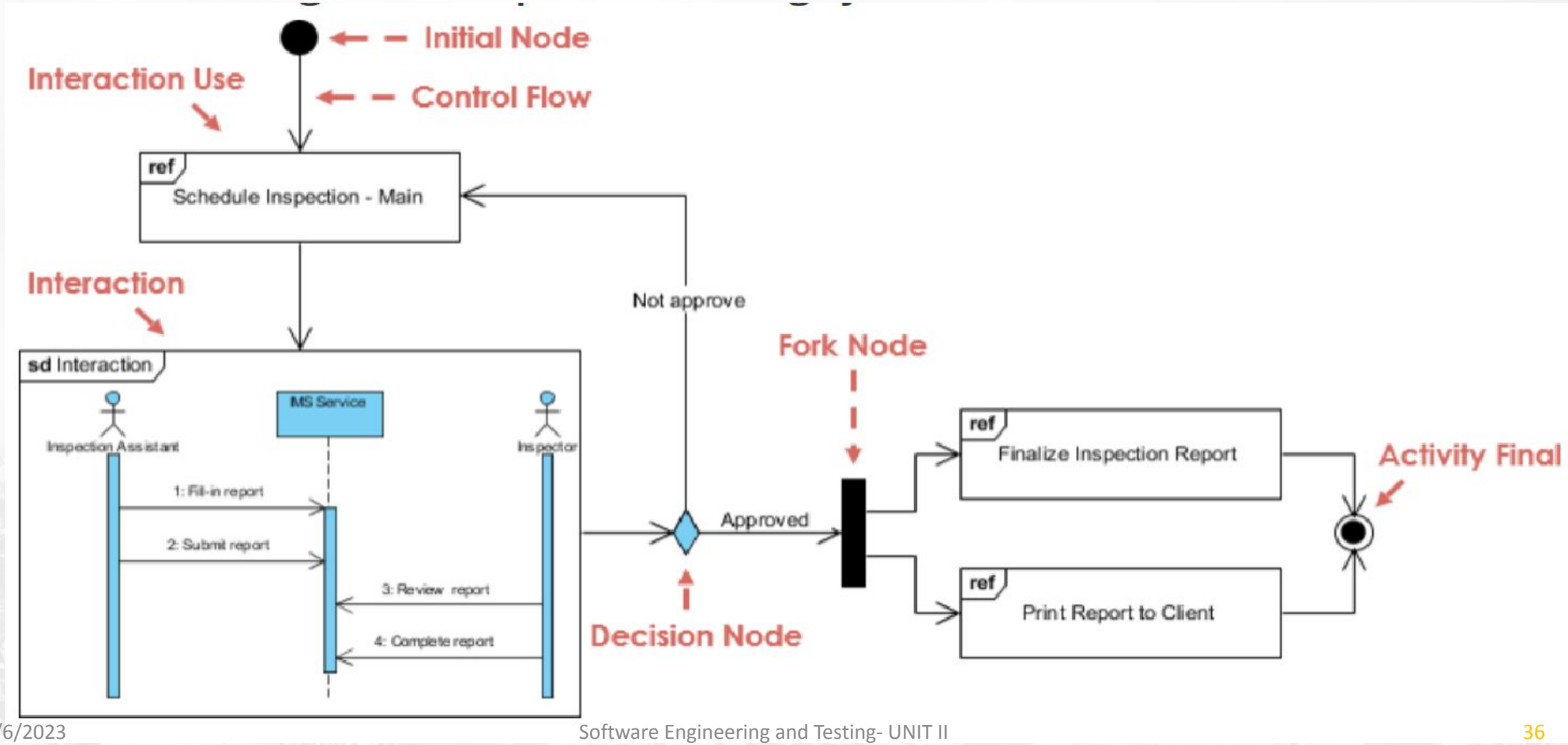
- Interaction overview diagrams focuses on the overview of the flow of control of the interactions.
- It is a variant of the Activity Diagram where the nodes are the interactions or interaction occurrences.
- It describes the interactions where messages and lifelines are hidden.

Interaction overview diagram

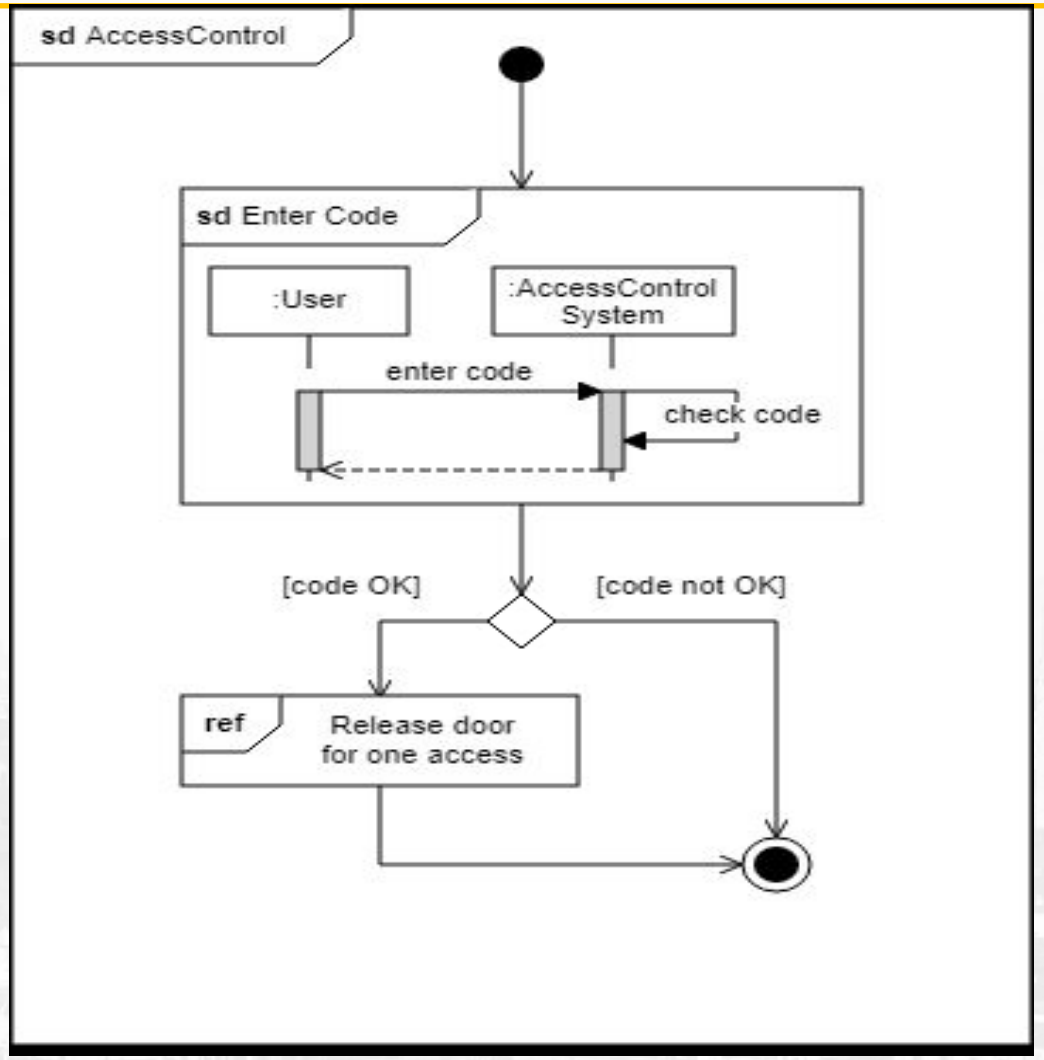
- The Interaction EstablishAccess occurs first with argument"Illegal PIN" followed by an interaction with the message CardOut which is shown in an inline Interaction.
- Then there is an alternative as we find a decision node with an InteractionConstraint on one of the branches.
- Along that control flow we find another inline Interaction and an InteractionUse in the sequence.



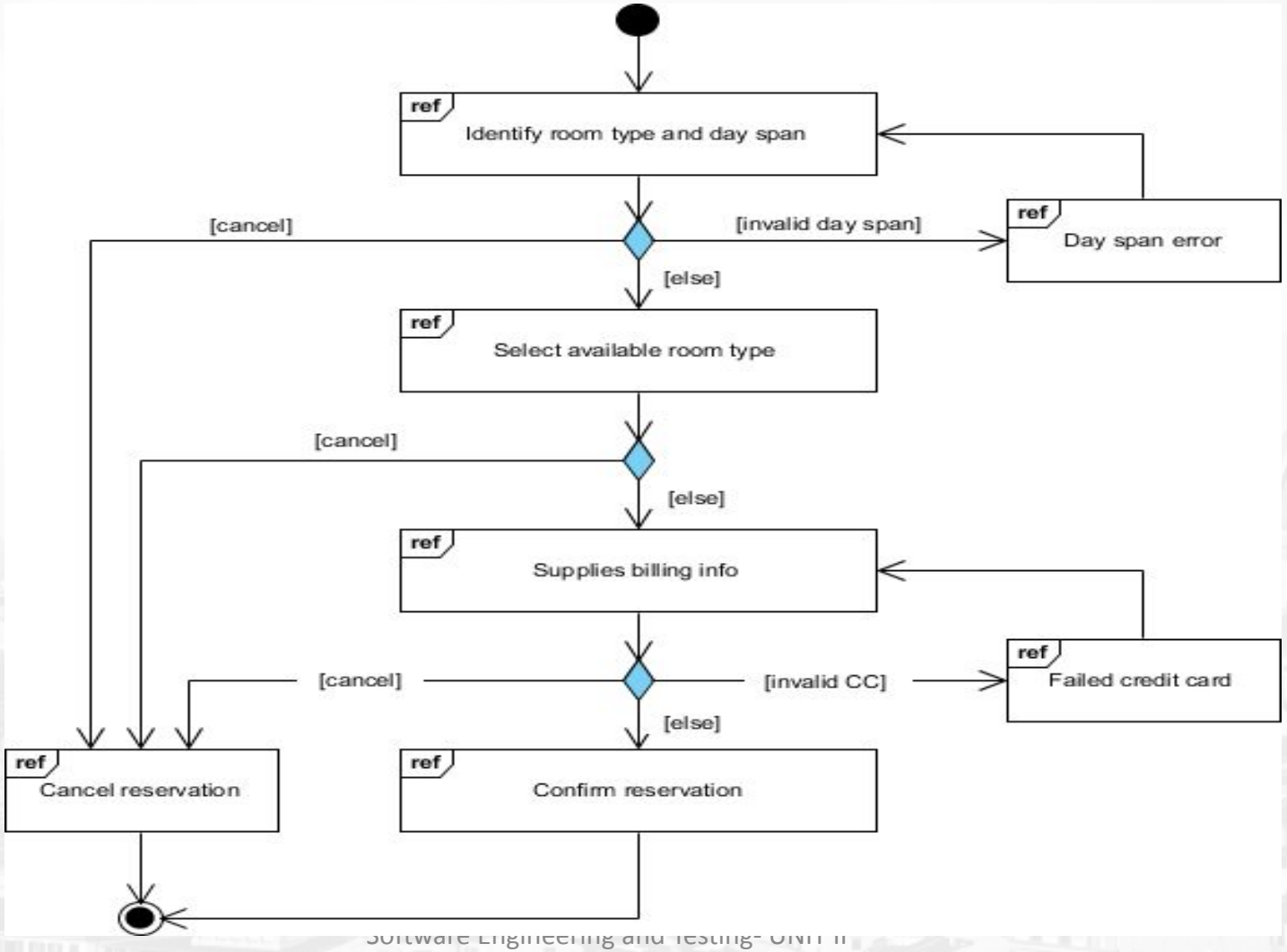
Interaction overview diagram- Scheduling System



Interaction overview diagram



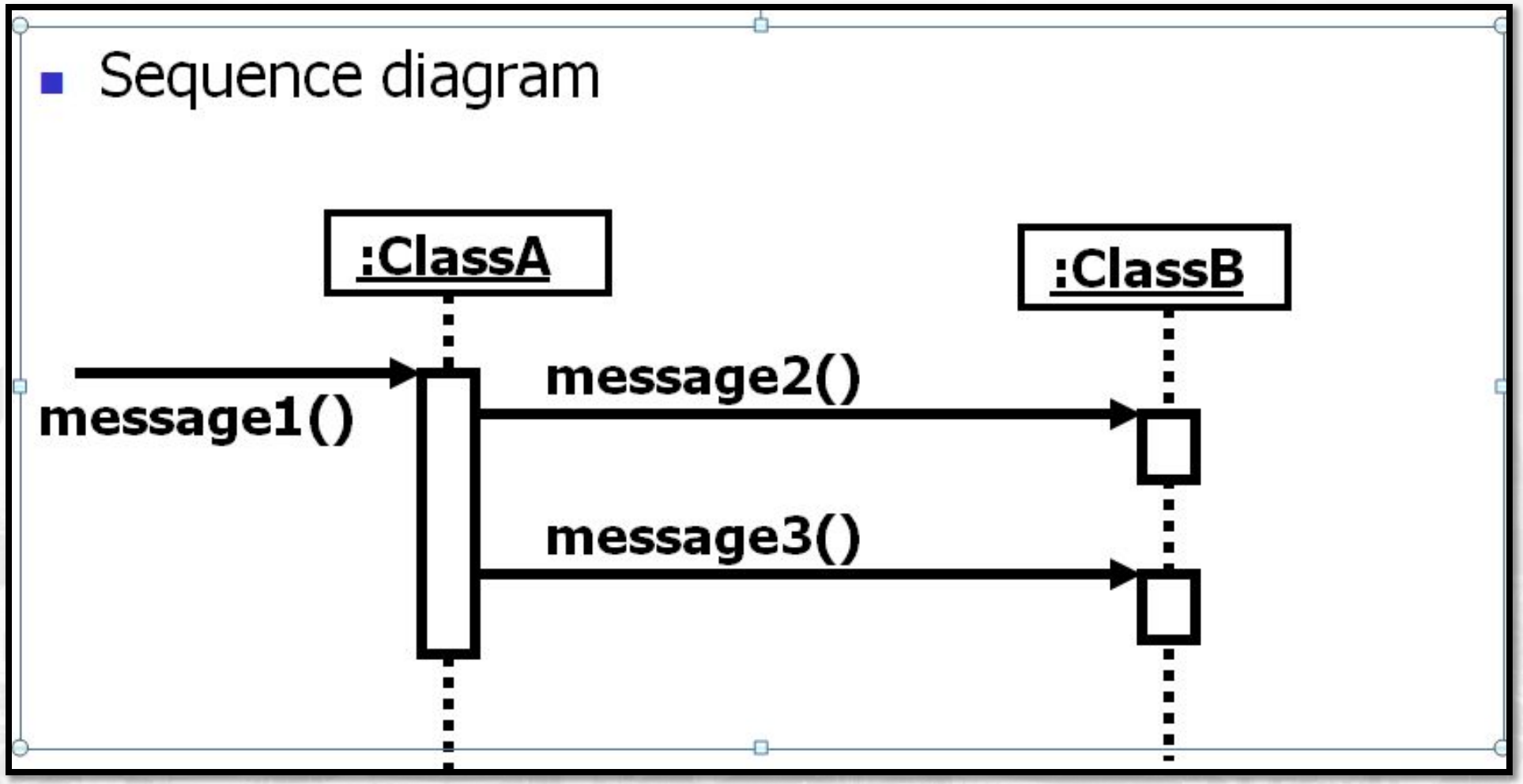
Interaction overview diagram



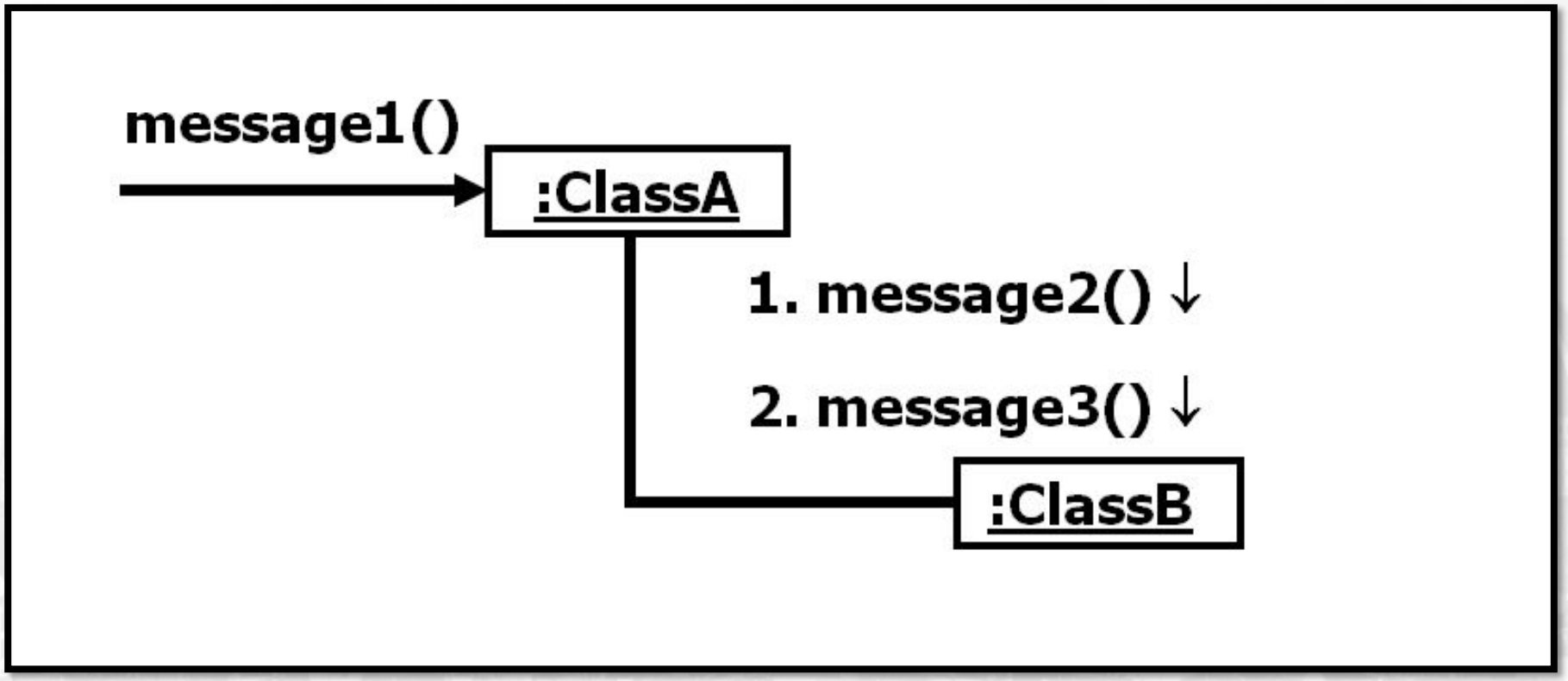
Communication diagrams

- UML Communication diagrams used to model the dynamic.
- When compare to Sequence Diagram, the Communication Diagram is more focused on showing the **collaboration of objects rather than the time sequence.**
- Components of the **communication process** include a sender, encoding of a message, selecting of a channel of **communication**, receipt of the message by the receiver and decoding of the message.
- It show the network and sequence of messages or communications between objects at run-time during a collaboration instance

Sequence Diagrams vs Communication Diagrams



Sequence Diagrams vs Communication Diagrams



Sequence Diagrams vs Communication Diagrams

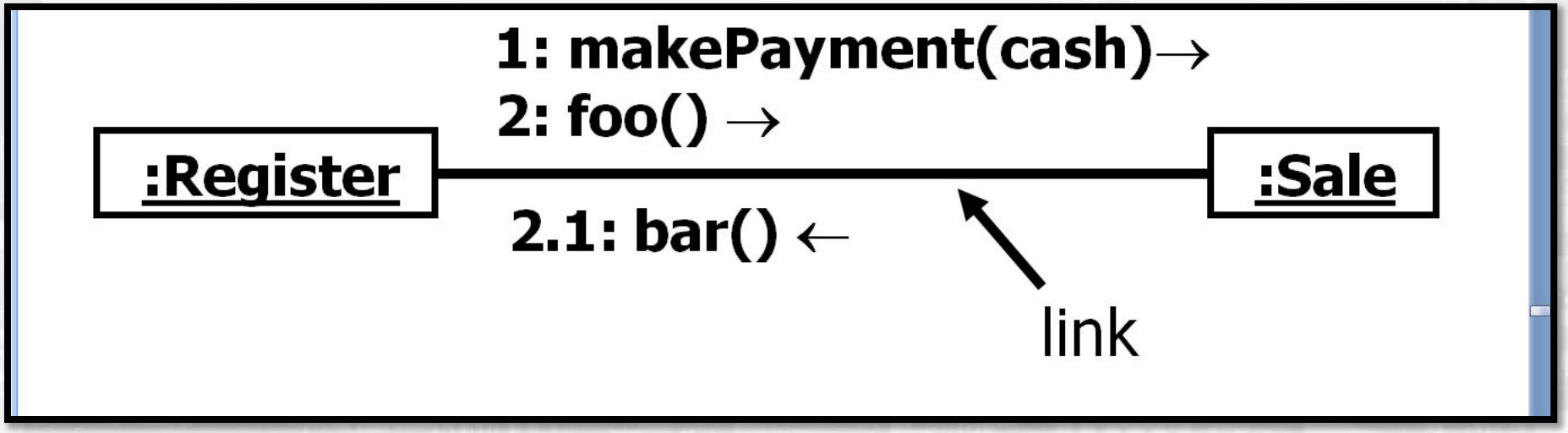
- **Sequence diagrams :**
  - Strength: clearly show sequence or time ordering of events, simple notation
  - Weakness: forced to extend to the right when adding new objects
- **Communication diagrams :**
  - Strength: space economical flexibility to add new objects in two dimensions, better to illustrate complex branching, iteration and concurrent behavior
  - Weakness: difficult to see sequence of messages, more complex notation

CommunicationDiagrams

- **Communication diagrams contain**
  - Classes
  - Associations
  - Message exchanges within a collaboration

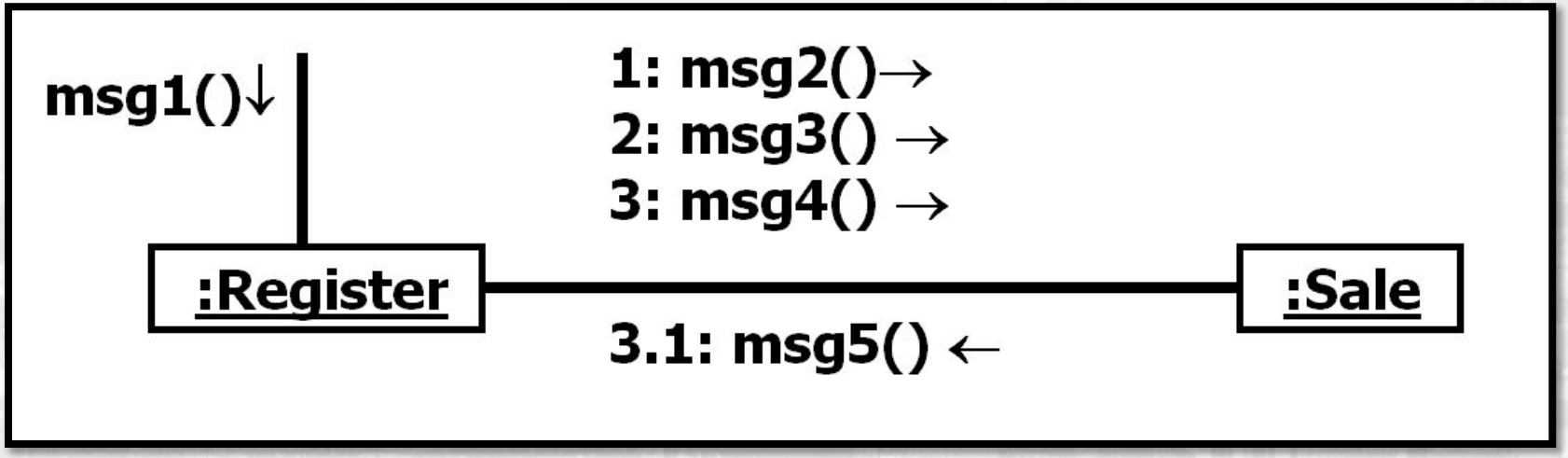
Communication Diagrams

- Link : A link is a connection path between two objects, it indicates some form of navigation and visibility between the objects. A link is an instance of an association.
- Note that multiple messages, and messages both ways can be exchanged along the same link.

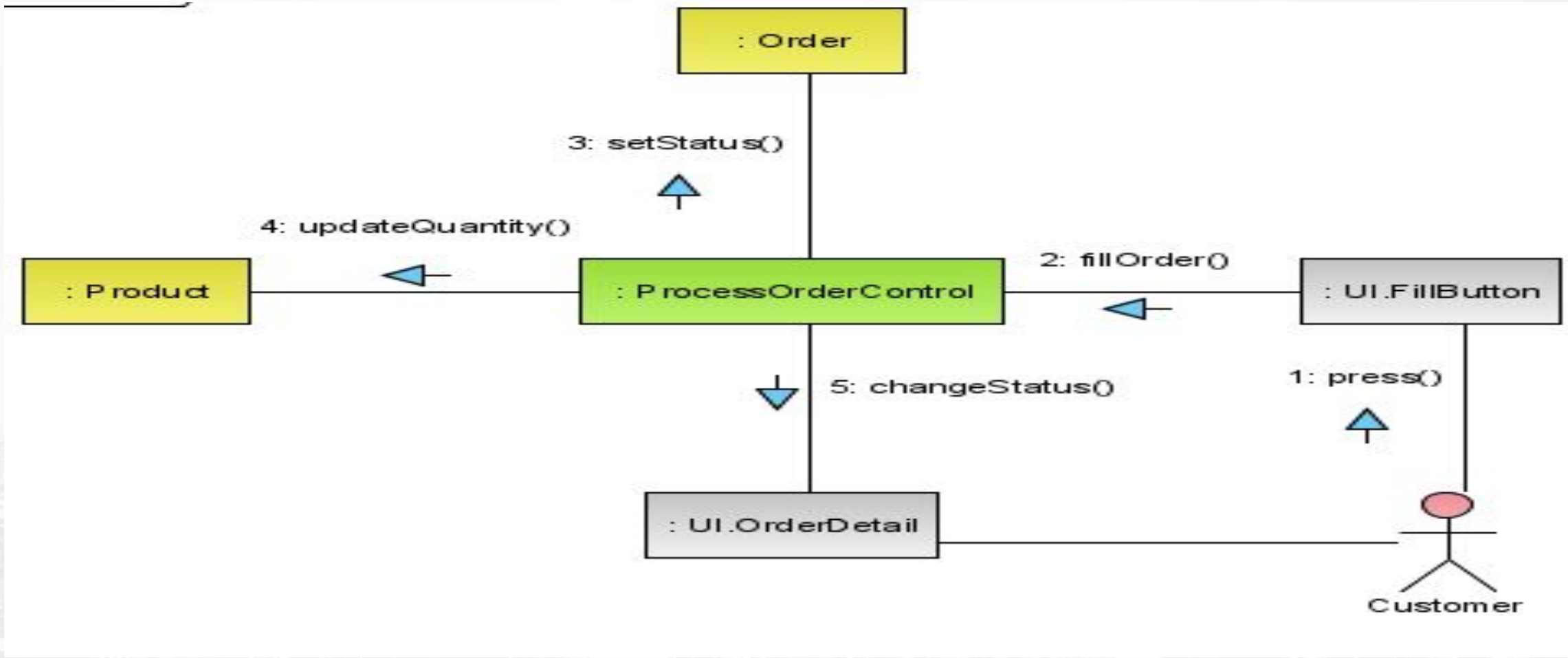


Communication Diagrams

- Messages : each message between objects is represented with a message expression and a small arrow indicating the direction of the message. A sequence number is added to show the sequential order of messages.



Process Order Control Communication diagram



Communication vs. Sequence Diagrams

	Communication Diagrams	Sequence Diagrams
Participants	✓	✓
Links	✓	
Message Signature	✓	✓
Parallel Messages	✓	✓
Asynchronous messages		✓
Message Ordering		✓
Create & Maintain	✓	

State Machine Diagrams…
Terms and Concepts

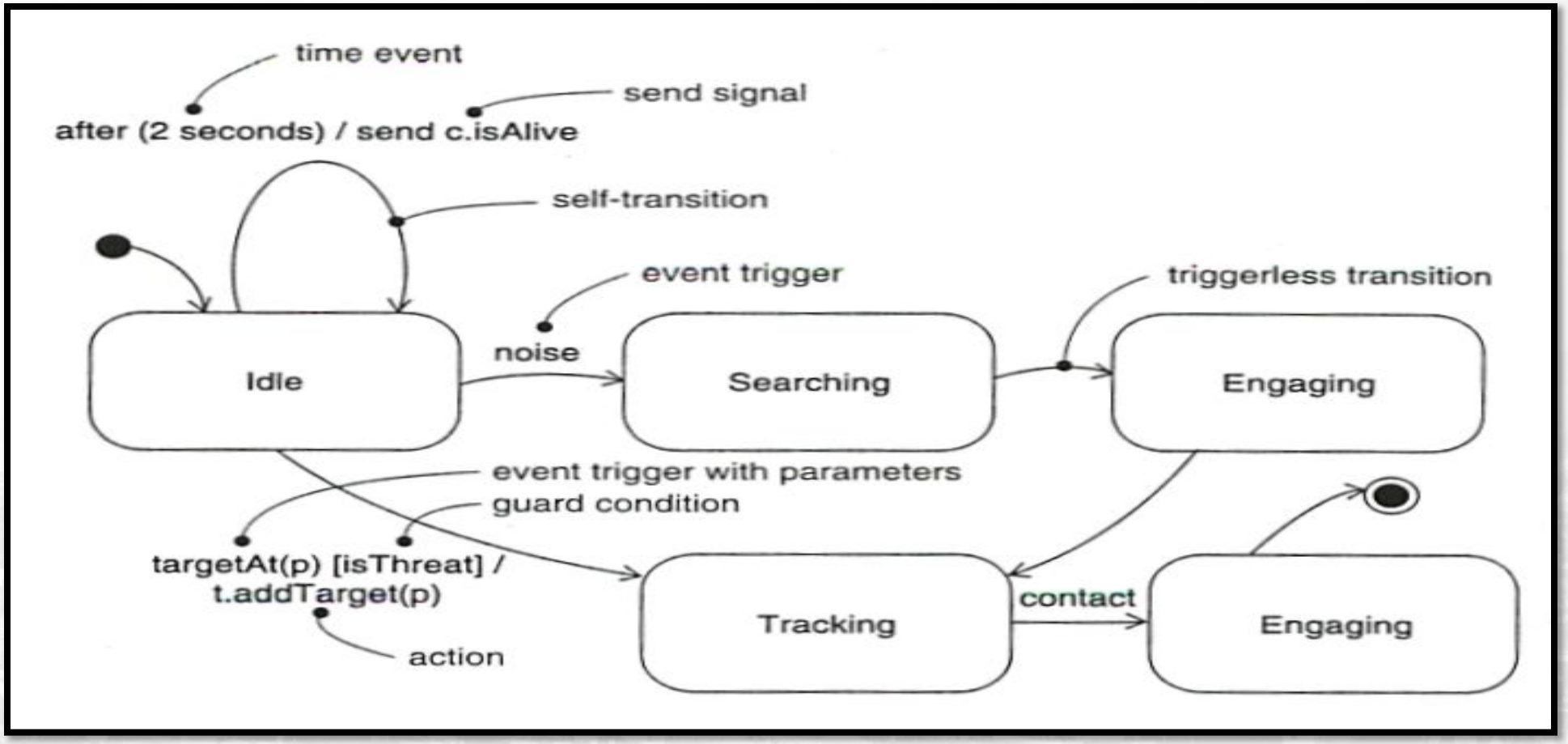
- *State*
  - A state is a condition or situation during the life of an object in which it satisfies some condition, performs some activity, or waits for some event.
  - A state may include …
    - Name
    - Entry/exit actions
    - Internal transitions
    - Activities
    - Substates - may sequential or concurrent
    - Deferred events (infrequently used)

Terms and Concepts

- ***State*** – Special categories of states
  - *Initial state* – indicates the initial starting state for for the state machine or a substate machine
  - *Final state* – indicates the statemachine's execution has completed.
  - Neither Initial or final states contain any of the parts found in traditional states.
- ***Transition*** – A directed relationship between two states
  - = **Contains five parts**
    - Event trigger – current state before transition fires
    - Guard condition – external and internal conditions that must be satisfied before a transition can fire.
    - Action – executable atomic computation that may directly act on the object that owns the transition.
    - Target state – new state after transition fires.

State Machine Diagrams

Example of Transition of states



Terms and Concepts

- **Advanced States and Transitions**
  - **Entry action** - Upon each entry to a state, a specified action is automatically executed.
    - Syntax (to be placed inside the state symbol): **entry / action**
  - **Exit action** - Just prior to leaving a state, a specified action is automatically executed.
    - Syntax (to be placed inside the state symbol): **exit / action**
  - **Internal Transitions** - The handling of an event without leaving the current state.
    - Used to avoid a states entry and exit actions.
    - Syntax (to be placed inside the state symbol): **event / action**
  - **Activities -** Ongoing work that an object performs while in a particular state. The work automatically terminates when the state is exited.
    - Syntax (to be placed inside the state symbol): **do / activity**
    - Activities are ongoing operations; actions are instantaneous operations.

Terms and Concepts

- **Advanced States and Transitions**
  - **Deferred Event** - An event whose occurrence is responded to at a later time.
    - Syntax (to be placed inside the state symbol): **event / defer**
    - Once an event has been deferred it remains deferred until a state is entered where
    - that particular type of event is not
    - listed as deferred.
    - The state diagram then responds
    - to the event as if it had just occurre

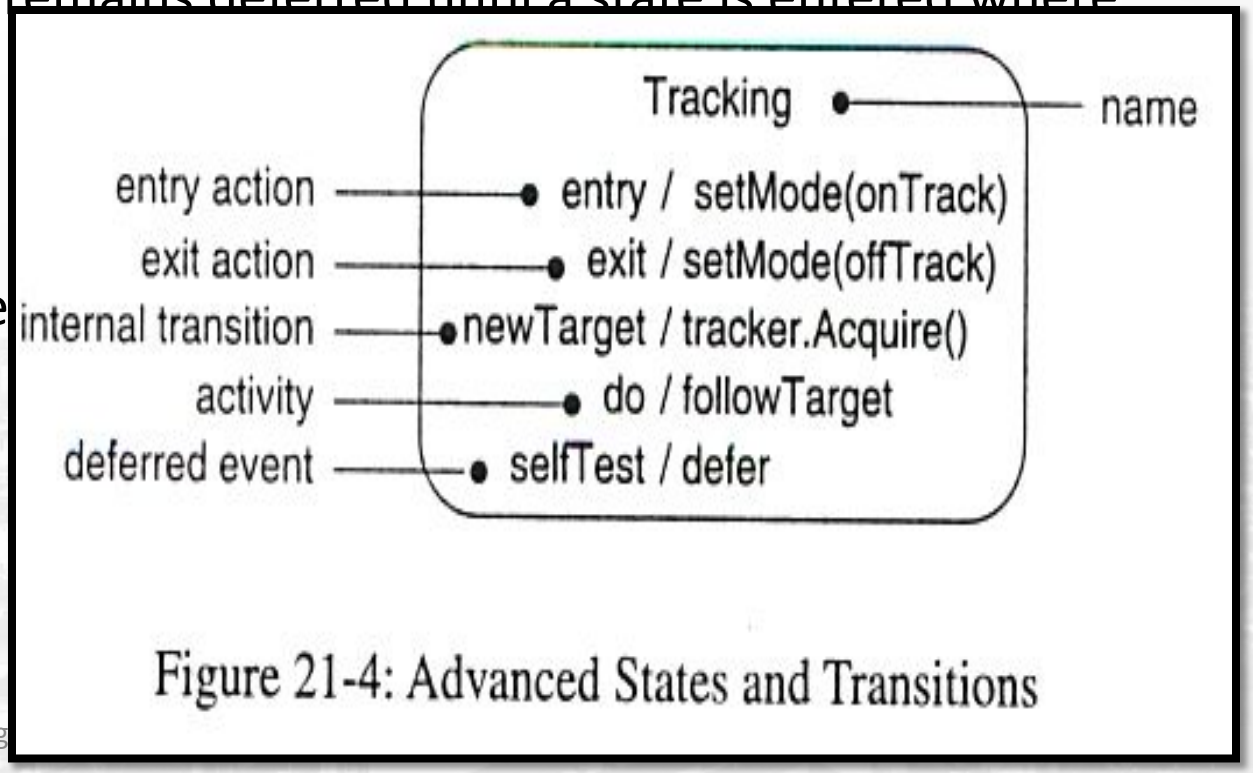
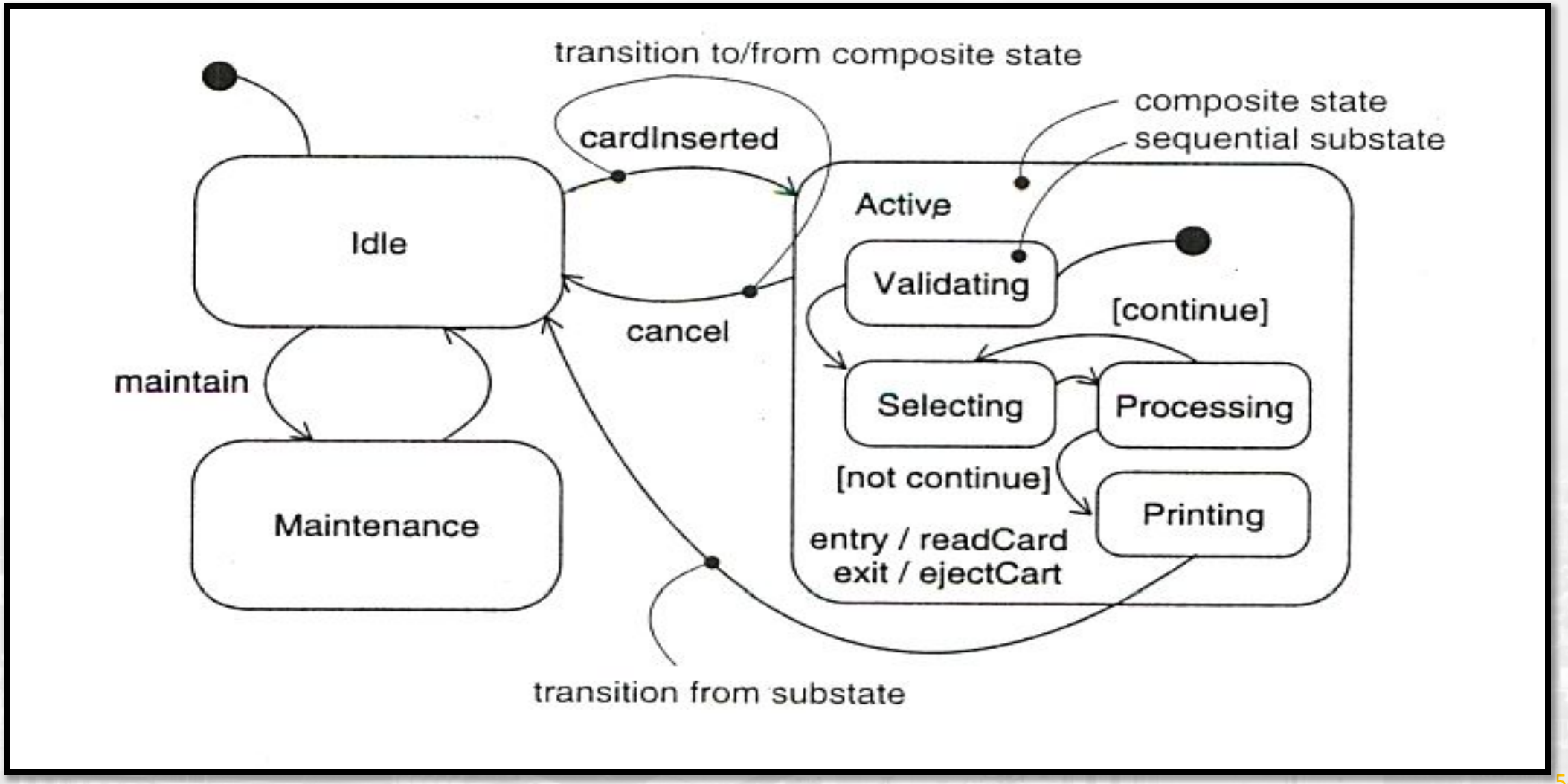


Figure 21-4: Advanced States and Transitions

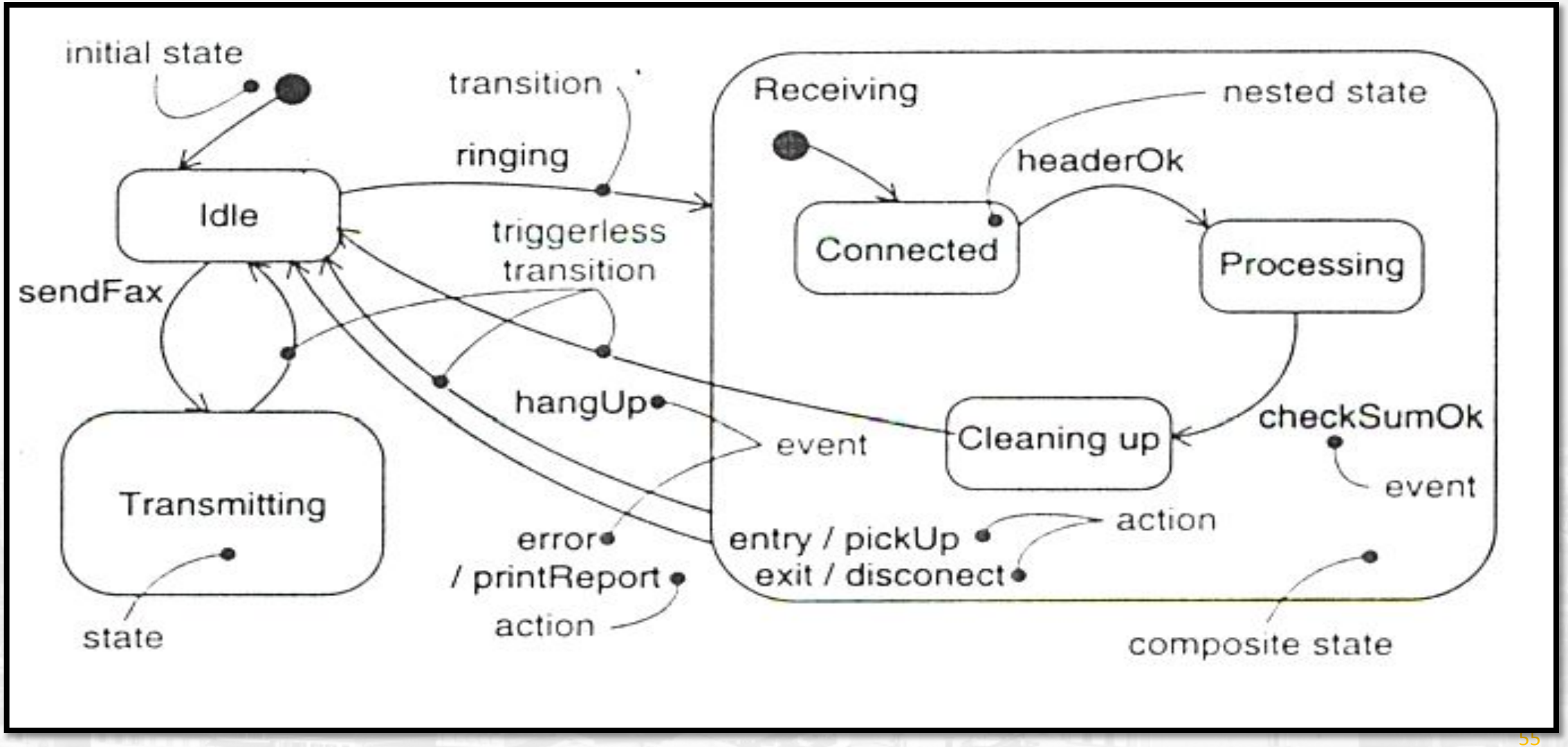
Terms and Concepts

- **Advanced States and Transitions**
  - Simple state - A state that contains no *substates*.
  - Composite state - A state that contains *substates*.
  - Substate - A state that is nested inside another state.
    - Substates allow state diagrams to show different levels of abstraction.
    - Substates may be sequential or concurrent.
    - Substates may be nested to any level.
- **Sequential Substates -** The most common type of substate. Essentially, a state diagram within a single state.
    - The “containing” state becomes an abstract state.
    - The use of substates simplifies state diagrams by reducing the number of transition lines.
    - A nested sequential state diagram may have at most one initial state and one final state.

ATM Machine Composite State Diagram



Phone Call Advance State Diagram



Terms and Concepts

- **Advanced States and Transitions**
  - History States - Allows an object to remember which substate was last active when the containing state was exited.
    - Upon re-entry to the containing state, the substate that was last active will be re-entered directly

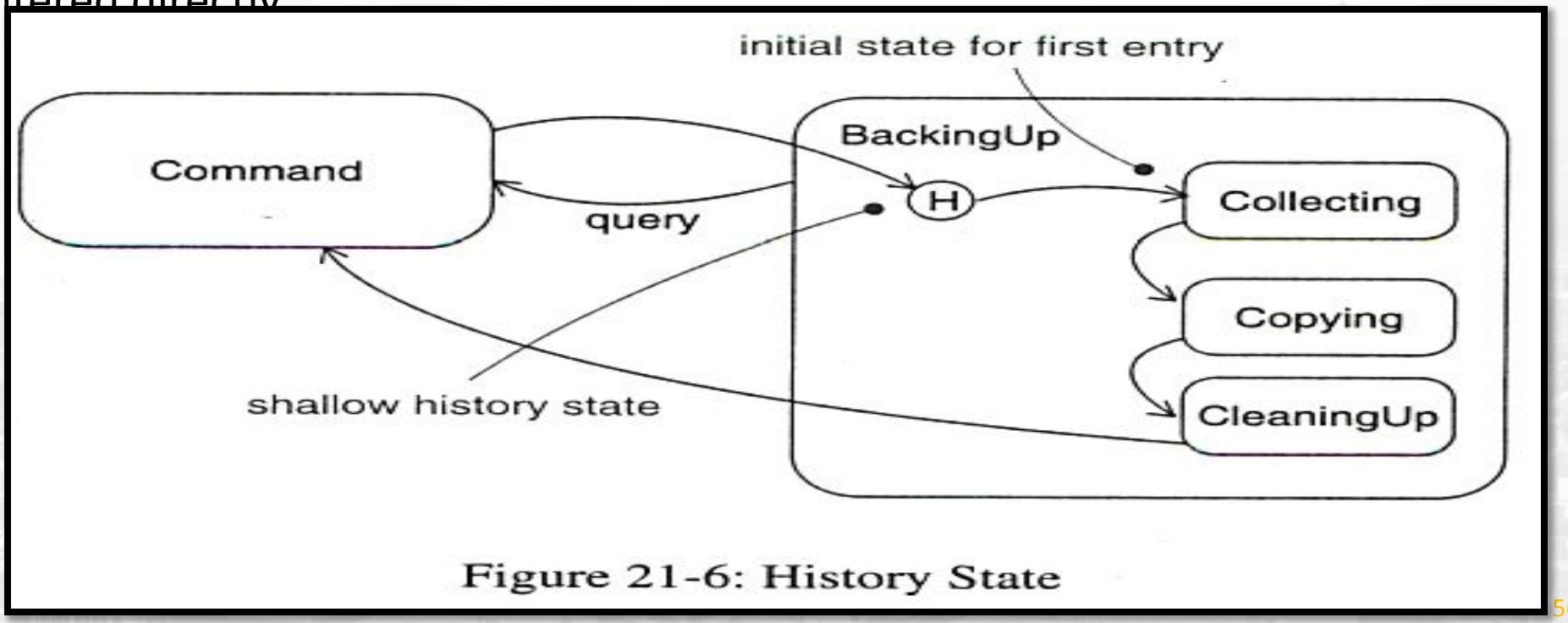


Figure 21-6: History State

Terms and Concepts

- **Advanced States and Transitions**
  - Concurrent Substates - Used when two or more state diagrams are executing concurrently within a single object.
    - Allows an object to be in multiple states simultaneously.
    - The concurrent state diagrams within a “containing” state must begin and end execution simultaneously.
    - If one concurrent state diagram finishes first, it must wait for the others to complete before exiting the containing state.

Concurrent Substrates Example

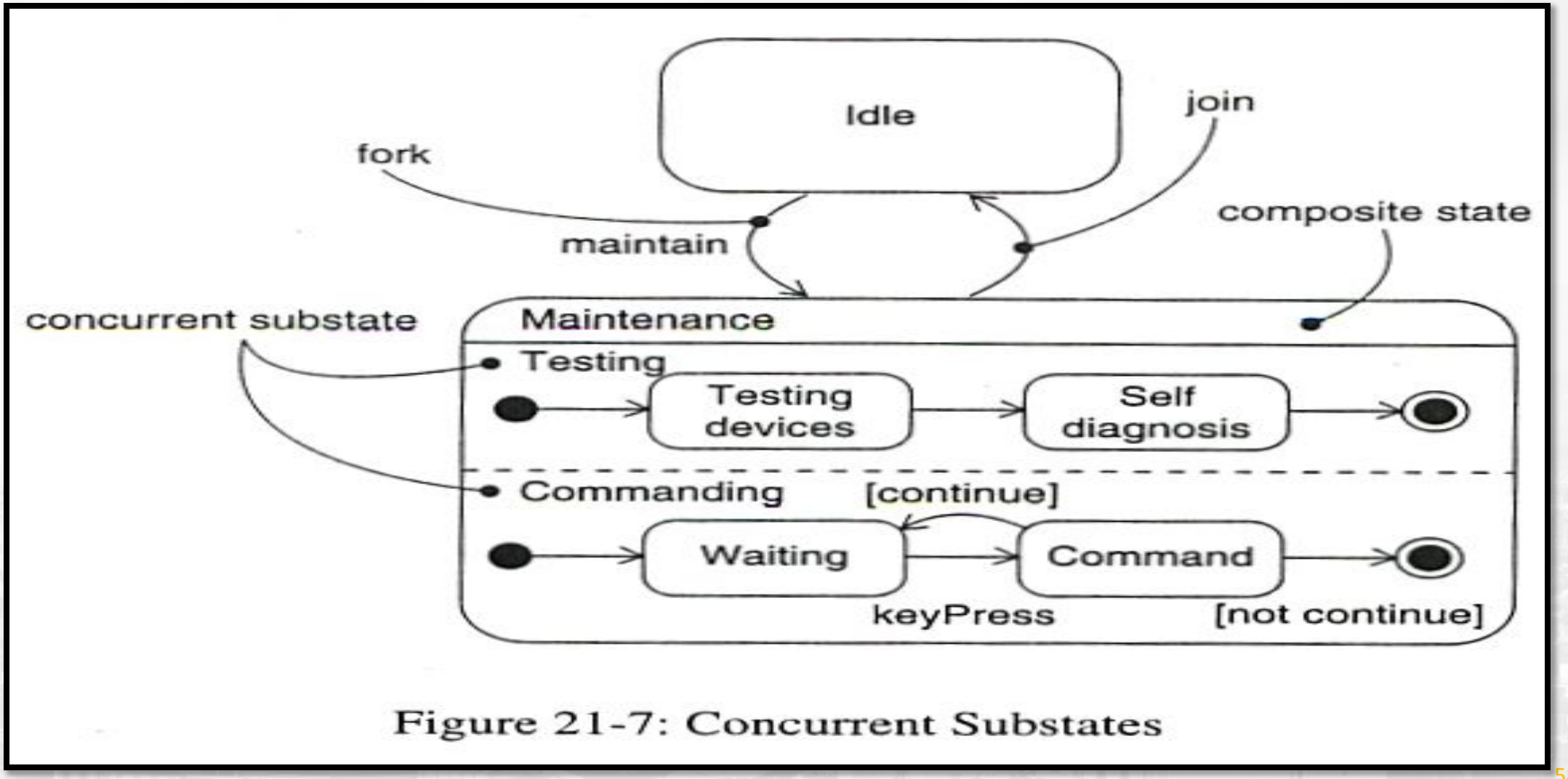


Figure 21-7: Concurrent Substates

State Diagram for a Phone Line

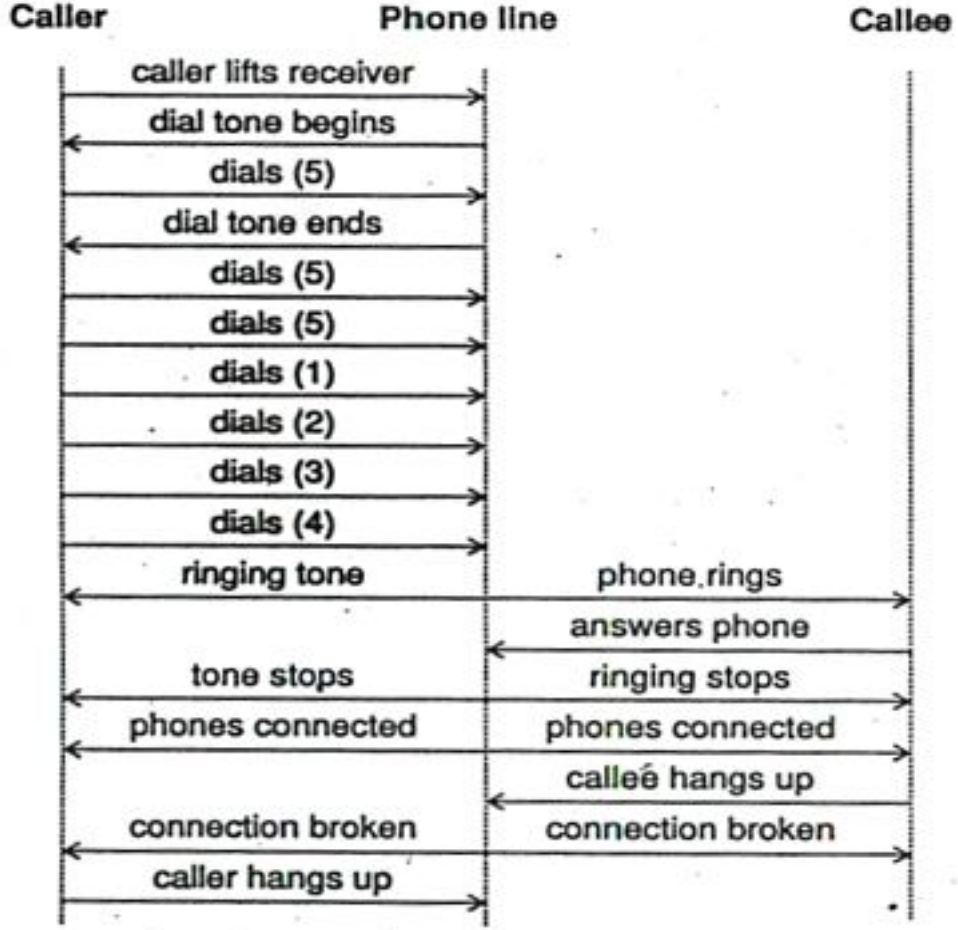


Figure 5.3 Event trace for phone call

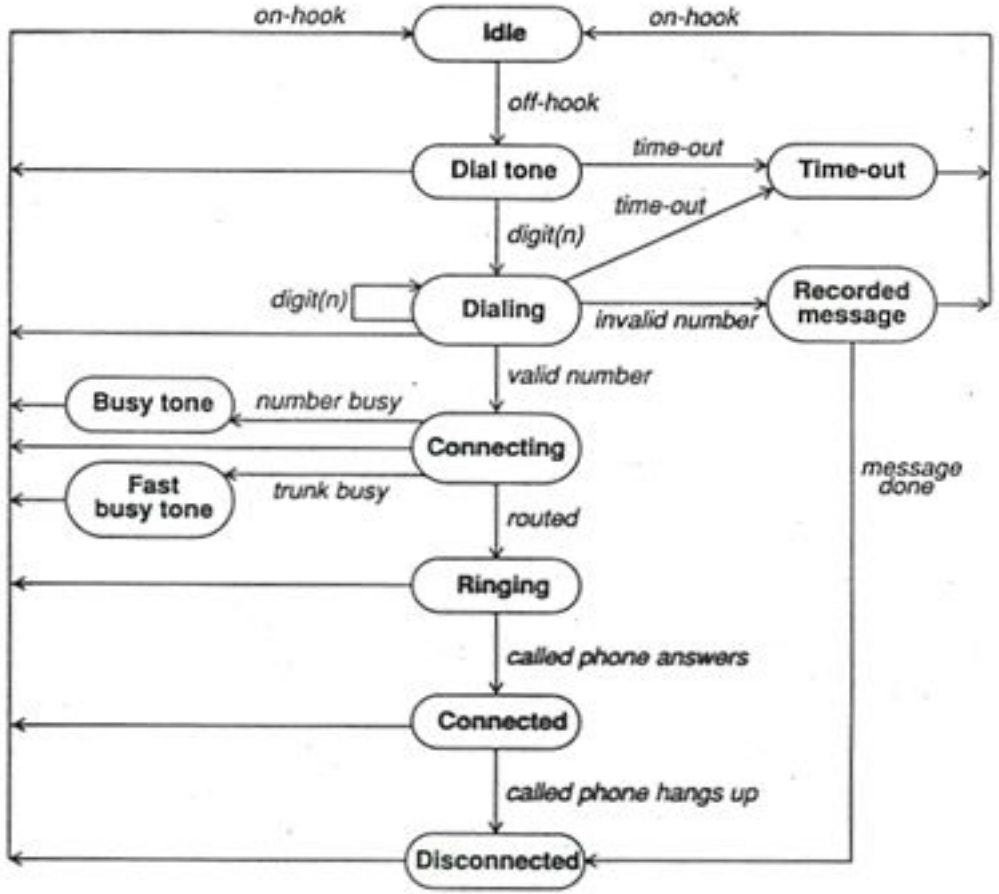


Figure 5.5 State diagram for phone line

References

- Ian Sommerville, — Software Engineeringǁ, Addison and Wesley. 9th Ed., 2011.
- Roger S Pressman, Software Engineering: A Practitioner's Approach, Mcgraw-Hill, ISBN: 0073375977, Seventh Edition, 2014
- Pankaj Jalote, Software Engineering: A Precise Approach, Wiley India.2010.

Disclaimer:

a. Information included in this slides came from multiple sources. We have tried our best to cite the sources. Please refer to the References to learn about the sources, when applicable.

b. The slides should be used only for academic purposes (e.g., in teaching a class), and should not be used for commercial purposes.

- SOFTWARE ENGINEERING AND PROJECT MANAGEMENT

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

UNIT III - Software Project Management

Project Management Principles, Process and Project Metrics, Function Point analysis, LOC, Make/Buy Decision, COCOMO II -Project Planning, SWOT analysis, Functions of manager, Team building & development, Risk Management.

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Project Management

Software Project Management

- Concerned with activities involved in ensuring that software is delivered
  - on time
  - within the budget
  - in accordance with the requirements
- Project management is needed because software development is always subject to budget, requirement and schedule constraints that are set by the development organisation or the customer

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Project Management

- Organising, planning and scheduling software projects
- Objectives
  - To introduce software project management and to describe its distinctive characteristics
  - To discuss project planning and the planning process
  - To show how graphical schedule representations are used by project management
  - To discuss the notion of risks and the risk management process

Management Activities

- Proposal writing
- Project planning and scheduling
- Project costing
- Project monitoring and reviews
- Personnel selection and evaluation
- Report writing and presentations

Failed Projects : **Airbus A380**

- Building the Airbus A380—the world's largest commercial aircraft at the time—required production facilities from across the globe to build individual parts of the aeroplane.
- All teams used different computer-aided design (CAD) programs.
- During installation, they discovered the parts designed by different teams didn't fit together.
- This cost the company $6 billion to put right and set the project back two years.

Failed Projects : **Ford Edsel**

- Ford did extensive market research before it released the Edsel.
- They spent 10 years and $250 million on research and planning—but by the time all this was completed, and the car was unveiled in 1957, Ford had missed its chance.
- The market had already moved on to buying compact cars, which didn't include the Edsel.

Failed Project: Ariane 5 Disaster

- On June 4th, 1996, the very first Ariane 5 rocket ignited its engines and began speeding away from the coast of French Guiana.
- 37 seconds later, the rocket flipped 90 degrees in the wrong direction, and less than two seconds later, aerodynamic forces ripped the boosters apart from the main stage at a height of 4km.
- This caused the self-destruct mechanism to trigger, and the spacecraft was consumed in a gigantic fireball of liquid hydrogen.

Failed Project: Ariane 5 Disaster

- The fault was a software bug in the rocket's Inertial Reference System. This system is used to determine whether it was pointing up or down, which is formally known as the horizontal bias.
- This value was represented by a 64-bit floating variable, which was perfectly adequate.
- However, when the software attempted to stuff this 64-bit variable, which can represent billions of potential values, into a 16-bit integer, which can only represent 65,535 potential values.
- For the first few seconds of flight, the rocket's acceleration was low, so the conversion between these two values was successful.
- However, as the rocket's velocity increased, the 64-bit variable exceeded 65k, and became too large to fit in a 16-bit variable.

A German pro basketball team was Failed Project : Relegated to a lower division due to a Windows update

In 2015, the Paderborn Baskets, a second division German basketball team, was relegated to a lower division for starting a game late, due to a necessary 17-minute Windows update to the scoreboard's laptop.

The game between the Chemnitz Niners and the Paderborn Baskets was set to begin as normal, when Paderborn connected its laptop to the scoreboard. According to Paderborn Baskets general manager Patrick Seidel, as reported in the Die Zeit journal, the laptop was connected by 6:00 p.m. (meaning 1 hour and 30 minutes before the game), and was set "as usual." However, according to Seidel, "As both teams warmed up, the computer crashed. When we booted it again by 7:20 p.m., it started downloading updates automatically." When the computer finished downloading and installing all the updates, the game finally began at 7:55 p.m.

Paderborn won the game 69 to 62.

As a result, Paderborn lost a point as a penalty and found itself relegated from the ProA to the ProB division

A $2 billion air traffic control System Failed due to insufficient er memory

On April 30, 2014, hundreds of LAX flights were delayed or canceled because all computers in the airport crashed due to a bug in the En Route Automation Modernization (ERAM) system.

The system failure was sparked due to a U-2 spy plane that was flying through the region.

The $2.4 billion system, made by Lockheed Martin Corp, cycled off and on trying to fix the error, which was due to a lack of altitude information in the airplane's flight plan.

After an air traffic controller entered an estimated altitude of the plane, the system calculated all possible flight paths to ensure that it was not on a crash course with other planes.

However, that process caused the system to fall short in memory and shut down every other flight processing function. Fortunately, no injuries or accidents were reported.

The ERAM system failed because it limits how much data each plane can send. Most planes have simple flight plans, so they do not exceed that limit. However, the U-2 operating that day had a complex flight plan that brought it close to the system's limit.

MIT-WPU

।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Project Management Principles

5 Project Management Principles

1. Address important questions at the beginning of the project
2. Sketch out a scope and goals for your project
3. Communicate roles, expectations, and objectives to the team
4. Monitor progress and identify roadblocks
5. Make sure all deliverables have been met and finalize the project

Project Management Principles

Address important questions at the beginning of the project	• Should this project happen and how will it help the company? • Can the company benefit from this project at this time? • What problem is this project solving?
Sketch out a scope and goals for your project	• What is this project's main goal and deliverables needed to reach that goal? • What risks exist for this project, and how can you avoid them? • What is the scope of this project and how will you manage expanding priorities and changing workloads if they happen mid-project
Communicate roles, expectations, and objectives to the team	• The progress you've made along the project timeline • The goals and advantages of the project • The project's roadblocks and successes
Monitor progress and identify roadblocks	• Do all team members understand the expectations and is the project on time? • What roadblocks exist? How can you remove them for your team? • Are you communicating and staying organized? • Does the project need to be redirected from its original scope?
Make sure all deliverables have been met and finalize the project	• Have all deliverables been met? • Were all deliverables carried out to a standard of quality the team is proud of? • What did your team do well? • How could a project like this function better next time?

People in the Project

- Projects don't get done by themselves. We need people to carry them out.
- This is where project managers and project management team come into the picture. When we look at the project management life-cycle, there are many people and groups involved.
- A project manager is a person who is responsible for leading the project. In other words, project managers are the spearheads of a project.
- A project team is a group of individuals teamed together. Their purpose is to achieve a specific business task or goal.

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Software Process & Project Metrics

Software process and **project metrics** are quantitative measures that enable **software** engineers to gain insight into the efficiency of the **software process** and the **projects** conducted using the **process** framework. In **software project** management, we are primarily concerned with productivity and quality **metrics**.

Software Measures: Lines of Code , Function Point

Software Metrics

Metrics is a quantitative measure of the degree to which a system, component or a process possesses a given attribute.

Software Metrics refers to a broad range of measurements for computer software.

Project Metrics :relate to **Project** Quality. They are used to quantify defects, cost, schedule, productivity and estimation of various **project** resources and deliverables.

Process and Project Metrics [1]

a) **Process Metrics**

These are metrics that pertain to Process Quality. They are used to measure the efficiency and effectiveness of various processes.

b) **Project Metrics**

These are metrics that pertain to Project Quality. They are used to quantify defects, cost, schedule, productivity and estimation of various project resources and deliverables.

Software Measurement

Measurement can be applied to the software process with the intent of improving it on a continuous basis.

Measurement can be used throughout a software project to assist in estimation, quality control, productivity assessment and project control.

Software Measurement

a) Direct Measures- cost and efforts applied i.e.
   i. Line of Code -LOC ,
   ii. Function Point -FP,
   iii. execution speed,
   iv. memory size,
   v. defect reported.

b) Indirect Measures- quality, complexity, efficiency, reliability, maintainability.

Direct Measurement : Function Point (FP)

- A function point calculates software sizewith the help of logical design and performance of functions as per user requirements.
- It also helps in determining the business functionality of a software application.
- It is derived from a software's requirements and can be estimated in the early phases of software development, before the actual lines of code can be determined.
- The number of function points in a code depends on function complexity.
- Measure software development and maintenance independently of technology used for implementation

Direct Measurement : Function Point (FP)

- Since Function Points measures systems from a functional perspective they are independent of technology.
- Regardless of language, development method, or hardware platform used, the number of function points for a system will remain constant.
- The only variable is the amount of effort needed to deliver a given set of function points.

Five Components of Function Point

A. DataFunction
   - Internal logic files
   - External Interface files

B. Transactional Functions
   - External Inputs
   - External Output
   - External Inquires


New Customer
Customer
Contact
Alt. Contact
Bill to
Phone
Fax
Alt. Phone
Ship to
Type
OK
Cancel
Next

Components of Function Point – Data Points

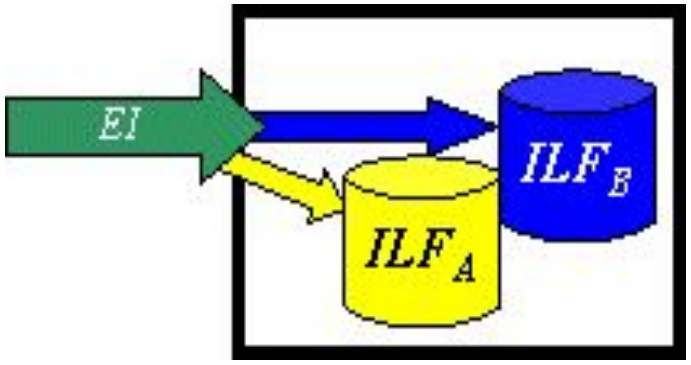
ILF

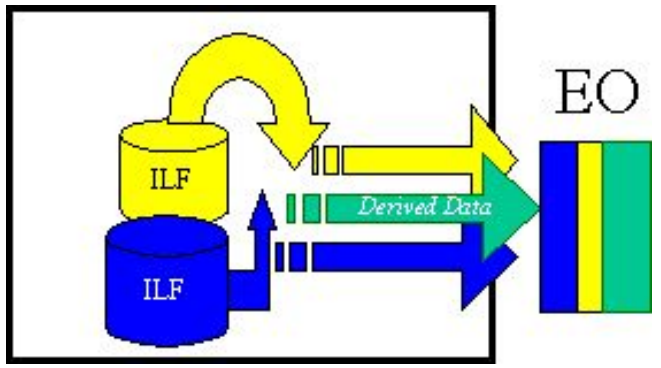
Internal Logical Files (ILF's) - a user identifiable group of logically related data that resides entirely within the applications boundary and is maintained through external inputs.

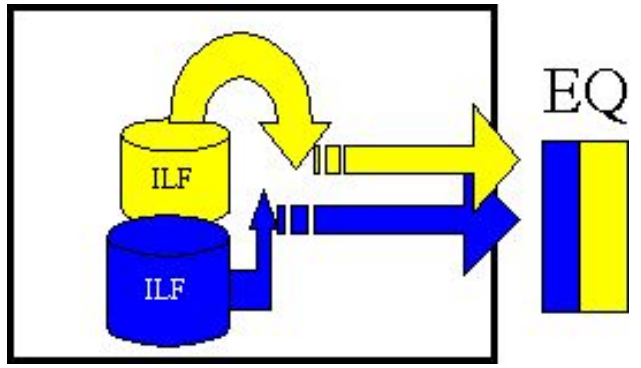
EIF

External Interface Files (EIF's) - a user identifiable group of logically related data that is used for reference purposes only. The data resides entirely outside the application and is maintained by another application. The external interface file is an internal logical file for another application.

Components of Function Point - Transactions







External Inputs External Output External Inquires

After the components have been classified as one of the five major components (EI's, EO's, EQ's, ILF's or EIF's), a ranking of low, average or high is assigned depending on the complexity.

Function Point (FP) Benefits

-Benefits :

- ✔ Function Point Analysis can be used to determine whether a tool, an environment, a language is more productive compared with others
- ✔ increase in productivity and
- ✔ reduction in the risk of inflation of created code.

Direct Measurement: Lines of Code (LOC)

Lines of code/statements (LOC)

- It is a software metric used to measure the size of a computer program by counting the number of **lines** in the text of the program's source **code**
- Studies show correlation between LOC and the overall cost and length of development, and between LOC and number of defects.
- The lower your LOC measurement is, the better it is.

Direct Measurement: Lines of Code (LOC)

Examples of use include:

- productivity — KLOC/person-month
- quality — faults/KLOC
- cost — $$/KLOC
- documentation — doc_pages/KLOC

LOC- Example

```
for (i = 0; i < 100; i += 1)
{
printf("hello");
} /* Now how many lines of code is this? */
```

In this example we have:

- 4 Physical Lines of Code (LOC)
- 2 Logical Line of Code (LLOC)
- 1 comment line

LOC- Example

C	COBOL

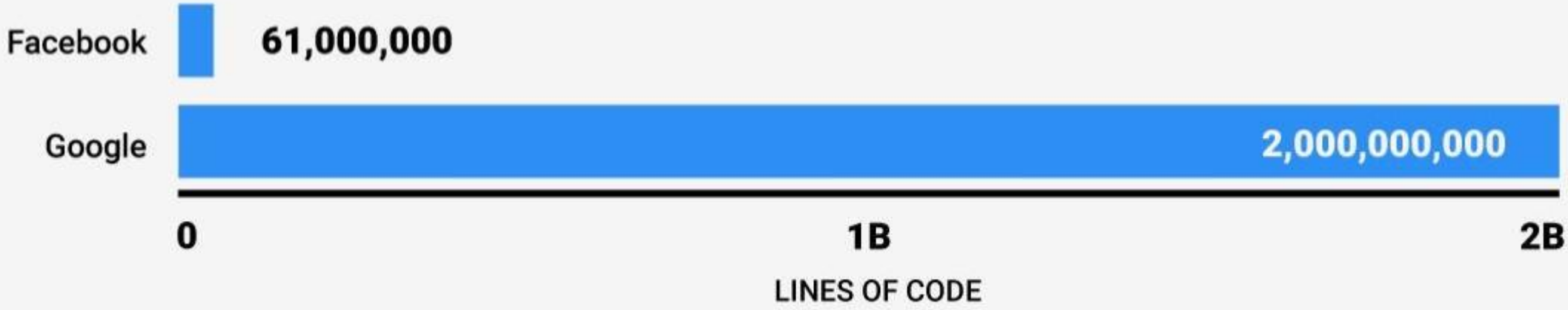
`# include <stdio.h>`
 	
`int main() {`
`  printf("\nHello world\n");`
`}`
	`identification division.`
	`program-id. hello .`
	`procedure division.`
	`display "hello world"`
	`goback .`
	`end program hello .`

Lines of code: 4 (excluding whitespace)	Lines of code: 6 (excluding whitespace)

The range is extraordinary: the average iPhone app has less than 50,000 lines of code, while Google's entire code base is two billion lines for all services.

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

HOW FACEBOOK'S CODE COMPARES TO ALL OF GOOGLE'S INTERNET SERVICES



Estimation Model

Estimation is the process of finding an estimate, or approximation.
Determines how much money, effort, resources, and time it will take to build a system.

Estimation is based on –

- Past Data/Past Experience
- Available Documents/Knowledge
- Assumptions
- Identified Risks

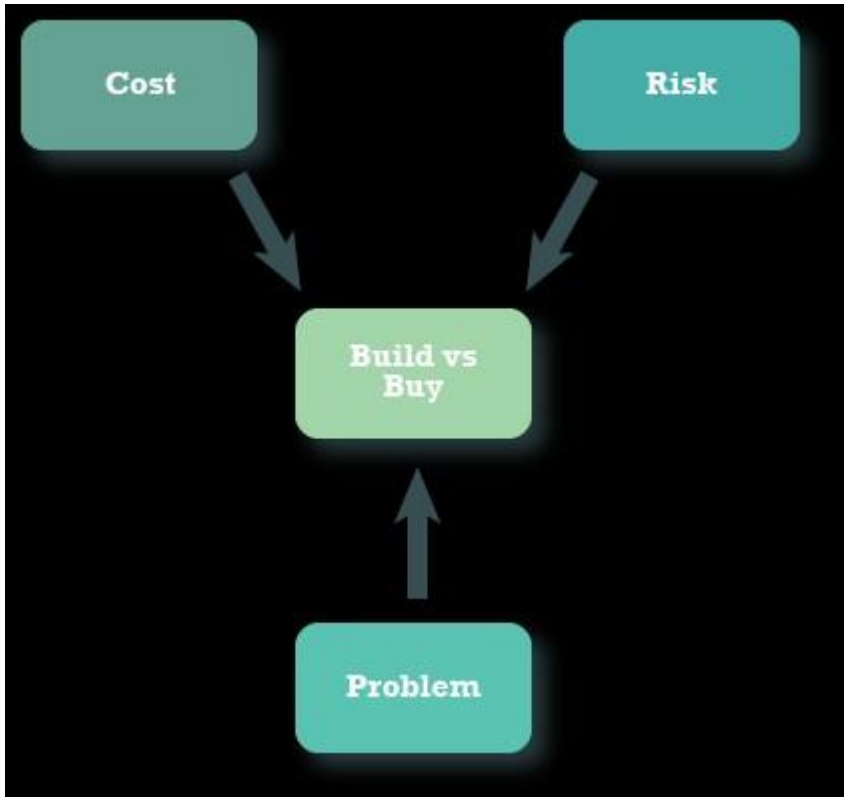
The four basic steps in Software Project Estimation are –

- Estimate the size of the development product.
- Estimate the effort in person-months or person-hours.
- Estimate the schedule in calendar months.
- Estimate the project cost in agreed currency.

-Make/Buy Decision

Make-or-Buy Decision

A make-or-buy decision is the act of choosing between manufacturing a product in-house or purchasing it from an external supplier.



Build to Compete. Buy to Standardize.

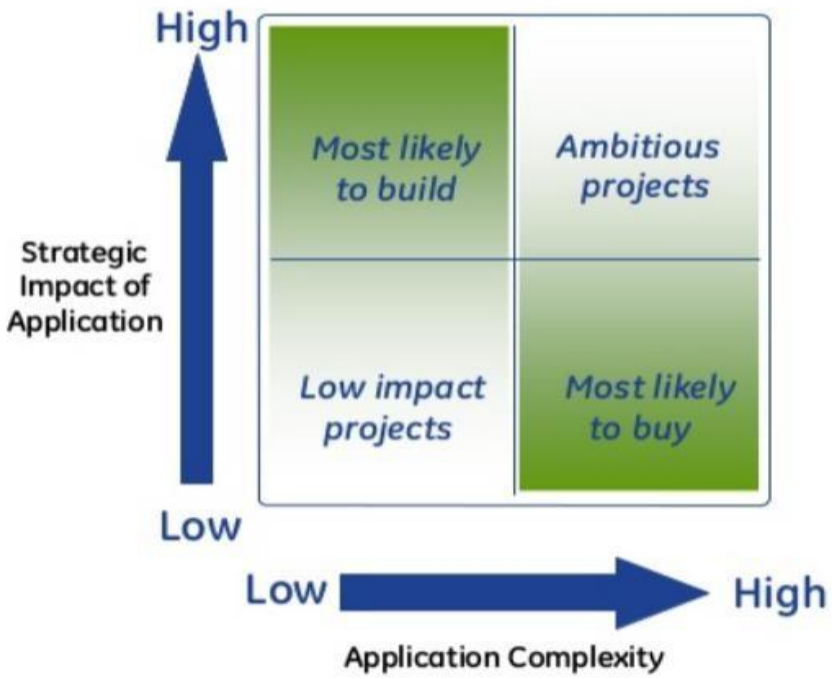
Make-or-Buy Decision

Options:

1. software may be purchased (or licensed) off-the shelf
2. “fully-experience” or “partially experience” software components may be acquired and then modified and integrated to meet specific needs.
3. software may be custom built by an outside contractor to meet the purchaser’s specification.

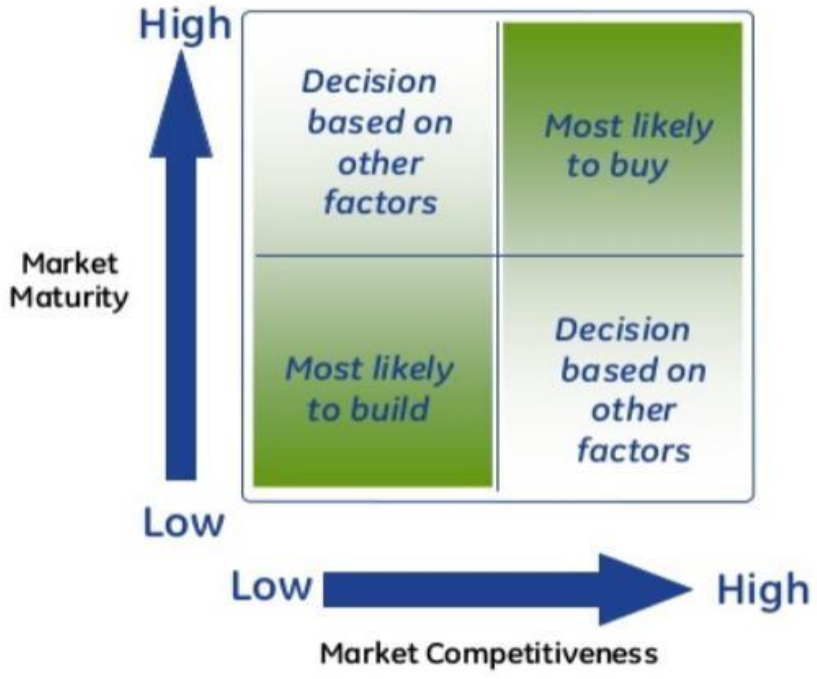
Make-or-Buy Decision : Strategic Considerations

Project Impact vs. Complexity



Application complexity is a factor in the decision, as IT organizations are more unlikely to build highly complex applications. However, several financial services firms build wealth-management software in-house, few organizations develop proprietary Enterprise Resource Planning (ERP) Systems.

Market Maturity vs. Competitiveness



Mature markets are more likely to offer sufficient, industry-specific functionality than newer less mature markets.
Competitive markets are more likely to offer low software pricing than markets dominated by vendors.

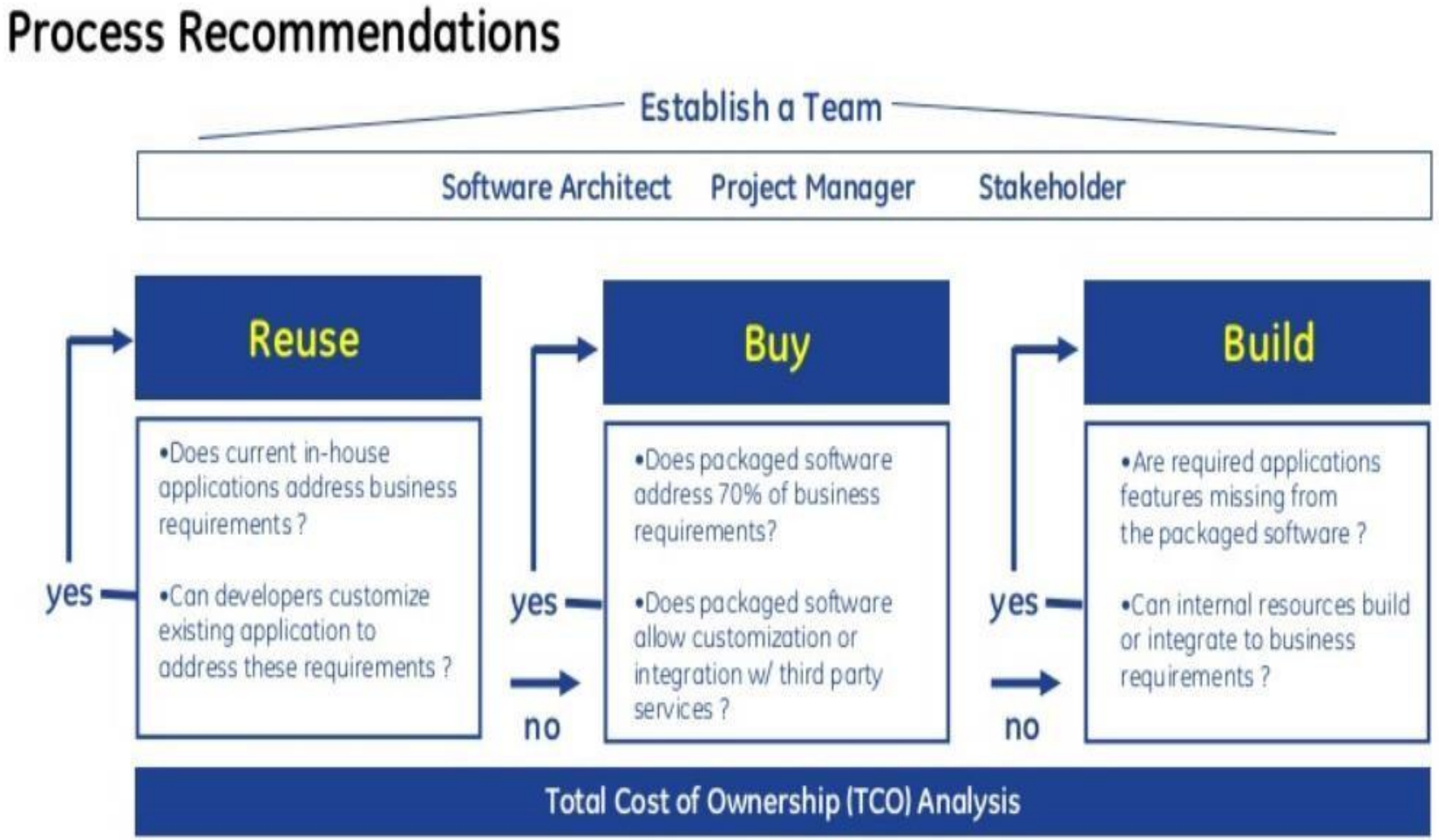
Make-or-Buy Decision : Evaluating Cost of Ownership

A simple method of evaluating the Total Cost of Ownership is to align Lifecycle Milestones side-by-side.

Where: TCO = Capital Costs (one-time) + Operating Costs (annual)

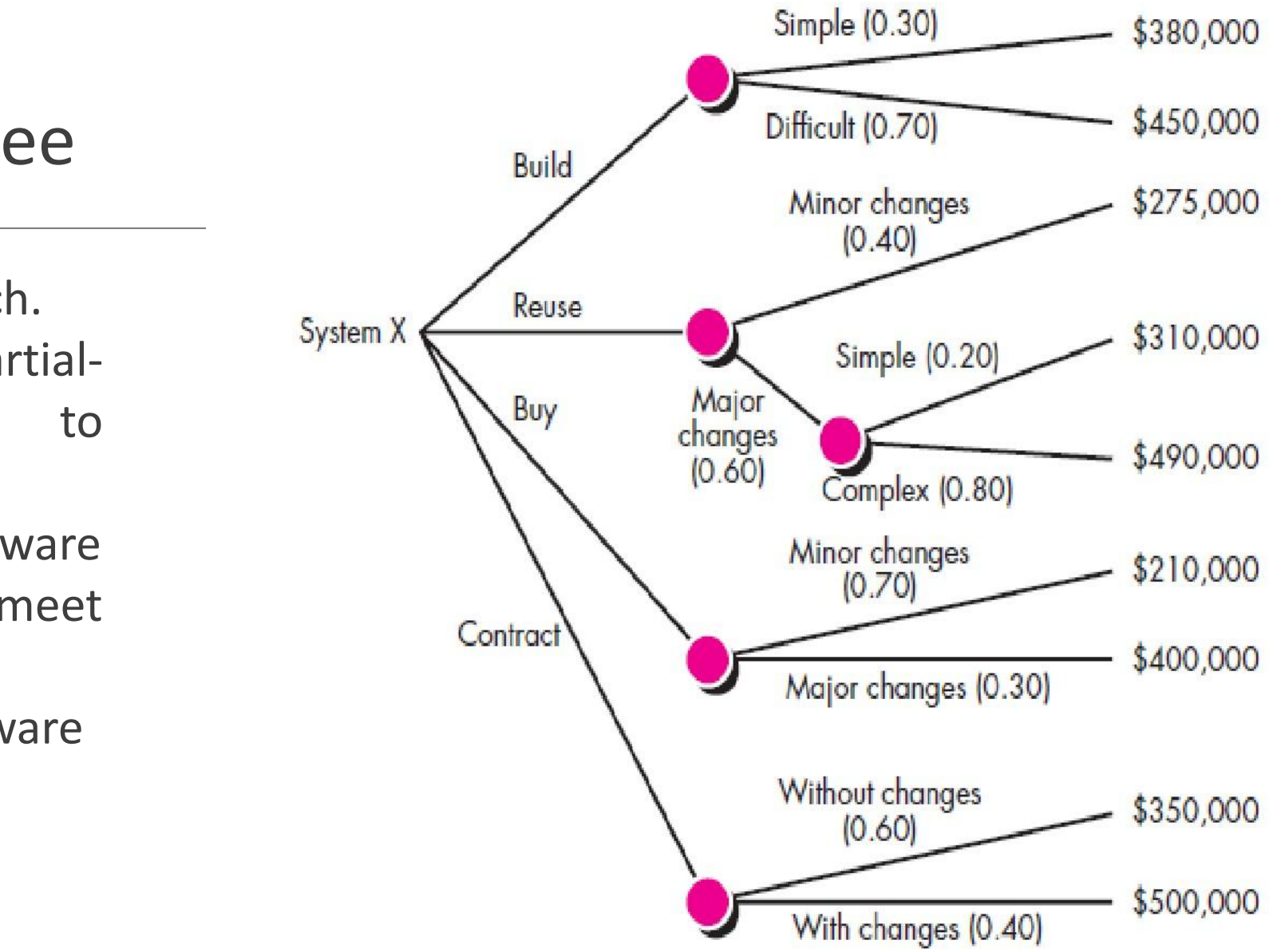
Build Cost Factors	Cost($)	Buy Cost Factors	Cost($)
Software Development Resources	$ -	Software Licensing	$ -
Software Quality Control & Testing	$ -	Software Implementation & Integration	$ -
Software Configuration & Deployment	$ -	Application Customization	$ -
End-User Training	$ -	End-User Training	$ -
Ongoing Maintenance & Enhancements (RTS)	$ -	Ongoing Maintenance & Support Fees	$ -
Software End of Life	$ -	Software End of Life	$ -

Make-or-Buy Decision : Process

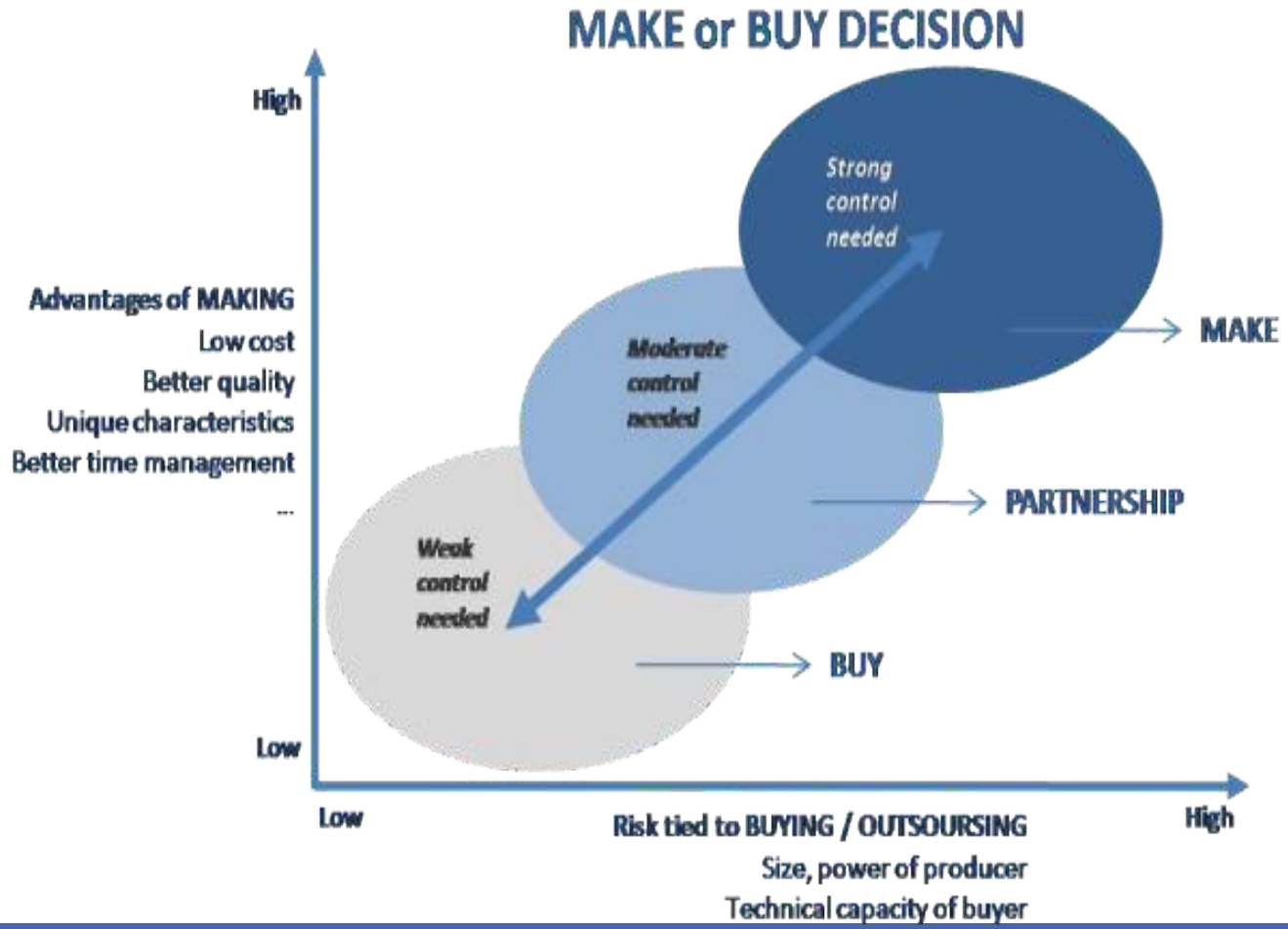


Creating a Decision Tree

1. Built system X from scratch.
2. reuse existing partial-experience to contractor the system.
3. Buy an available software product and modify it to meet local needs.
4. Contract the software development to an outside vendor.



Make-or-Buy Decision



MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

COCOMO II - Constructive Cost **Model**

COCOMO - Project Planning

The Constructive Cost **Model** (**COCOMO**) is a procedural software cost estimation **model** developed by Barry W. Boehm.

COCOMO-II by Barry Boehm

Has three types of model:-

Application Composition Model- Used during the early stages of software engineering.

Early design stage Model- Used once requirements have been stabilized and basic software architecture has been established .

Post-architecture stage Model –Used during the construction of the software.

The sizing information followed by this model is the direct software measure **'object points'.**

COCOMO II - Project Planning

- Object Point is computed using counts of the number of
  - Screens (at the user interface)
  - Reports
  - Components likely to be printed to build the application.
- Each object instance (e.g. a screen or report) is classified into one of three complexity levels (i.e. simple , medium, difficult)
- In essence, complexity is a function of
  - Number and source of the client and server data tables that are required to generate the screen or report.
  - Number of views or sections presented as part of the screen or report.

COCOMO II - Project Planning

Once complexity is determined, the number of screens reports, components are weighted according to following table:

Object type	Complexity weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component			10

COCOMO II - Project Planning

- The object point count is then determined by multiplying the original number of object instances by the weighting factor in the figure and summing to obtain a total object point count.
- When components based development or general software reuse is to be applied, the percent of reuse (% reuse) is estimated and the object point count is adjusted:

$$NOP = (\text{object points}) * [(100 - \%\text{reuse})/100]$$

Where New Object Points NOP is defined as new object points.

COCOMO II - Project Planning

- Now the estimate of project effort is computed as follows:

Estimated effort= NOP/PROD

Productivity rate (PROD) can be derived from following table based on developer experience and organization maturity.

Developer's experience/capability	Very low	Low	Nominal	High	Very high
Environment maturity/capability	Very low	Low	Nominal	High	Very high
PROD	4	7	13	25	50

COCOMO II Example

- Use the COCOMO II model to estimate the effort required to build software for a simple ATM that produces 12 screens, 10 reports, and will require approximately 80% as new software components. Assume average complexity and average developer/environment maturity. Use the application composition model with object points.

COCOMO II Example

- Given

Object	Count	Complexity	Weight Factor	Total Objects
Screen	12	Simple	1	12
Report	10	Simple	2	20
3GL Components	0	NA	NA	0
			Total Objects Points :	32

COCOMO II Example

- It is given that 80% of components have to be newly developed. So remaining 20% can be reused
- Now compute new object points as

NOP = (object points) * [(100 - %reuse)/100]

NOP = 32 * (100-20)/100 = 32*80 / 100

NOP = 25. 6 object points

- Since productivity is given average, we can assume PROD = 13
- Hence , effort = NOP/PROD

effort = 25.6/13

effort = 1.96 person months

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

COCOMO II Summary

- Size of project can be listed as object points, function points or source lines of code (SLOC)

Example 2 :

For a given IIST airline sales system, at the early stage, we need 3 screens and 1 report:

1. a booking screen to record a new advertising sale booking
2. a pricing screen showing the advertising rate for each day and each flight
3. an availability screen showing which flights are available

1.a sales report showing total sales for the month and year, and comparing them with previous months and years

Consider the developer experience and organization maturity as very low and requires 70% as a new software component.

The booking screen is classified as simple. Similarly, the levels of difficulty of the pricing screen, the availability screen and the sales report are classified as simple, medium and medium respectively. There is no 3GL component.

COCOMO II Example 2

Table 1 Ratings for IIST airline sales system

Objects		Complexity	Weight
Booking Screen	(1)	Simple	1
Pricing Screen	(1)	Simple	1
Availability Screen	(1)	Medium	2
Report	(1)	Medium	5
Total	**9**		

Object type	Complexity weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component			10

COCOMO II Example 2

NOP= (object points) * [(100-%reuse)/100]
NOP =9 * (100-30)/100
=9 * 0.7
= 6.3 object points.

Effort= NOP /PROD
= 6.3 / 4
= 1.575

Developer's experience/capability	Very low	Low	Nominal	High	Very high
Environment maturity/capability	Very low	Low	Nominal	High	Very high
PROD	4	7	13	25	50

According to COCOMO II, the project requires approx. 1.57 person-months.

MIT-WPU

।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

SWOT Analysis

SWOT Analysis

Purpose of a **SWOT Analysis**:

The **purpose** is to get managers thinking about everything that could potentially impact the success of a new project.

It is a study undertaken by an organization to identify its internal strengths and weaknesses, as well as its external opportunities and threats.

SWOT Analysis

Objectives :

1. To make a summary analysis of external and internal factors.
2. To identify key items for the management of the organization, which involves establishing priorities for action.
3. To prepare strategic options: risks and problems to solve.
4. To conduct a sales forecast in agreement with market conditions and study the capabilities of the company in general.

Internal and External Factors

Internal factors — the strengths and weaknesses **internal** to the organization.

External factors — the opportunities and threats presented by the environment **external** to the organization.

SWOT Analysis

Strengths: characteristics of the business or project that give it an advantage over others

Weaknesses: characteristics of the business that place the business or project at a disadvantage relative to others

Opportunities: elements in the environment that the business or project could exploit to its advantage

Threats: elements in the environment that could cause trouble for the business or project

Example of SWOT Analysis

Example of SWOT Analysis

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Functions of Manager

Functions of Manager

1. P- Planning
2. O- Organizing
3. S- Staffing
4. D- Directing
5. CO- Controlling
6. R- Reporting
7. B- Budgeting

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

PLANNING

PLANNING

- Planning means setting an organization’s goal and deciding how best to achieve them.
- Planning is decision making, regarding the goals and setting the future course of action from a set of alternatives to reach them.
- The plan helps to maintain the managerial effectiveness as it works as a guide for the personnel for the future activities.

MIT-WPU

।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

ORGANISING

ORGANISING

- Is the process of grouping together of men and establishing relationship among them, defining the authority and responsibility of personnel by using the company's other basic resources to reach predetermined goals or objectives.

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

STAFFING

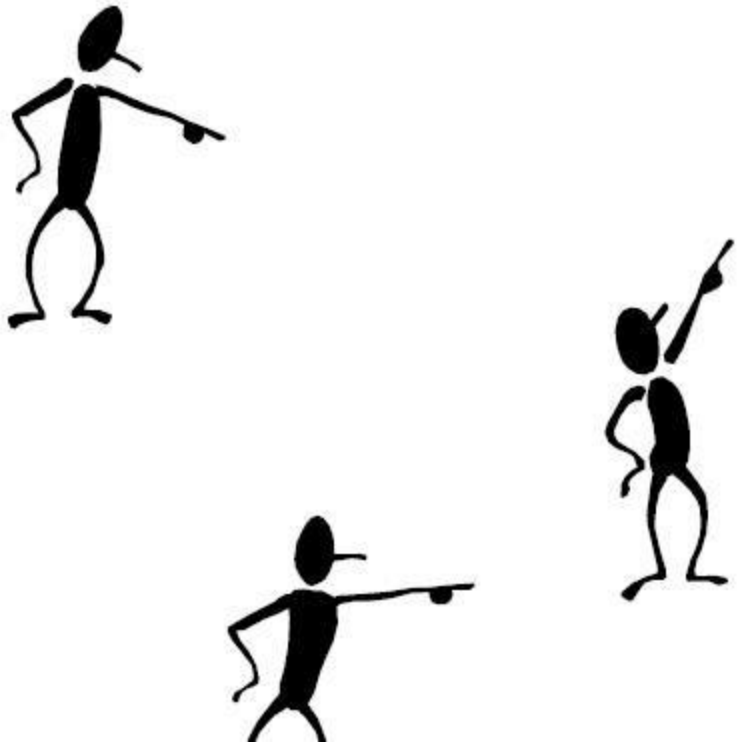
STAFFING

- Filling and keeping filled with qualified people all positions in the nosiness.
- Recruiting, hiring, training, evaluating and compensating are the specific activities included in the function.
- In the family business, staffing includes all paid and unpaid positions held any family members including the owner/operators.

Staffing Process

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

DIRECTING



DIRECTING

- Refers to the process of motivation, communication and leadership.
- The purpose of directing is to channel the behavior of all personnel to accomplish organization's mission and objectives while maintaining them. Helping them simultaneously

CONTROLLING

Process of evaluating and correcting activities to keep organization on course.

- Measuring performance
- Comparing performance against standards
- Identifying deviation from standards
- Investigating causes of deviations
- Taking Corrective Action

Reporting

- It includes daily follow up of activities with help of reports which are submitted by subordinates to his/her superior.
- Reporting keeps him/her informed with day to day activities.

Budgeting

- Budget is an estimate of future needs covering all the activities of an enterprise for a definite period of time.
- A budget is prepared for each separate activity of business.
- This is done to control the expenses of organizations within certain limit.

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Team Building & Development

Team Building and Development

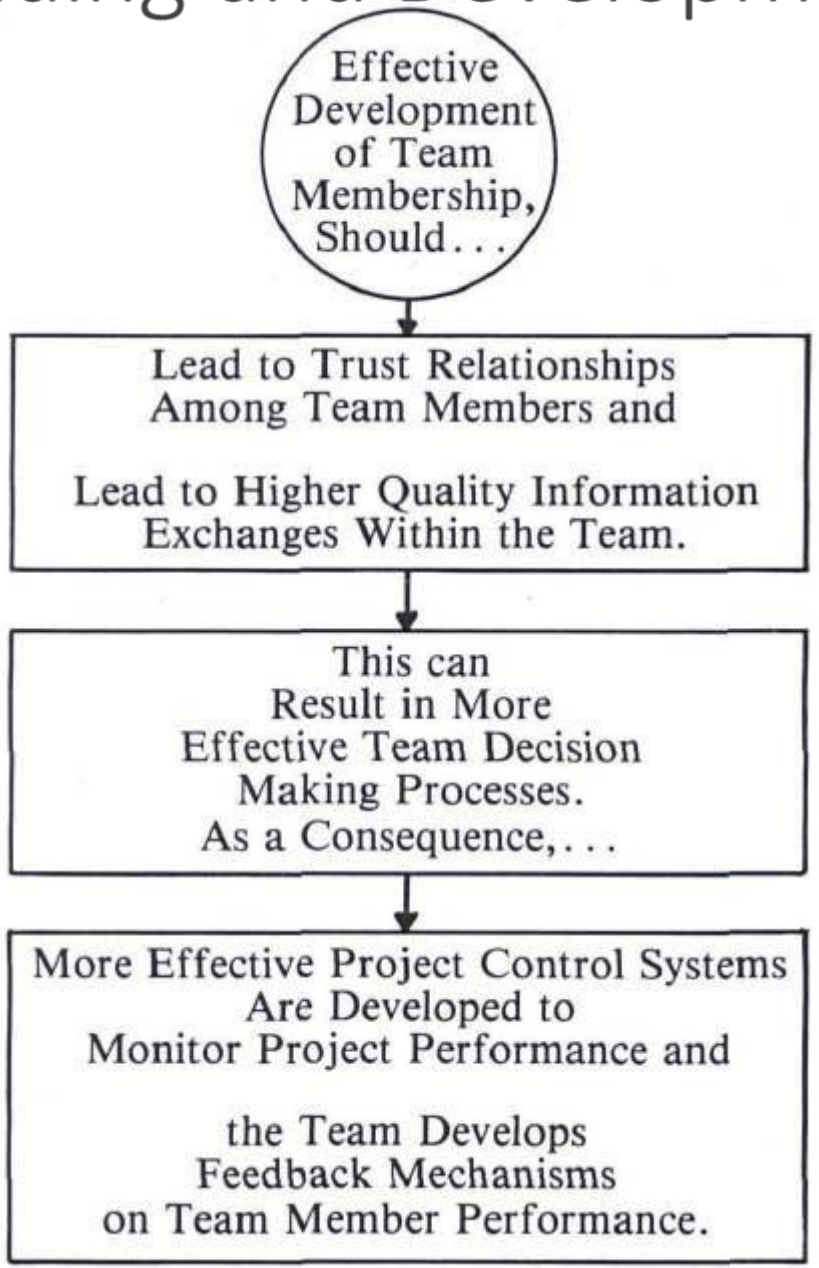
Selecting the project Team

- At what point is the project team formed?
- Who should select the project team?
- What types of people should be on the team?
- How do the team members get started?

Choosing Effective Team Leader

- Excellent Communicator
- Knowledge of Project Management Principles
- Highly Organized
- Strong Ability to Read People
- Accurate Estimating Skills
- Self-Assured

Team Building and Development



MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Characteristics of Effective Teams

An effective project team has:

- A clear understanding of the project objective
- Clear expectations of each person's role and responsibilities
- A results orientation
- A high degree of cooperation and collaboration
- An atmosphere of open communication
- A high level of trust

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Barriers to Team Effectiveness

- Unclear goals
- Unclear definitions of roles and responsibilities
- Lack of project structure
- Lack of commitment by team members
- Poor communication
- Poor leadership
- Turnover of team members
- Dysfunctional behavior

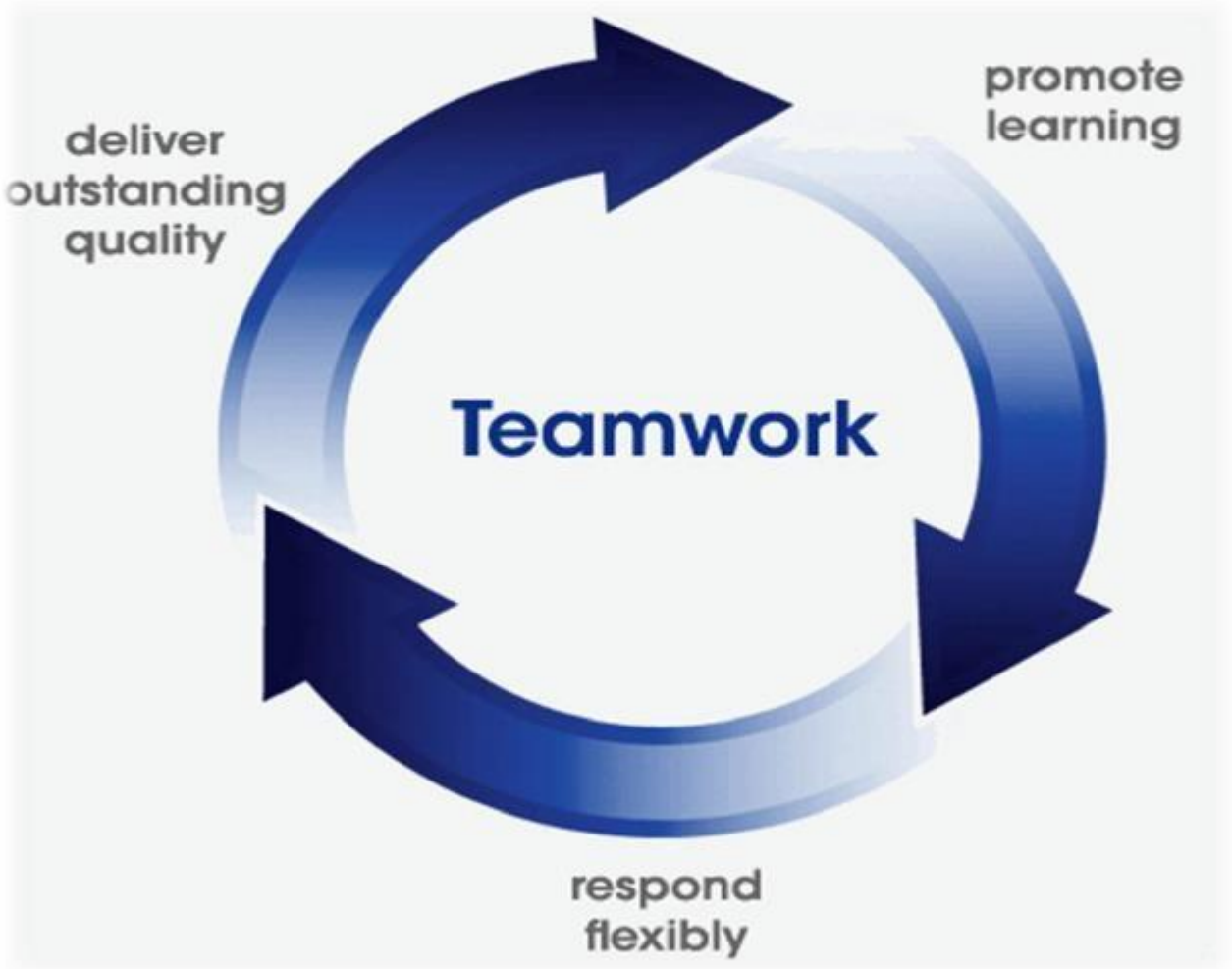
MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Teams are Most Effective When:

- There are 10 or fewer members on the team
- Members volunteer to serve on the team
- Members serve on the project from beginning to end
- Members are assigned to the project full time
- Organization culture promotes cooperation and trust
- Members report solely to the project manager
- All relevant functional areas are represented on team

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

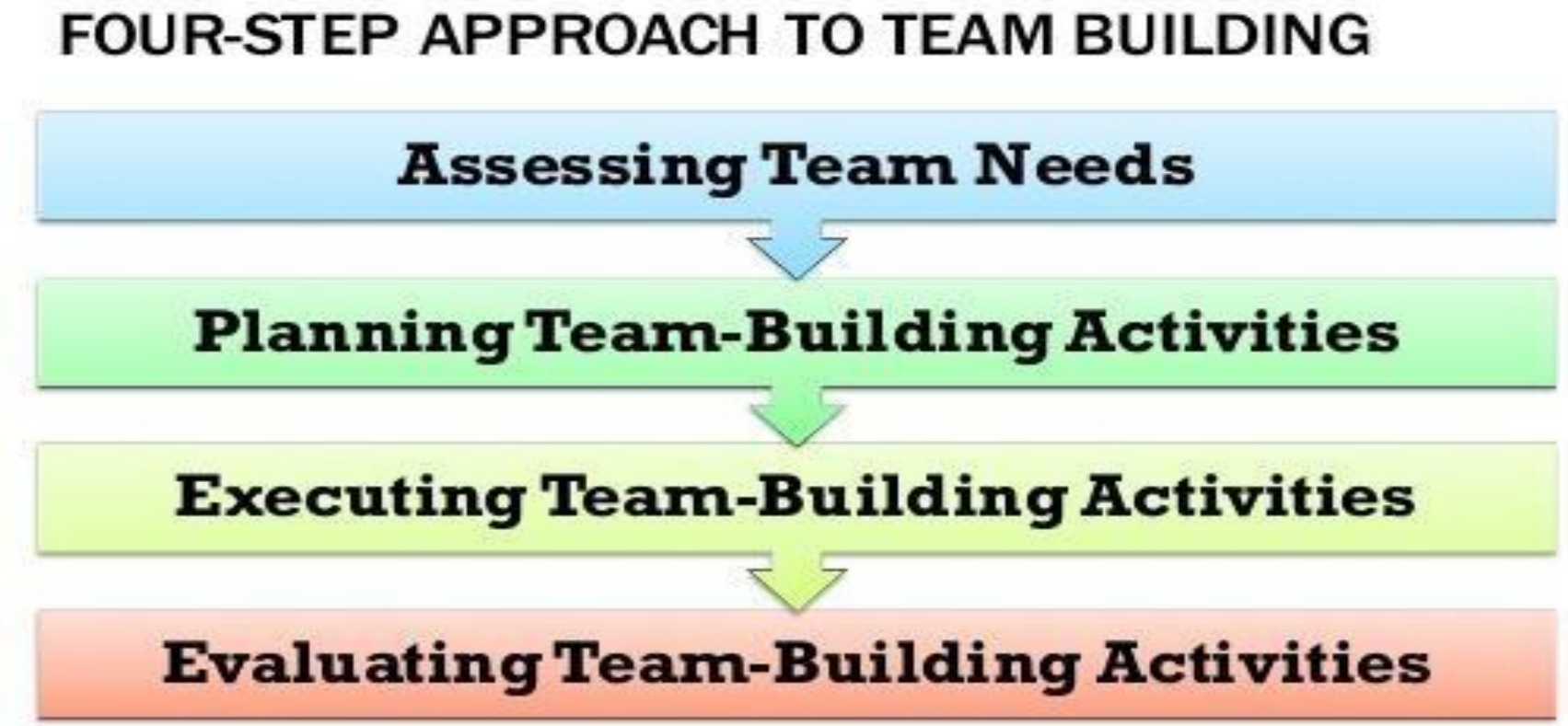
Teamwork Cycle



MIT-WPU

।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Team Building



Team Building -Assess

- Look for strengths and weaknesses in team members
- For a team to be successful the following characteristics are needed:
  - A clear direction that is understood by all team members
  - Team players
  - Understood and accepted accountability measures.

Team Building - Plan

- Planning
  - Based on the results of needs assessment
  - Activities should be based on the strengths and weaknesses of the needs assessment.

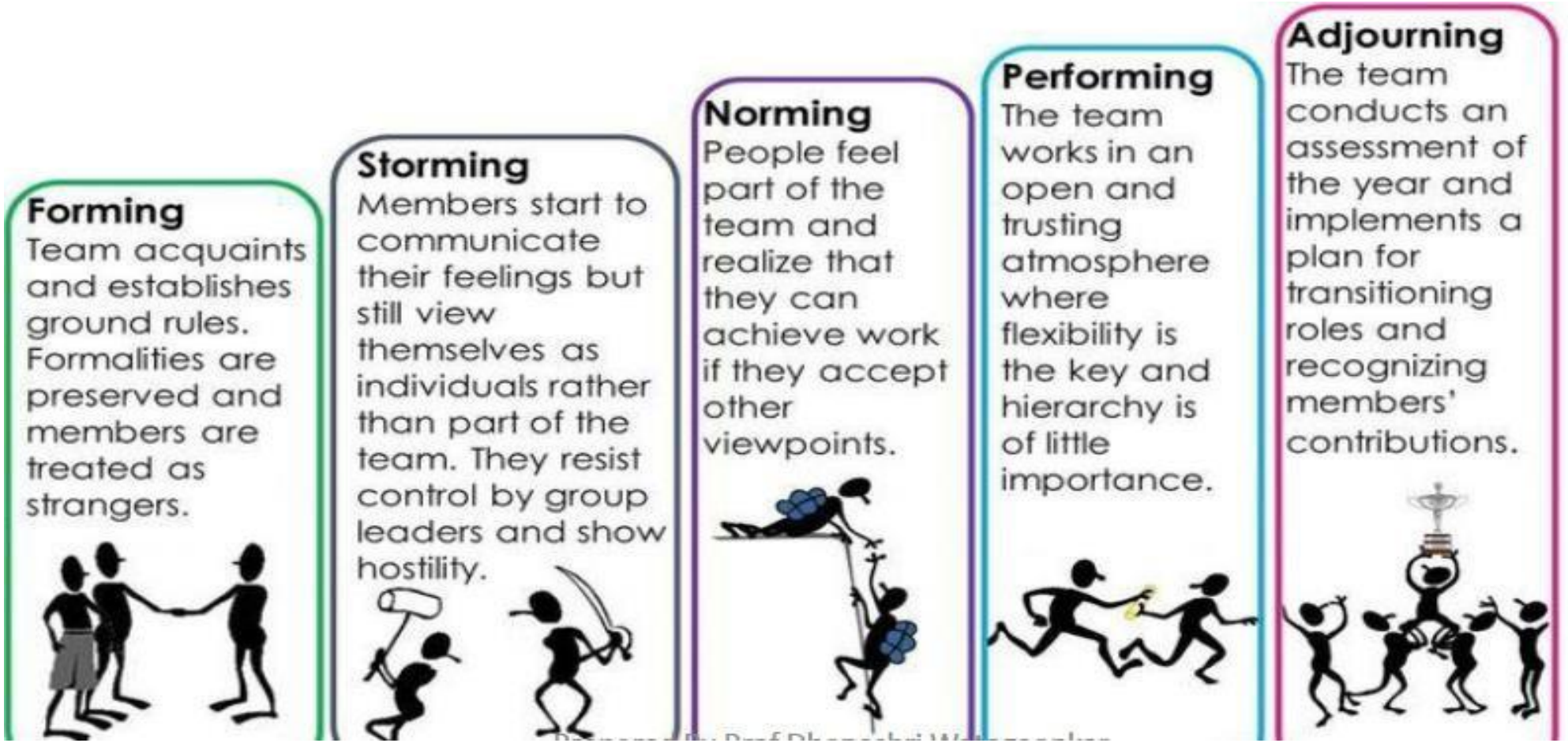
Team Building - Execute

- Execution
  - Just-in-time
  - Continuous improvement

Team Building - Evaluate

- Evaluation
  - Effectiveness can be measured based on how well weaknesses identified in the needs assessment were strengthened.
  - Re-administer the needs assessment.

Stages of Team Development

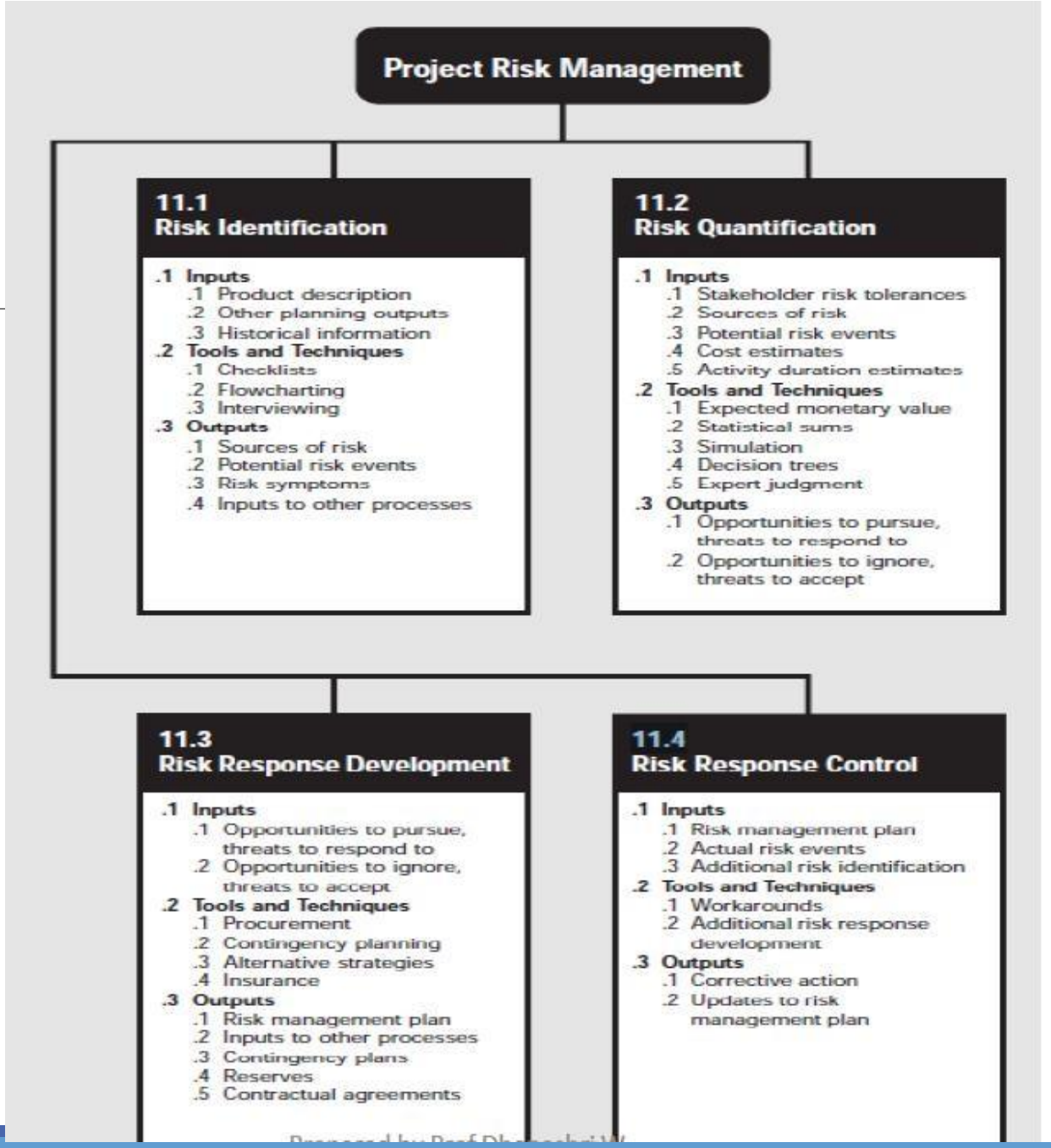


Risk Management

Risk management means risk containment and mitigation.

First, you've got to identify and plan. Then be ready to act when a risk arises, drawing upon the experience and knowledge of the entire team to minimize the impact to the project.

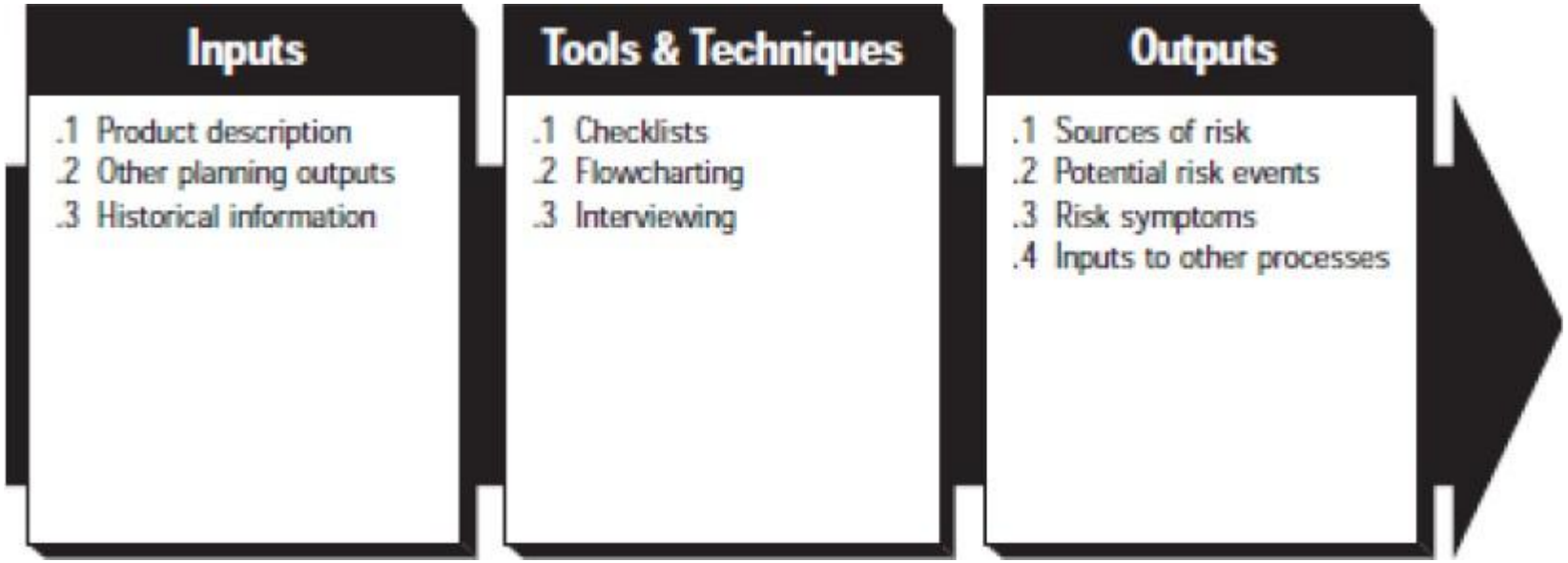
Risk Management



Risk Management

- **Risk Identification-** determining which risks are likely to affect the project and documenting the characteristics of each.
- **Risk Quantification-** evaluating risks and risk interactions to assess the range of possible project outcomes.
- **Risk Response Development-** defining enhancement steps for opportunities and responses to threats.
- **Risk Response Control-** responding to changes in risk over the course of project.

Risk Identification



Inputs

Inputs

- Product description
- Other planning outputs:
  - WBS
  - Cost Estimates
  - Staffing plan
  - Procurement management plan
- Historical Information

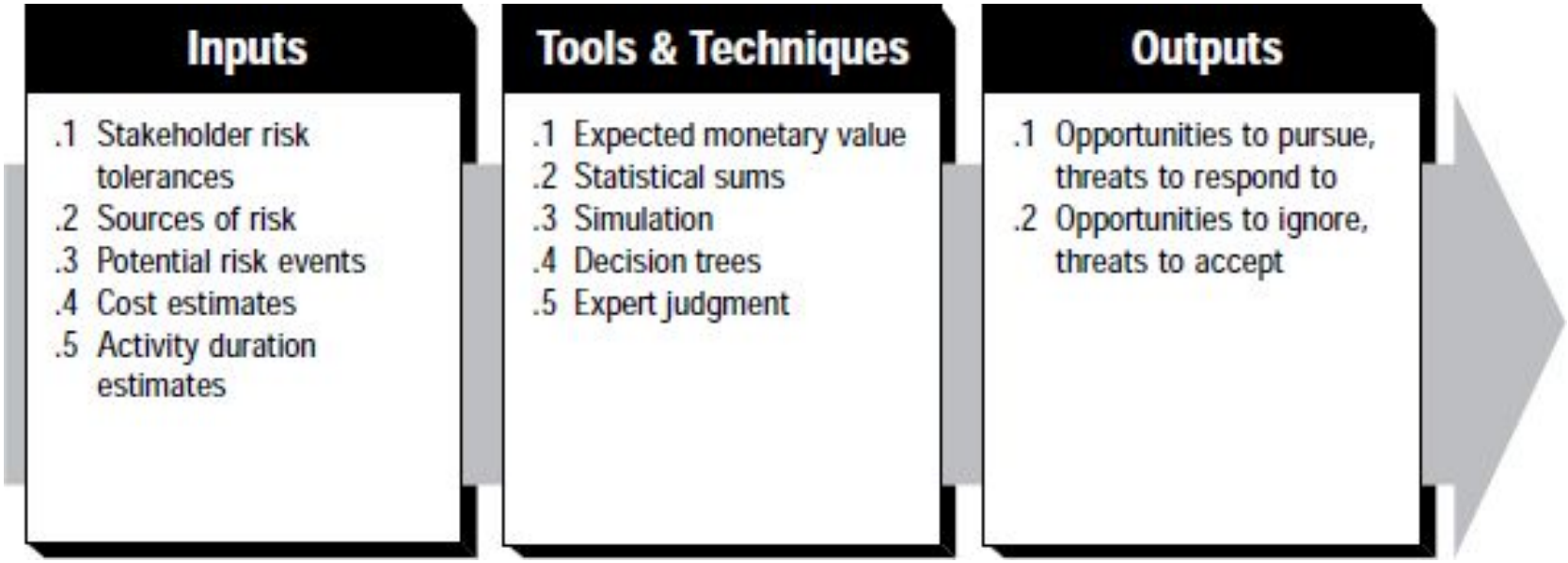
Tools & Techniques

- Checklists
- Flowcharting
- Interviewing : Risk oriented interviews with various stakeholders.

Outputs

- Sources of risk
  - -Change in requirements
  - -Design errors , omissions and misunderstandings
  - -Poorly defined or understand roles and responsibilities
  - -Poor estimates
  - -Insufficient skilled staff
- Possible risk events
- Risk symptoms
- Inputs to other processes

Risk Quantification



Risk Quantification

Inputs

- Stakeholder risktolerance: Different organizations and different individuals have different tolerances for risk.
- Source of risk
- Potential risk events
- Cost estimates
- Activity duration estimates.

Tools & Techniques

Expected monetary value.

- Risk event probability—an estimate of the probability that a given risk event will occur.
- Risk event value—an estimate of the gain or loss that will be incurred if the risk event does occur.

Statistical sums. The range of total project costs can be used to quantify the relative risk of alternative project budgets or proposal prices.

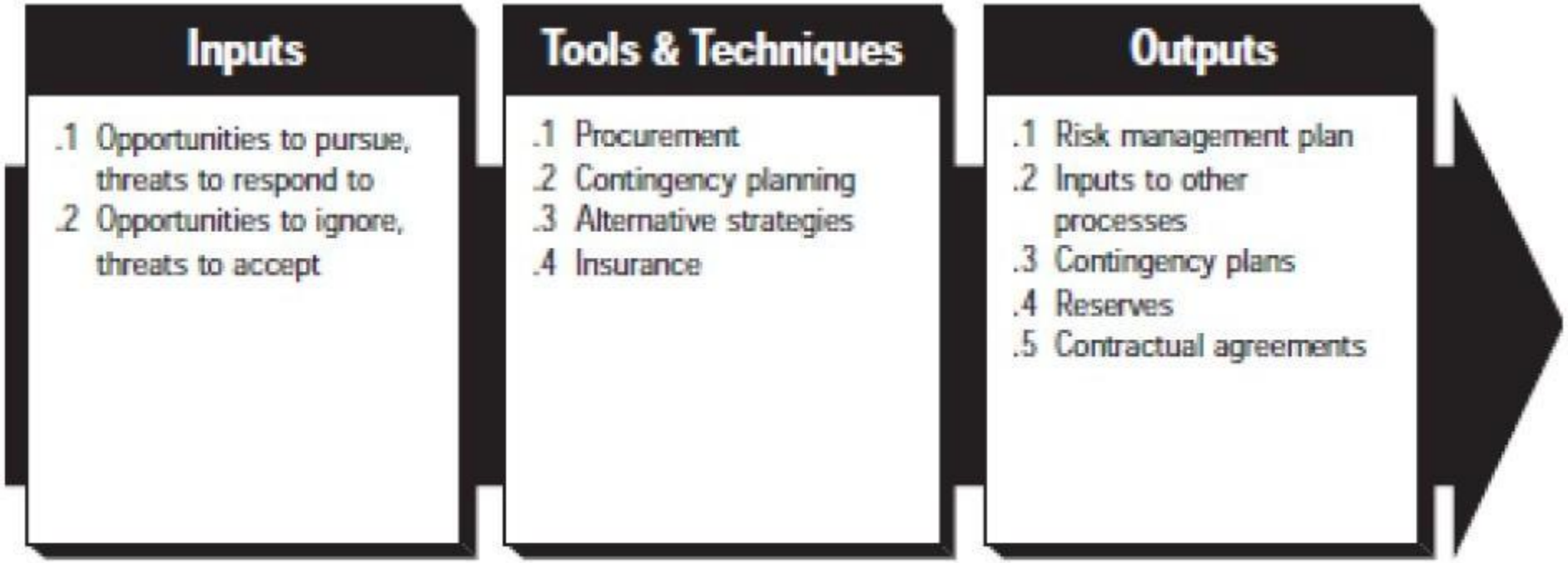
Simulation: It uses a representation or model of a system to analyze the behavior or performance of the system.

Decision trees. A decision tree is a diagram that depicts key interactions among decisions and associated chance events as they are understood by the decision

Outputs

- Opportunities to pursue, threats to respond to: The major output from risk qualification is a list of opportunities that should be pursued and threats that require attention.
- Opportunities to ignore, threats to accepts: The risk quantification process should also document
  - Those sources of risk and risk events that the project management team has consciously decided to accept or ignore.
  - Who made the decision to do so.

Risk Response Development



Inputs

- Opportunities to pursue, threats to respond to
- Opportunities to ignore, threats to accepts

Tools & Techniques

- Procurement
- Contingency planning
- Alternative Strategies: Risk event can often be prevented or avoided

  by changing the planned approach

- Insurance.

Outputs

Risk management plan. It should document the procedure that will be used to managed risk throughout the project.

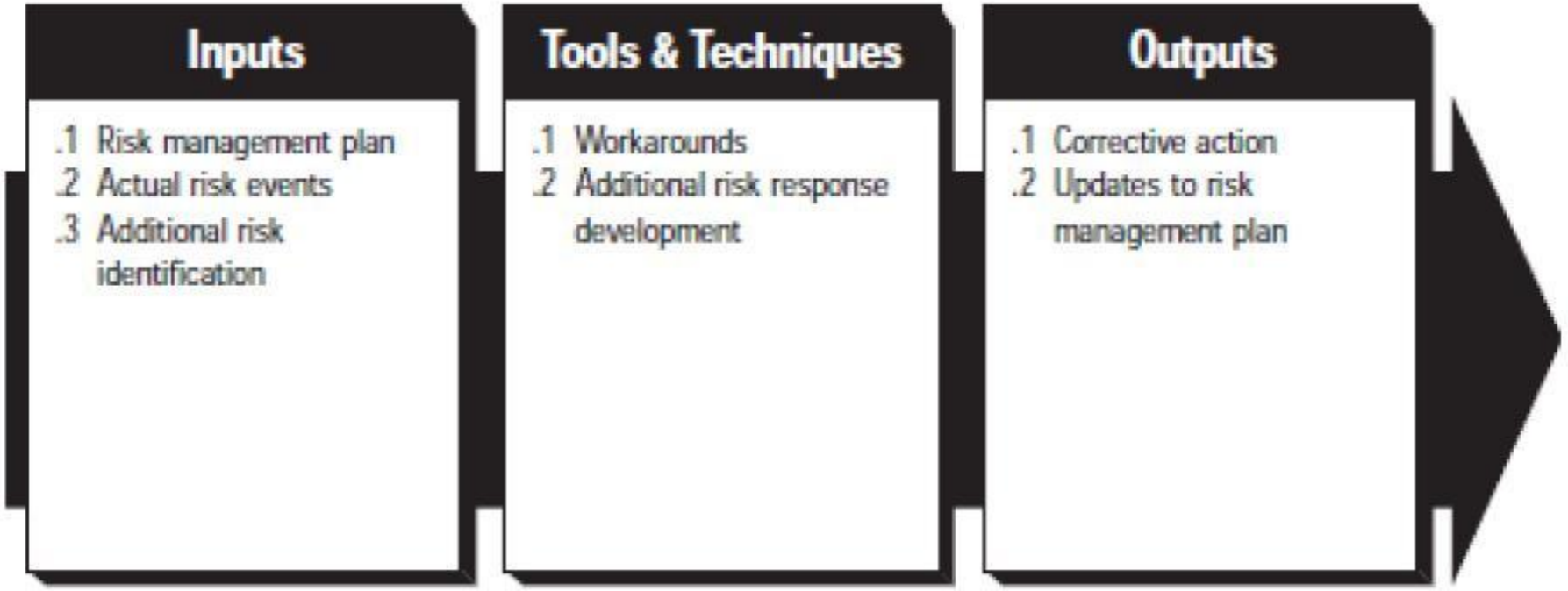
Inputs to other processes. Suggested alternative strategies, contingency plans, anticipated procurements, and other risk-related outputs must all be feedback.

Contingency plans. Contingency plans are pre-defined action steps to be taken if an identified risk event should occur.

Reserves. A reserve is a provision in the project plan to mitigate cost and/or schedule risk.

Contractual agreements. Insurance, services.

Risk Response Control



Inputs

- Risk management plan
- Actual risk events
- Additional risk identification

Tools & Techniques

- Workarounds: Workarounds are unplanned responses to negative risk events means the responses was not defined in advanced of the risk event occurring.
- Additional risk response development: If the effect is grater than expected, the planned responses may not be adequate and it will be necessary to repeat the response development process and perhaps the risk quantification process as well.

Outputs

- Corrective action
- Updates to risk management plan

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Gantt Chart

Gantt charts are **useful** for planning and scheduling projects. They help you assess how long a project should take, determine the resources needed, and plan the order in which you'll complete tasks. They're also **helpful** for managing the dependencies between tasks.

A **Gantt Chart** is a timeline that is used as a project management tool to illustrate how the project will run. You can view individual tasks, their durations and the sequencing of these tasks. View the overall timeline of the project and the expected completion date.

A **Gantt Chart** is a horizontal bar **chart** that visually represents a project plan over time. Modern **gantt charts** typically show you the status of—as well as who's responsible for—each task in the project. **Gantt charts** contain the following **features**: Start and end dates for tasks

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Gantt Chart

References

1 https://www.simplilearn.com/project-and-process-metrics-article

2 https://www.youtube.com/watch?v=mx3DejbTnSQ

3https://online.visual-paradigm.com/tutorials/swot-analysis-tutorial/ [4]

https://webcache.googleusercontent.com/search?q=cache:YZXp7NA4mocJ:https://www8.cs.umu.se/kurser/TDBB12/HT05/Lectures/PVKHT05project%2520management.ppt+&cd=10&hl=en&ct=clnk&gl=in

5https://webcache.googleusercontent.com/search?q=cache:J-emu92YVAQJ:https://www.softwareresearch.net/fileadmin/src/docs/teaching/SS06/PM/PMBOK11.PDF+&cd=9&hl=en&ct=clnk&gl=in

6 https://en.wikipedia.org/wiki/Source_lines_of_code

7 https://www.slideshare.net/nagendraamatya/applied-research-methodology-lecture-1-10496165

Disclaimer:

a. Information included in this slides came from multiple sources. We have tried our best to cite the sources. Please refer to the References to learn about the sources, when applicable.

b. The slides should be used only for academic purposes (e.g., in teaching a class), and should not be used for commercial purposes.