50.003
Elements of Software Construction
Lecture 1

Introduction to Software Engineering
Software Modelling
UML

Scope

- What is Software Engineering?
  - The Software Development Life Cycle (SDLC) model
- Software Engineering methodologies
  - Unified Process
  - Requirement Gathering
    - functional and non-functional requirements
- Software modelling
  - UML & UML models

Learning Outcomes

To be able to :

1. define Software Engineering.
2. describe the main issues in Software Engineering
3. explain SDLC, its phases and its objectives
4. explain what are software engineering methodologies
5. differentiate between functional and non-functional requirements
6. explain the purpose of software modelling
7. explain UML

What is Software Engineering?

Software Engineering

What is Software Engineering?

Software Engineering is a discipline whose goal is the production of:

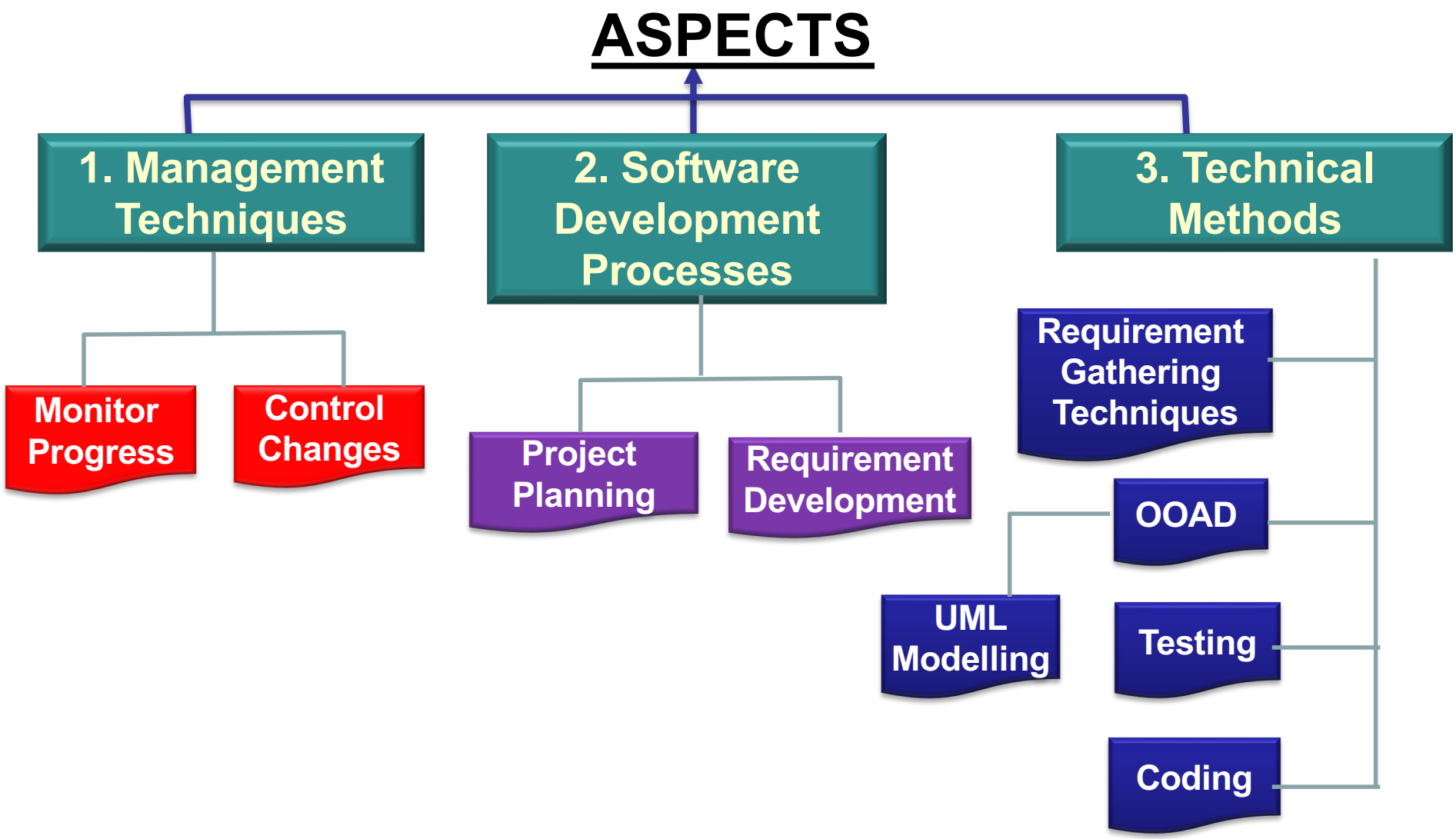
- **quality** software,
- delivered **on time** and
- **within budget**, that
- **satisfies the user's needs**

It encompasses:

- **processes**,
- **management techniques** and
- **technical methods**

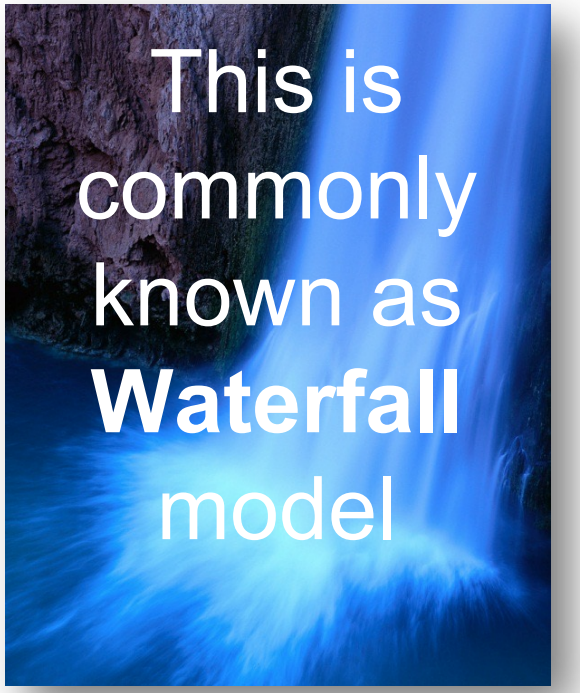
to achieve the software engineering goals.

Addressing Software Engineering Issues



Software Development Life Cycle (SDLC)

- **SDLC**: process of building, deploying, using, and maintaining a system
- **Five activities or phases** in a project
  1. **P**lanning,
  2. **A**nalysis,
  3. **D**esign,
  4. **I**mplementation,
  5. **S**upport



SDLC Phases and their Objectives

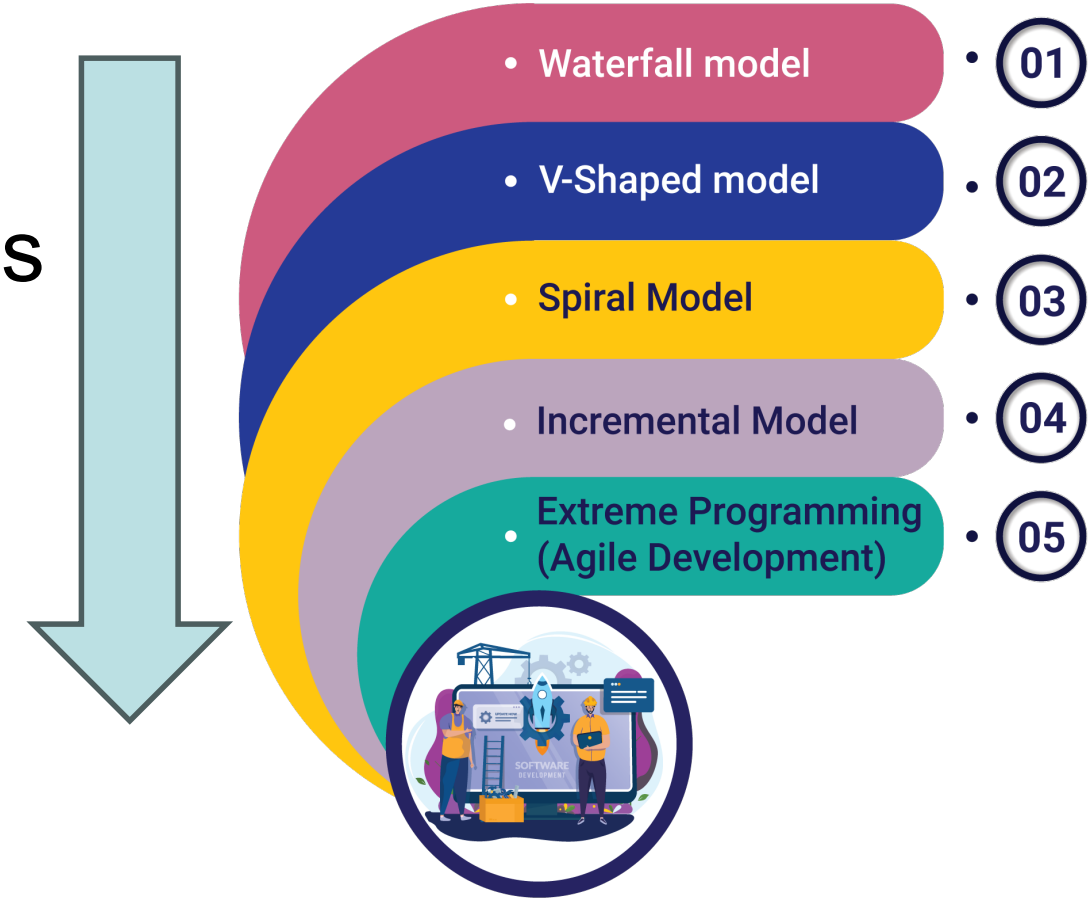
SDLC PHASE	OBJECTIVE
Project Planning	To identify the scope of the new system, ensure that the project is feasible, and develop a schedule, resource plan, and budget for the remainder of the project
Analysis	To understand and document in detail the business needs and the processing requirements of the new system
Design	To design the solution system based on the requirements defined and decisions made during analysis
Implementation	To build, test, and install a reliable information system with trained users ready to benefit as expected from use of the system
Support	To keep the system running productively initially and during the many years of the system's lifetime

Software Engineering Methodology

- defines the approach or framework used during software development.
- provide structured approaches for managing software development processes
- consist of a set of principles or rules
  - organize, plan, and execute development tasks toward the successful completion of a project
- tailored to specific project needs and contexts

Software Engineering Methodology

- Waterfall
- Unified Process (UP)
  - Agile
  - Scrum
- DevOps

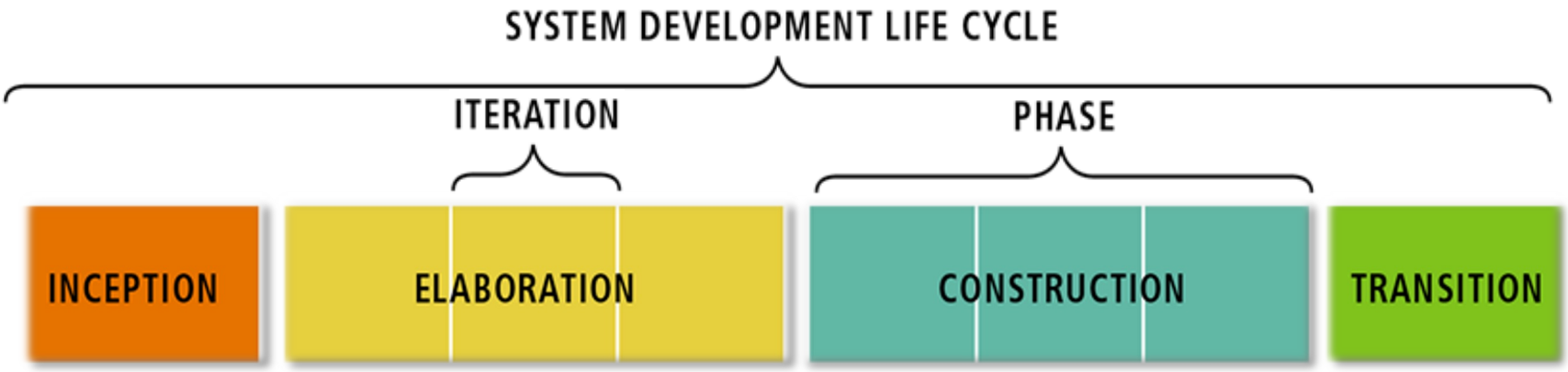


Unified Process (UP) – A methodology

- An iterative and incremental approach in software development
- Suited to tackling large and complex systems
- Based on the "divide and conquer" principles
  - plan out intermediate deliveries of system functionalities according to their priority
- Most influential OO system development
- Elaborate set of activities and deliverables for every step of the development process.

http://www.thefreedictionary.com/methodology

Unified Process (UP)
– A methodology



PHASES ARE NOT ANALYSIS, DESIGN, AND IMPLEMENT;

EACH ITERATION INVOLVES A COMPLETE
CYCLE OF REQUIREMENTS, DESIGN, IMPLEMENTATION, AND TEST DISCIPLINES

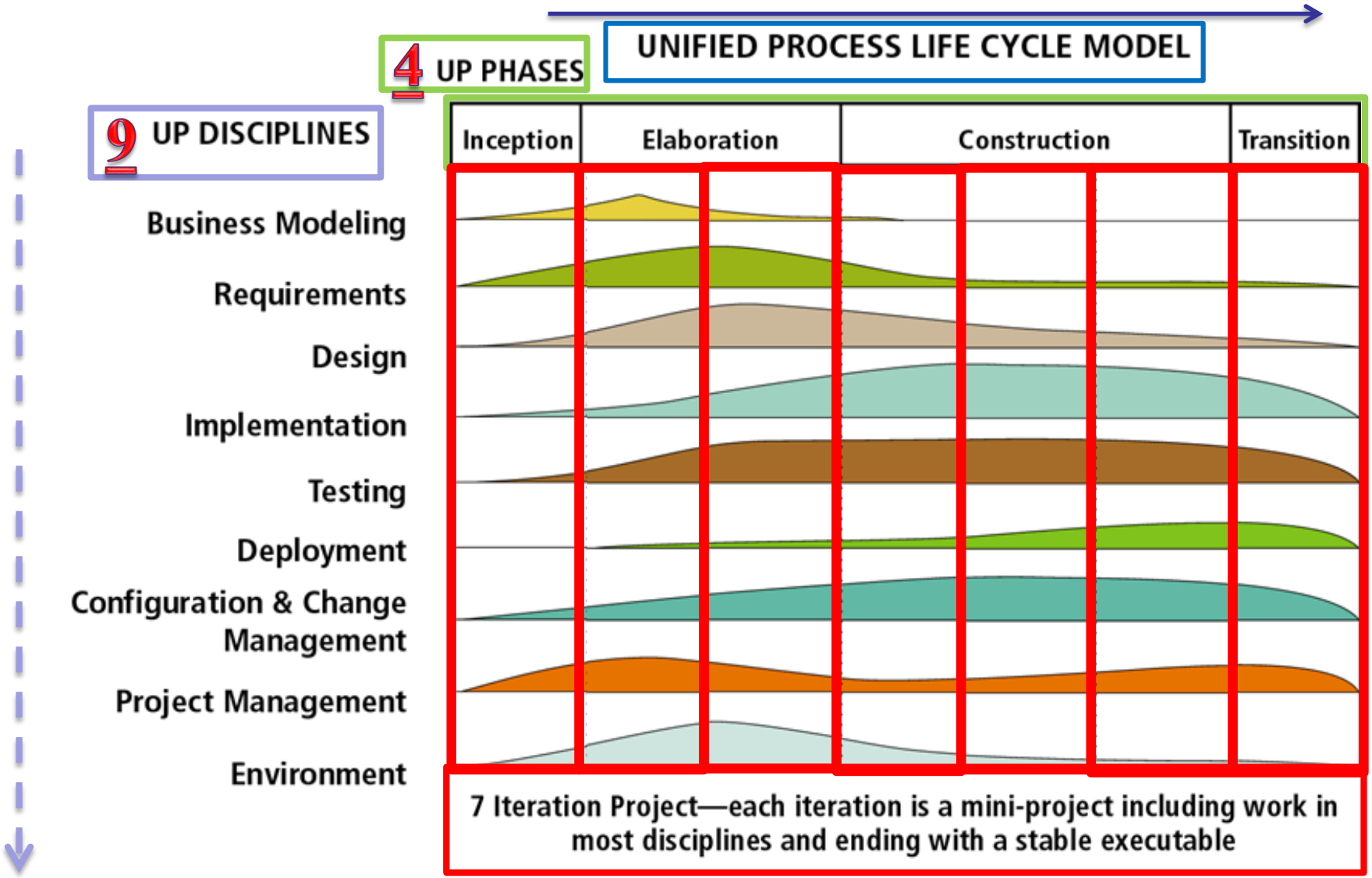
RUP Phases & their Objectives

UP PHASE	OBJECTIVE
Inception	Develop an approximate vision of the system, make the business case, define the scope, and produce rough estimates for cost and schedule.
Elaboration	Refine the vision, identify and describe all requirements, finalize the scope, design and implement the core architecture and functions, resolve high risks, and produce realistic estimates for cost and schedule.
Construction	Iteratively implement the remaining lower-risk, predictable, and easier elements and prepare for deployment.
Transition	Complete the beta test and deployment so users have a working system and are ready to benefit as expected.

Iterations in Unified Process

- Each phases can have 1 or more iterations
  - An **iteration** is like a mini project which has a duration to complete part of the system
  - There must be a clearly defined objective to be achieved at the end of each iteration
  - **Iterations** involves activities from all disciplines
- Each iteration involved **9 disciplines**

An example of UP Life Cycle Model



Disciplines in Unified Process

- A **discipline** is a set of functionally related activities.
- **Activities** are work to be carried out in each discipline that produce **artifacts.**
- **Artifacts** are UP work products.
  - Serves to document aspects of the system
  - Eg: model, a model element, a document, source code or executables.

The Unified Process Disciplines

- **6 main UP development disciplines**
  - 1. Business Modeling,
  - 2. Requirements,
  - 3. Design,
  - 4. Implementation,
  - 5. Testing, and
  - 6. Deployment
- These disciplines are used for a specific duration of the software development life cycle

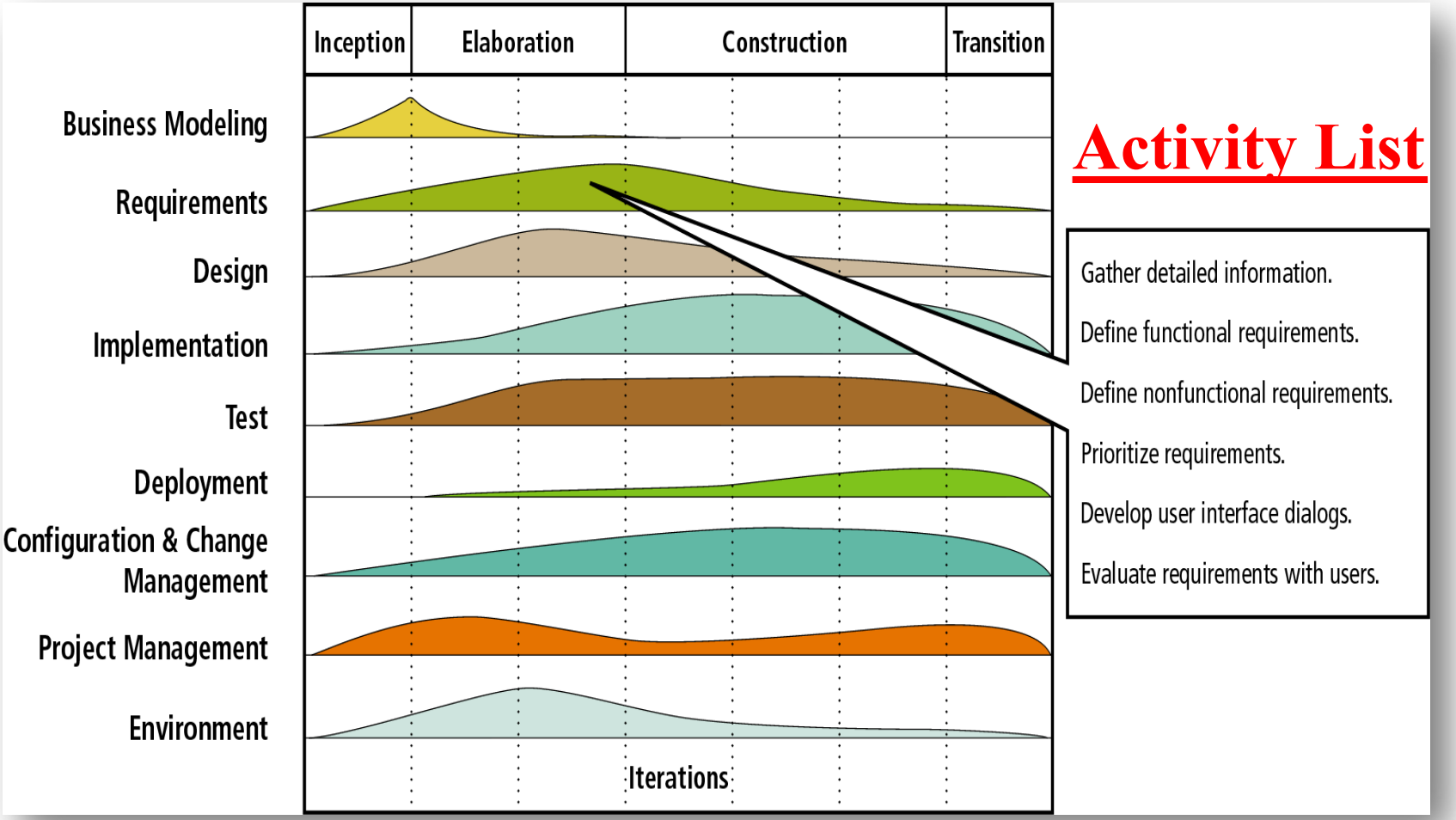
The Unified Process Disciplines

- **3 support disciplines**
  - 1. Project Management,
  - 2. Configuration and Change Management, and
  - 3. Environment
- These disciplines are used throughout the duration of the software development life cycle

The Requirements Discipline

- Establish and maintain agreement with customers and other stakeholders on what the system should do—and why!
- Provide system developers with a better understanding of the system requirements.
- Define the boundaries (delimit) of the system.
- Serve as a basis for planning the technical contents of iterations.

Activities of the Requirements Discipline



Functional Requirements

- activities or processes that system **MUST** perform
- are essential for successful software development
  - meets user needs
  - aligns with business goals
- keep all stakeholders aligned and working toward the same goal.

Example: ATM functional requirements

- ATM system shall check validity of the inserted ATM card
- ATM system shall validate the PIN number entered
- ATM system shall dispense money

Non-Functional Requirements

- **Constraints** on the system
  - **P**erformance,
  - **U**sability,
  - **R**eliability, and
  - **S**ecurity
- Example: ATM non-functional requirements
  - ATM system shall be written in C++ language
  - ATM system shall communicate with bank using 256bit encryption
  - ATM system shall validate the PIN number in 5 seconds.

Information Gathering Approaches

- **Q**uestioning, **O**bserving, **R**esearching, **M**odeling
- Good questions initiate **process**
- Questions center around **the themes of** Who? What? When? Why? and How?
  - **What** are business processes?
  - **Who** carry these processes
  - **What** information is required?
  - **Why** does this process must be completed with this time frame>
  - **How** is the business process performed?

Iteration is required for Information Gathering

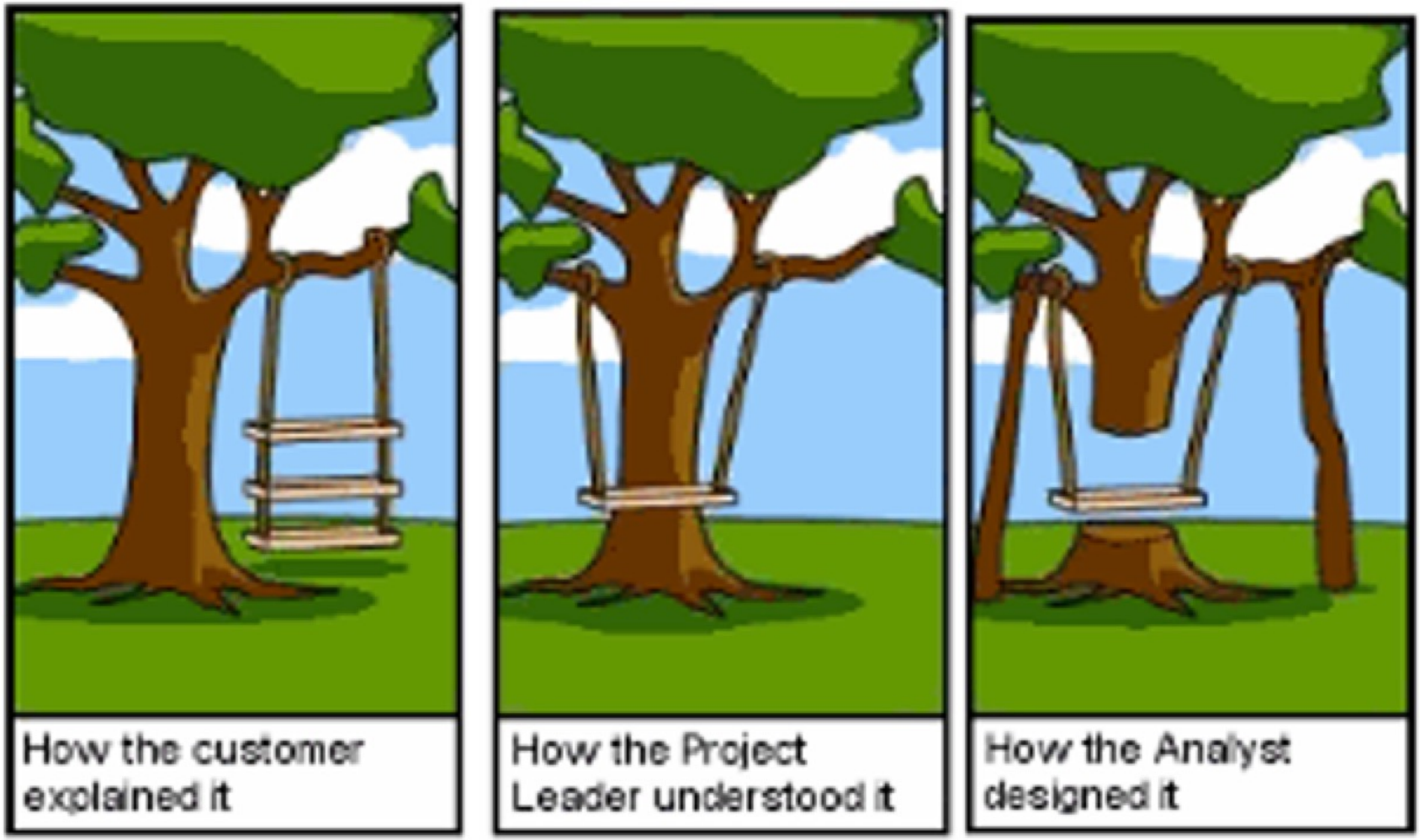
Common problems of requirements discover

- Incomplete or missing
- Conflicting between stakeholders
- Infeasible
- Overlapping
- Ambiguous

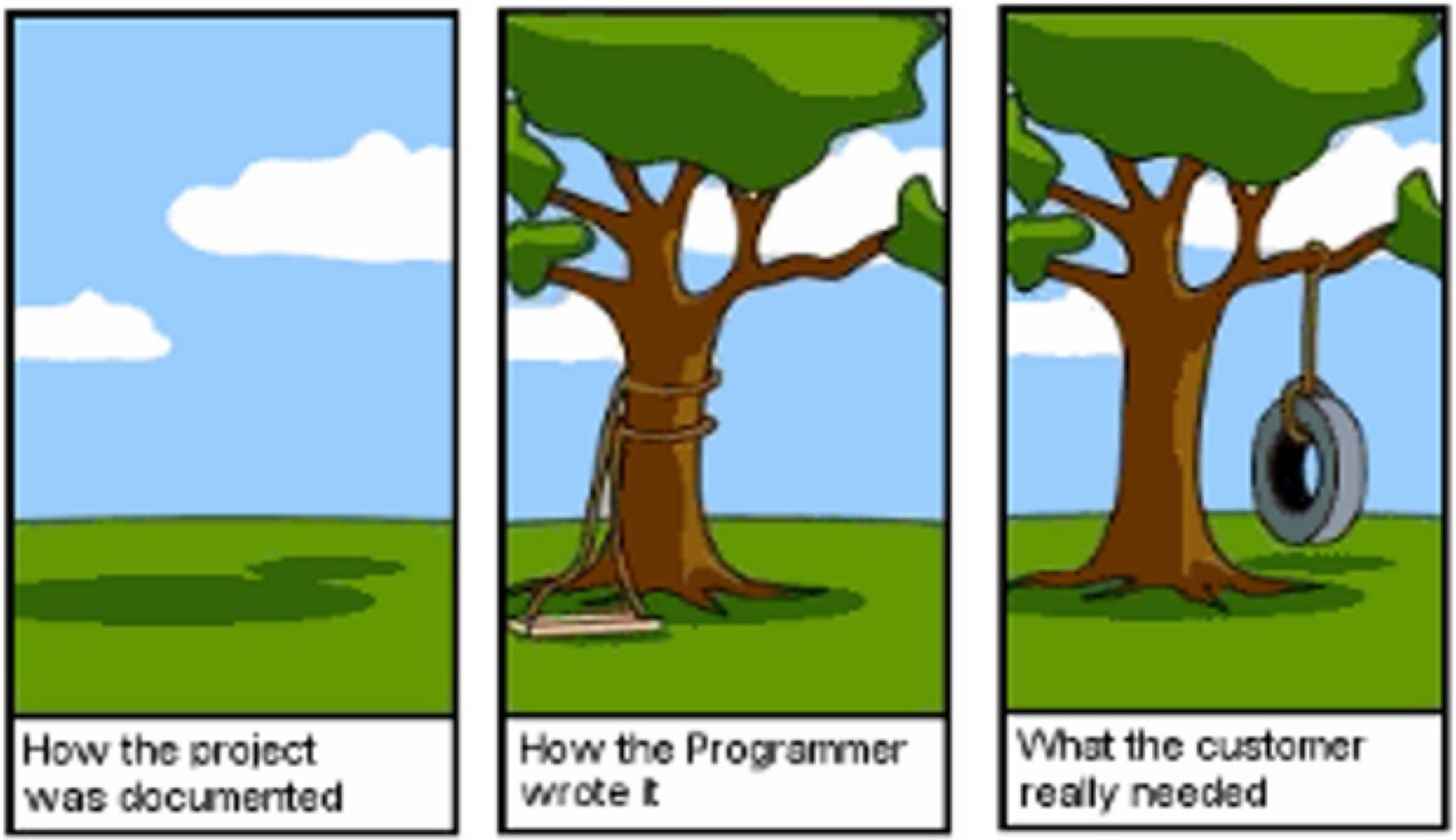
Techniques for Information Gathering

1. Conduct interviews and discussions with the users
2. Building effective prototypes
3. Distribute and collect questionnaires
4. Review reports, forms, procedure, descriptions
5. Observe business processes
6. Conduct Joint Application Design sessions (JAD)
7. Research vendor solutions

Software Engineering Problem



Software Engineering Problem



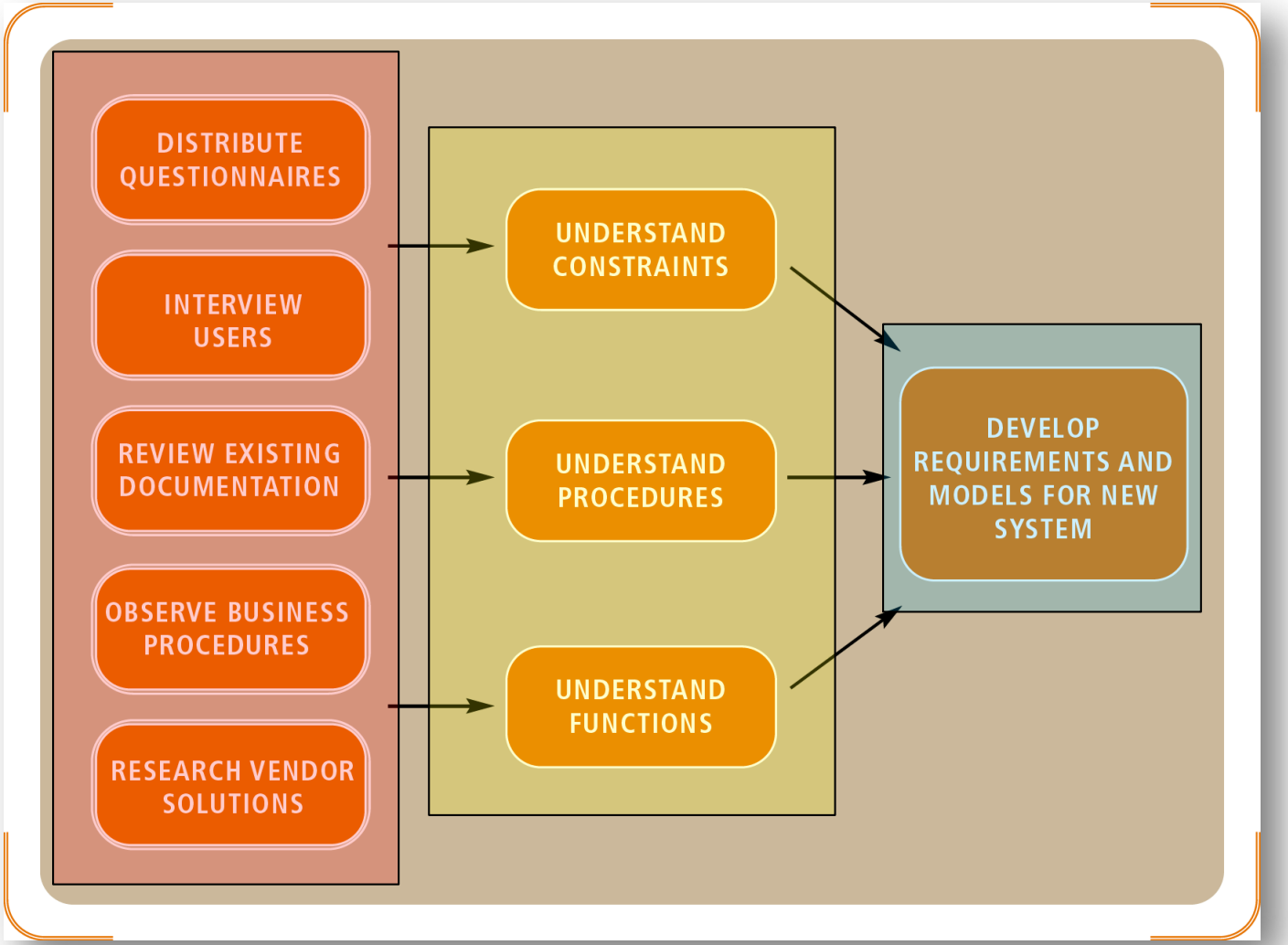
Software Models

- **What are Software Models?**
  - Representation of some aspect of the system being built
  - Encompasses the detailed information collected
- **Purpose of Software Modeling**
  - Documents the different parts of a problem and solution
  - Analyst uses different types of models to show different level of system details
  - Shows same problem in different perspective for new discovery
  - Tool for communication among stakeholders of a project

Purpose of Software Models

- **Reduce complexity** of components to essentials
- Act as **visual cues** to convey information to users and get feedback
- Allow analysts to **clarify and refine requirements** early in the development
- Provides a way to **store information** as documentation for later use and reference
- Provide platform for **effective teamwork** amongst project team members
- Promotes **informal training**

The Relationship between Information Gathering and Model Building



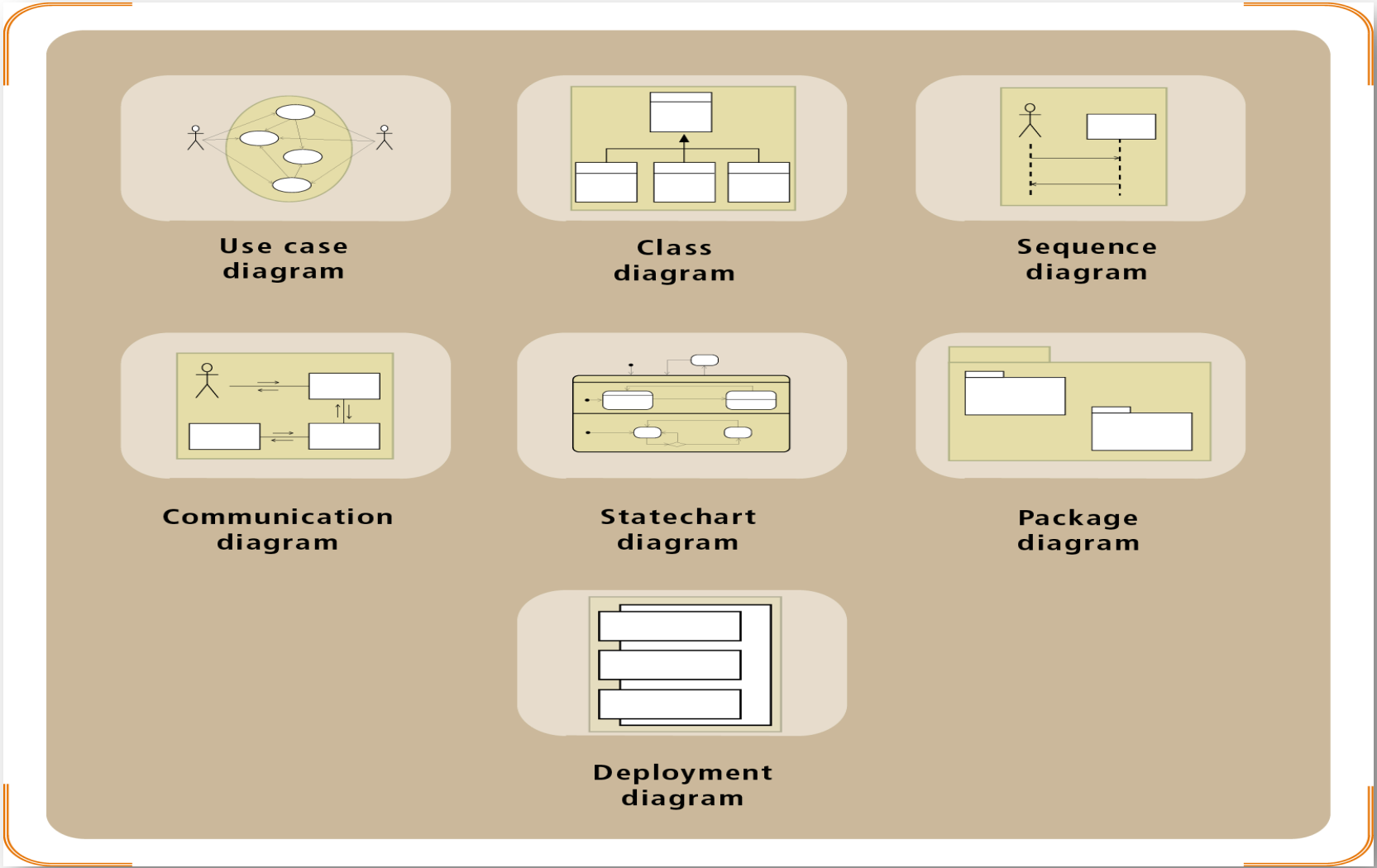
UML

- **U**nified **M**odeling **L**anguage
- Not a programming language like C++, C# or Java
- UML is a pictorial language used to make software blueprints.
- A standardized, general-purpose visual modeling “language” used in software engineering to visualize, specify, construct and document software system
- Includes a set of graphic notations to create visual models of object-oriented software systems

UML Diagrams

- Numerous diagrams
- Each diagram:
  - Contains a set of co-related graphical notations
  - Depicts different aspects of a system

UML Diagrams used for Modeling



Summary

- **Introduction to Software Engineering**
- **SDLC**: set of activities required to complete system development project
- **UP** : an iterative and incremental development life cycle
  - **iterations**
  - **disciplines**
- **UP Requirement Discipline**
  - Information gather
  - Functional and Non-functional requirements
- **Software Models**
- UML and software modelling

Cohort Class Task

- Explore how software engineering on how it can be applied beyond the realm of information technology
- What qualities do you think a good software engineer should possess? How do the qualities you description are relevant to you as an individual?
- Using your daily routine as the context, define functional and non-functional requirements from your perspective.