SE-2002
SOFTWARE DESIGN AND ARCHITECTURE

RUBAB JAFFAR
RUBAB.JAFFAR@NU.EDU.PK

Introduction

Overview and Basics

Lecture # 1, 2, 3
23, 27 Jan

TODAY'S OUTLINE

- Administrative Stuff
- Overview of SE-2002
- Basic Concepts---- Must have a grasp
- Software Design and its goal?
- Why software design is important?
- Software Architecture?
- Importance of software architecture?
- Who is responsible for design and Architecture?
- Summery of the lecture

ADMINISTRATIVE STUFF

ABOUT ME

- Graduated BS(SE) from FJWU, RWP
- Complete Masters MS(SE) from NUST, ISB
- Research interests:
  - Software engineering
  - Object Oriented Design
  - Database systems
  - Machine Learning
  - Algorithm analysis

ADMINISTRATIVE STUFF

OFFICE HOURS

- Office 6 (CS building)
- Office hours: 3:00 to 4:00 pm
  - (Monday, Thursday, Fridays)

OVERVIEW OF SE-2002
ABOUT THE COURSE

- Study application of a software engineering approach that models and designs a system as a group of interacting objects.
- Various cases studies will be used throughout the course to demonstrate the concepts learnt.
- A strong in class participation from the students will be encouraged and required during the discussion on these case studies.

OVERVIEW OF SE-2002

MAJOR TOPICS OF THE COURSE

- Course Introduction
- SDLC
- UML Relationships
  - • Association
  - • Aggregation
  - • Composition
- UML Packages
  - • Use Case
  - • Class Diagram
  - • Activity Diagram
  - • Collaboration Diagram
  - • Sequence Diagram
  - • State Transition Diagrams
- Design Patterns

OVERVIEW OF SE-2002

PRE-REQUISITES /KNOWLEDGE ASSUMED

- Programming language concepts
- Data structure concepts

OVERVIEW OF SE-2002

SKILLS ASSUMED

- We assume you have the skills to code in any programming language therefore you can design, implement, test, debug, read, understand and document the programs.

OVERVIEW OF SE-2002

TENTATIVE MARK DISTRIBUTION AND GRADING POLICY

- Assignments/Class Participation: 5 %
- Quizzes: 9%
- Mid Exams: 30 %
- Final: 50%
- Presentations + Projects: 6%
- Absolute Grading Scheme

COURSE OUTLINE

- - You may find the course outline on Google Classroom.
- - Assignments:
  - Assignment # 1 – Week 4
  - Assignment # 2 – Week 9
- - Quizzes
  - Quiz # 1 – Week 4
  - Quiz # 2 – Week 9
  - Quiz # 3 – Week 14

OVERVIEW OF SE-2002

COURSE ETHICS

Projects/Assignments

- Deadlines are always final
- No credit for late submissions
- One student per assignment at maximum

Quizzes

- Announced
- Unannounced

Honesty

- All parties involved in any kind of cheating in any assessment will get zero in that assessment.

OVERVIEW OF SE-2002

PROJECT

- An advance project
  - Provides interesting problem, realistic pressures, unclear/changing
- 3 member teams
- Emphasis on good SE practice/methodology Homework's and deliverables tied to the project

Deliverables	Deadlines
Project Proposal	(3rd week)
System Requirement Specifications (SRS)	(6th week)
Progress Report 02	(9th week)
System Design Specifications (SDS)	(11th week)
Final Report/User Manual	(13th week)
Presentations and Demo	(15th -16th week)

OVERVIEW OF SE-3002

COURSE MATERIAL

- You will have Presentations of each topic and ***reference books*** in PDF format will be available on GCR.

Text Books

1. Unified Modeling Language User Guide, The (Addison-Wesley Object Technology Series) by Grady Booch, 2nd Edition, 2017.
2. Software Modeling and Design: UML, Use Cases, Patterns, and SoftwareArchitectures, Hassan Gomaa, Cambridge University Press, 1st Edition, 2011.
3. Applying UML and Patterns 3rd Edition by Craig Larman, 2008.

Reference Books

1. Software Engineering by Ian Sommerville. Addison-Wesley Longman Publishing, 8th Edition, 2006

OVERVIEW OF SE-3002

COURSE GOALS/OBJECTIVES

- By the end of this course, you will
  - By the end of this course, you will be able to Perform analysis on a given domain and come up with a complete system and Design.
  - Practice various techniques which are commonly used in analysis and design phases in the software industry.
  - Use Unified Modeling Language (UML) as a tool to demonstrate the analysis and design ideas
  - Analyze, design and implement practical systems of up to average complexity within a team and develop a software engineering mindset

INTRODUCTION TO SOFTWARE DESIGN & ARCHITECTURE

WHY STUDY SDA?

WHAT IS A SYSTEM?

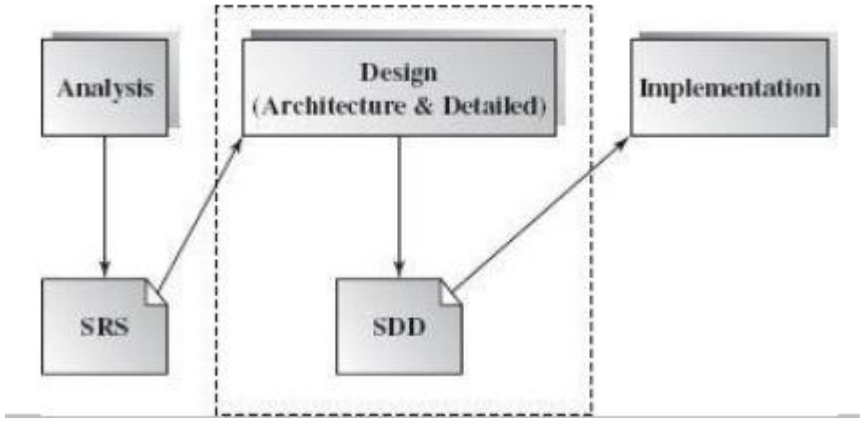
- Systems are created to solve problems
- It is an organized way of dealing with a problem
- Basically there are three major components in every system, namely input, processing and output.

SYSTEM TYPES

- A System can be a Application program or it can be an Information System.
- Computer App: is an application program (app for short) is a computer program designed to perform a group of coordinated functions, tasks, or activities for the benefit of the user.
- Information System: is software that helps you organize and analyze data. This makes it possible to answer questions and solve problems relevant to the mission of an organization.

WHAT IS SOFTWARE DESIGN?

- What is software design and its goal?
  - Identifying the objects of a system.
  - Identifying their relationships.
- Making a design, which can be converted to executables using OO languages.
- The goal of software design is to build a model that meets all customer requirements and leads to successful implementation.



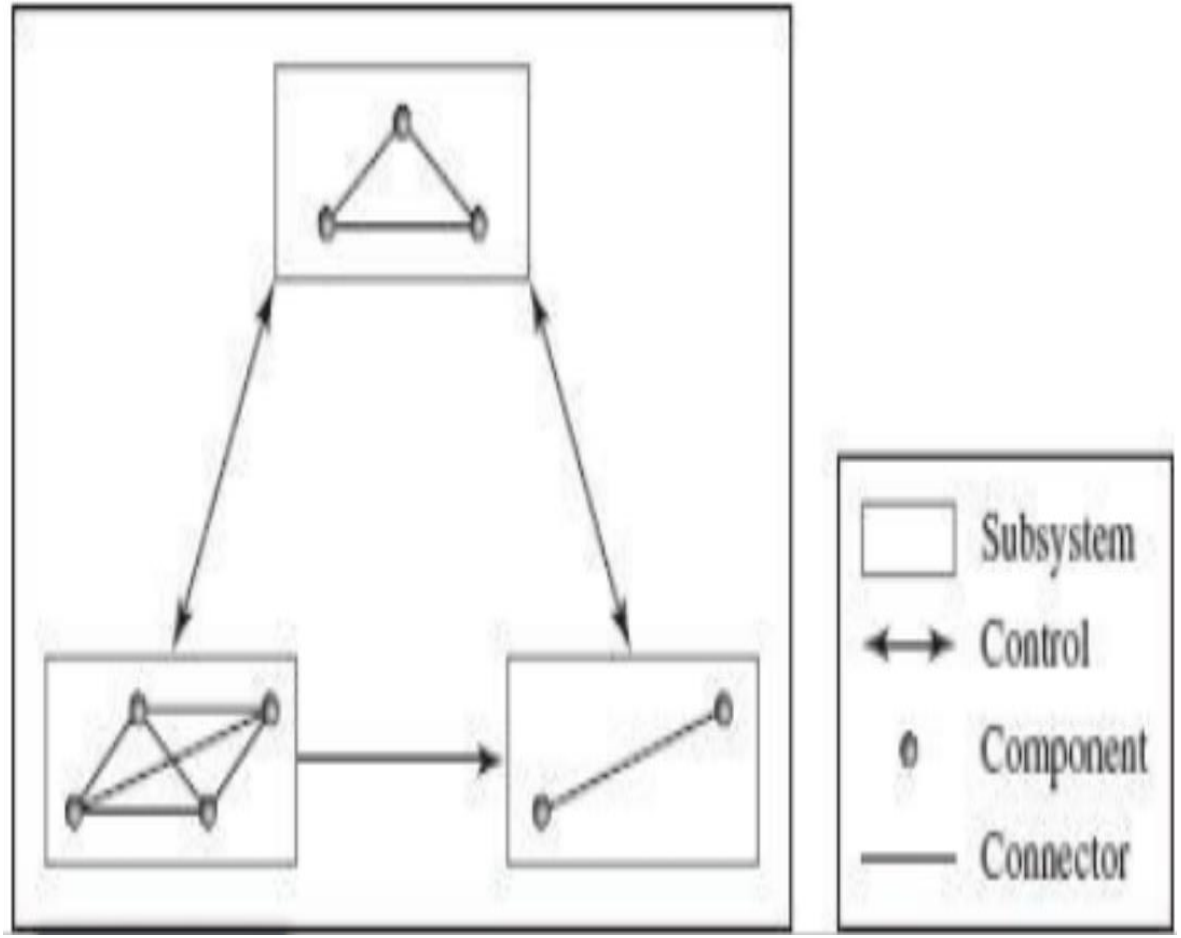
WHY SOFTWARE DESIGN IS IMPORTANT?

- As software systems continue to grow in scale, complexity, and distribution, their proper design becomes extremely important in software production. Any software, regardless of its application domain, should have an overall architecture design that guides its construction and development.

WHAT IS SOFTWARE ARCHITECTURE?

- Software architecture is **the organization of a system**.

- This organization includes all components, how they interact with each other, the environment in which they operate, and the principles used to design the software.



IMPORTANCE OF SOFTWARE ARCHITECTURE?

- The success of a software product or system largely depends on the success of its architecture design.
- What is the architecture design?
- "The architecture design defines the relationship between major structural elements of the software,
- the styles and design patterns that can be used to achieve the requirements defined for the system, and the constraints that affect the way in which architecture can be implemented" (Garlan and Shaw, 1996).

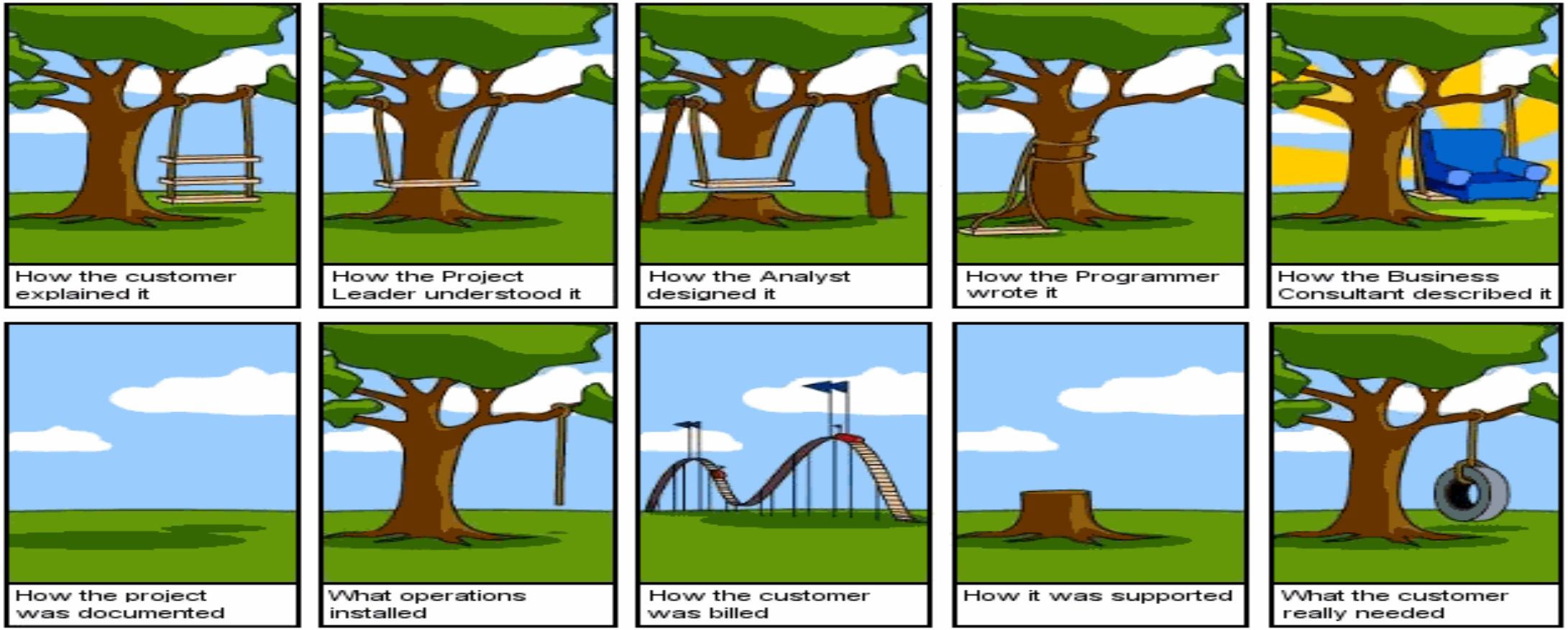
WHO IS RESPONSIBLE FOR DESIGN AND ARCHITECTURE?

- The architecture design representation is derived from the system requirement specification and the analysis model.
- Who is responsible for developing the architecture design? Software architects and designers are involved in this process. They translate (map) the software system requirements into architecture design.
- During the translation process, they apply various design strategies to divide and conquer the complexities of an application domain and resolve the software architecture.

WHY IS SOFTWARE ARCHITECTURE DESIGN SO IMPORTANT?

- There are several reasons.
- A poor design may result in a deficient product that does not meet system requirements,
- is not adaptive to future requirement changes,
- is not reusable,
- exhibits unpredictable behavior or performs badly.
- Without proper planning in the architecture design stage, software production may be very inefficient in terms of time and cost.

DEFINING THE PROBLEM IS THE PROBLEM



SOFTWARE CRISIS

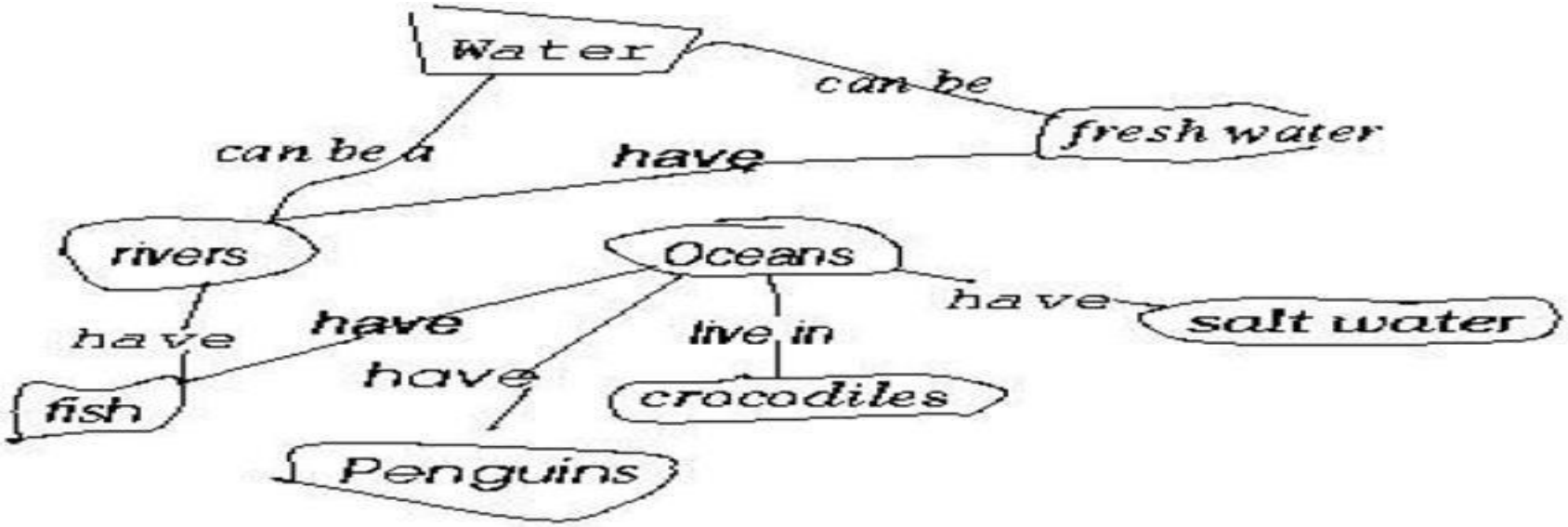
*"The "software crises" came about when people realized the major problems in software development were … caused by **communication** difficulties and the management of **complexity**" [Budd]*

What kind of language can alleviate difficulties with communication & complexity?

WHY OBJECT ORIENTED DESIGN?

Study of a first grade class [Martin & Odell] [Novak, 1984, Cambridge University Press]

When given a list of concepts (water, salt water, Oceans, Penguins,...),
Harry constructed a concept diagram through which he understands his world and communicates meaning



WHAT IS A MODEL AND WHY?

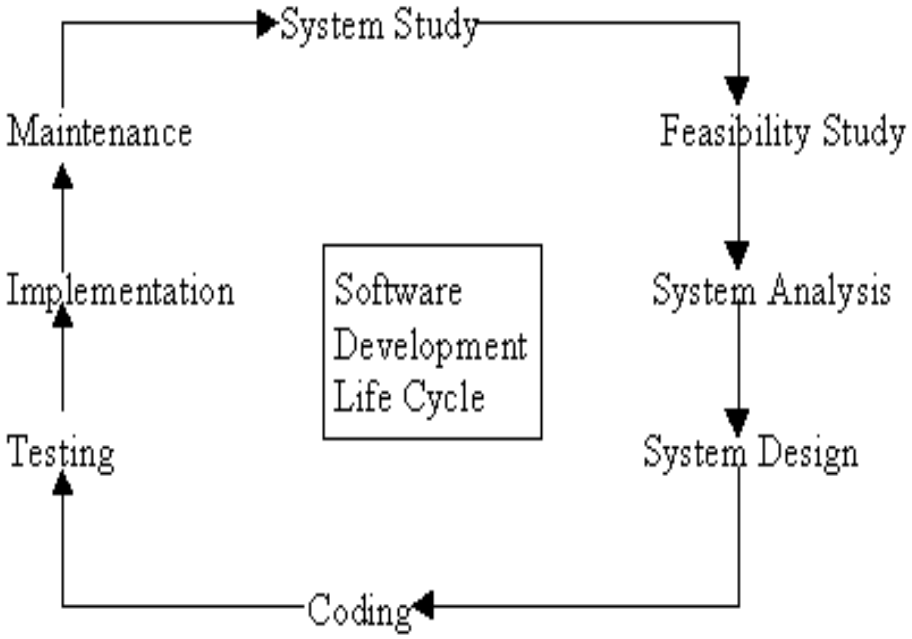
- A model is a simplification of reality.
- E.g., a miniature bridge for a real bridge to be built
- Well...sort of….but not quite
- A mental model is our simplification of our perception of reality
- A model is an abstraction of something for the purpose of understanding, be it the problem or a solution.

  - To understand why a software system is needed, what it should do, and how should do it.
  - To communicate our understanding of why, what and how.
  - To detect commonalities and differences in your perception, my perception, perception and her perception of reality.
  - To detect misunderstandings and miscommunications.

Basic OOP Concepts and Terms

SOFTWARE DEVELOPMENT LIFE CYCLE

- Software Development Life Cycle is an organizational process of developing and maintaining systems.
- It is a mixture of various activities.
- Following are the phases for SDLC:
  - System study
  - Feasibility study
  - System analysis
  - System design
  - Coding
  - Testing
  - Implementation
  - Maintenance



SYSTEM STUDY

- System study - 1st stage of system development life cycle.
- Gives clear picture of what actually the physical system is.
- **System study phases (I & II):**
  - I: initial survey of the system - helps in identifying the scope.
  - II: in-depth study - requirement identification / limitations & issues of current system.
- **Proposal:**
  - Prepared after completing the system study,
  - prepared by the System Analyst.
  - Contains the findings of the current system
  - Recommendations to overcome the limitations / issues of the current system.
- **Steps of System Study phase:**
  - problem identification and project initiation.
  - background analysis.
  - inference or findings.

FEASIBILITY STUDY

- Done on the basis of initial study.
- It is the test of the proposed system in the light of its workability, user's requirements, effective use of resources and the cost effectiveness.
- Goal : to achieve the scope.(not to solve issues)
- Advantage: Cost and benefits are estimated with greater accuracy.

SYSTEM ANALYSIS

- Analysis is detailed study of :
  - Current system, leading to specifications of a new system.
  - System operations & its relationships within & outside system.
- Data collection for: files, decision points, & transactions of present system.
- Tools of system analysis: Interviews, on-site observation & questionnaire.
- Steps to define boundary of the new system:
  - Keeping in view the problems and new requirements
  - Work out pros & cons including new system
- Analysis is documented in: detailed DFDs, data dictionary, logical data structures & miniature specifications.
- Includes sub-dividing of complex process, data store identification & manual processes.

SYSTEM DESIGN

- The new system must be designed based on the user requirements and the detailed analysis of a new system.

- This phase has two stages:
  - Preliminary or General Design
  - Structure or Detailed Design

TECHNIQUES FOR SYSTEM DESIGN

- There are several tools and techniques used for designing. These tools and techniques are:
  - Flowcharts
  - Data flow diagrams (DFDs)
  - Data dictionary
  - Structured English
  - Decision trees

MAKING DECISION TABLES

- A customer requests a cash withdrawal. One of the business rules for the ATM is that the ATM machine pays out the amount if the customer has sufficient funds in their account or if the customer has the credit granted.

Conditions	**R1**	**R2**	**R3**
Withdrawal Amount <= Balance	T	F	F
Credit granted	-	T	F
Actions			
Withdrawal granted	T	T	F

MAKING DECISION TREES

A supermarket has a loyalty scheme that is offered to all customers. Loyalty card holders enjoy the benefits of either additional discounts on all purchases or the acquisition of loyalty points, which can be converted into vouchers for the supermarket or to equivalent points in schemes run by partners. Customer without a loyalty card receive an additional discount only if they spend more than $100 on any one visit to the store, otherwise only the special offers offered to all customers apply.

CODING

- Coding the new system into computer programming language converts human instructions into a format that computer understands.
- Coding is stage where defined procedure are transformed into control specifications by the help of a computer language.
- This is also called the programming phase in which the programmer converts the program specifications into computer instructions, which we refer as programs.
- The programs coordinate the data movements and control the entire process in a system.
- Generally, programs must be modular in nature. This helps in fast development, maintenance and future change, if required.

TESTING

- Removing all the bugs, if any - Before implementing
- Test plan is developed and run on given set of test data.
- Output of test run should match expected results.
- Using test data following test runs are carried out:
  - **Unit test**: After coding / compiling programs, they are individually tested with test data. Any ambiguity must be noted & debugged.
  - **System Test**: done after unit test. Complete system is executed on actual data. Results or output of system is analyzed. If output is not matched with expected outputs, errors are identified and are fixed.

IMPLEMENTATION

- After UAT, the implementation phase begins.
- It is the stage during which theory is turned into practice.
- All programs of the system are loaded onto the user's computer.
- After loading the system, training of the users starts.
- Main topics of such type of training are:
  - How to execute the package
  - How to enter the data
  - How to process the data (processing details)
  - How to take out the reports
- After the users are trained about the computerized system, manual working has to shift from manual to computerized working.

SYSTEM RUN STRATEGIES

- **Parallel run:** Computerized & manual systems are executed in parallel. Advantages of Parallel run:
  - Manual results comparison with the computerized one.
  - Failure of the computerized system at the early stage, does not affect the working of the organization.

SYSTEM RUN STRATEGIES

- **Pilot run:** New system is installed in parts. Some part of the new system is installed first and executed successfully for considerable time period. Some advantages are:
  - When results are found satisfactory then only other parts are implemented.
  - This strategy builds the confidence and the errors are traced easily.

MAINTENANCE

- **System Review:** is necessary from time to time for:
- Knowing the full capabilities of the system
- Knowing the required changes or the additional requirements
- Studying the performance
- **Major change during the review:**
- If a major change to a system is needed, a new project may have to be set up to carry out the change.
- New project will then proceed through all above life cycle phases.

SYSTEM ENVIRONMENTS

- Development
- Test
- Staging
- Pre-Production
- Production
- Mirror

ROLES INVOLVED

- Developer
- Development Manager
- Project Manager
- Test Manager
- Configuartion Manager
- Deployment/Implementation Team

SYSTEM DEVELOPMENT STRATEGIES

- **Structured Analysis and Design:**
- "Structured analysis is a set of techniques and graphical tools that allow the analyst to develop a new kind of system specification that are easily understandable to the user. Analysts work primarily with their wits, pencil and paper." [Kendall 1996]

- It uses a set of process models to describe a system graphically.

SYSTEM DEVELOPMENT STRATEGIES

- **Philosophy of Structured Analysis and Design:**

- It focuses on processes that transform data into information, structured analysis is called a process-centered technique
- Analysts attempt to divide large, complex problems into smaller, more easily handled ones. "Divide and Conquer"
- Top-Down approach
- Functional view of the problem.
- Analysts use graphics to illustrate their ideas whenever possible

SYSTEM DEVELOPMENT STRATEGIES

- **Goals of Structured Analysis and Design :**

- Improve Quality and reduce the risk of system failure
- Establish concrete requirements specifications and complete requirements documentation
- Focuses on Reliability, Flexibility, and Maintainability of system

SYSTEM DEVELOPMENT STRATEGIES

- **Elements of Structured Analysis and Design:**
- **Essential Model:** Model of what system must do.
  - Does not define how the system will accomplish its purpose.
- **Environmental Model:** Defines the scope of the proposed system.
  - Defines the boundary and interaction between the system and the outside world.
  - Composed of: Statement of Purpose, Context Diagram, and Event List.

- **Elements of Structured Analysis and Design:**
- **Behavioral Model:**
  - Model of the internal behavior and data entities of the system.
  - Models the functional requirements.
  - Composed of Data Dictionary, Data Flow Diagram, Entity Relationship Diagram, Process Specification, and State Transition Diagram

- **Elements of Structured Analysis and Design:**
- **Implementation Model:**
  - Maps the functional requirements to the hardware and software.
  - Determines which functions should be manual and which should be automated.
  - Defines the Human-Computer Interface.
  - Defines non-functional requirements.
  - Tool: Structure Charts

SYSTEM DEVELOPMENT STRATEGIES

- **Advantages of Structured Analysis and Design:**
  - Visual, so it is easier for users/programmers to understand
  - Makes good use of graphical tools
  - A mature technique
  - Process-oriented approach is a natural way of thinking
  - Flexible
  - Simple and easy to understand and implement

SYSTEM DEVELOPMENT STRATEGIES

- **Object-Oriented Analysis and Design:**

  - Used to thoroughly represent complex relationships, as well as represent data and data processing with a consistent notation

  - It blends analysis and design in evolutionary process

  - It allows you to deal with the complexity inherent in a real-world problem by focusing on the essential and interesting features of an application

SYSTEM DEVELOPMENT STRATEGIES

Object-Oriented Analysis and Design:

- Process of progressively developing representation of a system component (or object) through the phases of analysis, design and implementation
- The model is abstract in the early stages
- As the model evolves, it becomes more and more detailed

SYSTEM DEVELOPMENT STRATEGIES

Object-Oriented SDLC:

- The Object-Oriented development life cycle consists of progressively developing an object representation through three phases:
  - Analysis
  - Design
  - Implementation

SYSTEM DEVELOPMENT STRATEGIES

Object-Oriented Analysis and Design:

Analysis Phase

- Object-oriented analysis is a popular approach that sees a system from the viewpoint of the objects themselves as they function and interact
- Model of the real-world application is developed showing its important properties
- Model specifies the functional behavior of the system independent of implementation details

SYSTEM DEVELOPMENT STRATEGIES

Object-Oriented Analysis and Design:

Design Phase

- Analysis model is refined and adapted to the environment
- Can be separated into two stages
  - System design
    - Concerned with overall system architecture
  - Object design
    - Implementation details are added to system design

SYSTEM DEVELOPMENT STRATEGIES

Object-Oriented Analysis and Design:

Implementation Phase

- Design is implemented using a programming language or database management system

SYSTEM DEVELOPMENT STRATEGIES

- **Structured Analysis and Design**
  vs.
  Object-Oriented Analysis and Design

- Both SAD and OOAD had started off from programming techniques
- Both techniques use graphical design and graphical tools to analyze and model the requirements.
- Both techniques provide a systematic step-by-step process for developers
- Both techniques focus on documentation of the requirements

SYSTEM DEVELOPMENT STRATEGIES

- **Structured Analysis and Design**
  vs.
  Object-Oriented Analysis and Design

- SAD is Process-Oriented
- OOAD combines data and the processes
- OOAD encapsulates as much of the systems' data and processes into objects, while SAD separates between them

That is all