# STRUCTURE OF COMPLEX SYSTEMS, UNIFIED PROCESS

Dr. Dharmendra Singh Rajput
Associate Professor

## STRUCTURE OF COMPLEX SYSTEMS

**Example of complex systems:**

- **The structure of a personal computer.**
  - **A personal computer is a device of moderate complexity**
- **The structure of plant and animals**
  - **Plants are complex multicellular organism it consists of various plant organ system arises, such as complex behaviour as photosynthesis and transpiration.**

## Contd…

- **Animals are multicellular, that exhibit a hierarchical structure similar to that of plants, collection of cells form tissues, tissues work together as organs , cluster of organs define systems and so on.**

## The five attributes of Complex Systems

- Frequently complexity takes the form of hierarchy, whereby a complex system is composed of interrelated subsystem that have in turn their own subsystem and so on, until some lowest level of elementary components is received.
- The choice of what components in a system are primitive is relatively arbitrary and is largely up to the discretion of the observer system.
- Intracomponent linkages are generally stronger than
- intercomponent linkages .

## Contd…

- Hierarchy systems are usually composed of only a few different kinds of sybsystems in various combinatons and arrangememnts.
- The complex system that works is invariably found to have evolved from a simple system that worked. A complex system designed from a scratch never works and cannot be patched.

## Decomposing complexity :

- When designing a complex software system it is essential to decompose it into smaller and smaller parts .
- Each of these smaller parts may then refine independently.

## Algorithmic Decomposition

- It is a top down structural design
- We approach decomposition as a simple matter of algorithmic decomposition.
- Each module denotes major step.

## Why is software inherently complex?

- The complexity of software is an essential property not an accidental one
- It consists of four elements
  - Problem domain
  - Difficulty of managing the development process
  - Flexibility possible through software
  - Problems of characterizing the behaviour of discrete systems.

- **Complexity of problem domain**
  - **It involves inescapable complexity.**
  - **It involves contradictory requirement**
  - **For e.g system of multiengine aircraft, cellular phone switching system or autonomous robot. The functionality of such system is difficult.**
- **Difficult of managing the development process**
  - **The fundamental task of software development team is engineer the illusion of simplicity**

## Object oriented decomposition

- Decomposing the system according to the key abstractions in the problem domain rather than decomposing the problem into steps such as Get formatted update and add checksum.

- Objects are identified such as master file and check sum which derive directly from the vocabulary of the problem domain.

## Advantages of Object oriented decomposition

- Reduces risk
- Resilient to change
- Reduces the risk of building complex software system

## The Meaning of design

- **The purpose of design is to create a clean and relatively internal structure, sometimes also called an Architecture.**

## Elements of analysis and design

**System involve incremental and iterative process**

- **Notation: The language for expressing each model**
- **Process: The activities leading to the orderly construction of the system model**
- **Tools: The artifacts that eliminate the tediousness of model building and enforcing rules about the models themselves, so that errors and inconsistencies can be expressed.**

## Typing

- It is a precise characterisation of structural or behavioural properties which a collection of all entities share
- Without type checking the program in most languages can crash at runtime.
- In most systems the edit-compile-debug cycle is so tedious that early error detection is indispensable.
- Type declaration helps to document the program.
- Most compilers can generate more efficient object code if types are declared
    - e.g of typing:
    - static and dynamic binding:

- Static binding
  - are early binding which means all variables and expressions are fixed.
- Dynamic binding
  - means the type of all variables and expressions are not known until runtime.

## Benefits of object model

- The main advantage of object oriented system is that you can build on what you already have.
- Reuse code and develop more maintainable systems in a shorter amount of time.
- The use of the object model produces systems that are build upon stable Intermediate form, which are more resilent to change.

## OBJECT ORIENTED ANALYSIS AND DESIGN

- To model the world is done by discovering the classes and objects that form the vocabulary of the problem domain.
- Classical approaches: The number of methodologies have proposed various sources of classes and objects.
- They derive primarily from the principles of classical categorization.
  - Tangible things- Cars, pressure sensors
  - Roles- mother, teacher, politician
  - Events- landing, interrupt, request
  - Interaction- loan, meeting, intersection

## Behaviour analysis:

- These approaches are more related to conceptual clustering
  - We form a class based upon groups that have similar behaviour.
  - This approach deals with analysing what takes place in the system these are the system behaviours.
  - System behaviour is closely related to function point.
  - A function point is a outwardly visible and testable behaviour of the system
  - It can include output, inquiry, input, file or interfaces.

## Domain analysis:

- It is an attempt to identify the objects, operations and
- relationship needed
- Use Case analysis, CRC cards, formal English description.

## Structured analysis

- **It is a front end to object oriented design**
- **We start with the essential model of the system**
- **It is described by DFD diagram**
- **It consists of external entities, data stores, control stores, control transformation**
- **Candidate class derive from two sources data flows and control flows.**

## Object:

- It represents entity the entities are as follows:
  - Physical entity
  - Conceptual entity
  - Software entity
- An object ids an entity with well defined boundary and identify that encapsulate state and behaviour state is represented by attribute relationship .
- Behaviour is represented by operations, methods and state machine.
- An object is a real world element each object has
  - Identity – that distinguishes from other objects in the system
  - State – It determines characteristic properties of an object.
  - Behaviour – that represents externally visible activities

## Object oriented programming

**Object oriented programming is a method of implementation in which programs are organized as a co-operative collection of objects, each of which represent instance of some class and whose classes are all members.**

## Unified Process:

- **The unified process is also called unified software development**
- ***it is the creation of pioneers of the OOAD i.e. BOOCH, JACOSON,RAMBAUGH**
- ***They merged together the best of their individual.**
- **The unified process has two basic characteristics**
  - **It is component based**
  - **The components are interconnected via interfaces**
  - **Unified process extensively uses UML terminology and notations**

## USE CASE DRIVEN:

- **It is a set of actions that the users of the system perform in order to get desired work done.**
  - **A use case represents functional requirements .**
  - **A use case model is the diagrammatic representation of the overall system requirements.**
  - **The term use case driven indicates that the unified process starts with the definition of use cases, then they are transformed into design and code, and finally validation during the testing phase**

## UNIFIED PROCESS LIFE CYCLE

- **Inception:**
  - **It is the first phase, it seeds idea for the development for entering into Elaboration phase.**
- **Elaboration:**
  - **it is the second phase of the process. The system requirement range from general vision statement, precise** evaluation criteria, each specifying functional and non- functional behaviour and basis for testing.
- **Construction**
  - **It is the third phase,** it is a software brought from an executable architectural base line ready to be transitioned to the user community, here it is constantly re-examined against the business needs of the project.
- **Transition**
  - **It is the fourth phase,** here the software is given in the hands of the user community, rarely the software process end here, during this phase the system is continuously improved , bugs are eradicated, some features which did not exist earlier are added.