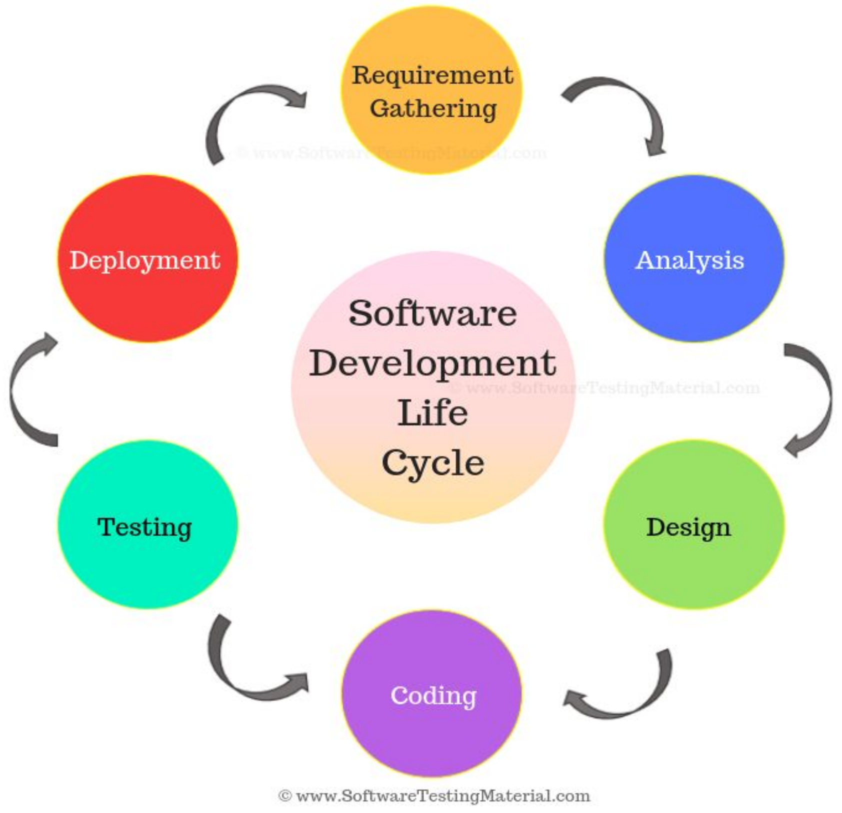
# Software Engineering

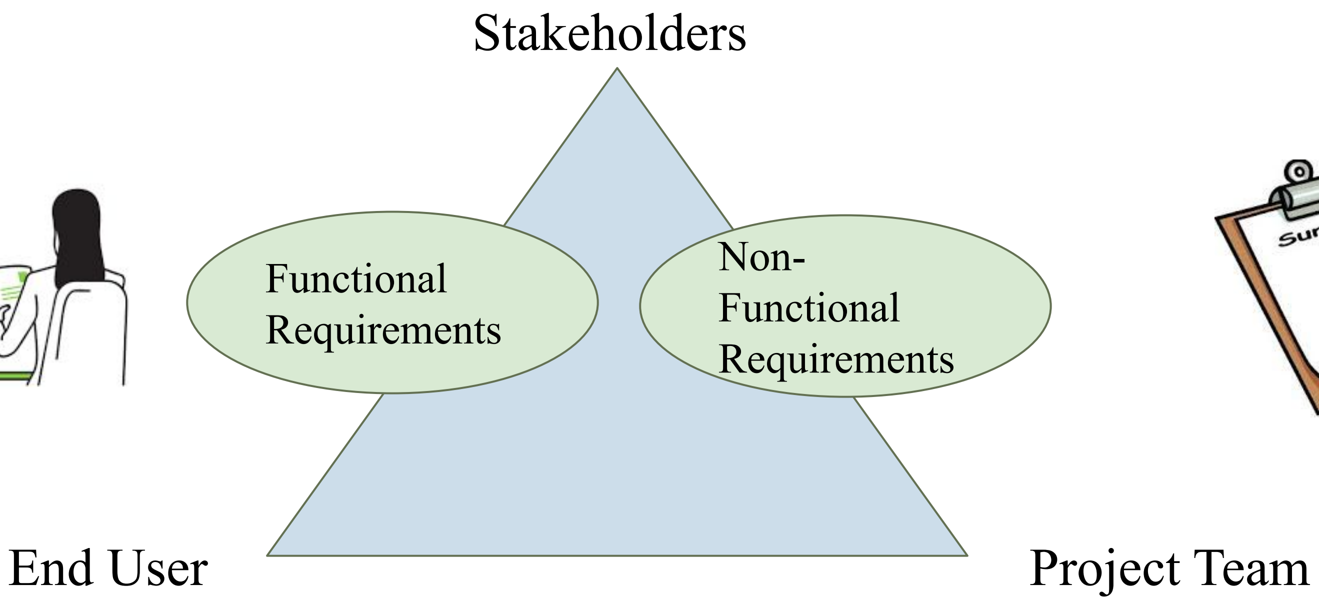
## SDLC

- SDLC is Software Development Life Cycle.
- Describes how to develop, design and maintain a software.
- SDLC contains methods to change existing systems.
- Also, can be used for creating new systems.
- SDLC helps to improve the quality of software project.

<ins>Video</ins>


Requirement Gathering
Analysis
Design
Coding
Testing
Deployment
Software Development Life Cycle
© www.SoftwareTestingMaterial.com

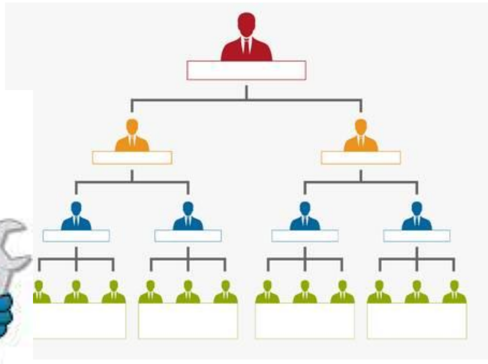
# Requirements Analysis



## Requirements Analysis

- This phase includes
  - Communication between project stakeholders, end users and project team.
  - Lock in functional and nonfunctional requirements for the project
  - Identify and capture requirements from stakeholders through client interviews and surveys.
  - Clearly define and document the SRS (Software Requirement Specification)

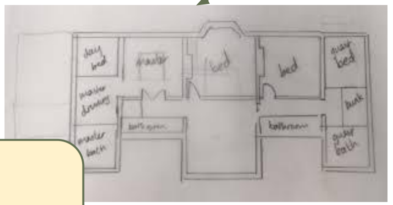
## Feasibility Analysis



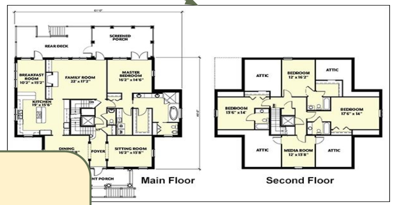
## Feasibility Analysis

- The feasibility study is basically the test of the proposed system in accordance with its:
  - workability,
  - meeting user's requirements,
  - effective use of resources and
  - the cost effectiveness.
- Economic feasibility:The likely benefits outweigh cost of solving the problem.
- Operational feasibility: Whether the problem can be solved in the user's environment with existing and proposed system workings?
- Organizational feasibility: Whether the proposed system is consistent with the organization's strategic objectives?
- Technical feasibility: Whether the problem be solved using existing technology and resources available?

## Design Phase



Logical Design



Physical Design

## Design Phase

- The software development process starts moving from what to how of the software.
- Logical Design
  - An abstract design of the project
  - Finding the type of information needed
  - The process involves arranging data into a series of logical relationships called entities and attributes
- Physical Design
  - Contains the physical implementation details
  - Deciding the programming, hardware and software details

## Implementation Phase

- Planning and announcing the implementation
- Acquire hardware resources
- Acquire software resources
- Prepare physical facilities
- Educate participants and users
- Prepare implementation schedule

## Testing Phase

- Verifies that the system works and meets all of the business requirements defined in the analysis phase
- Two primary testing activities:
  - Write the test conditions
    - **Test conditions** - the detailed steps the system must perform along with the expected results of each step
  - Perform the testing of the system
    - **Unit testing**
    - **System testing**
    - **Integration testing**
    - **User acceptance testing (UAT)**

## Maintenance Phase

- Monitor and support the new system to ensure it continues to meet the business goals.
- Primary activities:
  - **Help desk** - a group of people who responds to knowledge workers' questions
  - **Auditing the System** - Including a post implementation review of the system
  - Maintaining the system
  - Re-engineering proposals

## In Class Activity 1

- What is the appropriate pairing of items in the two columns listing various activities encountered in a software life cycle in the next slide?

## In Class Activity 1

| | |
|---|---|
| P. Requirements Capture | 1.Module Development and Integration |
| Q. Design | 2.Domain Analysis |
| R. Implementation | 3.Structural and Behavioral Modeling |
| S. Maintenance | 4.Performance Tuning |

A) P-3, Q-2, R-4, S-1

B) P-2, Q-3, R-1, S-4

C) P-3, Q-2, R-1, S-4

D) P-2, Q-3, R-4, S-1

## In Class Activity 1

**Answer: (B)**

**Explanation:** Module Development and Integration is clearly implementation work
Performance Tuning is clearly maintenance work
Domain Analysis is clearly Requirements Capture

## Case Study

**A Case Study of the Application of the Systems Development Life Cycle (SDLC) in 21st Century Health Care: Something Old, Something New?**

*Mark McMurtrey*

UNIVERSITY OF CENTRAL ARKANSAS

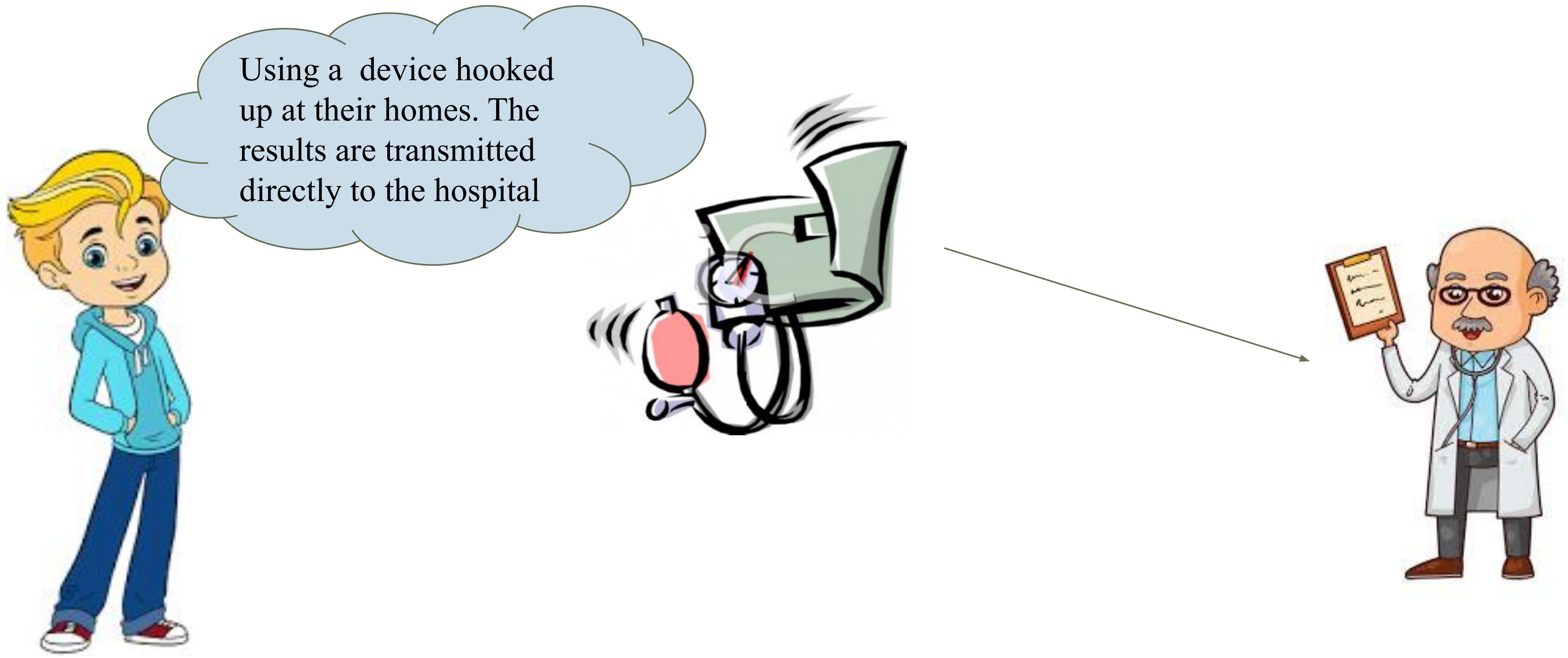
## Home Health and Study Overview

Home Care is the portion of health care carried out inside patient’s home.

## Home Health and Study Overview

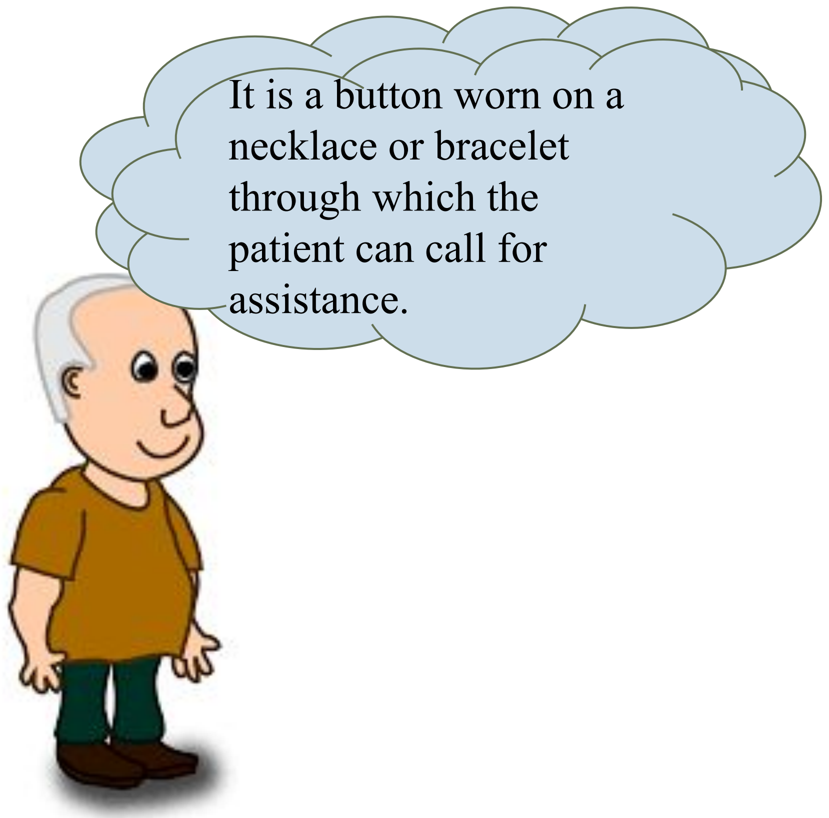
No need of constant trips to the hospital for routine procedures like Blood Pressure

## Home Health and Study Overview

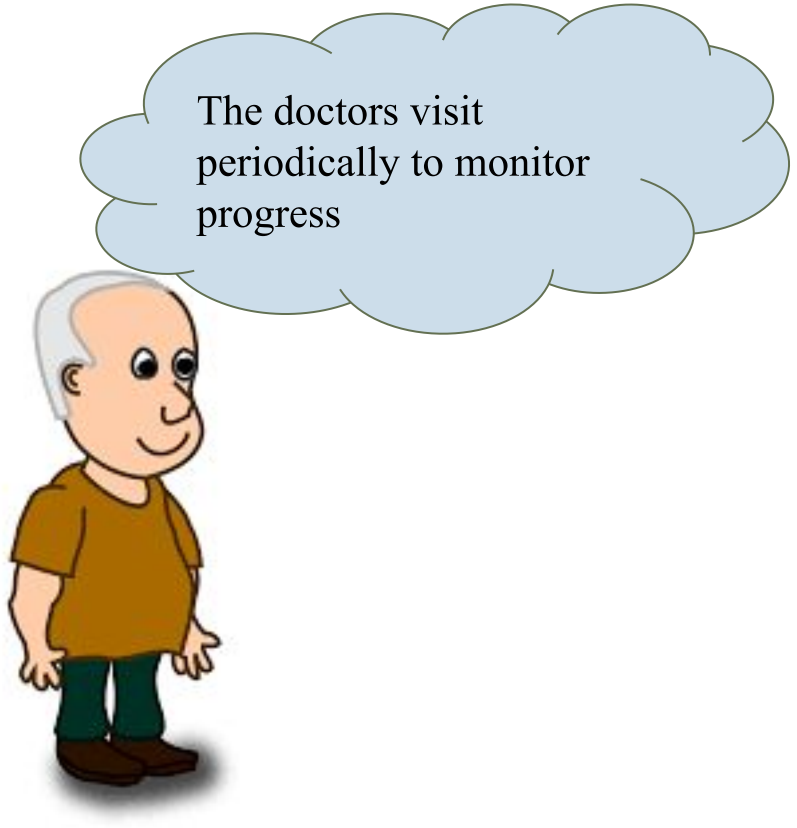


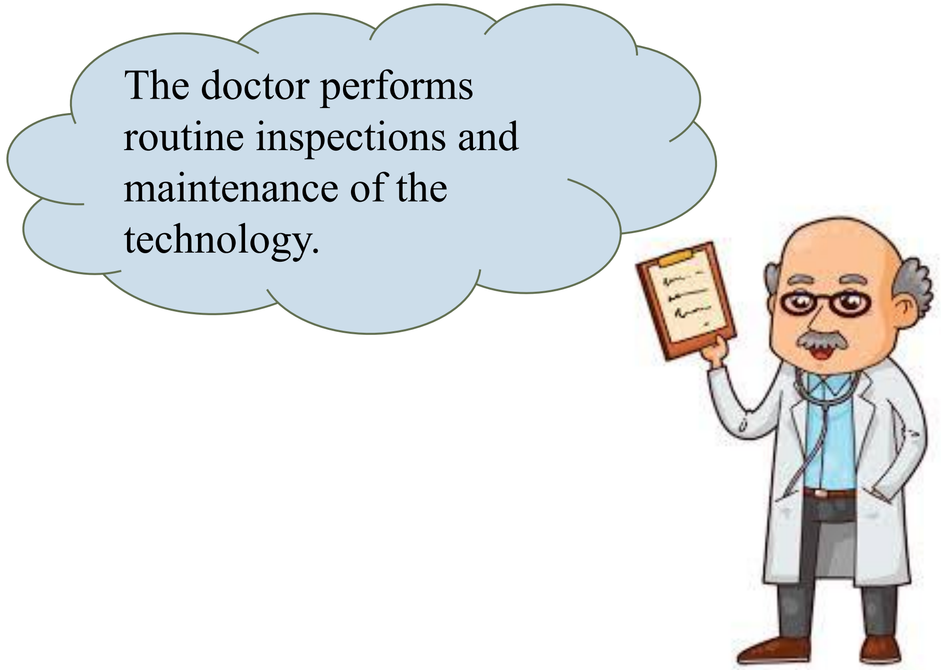
Using a  device hooked up at their homes. The results are transmitted directly to the hospital

## Home Health and Study Overview


It is a button worn on a necklace or bracelet through which the patient can call for assistance.

## Home Health and Study Overview




The doctor performs routine inspections and maintenance of the technology.

## Home Health and Study Overview

- We need to create a software to facilitate and help coordinate the Home health care portion.
- Other specifications of the product can be discussed.

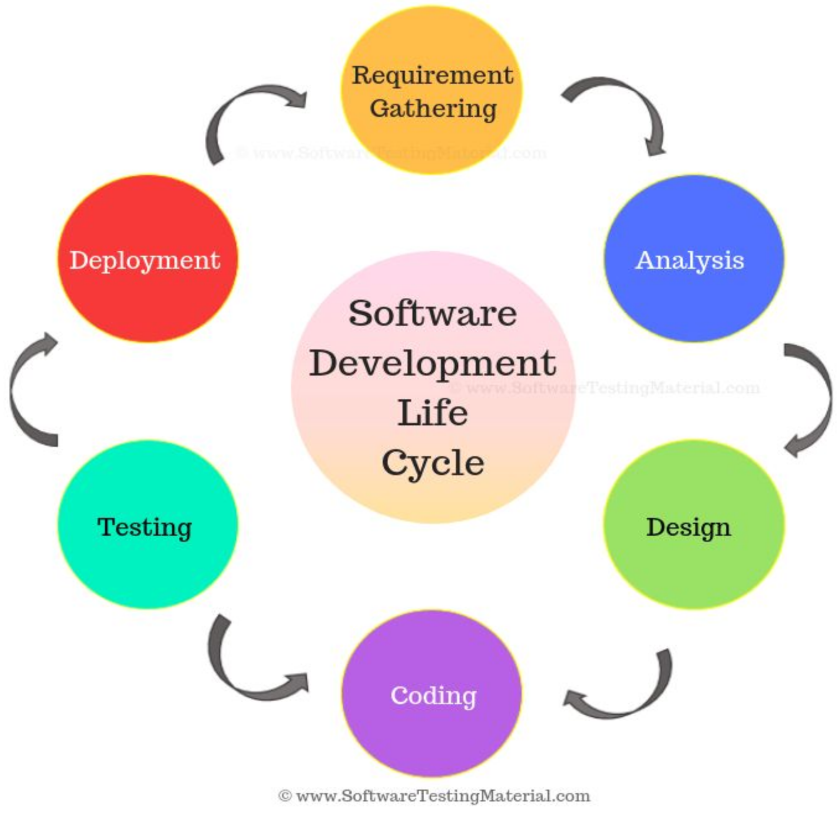
## References

| S No. | Link | Description |
|---|---|---|
| 1 | SDLC in 9 minutes (https://www.youtube.com/watch?v=i-QyW8D3ei0&feature=youtu.be) | Video covers SDLC in detail |
| 2 | SDLC simplified (https://www.youtube.com/watch?v=DRDD7UWX2y4) | Covers SDLC while telling about the process followed in a company for software development. |

## SDLC Models

## Software Life Cycle Model

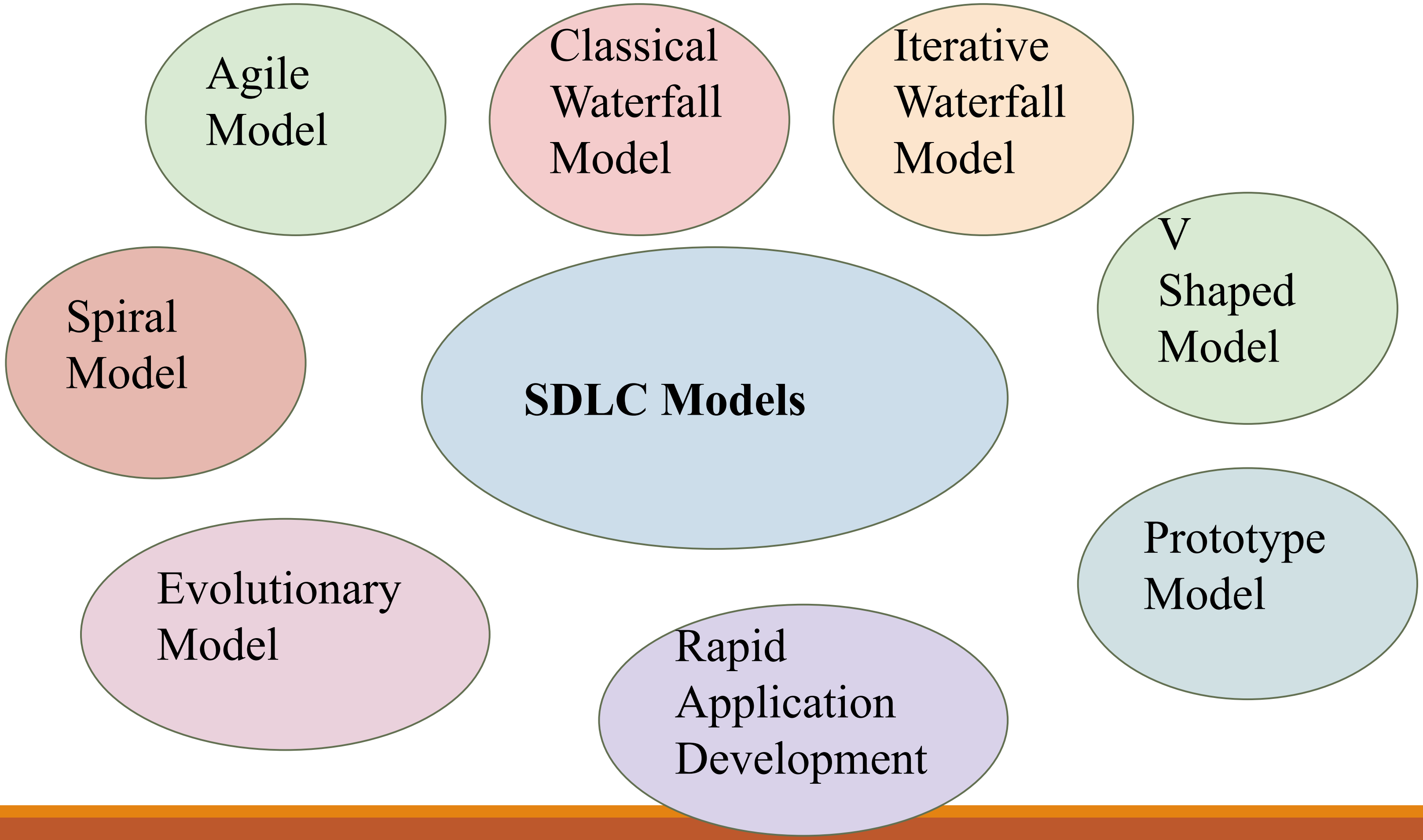
- A software life cycle model (or process model):
  - a descriptive and diagrammatic model of software life cycle:
  - identifies all the activities required for product development,
  - establishes a precedence ordering among the different activities,
  - Divides life cycle into phases.
  - defines entry and exit criteria for every phase.


Requirement Gathering
Analysis
Design
Coding
Testing
Deployment
Software Development Life Cycle
© www.SoftwareTestingMaterial.com

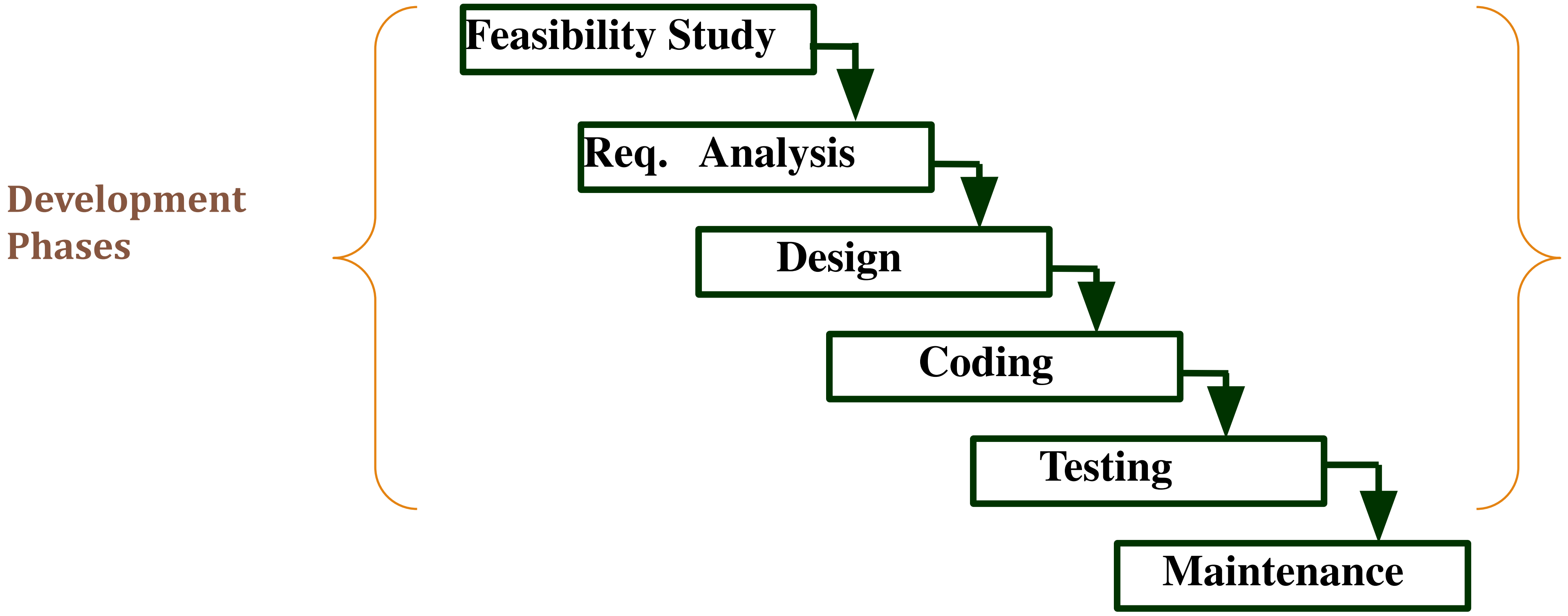
## Why Model  Life Cycle ?

- A written description:
  - Forms a common understanding of activities among the software developers.
  - Helps  in identifying inconsistencies, redundancies, and omissions in the development process.
  - Helps in tailoring a process model for specific projects.
  - The development team must identify a suitable life cycle model  and then adhere to it.
  - Helps monitor the progress of the project otherwise the project manager would have to depend on the guesses of the team members. This usually leads to a problem known as the  99% complete syndrome.

## Different Life Cycle Models



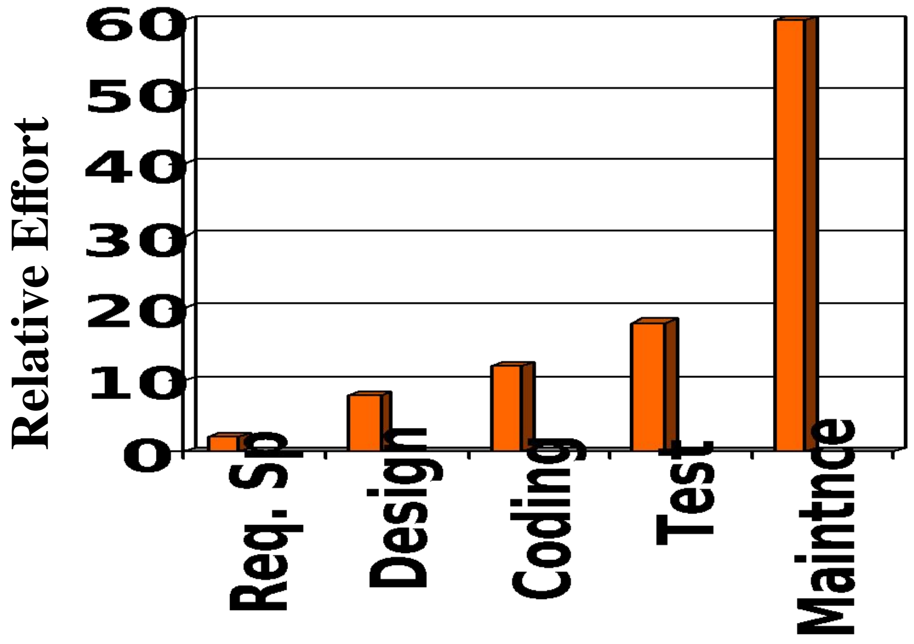
## Classical Waterfall Model



(https://www.youtube.com/watch?v=ic52qU7pLoQ)

# Activities during Feasibility Study

- Determine whether developing the product is
  - financially worthwhile
  - technically feasible
- Roughly understand what the customer wants
- Work out an overall understanding of the problem.
- Formulate different solution strategies.
- Examine alternate solution strategies in terms of:
  - resources required,
  - cost of development, and
  - development time.



## Requirements Analysis and Specification

- To understand the exact requirements of the customer and document them properly in form of Software Requirement Specification document.
- Consists of two distinct activities:
  - requirements gathering and analysis
  - requirements specification

## Requirements Gathering

- Gathering relevant data:
    - usually collected from the end-users through interviews and discussions.
    - For example,  for a business accounting software: interview all the accountants of the organization to find out their requirements.
- The data you initially collect from the users:
    - would usually contain several contradictions and ambiguities.
    - each user  typically has only a partial and incomplete view of the system.

## Requirements Analysis

- Ambiguities and contradictions:
  - must be identified
  - resolved by discussions with the customers
- Next, requirements are organized into a Software Requirements Specification (SRS) document.
- Engineers doing requirements analysis and specification are designated as analysts.

## Design

- Identify all the functions to be performed.
- Identify data flow among the functions.
- Decompose each function  recursively into sub-functions.
- Identify data flow among the  sub-functions as well.
- Identify the design required:
    - High-level design:
        - decompose the system into modules,
        - represent invocation relationships among the modules.
    - Detailed design:
        - different modules designed in greater detail:
        - data structures and algorithms for each module are designed.

## Implementation

- Purpose of implementation phase is to translate software design into source code.
- During the implementation phase:
  - each module of the design is  coded,
  - each module is tested independently as a stand alone unit, and debugged ( Unit Testing).
  - each module is documented.

## Integration and System Testing

- Different modules are integrated in a planned manner:
  - modules are almost never integrated in one shot.
  - Normally integration is carried out through a number of steps.
  - During each integration step, the partially integrated system is tested.
- After all the modules have been successfully integrated and tested: system testing is carried out.
  - System testing ensure that the developed system functions according to its requirements as specified in the SRS document.

## Maintenance

- Corrective maintenance:
  - Correct errors which were not discovered during the product development  phases.
- Perfective maintenance:
  - Improve implementation of the system
  - enhance functionalities of the system.
- Adaptive maintenance:
  - Port software to a new environment,
  - e.g. to a new computer or to a new operating system.

## Waterfall Strengths

- Easy to understand, easy to use
- Provides structure to inexperienced staff
- Milestones are well understood
- Sets requirements stability
- Good for management control (plan, staff, track)
- Works well when quality is more important than cost or schedule

## Waterfall Deficiencies

- All requirements must be known upfront
- Deliverables created for each phase are considered frozen – inhibits flexibility
- Can give a false impression of progress
- Does not reflect problem-solving nature of software development – iterations of phases
- Integration is one big bang at the end
- Little opportunity for customer to preview the system (until it may be too late)

## When to use the Waterfall Model?

- Requirements are very well known
- Product definition is stable
- Technology is understood
- New version of an existing product
- Porting an existing product to a new platform.
- Applications: Waterfall model was used to develop enterprise applications like
    - Customer Relationship Management (CRM) systems,
    - Human Resource Management Systems (HRMS),
    - Supply Chain Management Systems, Inventory Management Systems,
    - Point of Sales (POS) systems for Retail chains etc.

## Example

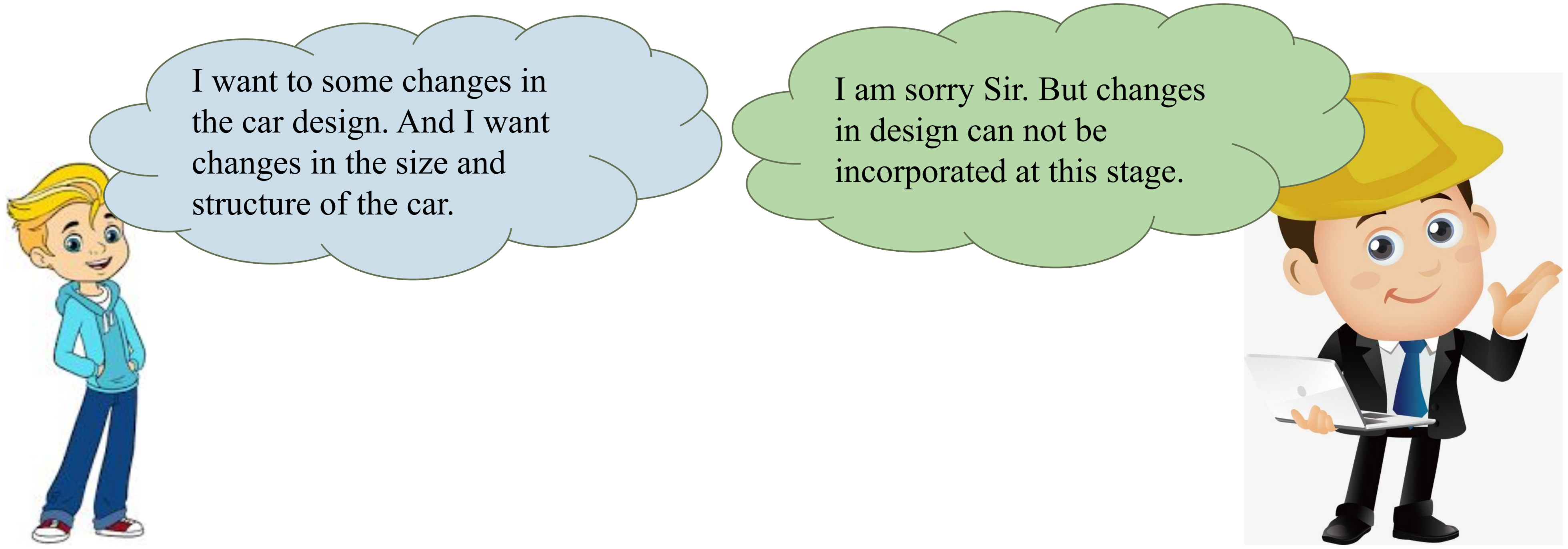
- Take the example of automobile building.
- We need to make a car as specified by the client company.

## Example

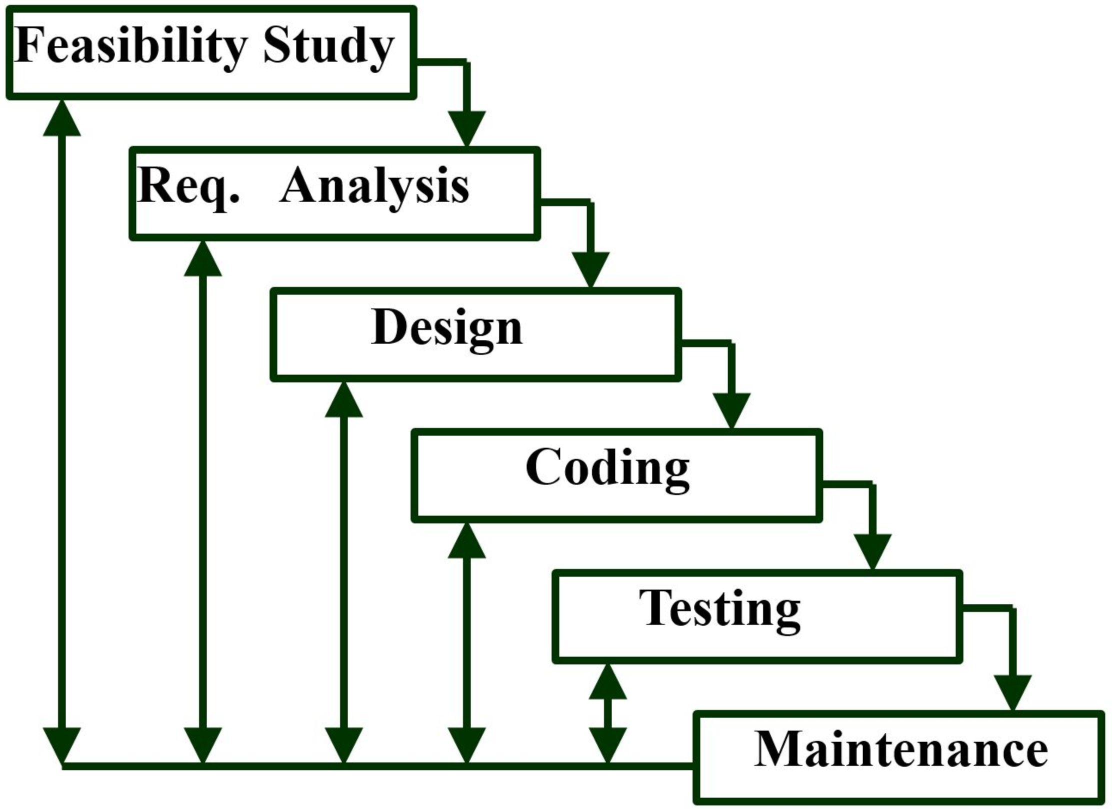
- So steps will be followed as:
    - **Requirement Analysis:** To analyse the type of car client company needs.
    - All specifications regarding the car size, its color, its functionalities are locked in this phase.
    - **Design:** The company creates a design based on the type of requirements specified by the user.
    - **Implementation:** The actual making of car parts and assembling is done in this process.
    - **Testing:** The car is tested by us and by the customer as well.

## Example

- Let’s say in middle of implementation phase while the car engine is being made, the client comes,


I want to some changes in the car design. And I want changes in the size and structure of the car.
I am sorry Sir. But changes in design can not be incorporated at this stage.

## Iterative Waterfall Model



## Iterative Waterfall Model

- Classical waterfall model is idealistic and assumes that no defect is introduced during any development activity.
- In practice, defects do get introduced in almost every phase of the life cycle.
- For example, a design defect might go unnoticed till the coding or testing phase. Therefore we need feedback paths in the classical waterfall model.

## Example

- Let’s say in middle of implementation phase while the car engine is being made, the client comes,

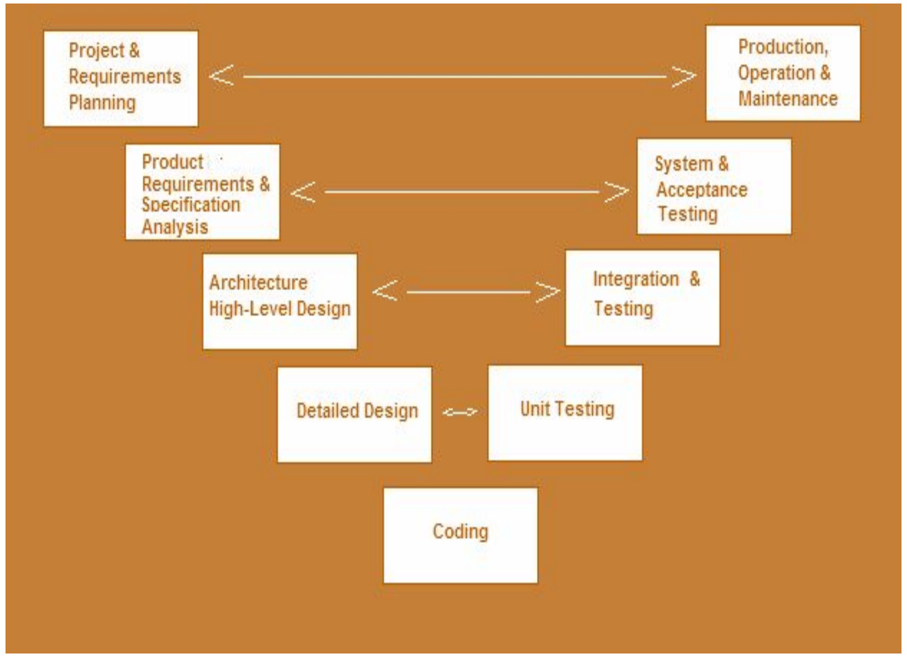
I want to some changes in the car design. And I want changes in the size and structure of the car.

Sure Sir, Let’s see what changes can be incorporated in the design.

**The design stage is revisited and implementation is restarted along with the required changes**

## V-Shaped SDLC Model

- A variant of the Waterfall that emphasizes the verification and validation of the product.
- Testing of the product is planned in parallel with a corresponding phase of development.


Project & Requirements Planning
Production, Operation & Maintenance
Product Requirements & Specification Analysis
System & Acceptance Testing
Architecture High-Level Design
Integration & Testing
Detailed Design
Unit Testing
Coding

## V-Shaped Strengths

- Emphasize planning for verification and validation of the product in early stages of product development
- Each deliverable must be testable
- Project management can track progress by milestones
- Easy to use

## V-Shaped Weaknesses

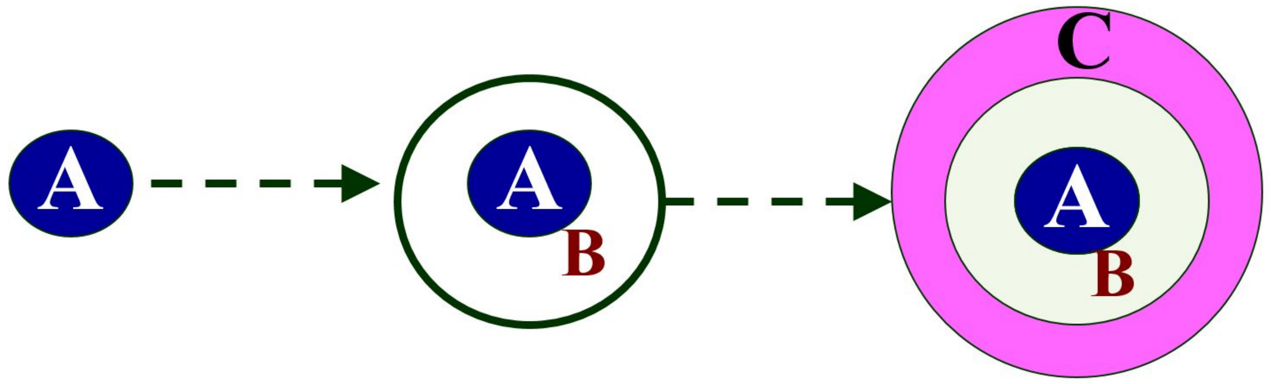
- Does not easily handle concurrent events
- Does not handle iterations or phases
- Does not easily handle dynamic changes in requirements
- Does not contain risk analysis activities

## When to use the V-Shaped Model?

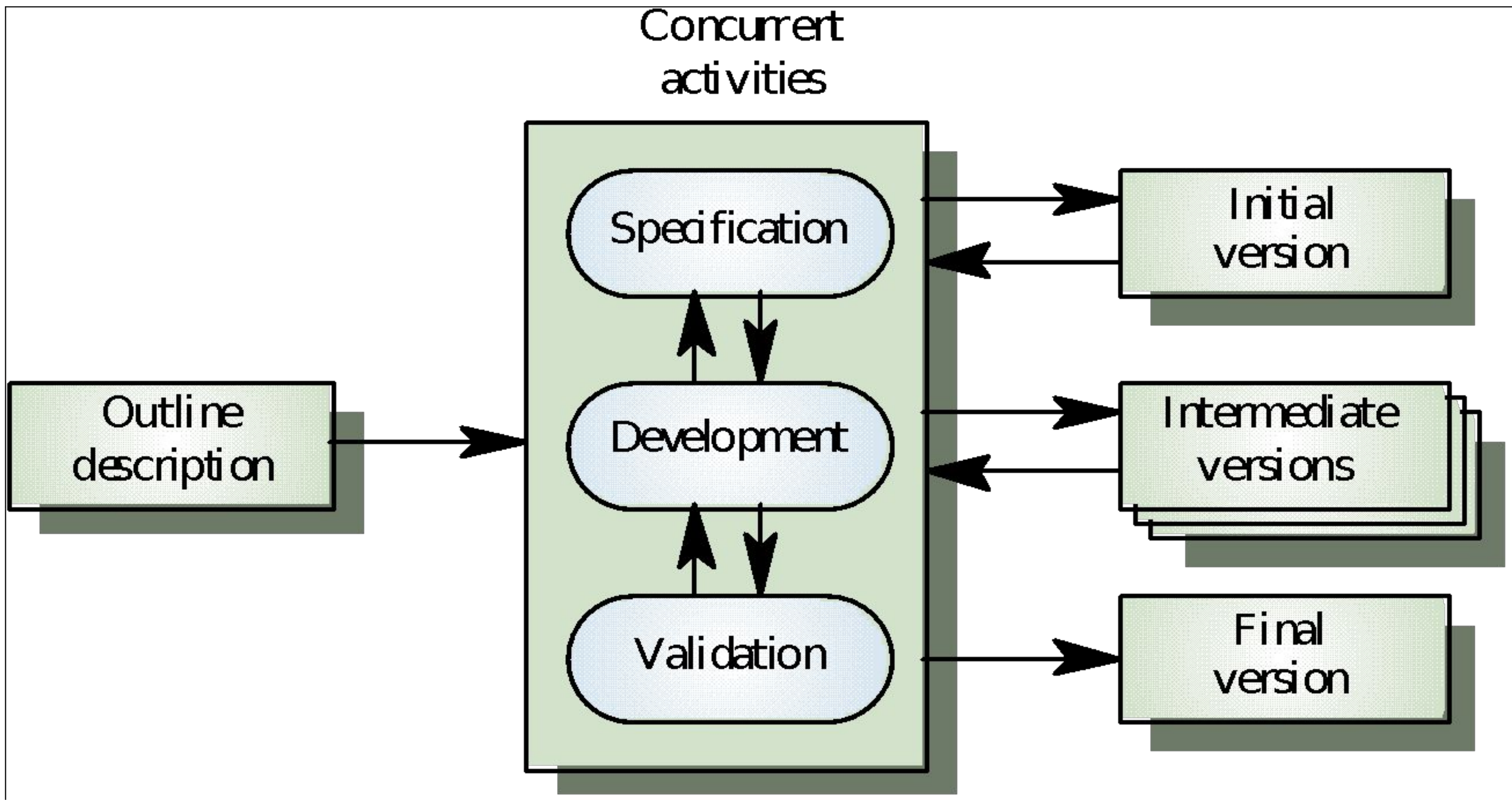
- Excellent choice for systems requiring high reliability – hospital patient control applications
- All requirements are known up-front
- When it can be modified to handle changing requirements beyond analysis phase
- Solution and technology are known

## Evolutionary development

- Exploratory development
    - Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements
- Throw-away prototyping
    - Objective is to understand the system requirements. Should start with poorly understood requirements



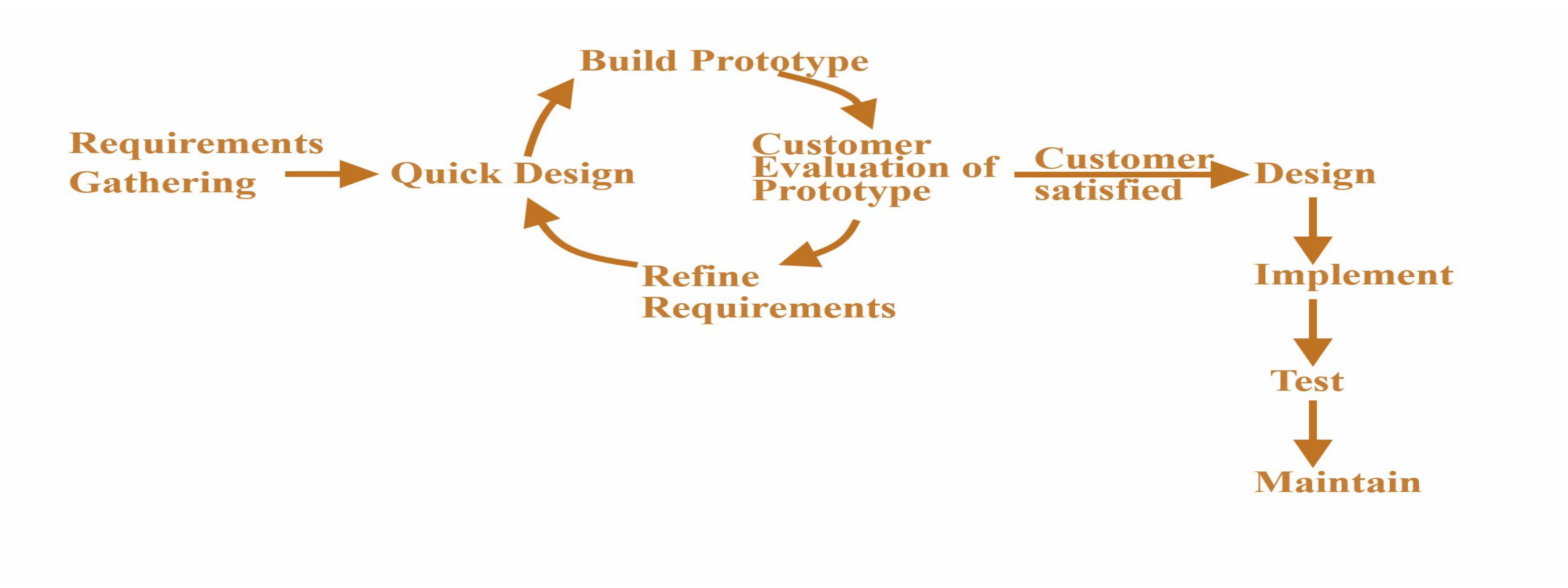
## Evolutionary development



## Evolutionary development

- Problems
  - Lack of process visibility
  - Systems are often poorly structured
  - Special skills (e.g. in languages for rapid prototyping) may be required
- Applicability
  - For small or medium-size interactive systems
  - For parts of large systems (e.g. the user interface)
  - For short-lifetime systems

# Prototyping Model



## Prototyping Model

- A prototype is a toy implementation of a system is built before actual development with :
  - limited functional capabilities,
  - low reliability,
  - inefficient performance.
- Start with approximate requirements.
- Carry out a quick design.

## Prototyping Model

- Prototype model is built using several  short-cuts that might involve using inefficient, inaccurate,  or dummy functions.
- For ex: A function may use a table look-up rather than performing the actual computations.
    - The developed prototype is submitted to the customer for his the user feedback, requirements are refined.
    - The actual system is developed using the classical waterfall approach.

## Reasons for developing a prototype

- Illustrate to the customer:
  - input data formats, messages, reports, or interactive dialogs.
- Examine technical issues associated with product development:
  - Major design decisions  depend on issues like:
    - response time of a hardware controller,
    - efficiency of a sorting algorithm, etc.
- It is impossible to ``get it right'' the first time.
  - we must plan to throw away  the first product ,if we want to develop a good product.

## Advantages

- Even though construction of a working prototype model involves additional cost --- overall development cost might be lower for:
  - systems with unclear user requirements,
  - systems with unresolved technical issues.
- Many user requirements get properly defined and technical issues get resolved:
  - these would have appeared later as change requests and resulted in incurring massive redesign costs.

## Disadvantages

- Requirements analysis and specification phase becomes redundant:
  - final working prototype  (with all user feedbacks incorporated) serves as an animated requirements specification.
- Design and code for the prototype is usually thrown away:
  - However, the experience gathered from developing the prototype helps a great deal while developing the actual product.

## When to use Prototyping?

- Requirements are unstable or have to be clarified
- As the requirements clarification stage of a waterfall model
- Develop user interfaces
- Short-lived demonstrations
- New, original development

## Example

- Let’s say a client asked you to create a website for his business and specified the requirements.

Sure Sir, We will incorporate all your changes and create a new prototype.

But this is

I

I also need to change the main section to include more new images. Also change the footer. I don’t like this one.

**The client asks for too many changes.**
**Requirements have changed completely.**
**No worries, Prototype model can incorporate all the changes.**

## In Class Questions

- What are the two important elements of the Waterfall Model?
- What are the seven products described by the Waterfall Model?
- When is software testing performed in the Waterfall Model?
- Explain the purpose of iteration in the Waterfall Model.
- Which process is the only process that does not involve iteration?
- Why is Waterfall advantageous when a project requires specialized staff and skills that are in short supply?

## In Class Exercise

Q1 Which of the following is not true of the conversion phase of the development life cycle?

- A. the user and systems personnel must work closely together.
- B. steps must be taken to phase out the old system
- C. documentation should be emphasized
- D. the non machine components of the system should be considered
- E. None of the above

## In Class Exercise

Q2:In phase 1 of the system development life cycle, which of the following aspects are usually analyzed?

- A. outputs
- B. input (transactions)
- C. controls
- D. All of the above
- E. None of the above

## In Class Exercise

Q3: During the maintenance phase

- A. System requirements are established
- B. System analysis is carried out
- C. Programs are tested
- D. All of the above
- E. None of the above

## Question

Explain why systems developed as prototypes should not normally be used as production systems.

## Solution

Prototypes should be discarded after development as they are not a good basis for a production system:

1. It may be impossible to tune the system to meet non-functional requirements;
2. Prototypes are normally undocumented;
3. The prototype structure is usually degraded through rapid change;
4. The prototype probably will not meet normal organizational quality standards.

## Home Work

- For each of the following documents, indicate in which phase(s) of the software life cycle it is produced:
    - final user manual,
    - architectural design,
    - SQA plan,
    - module specification,
    - source code,
    - statement of work,
    - test plan,
    - preliminary user manual,
    - detailed design,
    - cost estimate, project plan, test report, documentation.

## References

| S No. | Link | Description |
|---|---|---|
| 1 | Prototyping Model (https://www.youtube.com/watch?v=v8WzC5bCqY0) | Video covers Prototype model and the issues that are involved in the model in detail |
| 2 | Waterfall Model (https://www.youtube.com/watch?v=Y_A0E1ToC_I) | Covers Waterfall model in detail. |
| 3 | Evolutionary Model (https://www.youtube.com/watch?v=2S8lYZQF3I4) | Explains Evolutionary model and its working |

“YOU’VE GOT TO BE VERY CAREFUL BECAUSE IF YOU DON’T KNOW WHERE YOU’RE GOING, YOU MIGHT NOT GET THERE.”

-YOGI BERRA

Thank You