# SOFTWARE PROTOTYPING

# Prototyping

- Its an early sample, model of the product built to test a process to act as a thing to be learned from.

# Introduction

- what is software prototyping ?

  It is the process of implementing the presumed software requirements with an intention to learn more about the actual requirements or alternative design that satisfies the actual set of requirements .

- Need for software prototyping

  -To assess the set of requirements that makes a product successful in the market

  -To test the feasibility without building the whole system.

  -To make end-user involved in the design phase

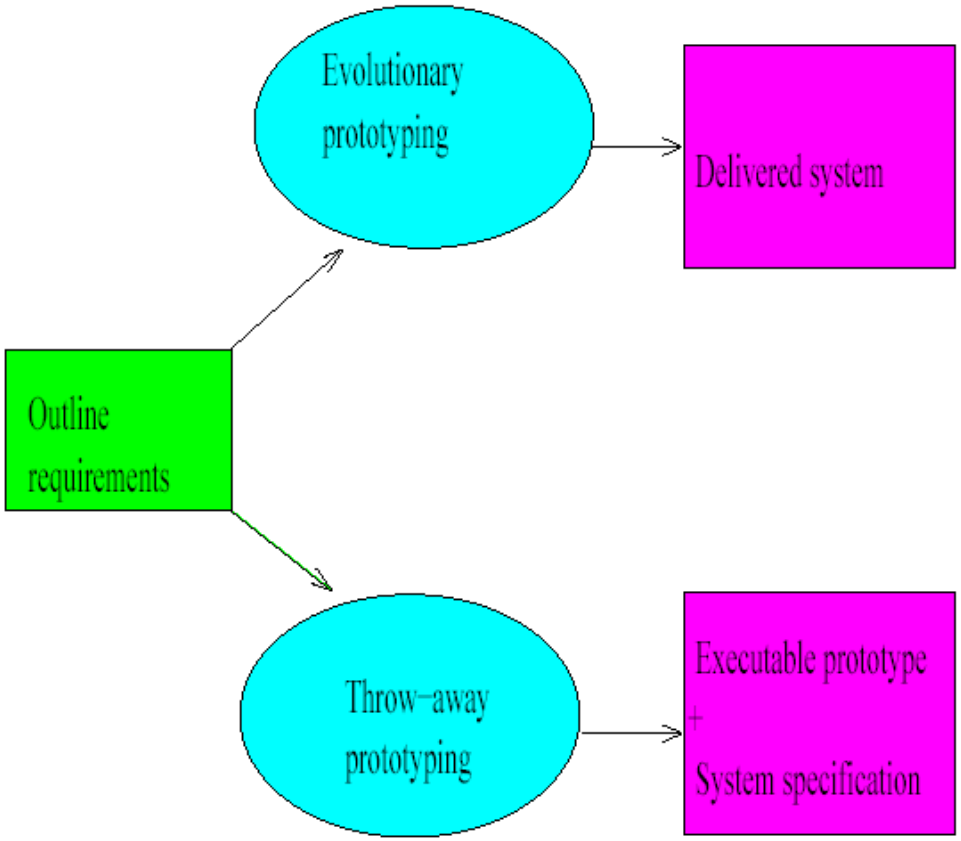
# Benefits of Software Prototyping

- It makes the developers clear about the missing requirements. Lets the developers know what actually the users want.

- Reduces the loss by bringing the manufacturer to a conclusion weather the system which we are about to build is feasible or not rather than building the whole system and finding it .

- One can have a working system in before hand.
- It brings the user to get involved in the system design

# Types of Prototyping



# Types of Prototyping

- Throw away prototyping
- Evolutionary prototyping

.

- Operational prototyping

# Throw away prototyping

- Objective - Derive end system requirements
- **Throw away prototyping** is one type of approach where an initial prototype is built mainly focusing on the poorly understood requirements
- Once the requirements are understood requirements document is updated and a conventional development process is followed to build system

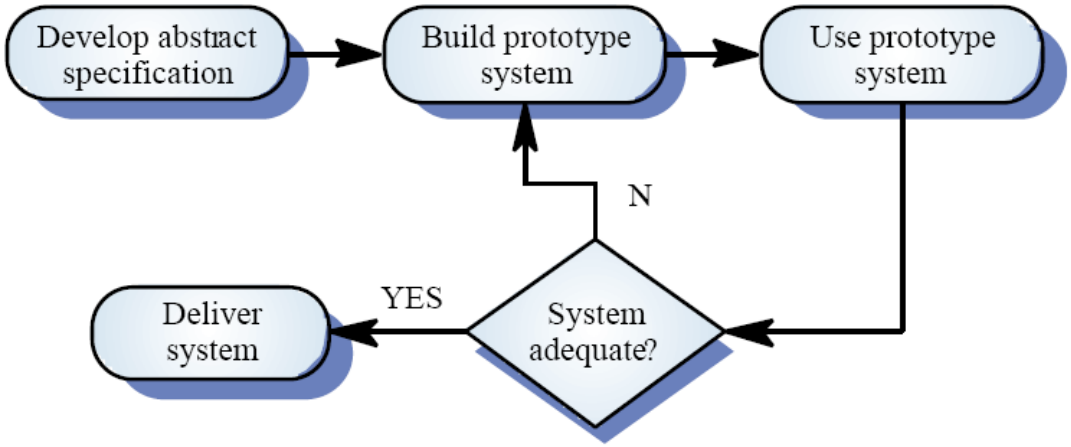
# Throw-away prototyping

- Used to reduce requirements risk
- The prototype is developed from an initial specification, delivered for experiment then discarded
- The throw-away prototype should NOT be considered as a final system
  - Some system characteristics may have been left out
  - There is no specification for long-term maintenance
  - The system will be poorly structured and difficult to maintain

# Evolutionary Prototyping

- **Objective – Deliver a working system + requirements**

- **Evolutionary prototyping** is the one in which a system is build using the well understood requirements.



# Evolutionary prototyping

- Specification, design and implementation are inter-twined
- The system is developed as a series of increments that are delivered to the customer
- Techniques for rapid system development are used such as CASE tools and 4GLs
- User interfaces are usually developed using a GUI development toolkit

# Evolutionary Prototyping

- Advantages –

  Quickly Delivered

  Makes User Commit

  Look like feel

- Disadvantages –

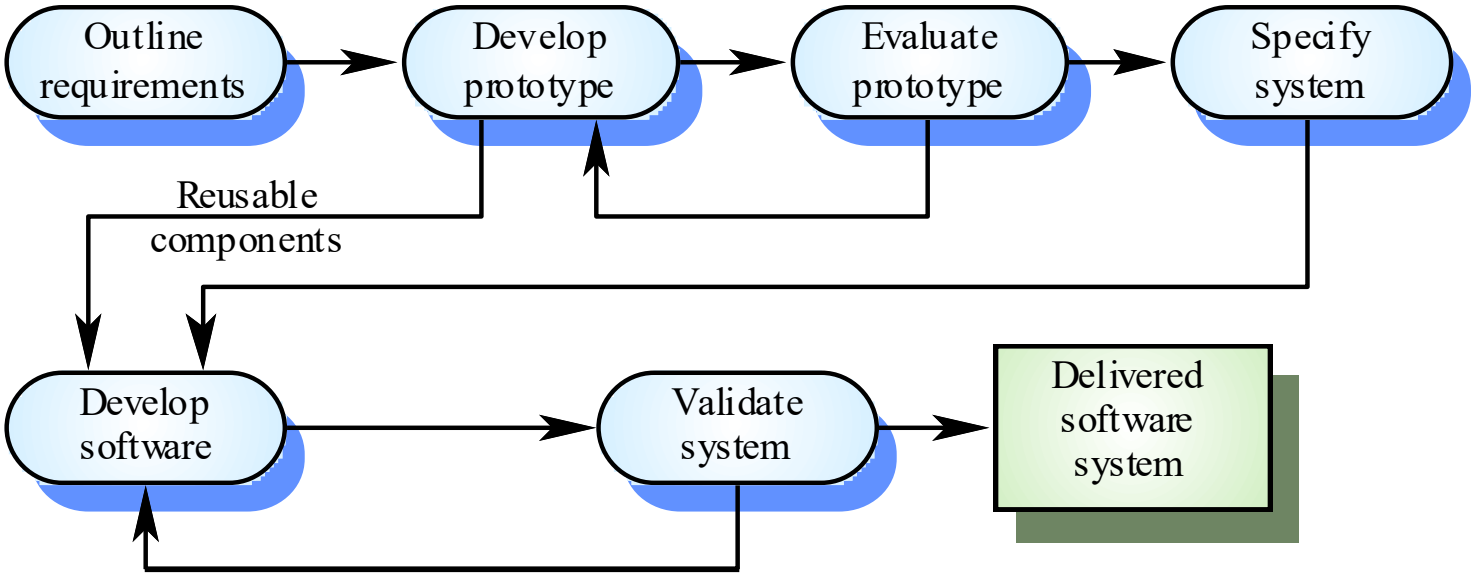
  Availability of specialist skills

  Maintenance over long term

# Operational Prototyping

- Used  when requirements are either critical and understood or not critical and poorly understood .
- Throw away prototypes are selectively built on top of evolutionary prototype
- A trained  prototyper  keeps track of user .

# Throw-away prototyping



Outline the requirements all should be mentioned

The Design according to the Requirements and develop the prototype

Evaluate means user experiences and then specify the new requirements.

REPEAT the above step if necessary

Then using system specification and developed prototype, develop the software

Then validate the system and check whether any error is coming or not. If yes then again develop the software.

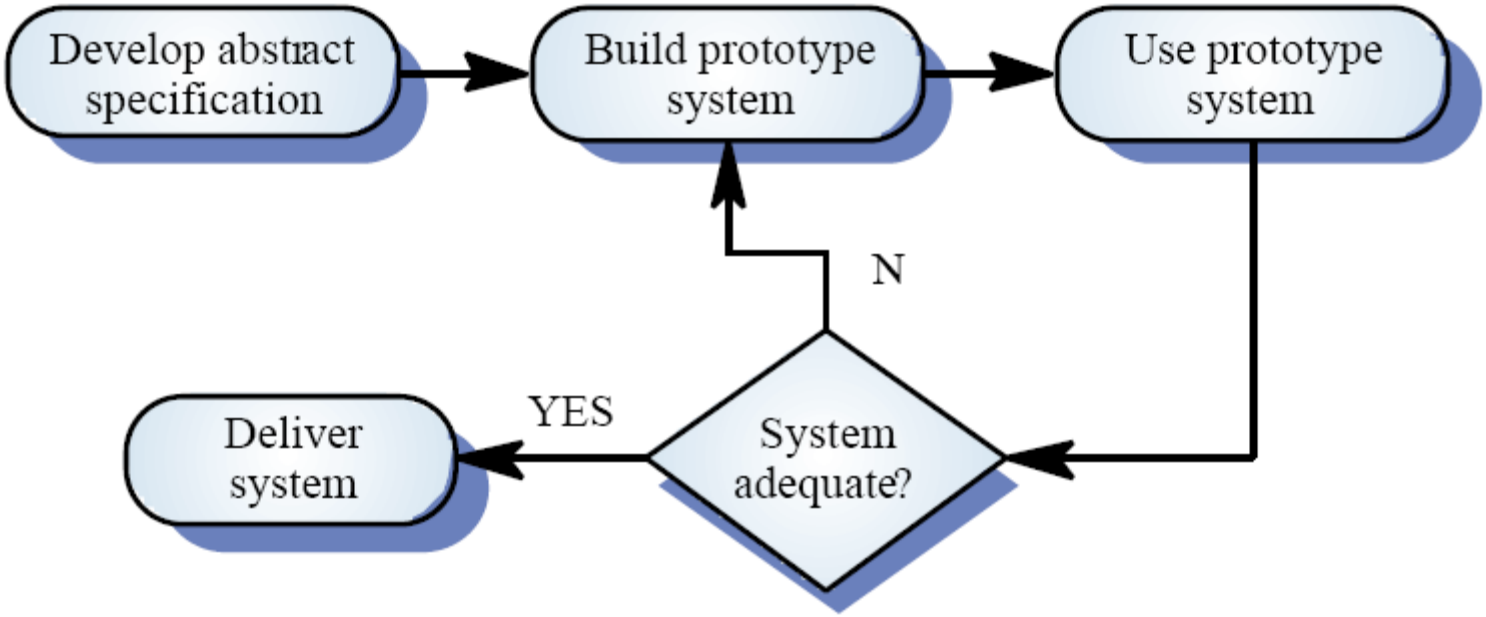
# Advantage

- Used to check the design of the system
- Can be developed very quickly
- Used for hardware system which helps in cost reduction.

# Disadvantage

- This prototype fails to reflect the FUNCTION of the system
- Fails to clarify interface and the relationship
- The training time during prototype evaluation may not be enough
- Important features may be left out of the prototype.

# Evolutionary Prototyping



# Objective – Deliver a working system + requirements

- Giving the user a system which is INCOMPLETE and then modifying and augmenting it as the user requirements becomes CLEAR
- This start with the limited understanding of the system requirements and changed as NEW REQUIREMENTS are discovered

- Develop as abstraction
- Build the system
- By using prototype system
- And a system adequate gives the delivered system
- Or rebuilt again the prototype system

# Advantages

- Rapid delivery of the system and deployment are sometime more important than long them maintainability

- System should not only meet the requirements of the system, but also commit to use the system.

## Disadvantages:

Management problems---- Specialist skills not available

Maintenances problems---- continuous changes may corrupt the sys structure

So long term maintenances is expensive

# PROTOTYPING TECHNIQUES

- Development techniques used to develop prototype RAPIDLY are called RAPID PROTOTYPING techniques.

- It concentrates only on speed of delivery

- It does not rely on other characteristics such as performance, maintenance or reliability

# Three Rapid Development techniques used are:

1) Dynamic high level language development

2) Data base programming

3) Component and application assembly

# Dynamic high level language development

- Dynamic HLL are programming language that simplify program development since they reduce problems of storage allocation and management

| Language | type | Application domain |
|---|---|---|
| Small talk | Object oriented | Interactive system |
| Java | Object oriented | Interactive system |
| Prolog | Logic | Symbolic processing |

# Characteristics of DHLL

- They are languages which include powerful data management
- They need a large run time support system, normally not used for large system development
- Some languages have IDE ( integrated development) whose facilities may be used in the prototype

- CHOICE OF PROTOTYPE LANGUAGE:

DHLL can be selected for prototyping based on:-

1) Application domain
2) Type of user interface
3) Environment support

# DATABASE PROGRAMMING

- It creates data and maintains the records in a specified fashion
- It manipulates the data from database and produce outputs in organized and formatted fashion

- The database programming language along with the environment is known as **FOURTH-GENERATION language (4GL)**

**1) standard forms for input and output**

**2) Abstract information's from a database and present it to end user on their screen**

**3) Update the database with changes made by the user**

**4) User interface containing a set of standard forms or a spread sheet**

# Tools and Techniques

- Low level tools
- High Level languages
- Fourth Generation Languages (4GL)
- Visual programming .

# Troubles of Software Prototyping

- Developers may loose the focus on real purpose of prototype and comprise with the quality of system .
- New born ideas will be plundered at the initial stages
- Prototyping will not reveal the non functional requirements like robustness, safety etc .

# CHAPTER 5
# SOFTWARE DESIGN

- What is software Design ?

Software Design is what virtually every engineer wants to do, it's the place where creativity rules, Customer requirements, Business needs and Technical considerations come together in the development of a system.

It's the bridge between requirements specification and the final solution for satisfying the requirements

# Why software Design

- It produce various models that form a kind of blueprint of the solution to be implemented.
- Design allows a s/w engineer to model the system that is to be developed.
- These models can be used for improving the quality of the system before code is generated and tests are conducted.
- We can analyse and evaluate these models to determine whether or not they will allow us to fulfil the various requirements.

**Design Process**: is a sequence of steps that enable the designer to describe all aspects of the software to be built.

## Design Process for s/w system (2 levels) are

1) Deciding which model/module are needed for the system.

2) Internal Design of the model is decided upon. This is called detailed design or Logic Design

**Design Model :** its like architect's plan for a house. The design model represents the completeness of the things to be built.

# Design Objectives:

- The goal of the design process is to find the best possible design, within the limitation imposed by the requirements.

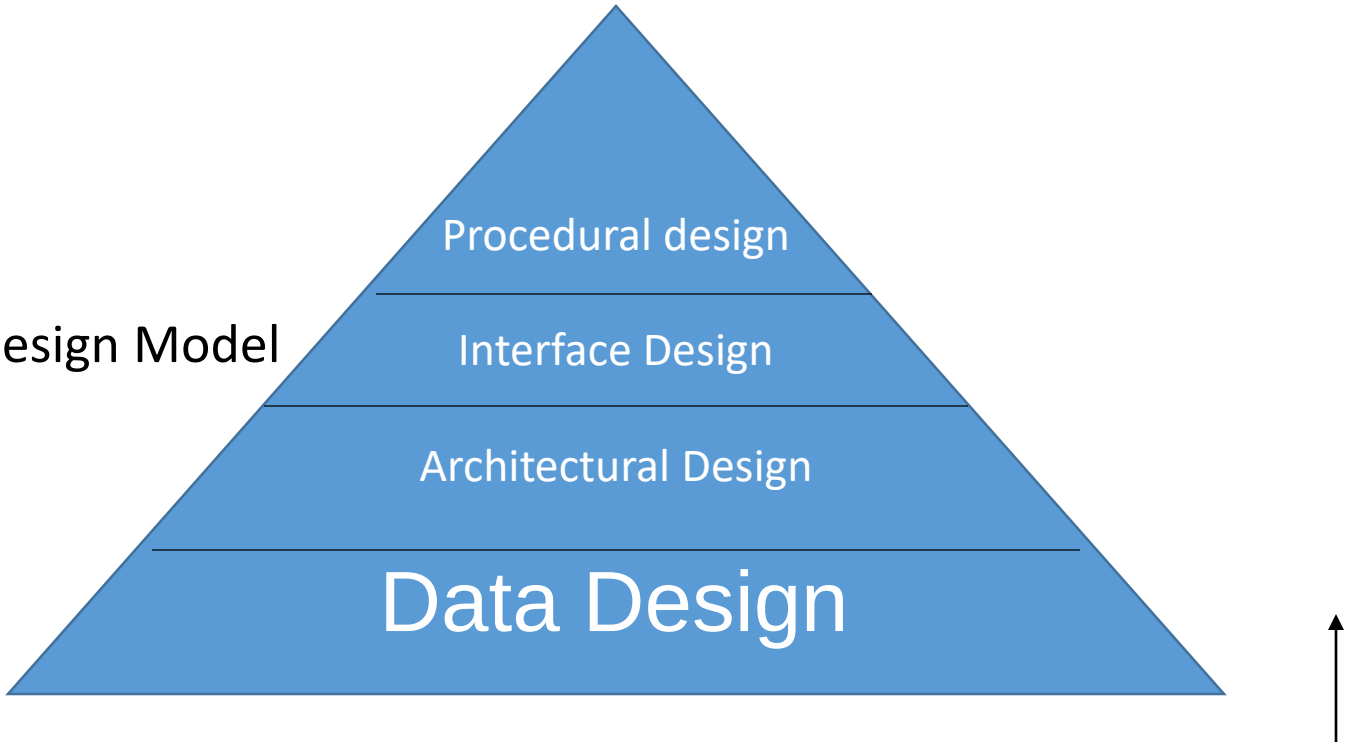
- Properties are:

1) Correctness
2) Verifiability
3) Completeness
4) Consistency
5) Efficiency
6) Traceability
7) Uderstandability
8) Maintainability

# Design Architecture

- It refers to the overall structure of the s/w and the ways in which that structure provides conceptual integrity for a system.

- Design Model



# Design Principals

- The design process should not suffer from tunnel vision
- The design should be traceable to analysis model
- The design should not re-invent the components
- The design should minimize the intellectual distance
- The design should exhibit uniformity and distance

- The design should be structured to accommodate changes
- The design should be structured to degrade gently, even when data events or operating conditions are encountered.
- Design is not coding , coding is not design
- The design should be assessed for quality as it is being created not after
- The design should be reviewed to minimize semantic errors.

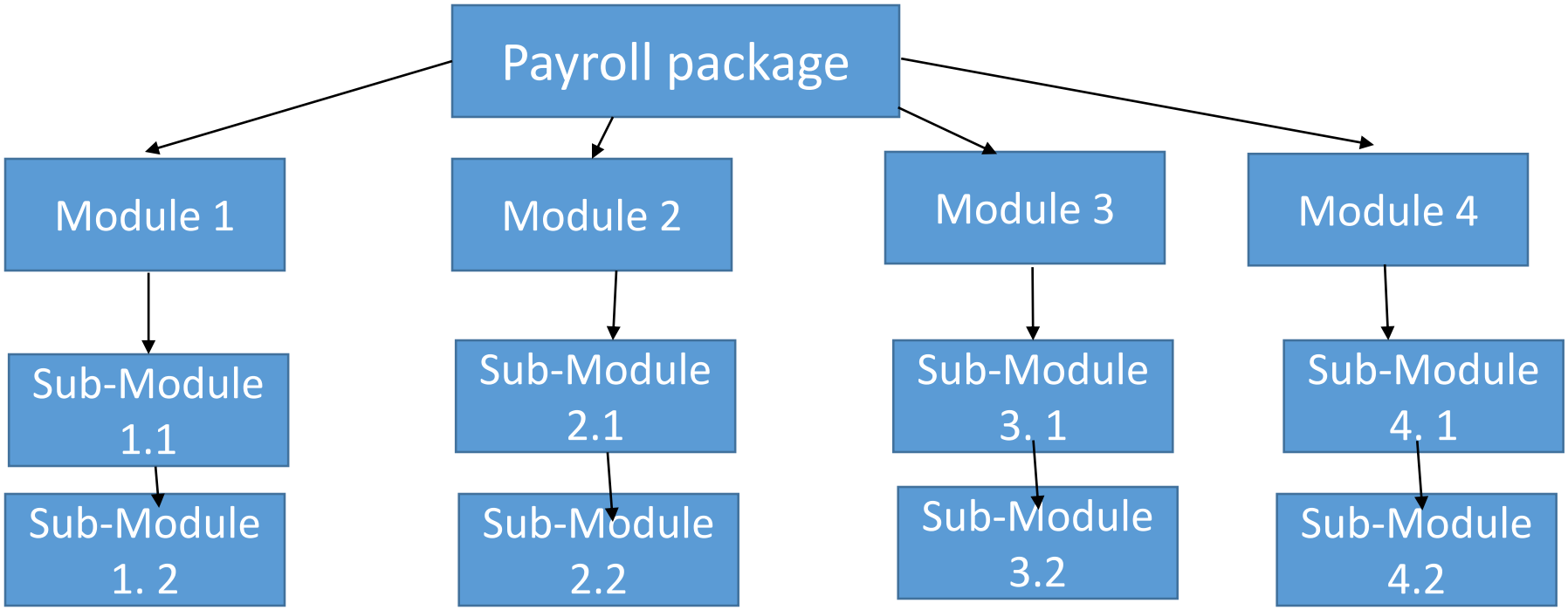
# DESIGN TECHNIQUES

1) Problem partitioning
2) Abstraction
3) Top – Down and Bottom – up
4) Modularity

# PROBLEM PARTITIONING

- Problem partitioning is s method of adopting the principle od divide and conquer to get the solution to the problem.

- Complex problems can be divided into small sub-program.
- Each program cab be handled independently.
- It can be integrated to form a whole executable program.



# ABSTRACTION

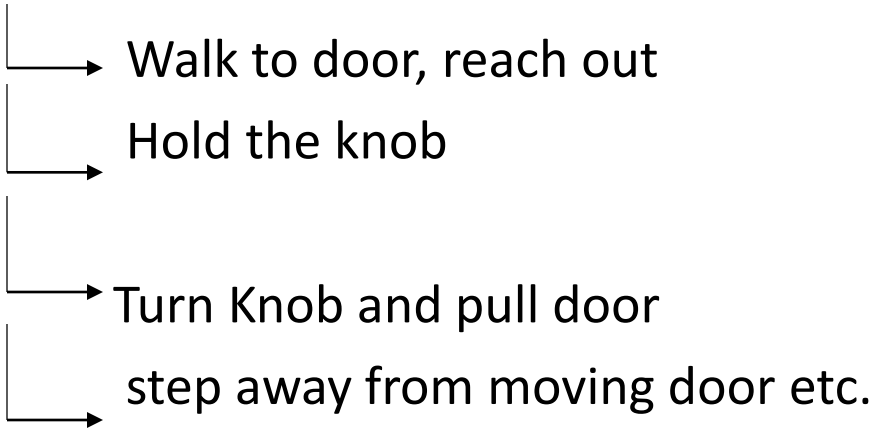
- Abstraction is the method of describing a program function.
- High level of abstraction states the solution to the problem
- Low level of abstraction deals with procedural details.

- DIFFERENT LEVELS OF ABSTRACTION ARE:

1) Data abstraction
2) Procedural Abstraction
3) Control Abstraction

# DATA ABSTRACTION

- It refers to the collection of data describes
- Data types, objects, operations on the objects by suppressing representation and manipulation details.
- Eg: open the door



- Procedural Abstraction:-
    * It reflects functional Data Base.
    * it's a sequence of instructions that has specification and relates to a particular function, internals of the object are hidden

Eg: Starting the car ------ steering, engine
------- gear, driver ( all others are hidden)

- Control Abstraction: -
    * It controls the program without specifying the internal details.

Eg: Apply car break.

# MODULARITY

- Software is divided into components called Modules that are integrated to satisfy problem requirements.
- Modules contains instructions, processing logic and data structures
- Modules cab be separately compiled and stored in a library and included in a program.
- Modules can be invoked by Name and some parameters

MODULARITY HAS 5 CRITERIA ARE:-

1) Modular Decomposability --- decomposing a problem into sub-problem
2) Modular Composability---- it's a design method, existing design to new sys
3) Modular Understandability---its understood as single unit without referring to other modules, easier to build a module and make changes easily.
4) Modular Continuity----small change to the sys requirements result in changes
5) Modular protection ---- Error will be minimized

# TOP- DOWN AND BOTTOM – UP STRATEGIES

Top-Down Design approach starts by identifying the major components of the system.

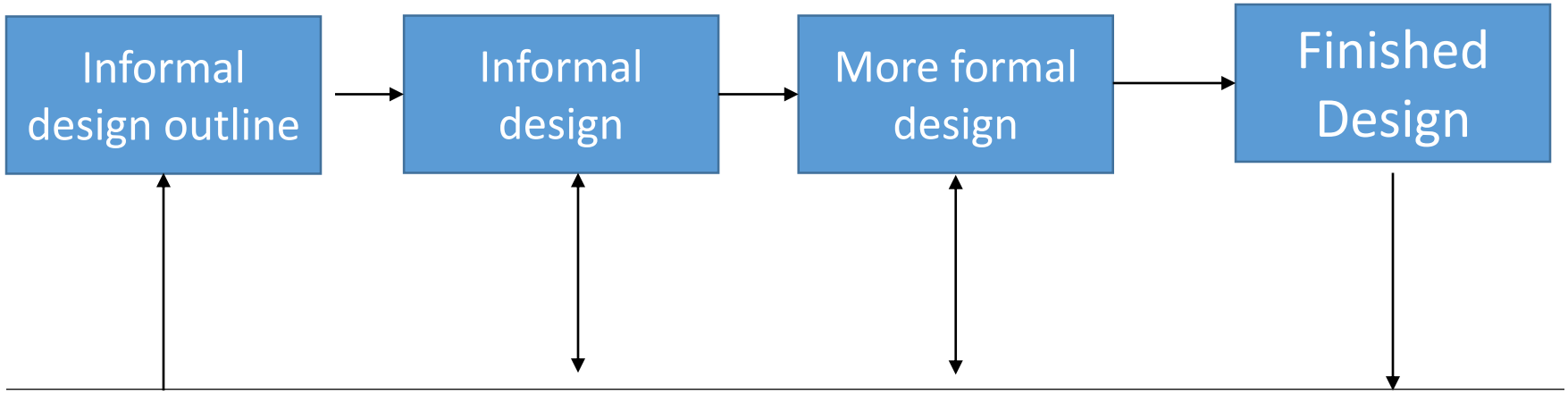
Decomposing them into Lower-level components.

Bottom- up design approach starts with designing the most basic components and proceeds to the highest level component.

Bottom approach can be used to improve system features and functions.

**Design process** is an activity of converting the system specification into executable design

Progression from an informal to a detailed design




- The system should be described at several different levels of abstraction
- Design takes place overlapping. For that we have the general model

# Design process involves 6 main design activities are:-

1) Architectural Design
   - system structuring
   - control modelling
   - modular decomposition
2) Abstract Specification
3) Interface Design
4) Component Design
5) Data structure Design
6) Algorithm Design

# DESIGN STRATEGIES

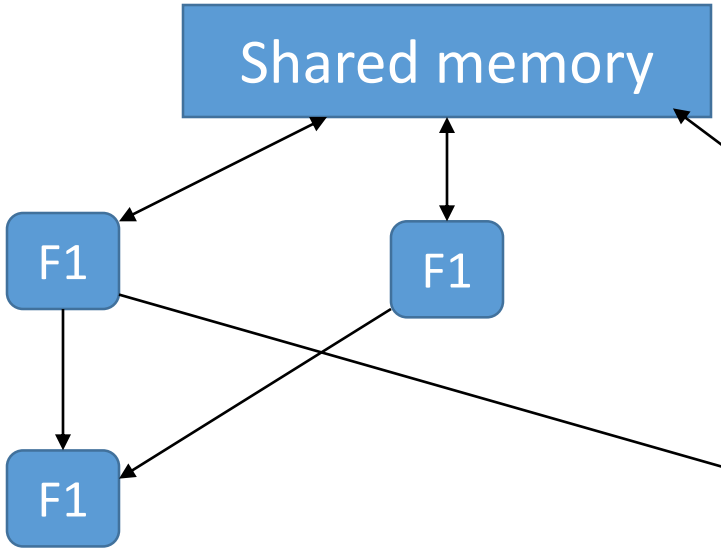
- Large computer systems can be broadly divided into number of sun-systems.
- Each sub-system include functional component with system state information and shared data area structures

There are 2 most commonly used software design strategies :-

1) Functional Design
2) Object-Oriented Design

# Functional Designs

- It involves decomposing the system into set of interacting functions which share a centralised system memory.
- Its designed using Data flow diagram
- It start with high level abstraction and then slowly progress to detailed design
- Eg:




Data flow design

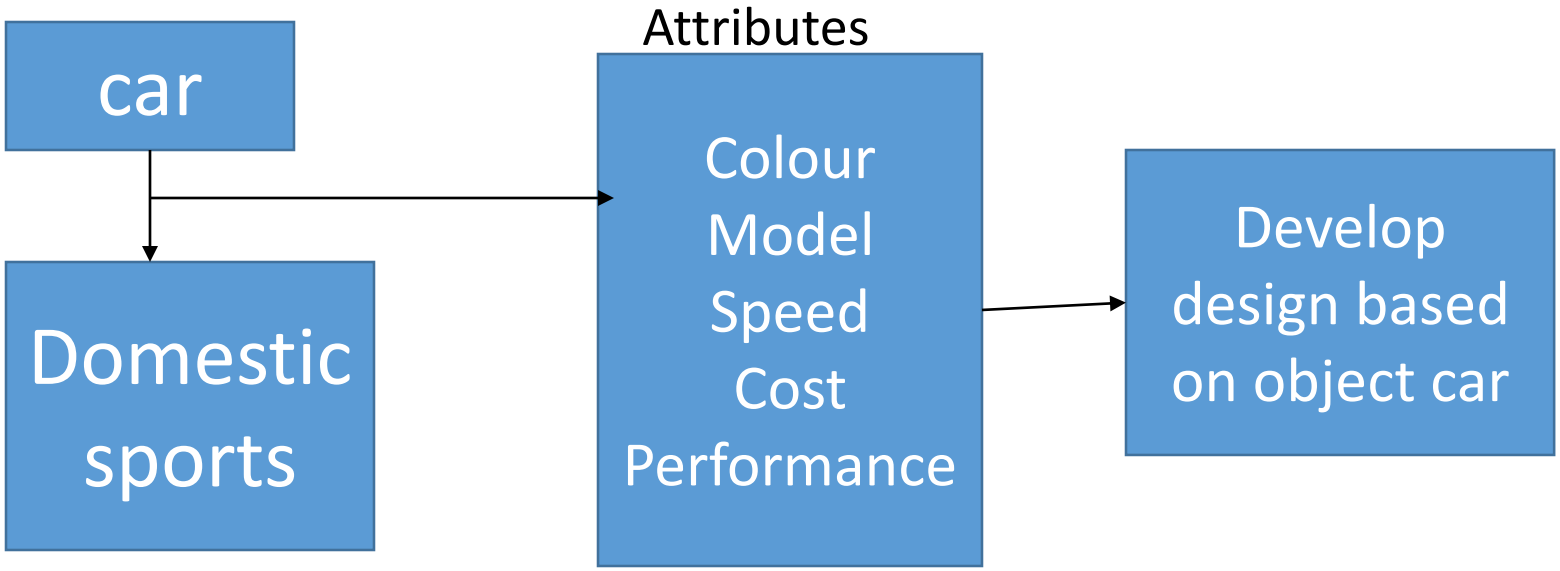
Structural Decomposition

Detailed Design

functional Design process has a set of activities

# Object Oriented Design

- Design is viewed as set of objects.
- Each object has its own attributes and operations performed on the object.
- It is based on information hiding
- The system state is de-centralised and each object manages its own state



# DESIGN QUALITY

- Good software design depends on the codification.
- Coding must be efficient. Adaptability, clarity, compactness and flexibility are the important design qualities

- Design Quality is measured using 2 qualities:

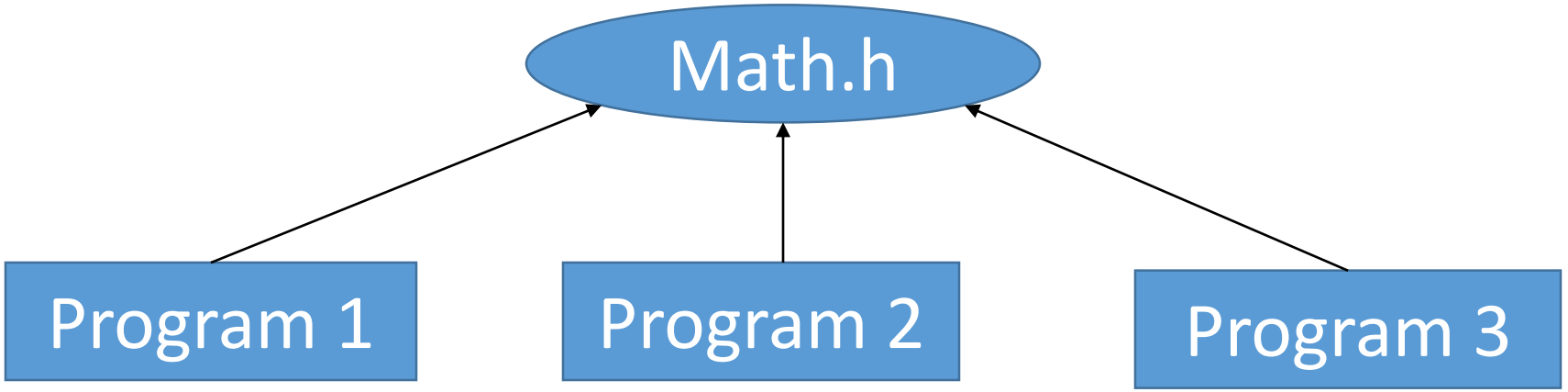
1) Cohesion
2) Coupling

Cohesion :
it’s a module represent how tightly bound the internal elements of the module are to one another.
Strong cohesion is desirable.

There are 7 levels of Cohesion

1) Coincidental Cohesion: can occur if an existing program is modularized by chopping into pieces and making different pieces to be modules.
2) Logical association : Components which perform similar function are grouped together and there exists some logical relationship.

Eg: in c language ----- math.h header file



3) Temporal Cohesion: Components which are activated at the same time are grouped together

Eg: call init-terminal, call init_calculations etc

4) Procedural Cohesion: it contains elements that belongs to a common procedural unit module performs a series of steps in specific order.

5) Communicational Cohesion: All the elements of a component operate on the same data

Eg: print and punch record

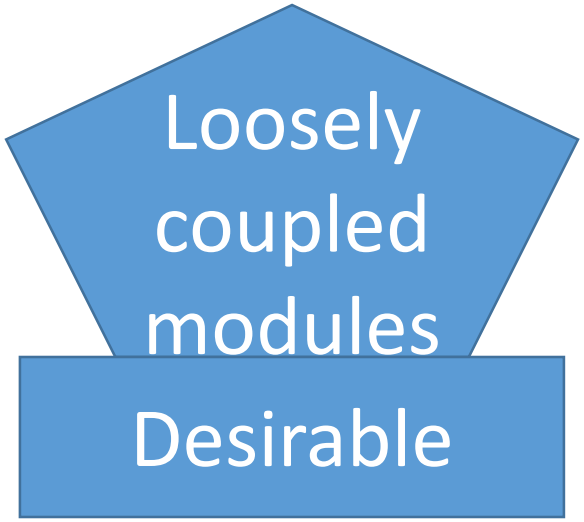
6)Sequential Cohesion: the output form one part of a component is the input to another part.

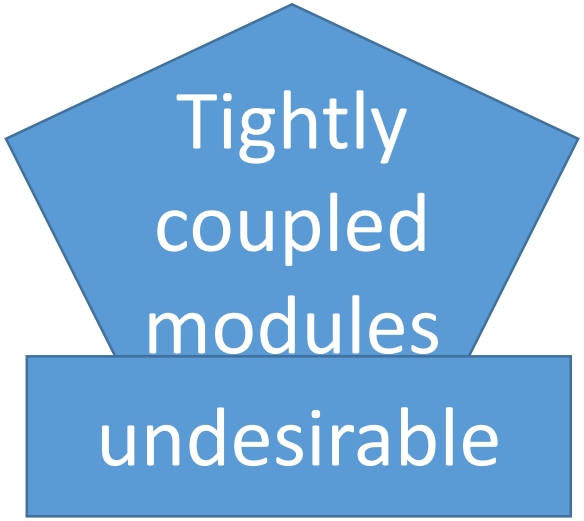
7) Functional Cohesion: its strongest Cohesion. Here modules contain elements that perform exactly one function specified

8) Object Cohesion : each operation provides functionality which allows attributes of the object to be modified.

# COUPLING

- Coupling is defined as the degree to which module interacts and communicates with another module to perform certain task.



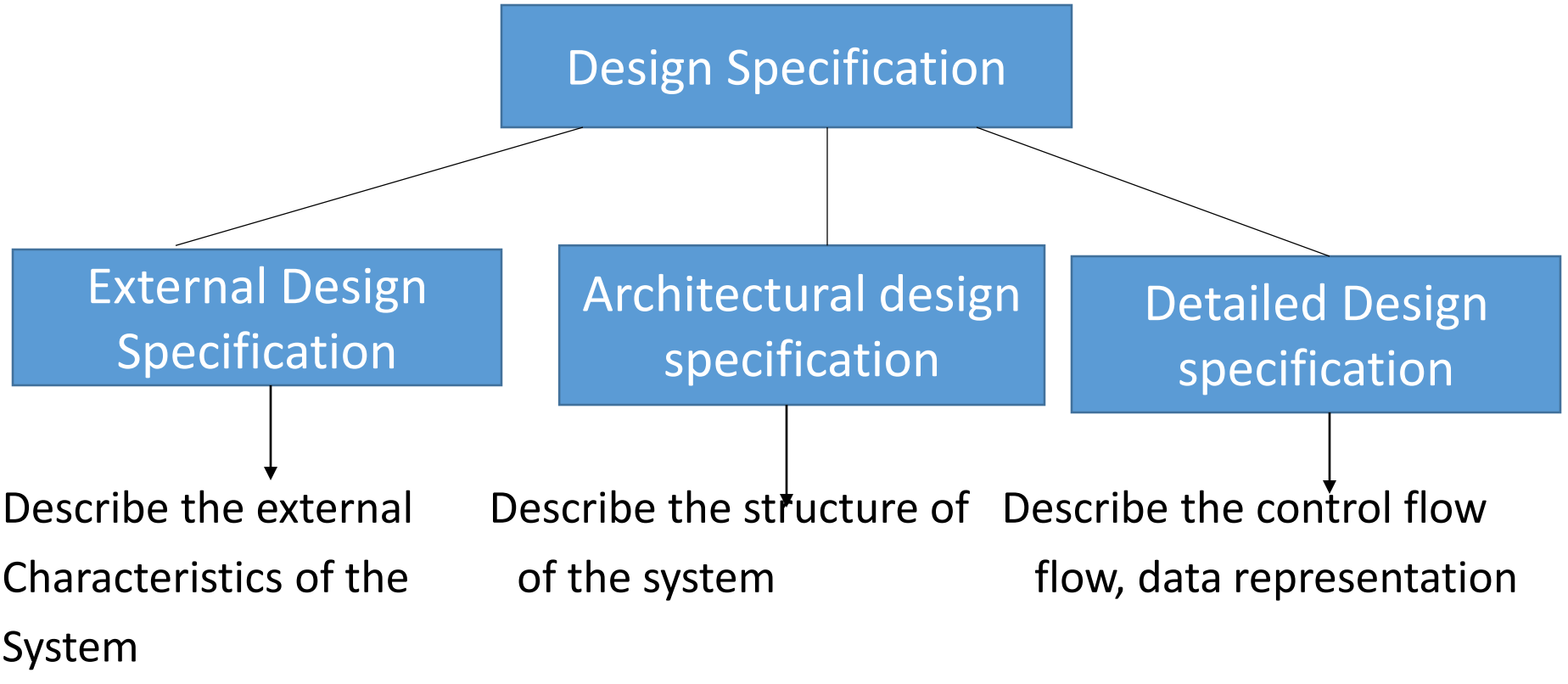



# Types of coupling

1) Data coupling
2) Stamp coupling
3) Control coupling
4) External coupling
5) Common coupling
6) Content coupling

# Design Notation and Specification

- Design representations are important in software design since they classify the inter-relationship and the interactions of the system.



# Design Notations

1) Data flow diagrams
2) Structure charts
3) HIPO diagrams (Hierarchy-input-process-output)
4) Pseudo code
5) Structured flowcharts
6) Box Diagrams
7) Structured English
8) Decision tables
9) PDL---- Program Design Language

# DOMAIN SPECIFIC ARCHITECTURES

- Architectural models may be specific to some application domain and can be reused while developing new system. These types of models are known as Domain specification architecture .

- Types are:

1) Generic Model
2) Reference Model

# Generic Model

- Are abstractions from a number of real systems. They may serve as a reference model and can be reused directly in a design.

- Generic models are bottom-up models, means they are derived from existing systems

# Reference Model

- Are derived from study of application domain rather than existing system
- OSI references model

# Difference between Generic and Refernces

| Generic | References |
| --- | --- |
| 1. Bottom up model | 1. top down model |
| 2. Derived from a study of application domain | 2. Derived from existing systems |
| 3. Used to compare possible architectures | 3. can be reused directly in a design |