Software Development Life Cycles

Chapter 3
Outcomes & Software Process

Outcomes of Chapter 3

- Examine Software Development Life Cycle (SDLC)
- Discuss the various models used for Software Development
- Compare the different models used for Software development

The Software Process

- A structured set of activities required to develop a software system.
- Many different software processes but all involve:
  - Specification – defining what the system should do;
  - Design and implementation – defining the organization of the system and implementing the system;
  - Validation – checking that it does what the customer wants;
  - Evolution – changing the system in response to changing customer needs.

The Software Process (Cont'd)

- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

SDLC Introduction

Software Development Life Cycle

- Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software.
- It consists of a detailed plan describing how to develop, maintain, replace and change or improve specific software.

Software Development Process Activities

- The major software development activities include gathering requirements, software design and implementation, code testing, documentation and maintenance.
- There are various ways in which the above activities can be done to achieve the final software. These are known as software development models.

SDLC Models

- Code and Fix Model
- Waterfall Model
- V-Model
- Incremental Implementation
- Prototyping
- Spiral Model

SDLC Models

Code and Fix Model

Code and Fix Model

- The code-and-fix model is the concept of jumping right in and building the software, only to fix any problems along the way.
- It consists of two major steps: Coding and Fixing.

Advantages of Code and Fix Model

- Convenient for small projects
- Time saver and Low budget
- Common when team consists of entry level developers
- Helps with greater experimentation and observation
- Developers have full control over development process

Disadvantages of Code and Fix Model

- Costs more time and money
- End products are difficult to maintain
- Lower quality
- Doesn't work for medium to large team sizes
- Difficult to estimate the work delivery

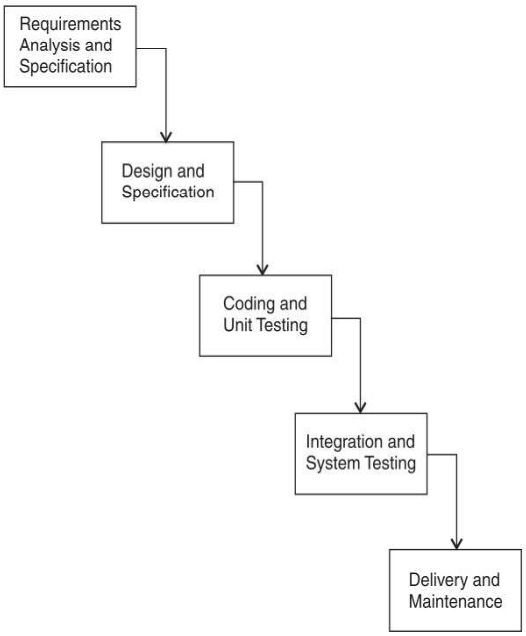
SDLC Models

Waterfall Model

Waterfall Model

- The waterfall model is a basic software development model where the life cycle of a software product is divided into a set of sequential phases.
- Structural / Geometric Similarity with a Waterfall

Classic Waterfall Model



Phases of Software Development

- Requirement Analysis and Specification
- Design and Specification
- Coding and Unit Testing
- Integration and System Testing
- Delivery and Maintenance

Characteristics of Waterfall Model

- Sequential phases
- One way downward flow of information
- Work divided as per phases
- Goal/Deliverable associated with each phase
- Output of one phase is input to the next phase
- Different developmental tools can be used for each phase
- Easy to manage

Advantages of Waterfall Model

- Simple and easy
- Clearly defined phases and outputs
- Phases processed one at a time
- Well suited for software systems with well defined requirements
- All phases well documented, easy maintainability
- Planned component interaction
- Project management is easier
- Reduces development and maintenance costs
- Enables organizations to be more structured and manageable

Disadvantages of Waterfall Model

- Rigid. No overlap Permissible
- Not suitable for continually changing or long ongoing systems
- Difficult to accommodate changing requirements
- Errors cannot be caught early and fixed
- No working software till the end
- Bureaucratic nature

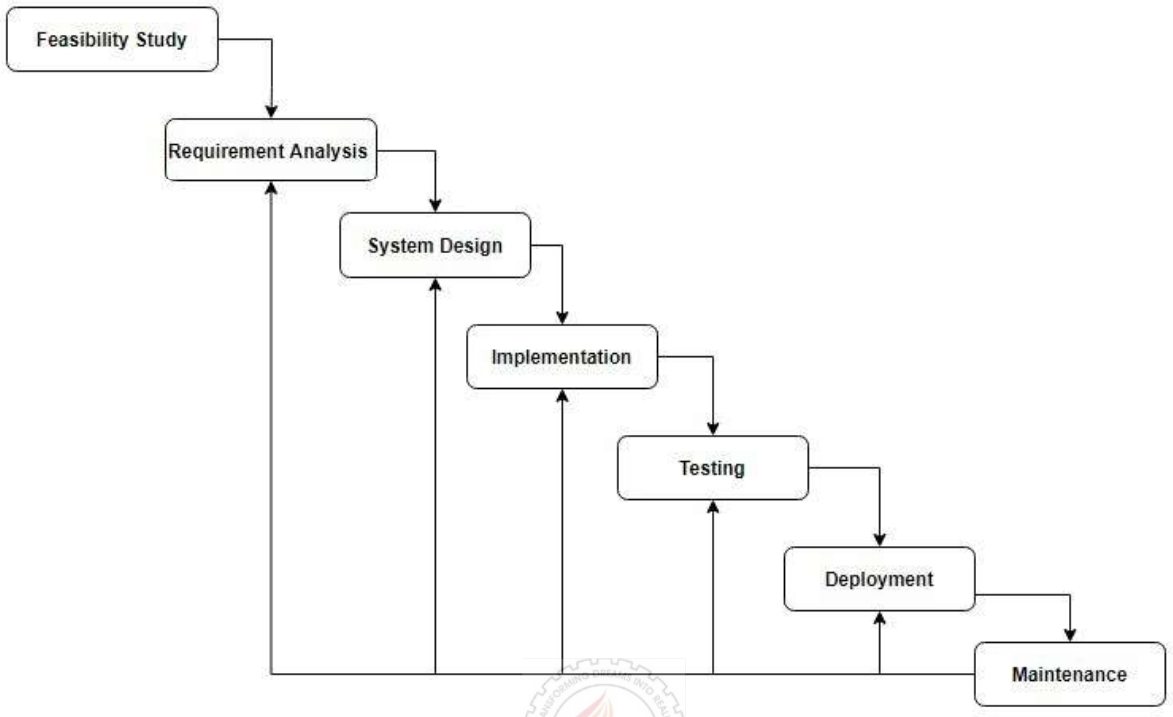
SDLC Models

Iterative Waterfall Model

Iterative Waterfall Model

- Classical waterfall model is idealistic
- Assumes no Changes or Defects
- Change Management is Expensive
- Feedback Mechanism to Classical Waterfall Model
- Phase Containment of Errors (Detect in the that Phase itself)

Iterative Waterfall Model



Drawbacks of Iterative Model

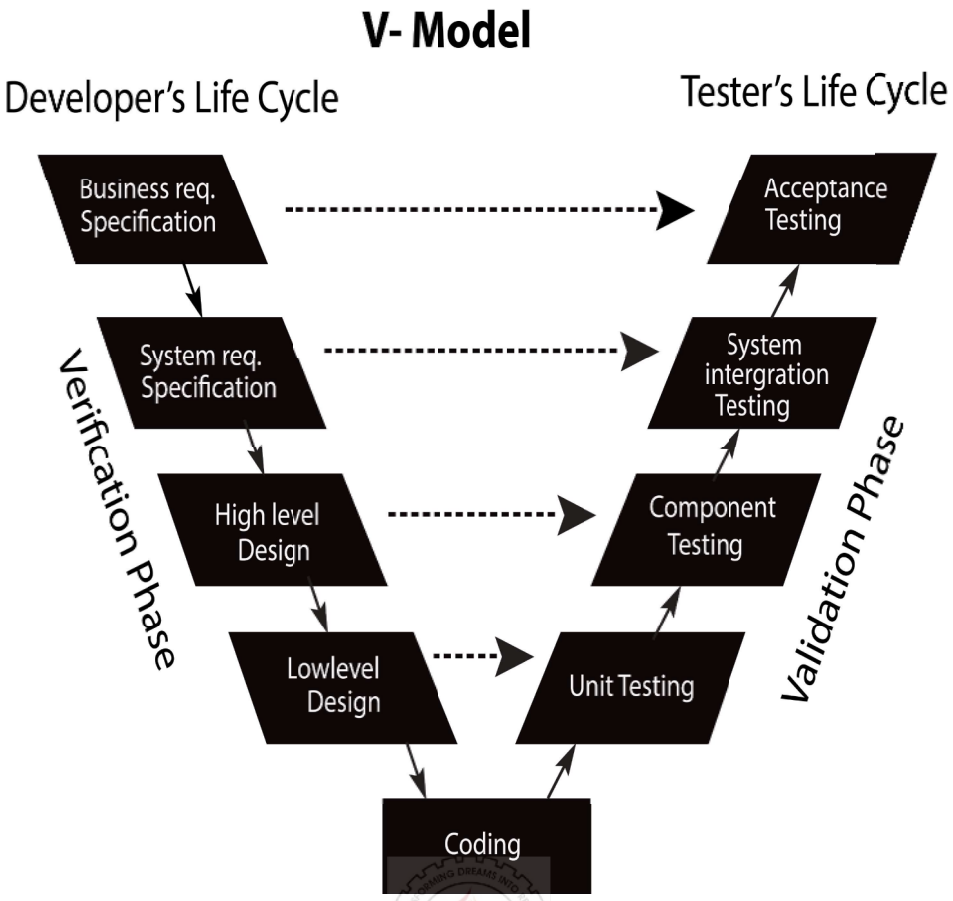
- Carries over most of the drawbacks of the Classical Waterfall Model
- Difficult to incorporate change requests
- Incremental delivery not supported
- Overlapping of phases not supported
- Risk handling not supported
- Limited customer interaction

SDLC Models

V-Model

V-Model

- The V-model is a type of SDLC model where process executes in a sequential manner in V-shape.
- It is also known as Verification and Validation model.



Characteristics of V-Model

- Testing phase associated with each corresponding development stage
- Next phase only after completion of previous phase
- Testing phases planned in parallel with development
- Verification phases on one side of the 'V' and Validation phases on the other side.
- Verification and Validation phases are joined by coding phase in V-shape.

Verification Phase

- Static analysis method (review) done without executing code.
- Evaluation of the product development process to find whether specified requirements are met
  - Requirement Analysis
  - System Design
  - High Level Design
  - Coding Phase

Validation Phase

- Dynamic analysis method (functional, non-functional), testing is done by executing code.
- After the completion of the development process to determine whether the software meets the customer expectations and requirements.
  - Unit Testing
  - Integration Testing
  - System Testing
  - Acceptance Testing

Advantages of V-Model

- Highly disciplined model , phases completed one at a time
- Suitable for small projects with clearly defined requirements
- Better quality software due to verification and validation
- Accurate tracking of progress

Disadvantages of V-Model

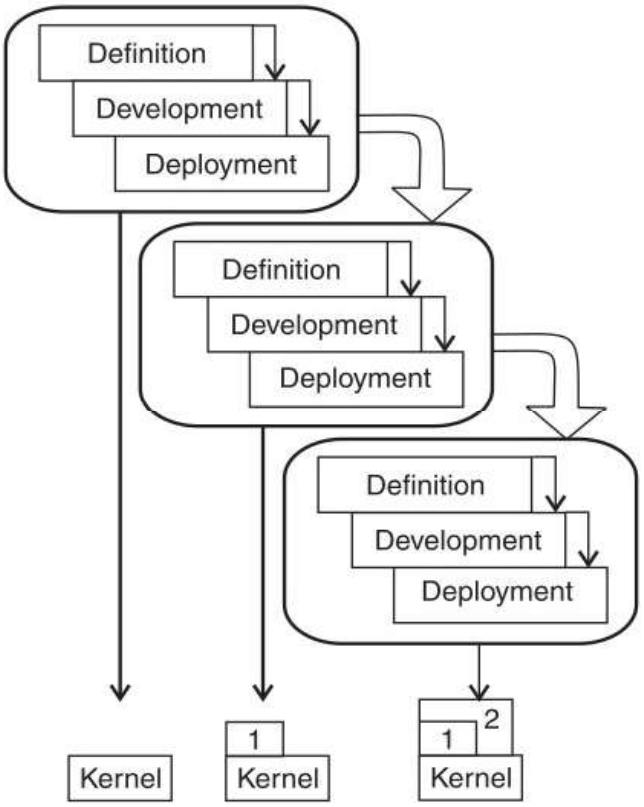
- High risk and uncertainty
- Not suitable for complex systems
- Not suitable where requirements are not clear and changing
- No support for iteration of phases
- Cannot handle concurrent activities

SDLC Models

Evolutionary Models - Incremental Implementation

Incremental Implementation

- In this approach a software product is developed in increments or steps.
- First a few functions are developed to get a working model of the software.
- Later more functions are added.



Advantages of Incremental Implementation

- Users changes can be incorporated
- Most important functionality developed first with better quality
- Time taken to show working software is reduced
- Testing, error detection and correction become easy

Disadvantages of Incremental Implementation

- The overall architecture or design of the software product must be done completely at the beginning of the life cycle.
- All increments or steps should be a part of this design.
- Such kind of customer-developer contracts are not very usual.

SDLC Models

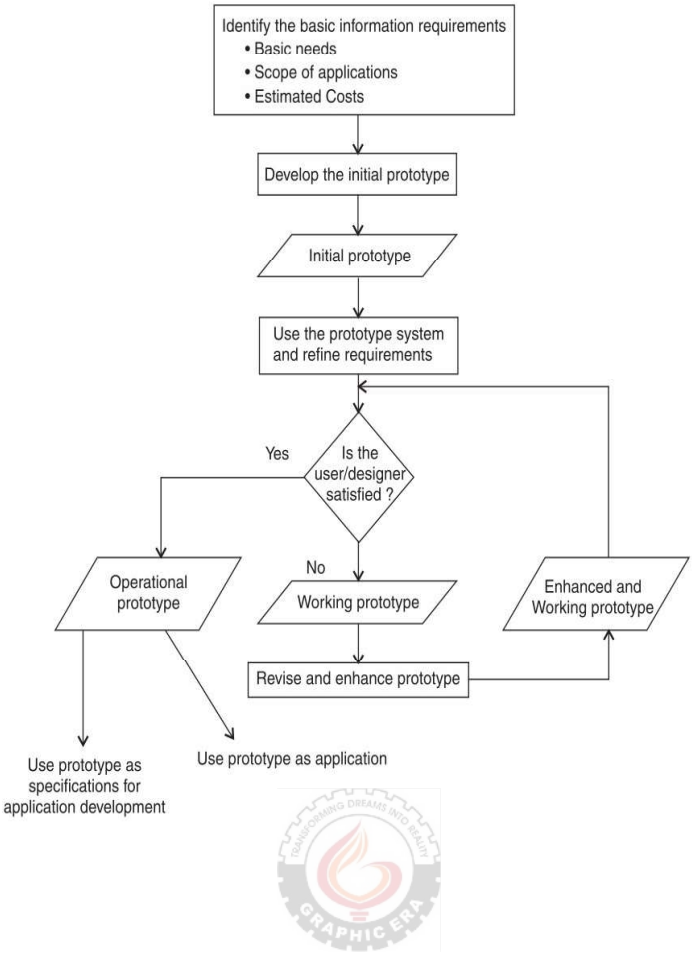
Prototyping

Prototyping

- In this model a working prototype of the software is given to the user so that we can get all their requirements in more detailed fashion.
- Prototyping can be of two types:
  - Throwaway prototyping (Scaffolding)
  - Evolutionary prototyping

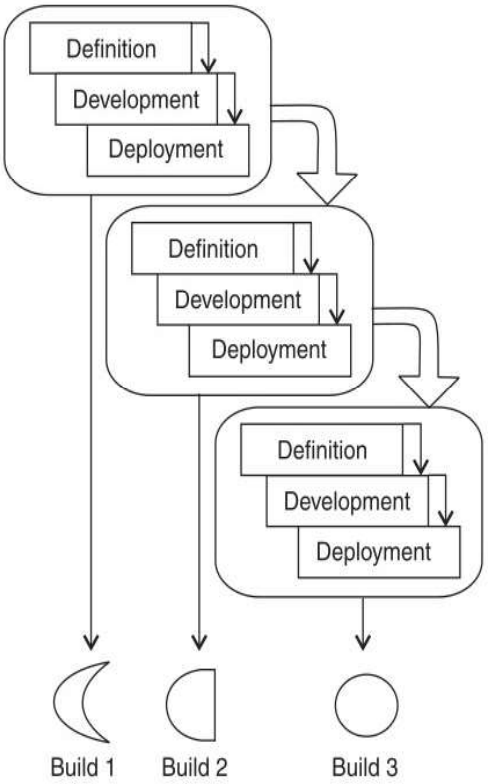
Throwaway Prototyping

- In this model an initial version of the software is developed to get detailed requirements.
- This version is thrown away and second version is developed using waterfall model.



Evolutionary Prototyping

- In this model the initial version or prototype is not thrown away.
- Changes are made to this version to get the final software product.



Advantages of Prototyping

- Very good for eliciting requirements clearly
- Helps gain user confidence
- Lower development costs due to correct specifications for requirements reducing errors
- Helps with user training even before delivery of final product
- Test cases for prototyping can be used for the final software product.

Disadvantages of Prototyping [1/2]

- Costly model
- Poor documentation due to changing requirements
- Too many variations in requirements
- Customer may expect the product to be delivered sooner after seeing early prototype

Disadvantages of Prototyping [2/2]

- Sub optimal solutions in a hurry to build prototype
- Customer may lose interest in product after viewing initial version
- Uncertainty in number of iterations
- Increased complexity of system

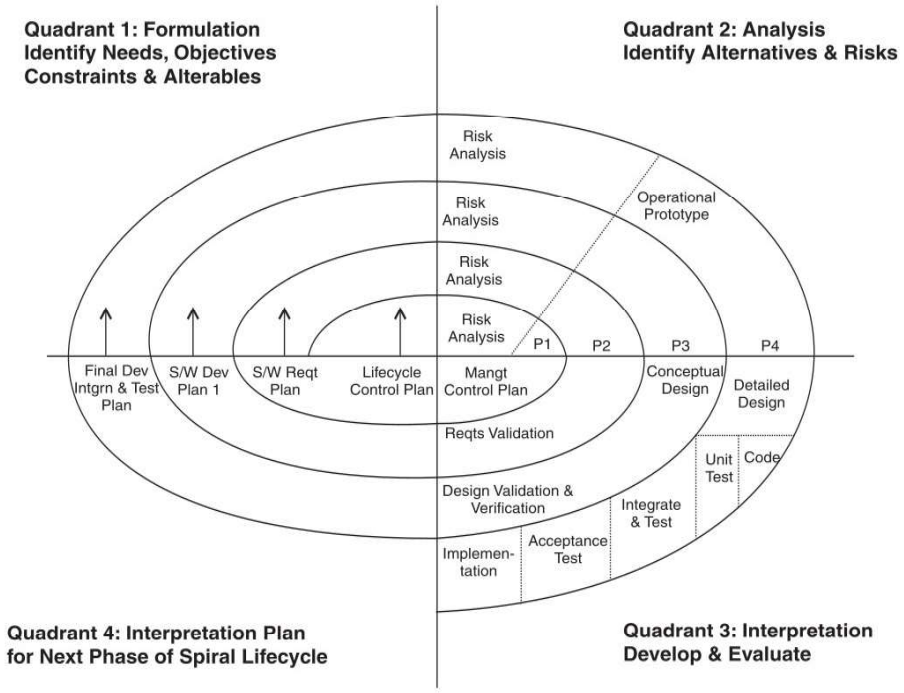
SDLC Models

Spiral Model

Spiral Model

- The Spiral Model integrates the characteristics of the waterfall model, the incremental implementation, and the evolutionary prototyping approach.
- It is also called the Meta Model.

Spiral Model



Features of Spiral Model

- No fixed phases.
- Each quadrant of the spiral corresponds to a set of activities for all phases -
  - Formulation
  - Analysis
  - Interpretation
  - Plan next phases
- Risk assessment is an important activity

Advantages of Spiral Model

- Software produced early
- Suitable for high risk projects
- Flexibility in requirements
- Good for large and complex projects
- Good for customer satisfaction
- Strong approval and documentation control

Disadvantages of Spiral Model [1/2]

- Expensive for small projects
- More complex
- Requires high expertise
- Difficult in time management
- Spirals may go on indefinitely

Disadvantages of Spiral Model [2/2]

- End of project may not be known early
- Not suitable for low risk projects
- Hard to define objectives and milestones
- Requires excessive documentation due to large number of intermediate stages

SDLC Models

Software Reuse

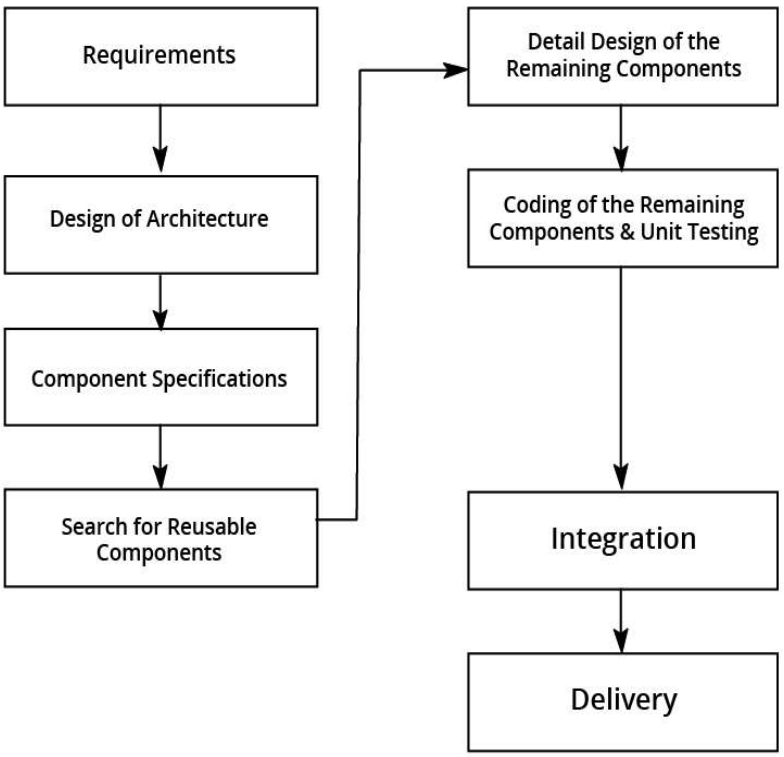
Software reuse

- Software firms develop libraries of generic reusable software components that can be assembled to create new software products.
- Example for the Hospitals

Software Reuse

- What can be reused ?
  - The whole application system.
  - Major subsystems of an application.
  - Modules or objects.
  - Functions.

Software reuse- process



Advantages of Software Reuse

- Increased system reliability.
- Reduced project risk due to less uncertain cost estimates.
- Effective use of specialists in developing generic components rather than in a wide number of projects.
- Enforcement of organizational standards in reusable components, such as common user interface and error handling procedures.
- Reduction of software development time.

Guidelines to make a reusable software component

- The software component should be generalized by having general names, operations and exceptions.
- Complete documentation should be created for a reusable component.
- Test cases should be made available and used while integrating it with the remaining developed components.
- PORTABILITY & CUSTOMIZATION