SOFTWARE PROCESS MODELING

Software Development Life Cycles

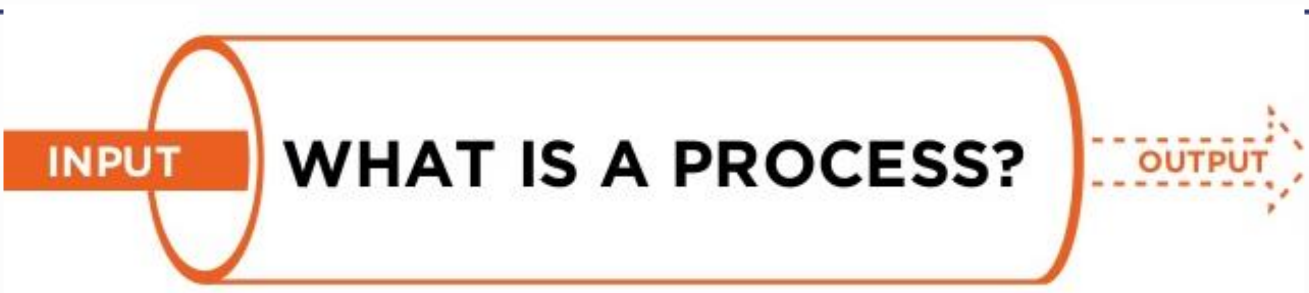
Group and Topic Allocation- Assignments- 2024

- Group registration link will be published on the first-year supportive page.
- Students should form a group with five members and the Group leader should register a group.
- Once students registered groups, Topic allocation will be done randomly
- Then, Topic allocation will be published on the first-year supportive page.

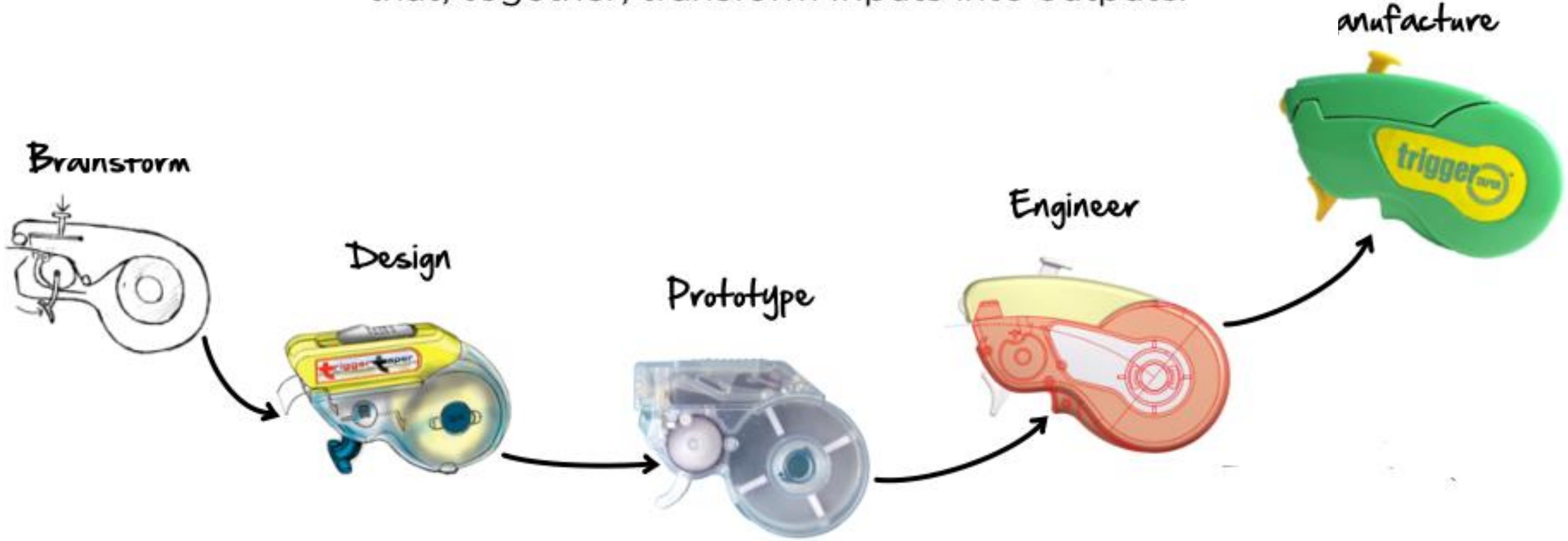
Session Outcomes

- What is a
  - Software Process
  - SDLC
  - Life Cycle Model
- Life Cycle Models
  - Waterfall Model
  - Prototyping Model
  - Incremental Model
  - Spiral Model
- Comparison and Selection

What is an Engineering Process?

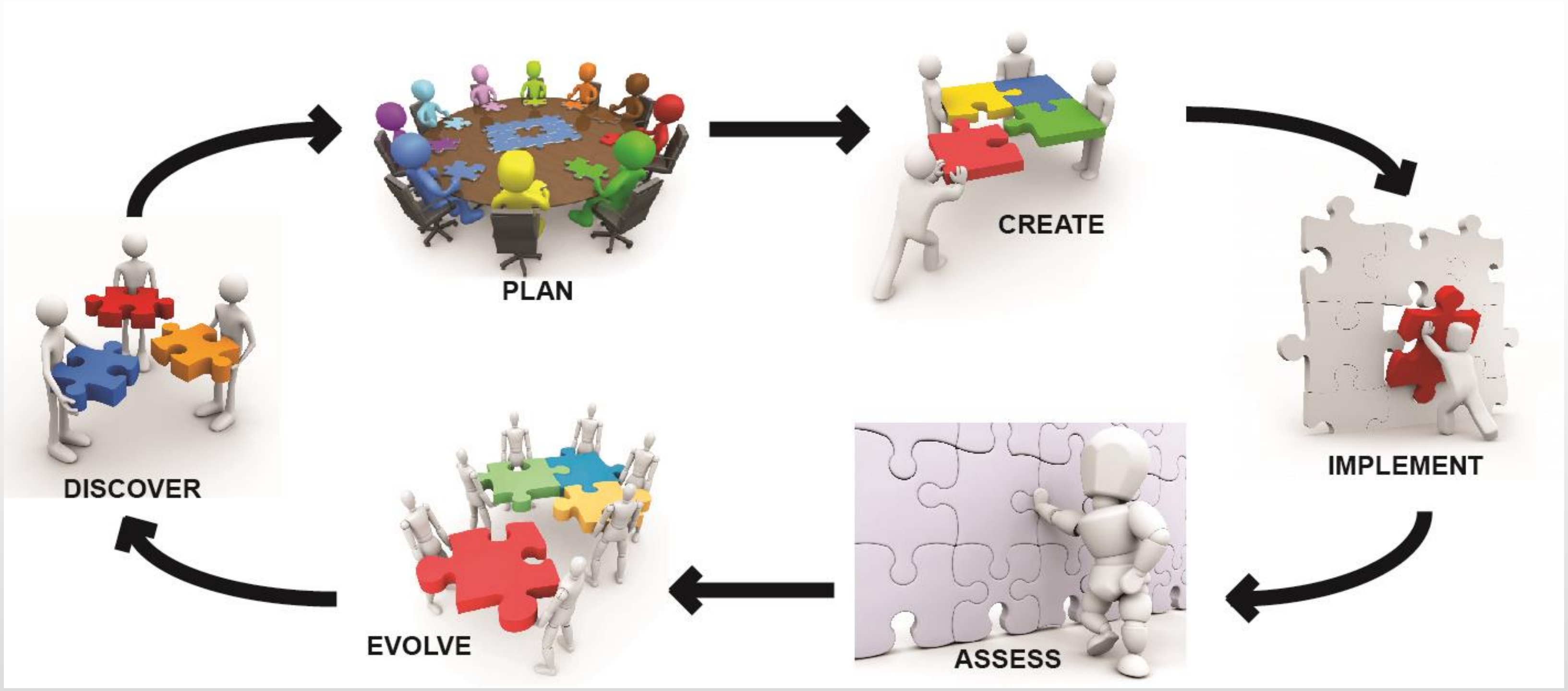


In engineering a process is a set of interrelated tasks that, together, transform inputs into outputs.



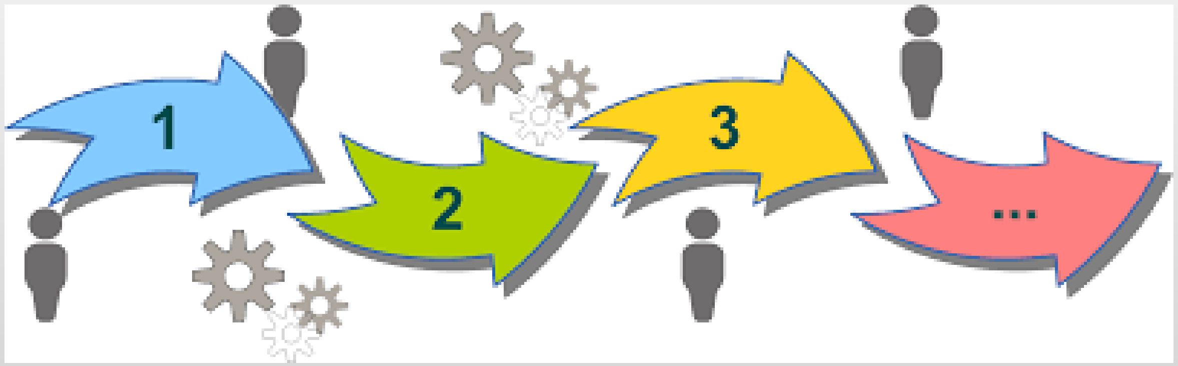
What is a Software Development Process?

- A set of Activities and Associated Results that produce a Software.



Fundamental Process Activities

1. Software Specification
2. Software Development
3. Software Validation
4. Software Evolution



1. Software Specification

- Specification involves clearly documenting the expectations on the system to be built
- Lays out requirements, and may include written and diagrammatic description of the services that the future system must provide.

2. Software Development

- Software development involves designing and implementing the system according to the software specification

3. Software Validation

- Software validation involves checking and verifying whether the system fulfills the requirements.

4. Software Evolution

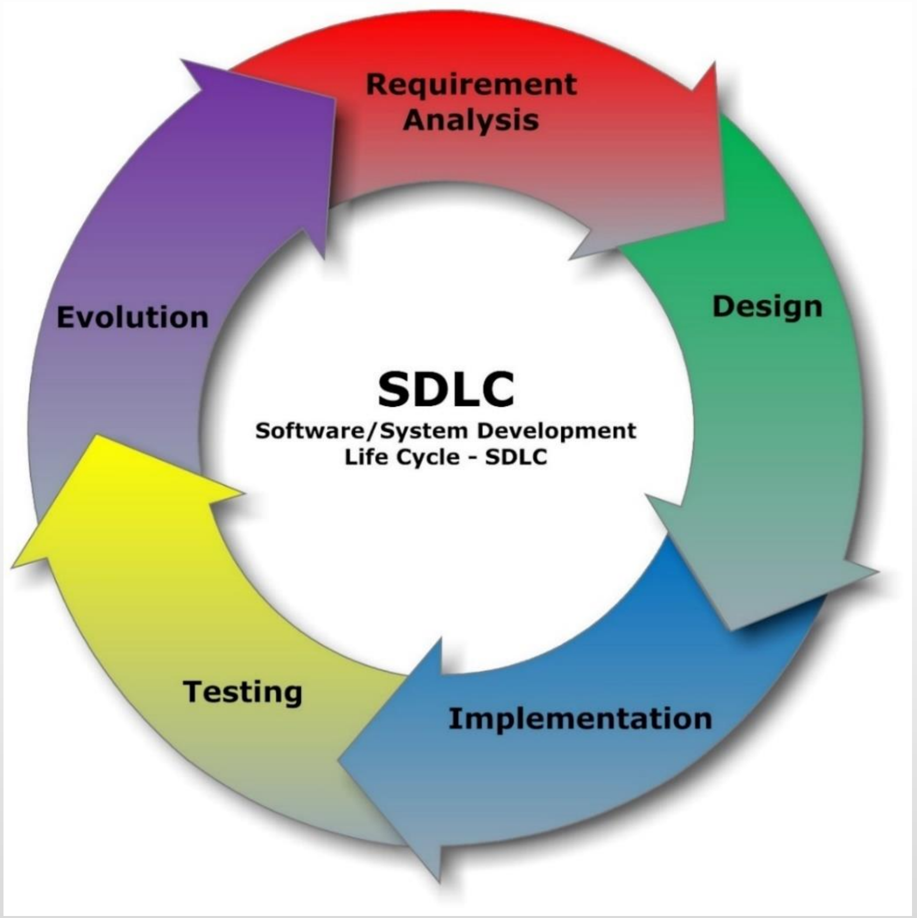
- Software needs to be changed and upgraded with time.

Software Development Evolution
Waterfall
Agile
DevOps
www.heliossolutions.in

SLIIT Discover Your Future

What is a SDLC?

- The Software Development Life Cycle (SDLC) is a framework that defines activities performed throughout the software development process



Life Cycle Model (Process Model)

- A software life cycle (process) model:
  - is a descriptive and diagrammatic model of the life cycle of a software product;
  - identifies all the activities and phases necessary for software development;
  - establishes a precedence ordering among the different activities.
- Life cycle models encourage systematic and disciplined software development.

Life Cycle Models

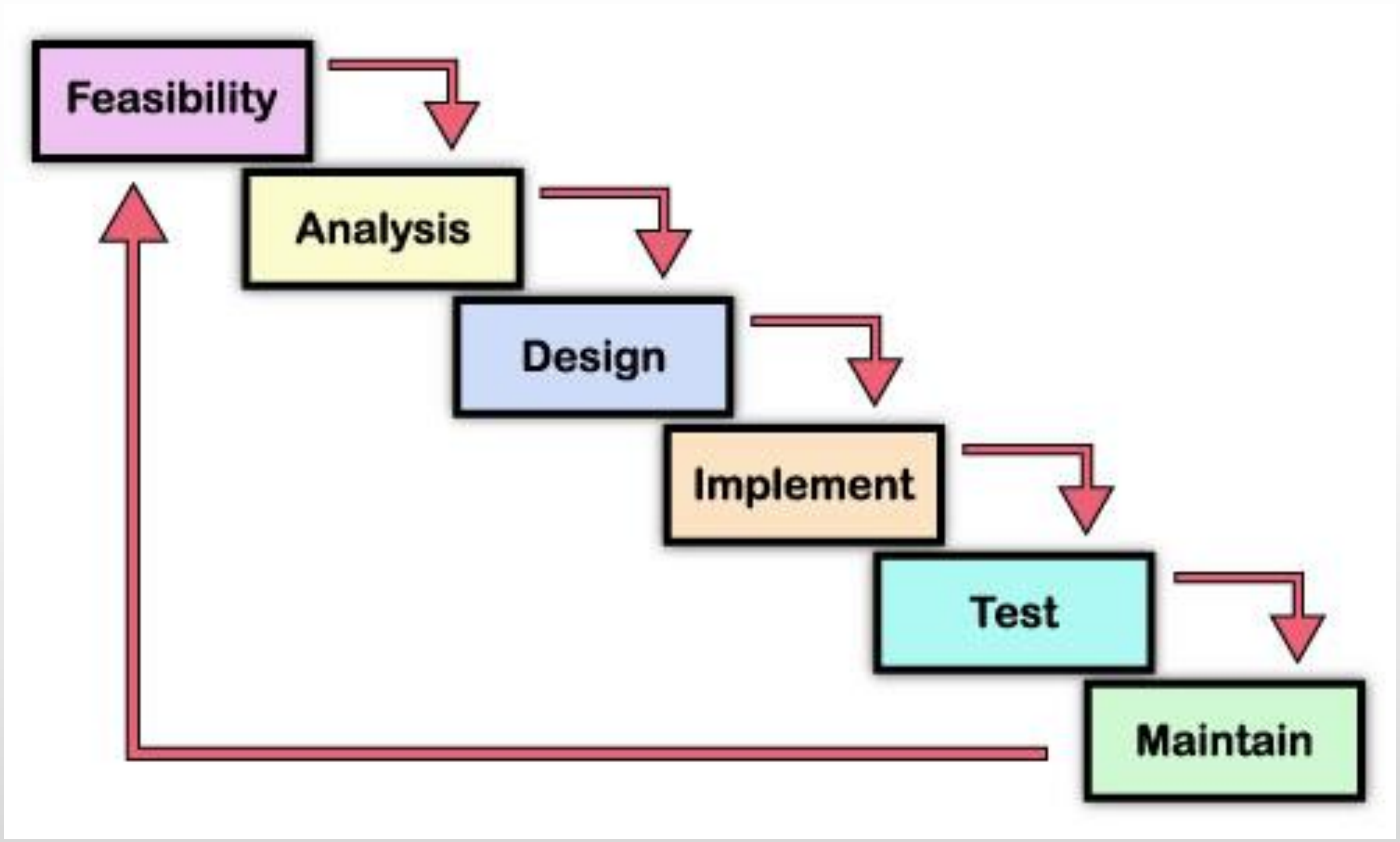
- Traditional Approaches
  1. Waterfall Model
  2. Incremental Model
  3. Prototyping Model
  4. Spiral Model
  5. Unified Process
- Modern Approaches
  - Agile Methods (XP, Scrum)
  - Secure Software Development

Life Cycle Models

1

Waterfall Model

Waterfall Model



Waterfall model

- Waterfall model is the most well known SDLC.
- It is very simple to understand and use.
- Each phase begins only after the previous phase is over.
- Also called **Linear Model**
- A document driven process
- This model specifies what the system is supposed to do (i.e. define the requirements) before building the system (i.e. designing)

1. Feasibility Study

- This is the first phase of any SDLC model.
- The project objective is determined during this phase.
- The client and company developing the software decide if they should ;
    - Keep the existing system as is, or
    - Build a new software

Why do a feasibility study?

- To provide management with enough information to know:
  - Whether the project can be done
  - Whether the final product will benefit its users
  - What the alternative solutions are
  - Whether there is a preferred alternative

Example – Library System

- What are the feasibility criteria for a ‘Library System’?

Book ID	Book5			
Book Name	IBM			
[illegible]	java			
[illegible]	ravishastri			
[illegible]	8			

	BookID	BookName	Copies	Subject
	101	bibliography	0	biology
	Book1	Bible	1	chemistry
	Book2	JAva	0	jk
	Book3	OOPS	5	C++
	Book4	C Programming	8	C
▸	Book5	IBM	1	java
*				

2. The Requirements Phase

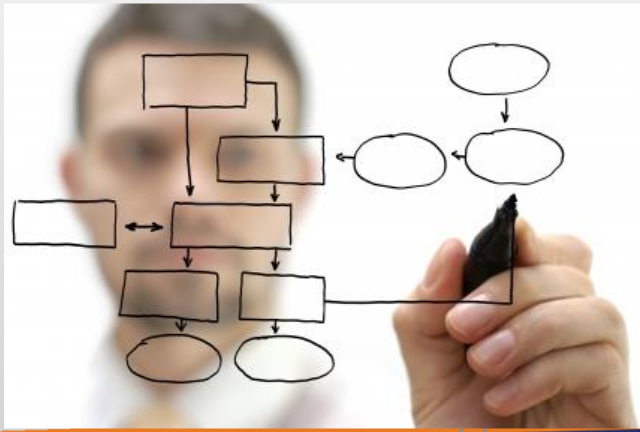
- Aim: to understand the customer's requirements:
- A customer may be a single person, a group, a department, or an entire organization:
- This phase involves two distinct activities:
  1. **Requirements Gathering and Requirements Analysis**
  2. **Requirements Specification**

2a.1 Requirement Gathering

- The goal of this phase is for the stakeholders to find out the ‘what to be done’.
- Questions answered during this phase include:
    - Who will use the system?
    - How will they use the system?
    - What will be the input for the system?
    - What will be the output from the system?
- Requirement Gathering involve collecting information through meetings, interviews and discussions

2a.2 Requirements Analysis

- **Aim:** To understand exactly what the customer needs.. which may not be what they ask for:
    - data to be **input** to the system;
    - **processing** to be performed on these data;
    - data to be **output** from the system;
    - **characteristics** of the system as a whole;
    - **constraints** on the system/project.
- WHAT, not HOW!



2b Requirements Specification

- Requirements are documented in a Software Requirements Specification (SRS).
- The SRS forms the basis of a legal contract with the customer:
- Software Engineers who specialize in requirements gathering, analysis, and specification are called (Systems/ Business / Requirement) Analysts.

3. Design

- Architects and Designers craft a high-level and low level design of the software.
  - Architectural Design
  - Low level Design
- Decisions are made about hardware, software and the system architecture.
- A design specification document (DSD) records this information.

4. Development

- On receiving system design documents, the work is divided in modules.
- A set of developers code the software as per the established design specification, using a chosen programming language
- Programmers carry out some program testing to discover faults in the program and remove these faults in the debugging process

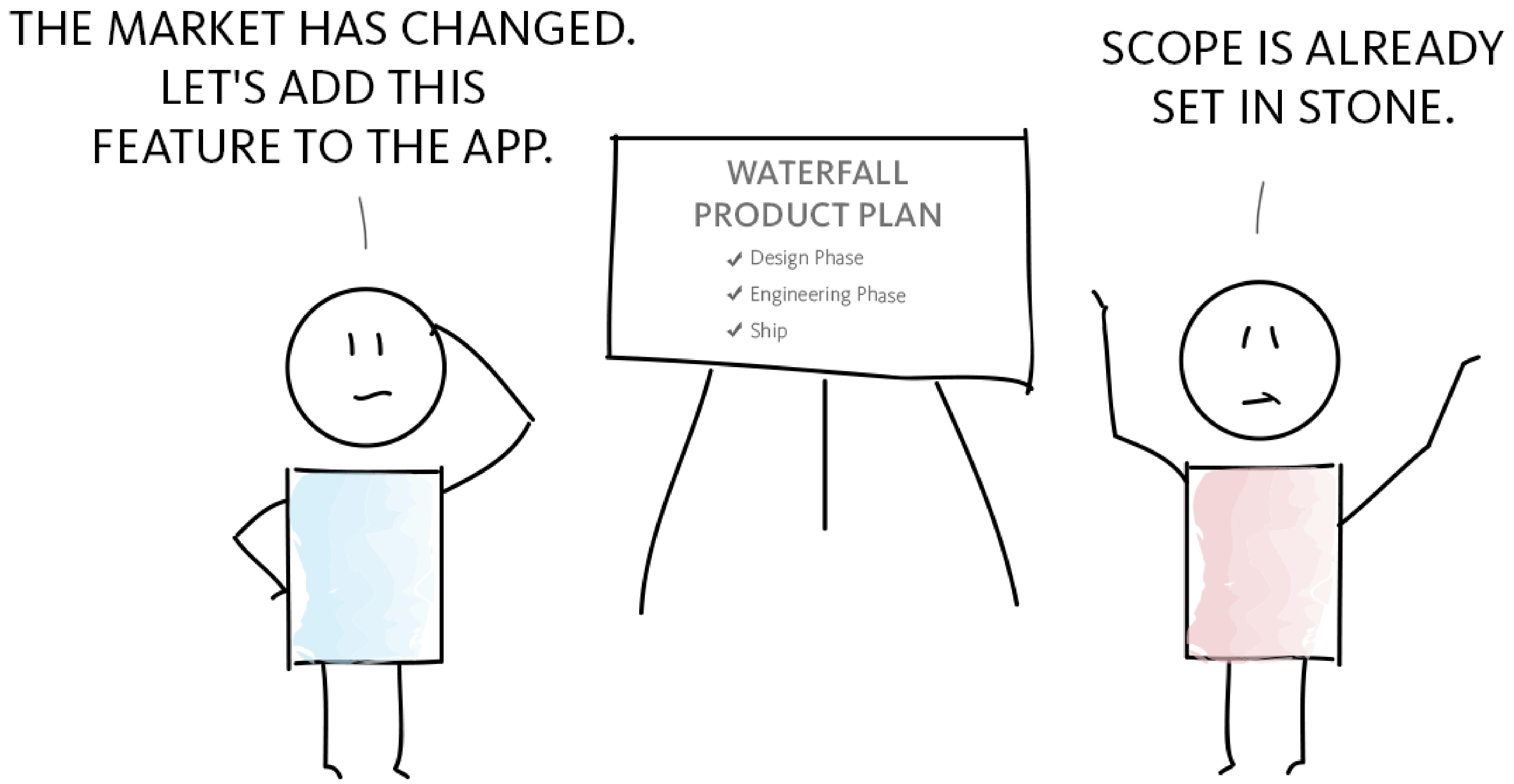
5. Testing

- The testing phase ensures that the software requirements are in place and that the software works as expected.
- When a defect is identified, testers inform the developers.
- If the defect is valid, developers resolve it and create a new version of the software which then repeats the testing phase.
- The cycle continues until all defects are mitigated and the software is ready for deployment into the production environment.

6. Deployment and Maintenance.

- Once the software is error free, it is deployed into the operating environment.
- While the customers are using the software, any issues will be brought to the attention of the maintenance team
- They work to resolve them immediately.

Waterfall in practise


THE MARKET HAS CHANGED. LET'S ADD THIS FEATURE TO THE APP.
WATERFALL PRODUCT PLAN
✓ Design Phase
✓ Engineering Phase
✓ Ship
SCOPE IS ALREADY SET IN STONE.

SLIIT
Discover Your Future

Waterfall Model - Strengths

- Simple and easy to manage– each phase has specific deliverables.
- Milestones are better understood
- Sets requirements stability
- Works well for smaller projects where requirements are very well understood.
- A schedule can be set with deadlines.

Waterfall Model - Weaknesses

- No working software is produced until end.
- High uncertainty.
- Delays discovery of serious errors.
- After requirements phase, there is no formal way to make changes to the requirements.
- Not a good model for
  - complex projects
  - projects where requirements are at a moderate to high risk of changing

SLIIT Discover Your Future

When to use Waterfall Model

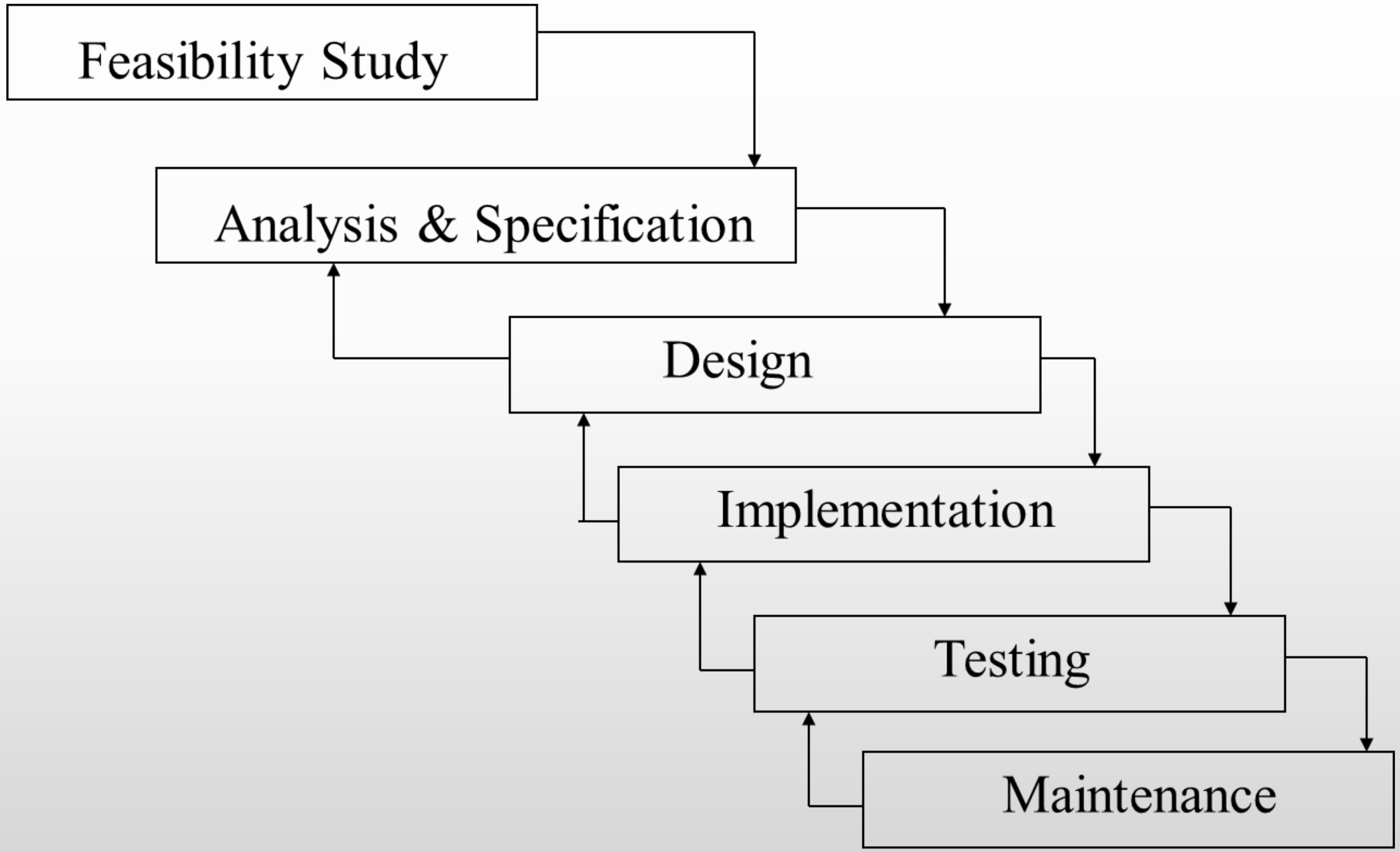
- Software requirements clearly defined and known
- Product definition is stable
- New version of the existing software system is created
- Software development technologies and tools are well known
- Ample resources with required expertise are available

2
Iterative Model

Iterative Waterfall Model

- The classical waterfall model is idealistic:
  - It assumes that no defects are introduced during any of the development phases.
- In practice, defects are introduced during every phase of the software life cycle:
  - Hence feedback paths must be added to the classical waterfall model.
- The resulting Iterative Waterfall Model is one of the most widely used process models….

Iterative Waterfall Model



Iterative Waterfall Model

- Strengths
  - Defects are detected and fixed early through the feedback path
- Weaknesses
  - Limited customer interactions
  - Difficult to incorporate change requests

Discover Your Future

3
Incremental Model

new.png

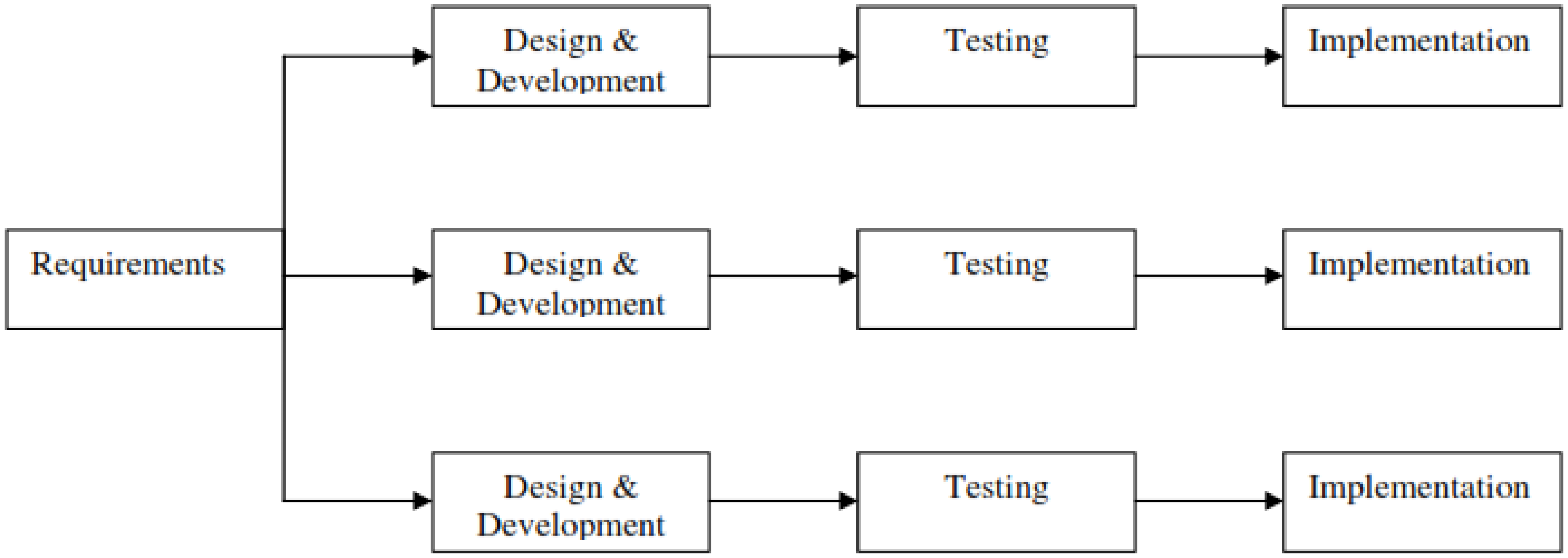
What is the OS you work on?

Windows1
1985
Windows 3.1
1992
Windows 95
1995
Windows XP
2001
Windows
Vista 2006
Windows 7
2009
Windows 8
2012
Windows 10
2015

Incremental model

- Incremental model is an evolution of waterfall model.
  - The product is designed, implemented, integrated and tested as a series of incremental builds.
- The incremental model prioritizes requirements of the system and then implements them in groups.
- It is the process of constructing a partial implementation of a total system and slowly adding increased functionality or performance.

Incremental model



Incremental model - Strengths

- Generates working software quickly and early during the software life cycle.
- More flexible - less costly to change scope and requirements.
- Easier to test and debug.
- Easier to manage risk.
- Lowers initial delivery cost.
- Less stress for the development team.

Incremental model - Weaknesses

- Requires good planning and design
- Each phase of an iteration is rigid and do not overlap each other.
- Demarcation of increments can be difficult in a practical application.
- Total cost of the system might not be lower.
- Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.

SLIIT
Discover Your Future

When to use Incremental model

- On projects which have lengthy development schedules
- A need to get basic functionality to the market early
- Most of the requirements are known up-front but are expected to evolve over time
- On a project with new technology

Mini-Case 1

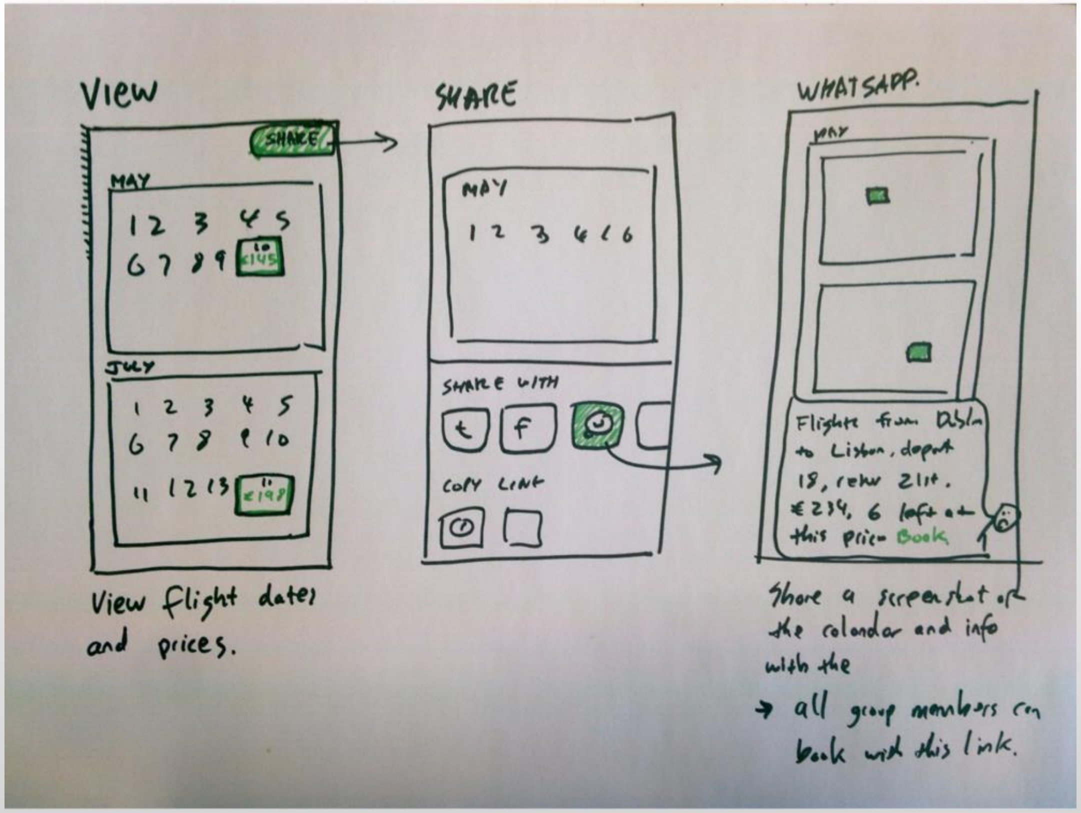
- The project is to develop an inventory control system for a new super market in town. It should have all the regular functionalities such as adding new stocks, getting reports such as inventory re-order report and daily sales report etc. A project team has previous experience developing inventory systems for other clients.

4
Prototyping Model

Farming Software

agrivi
WHY AGRIVI?
FORGET ABOUT PEN AND PAPER
INCREASE YOUR FARM PRODUCTIVITY WITH SMART DECISIONS!
SIMPLE, EFFICIENT AND AFFORDABLE FARM MANAGEMENT SOFTWARE
All in one place
Farm data management
Knowledge base
Farm analytics
TRUSTED BY THOUSANDS OF FARMERS WORLDWIDE
TOMATO
OÓTA Consulting Ltd (Agrivi, Reseller Partner), Lusaka, Zambia
e-mail: kahootab@gmail.com, phone: +260 977 611 779
Google Play
App Store

Explain Whatsapp (before implementation) to a traditional farmer



View flight dates and prices.

Flights from Dublin to Lisbon, depart 18, return 21st. €234, 6 left at this price Book

Share a screenshot of the calendar and info with the

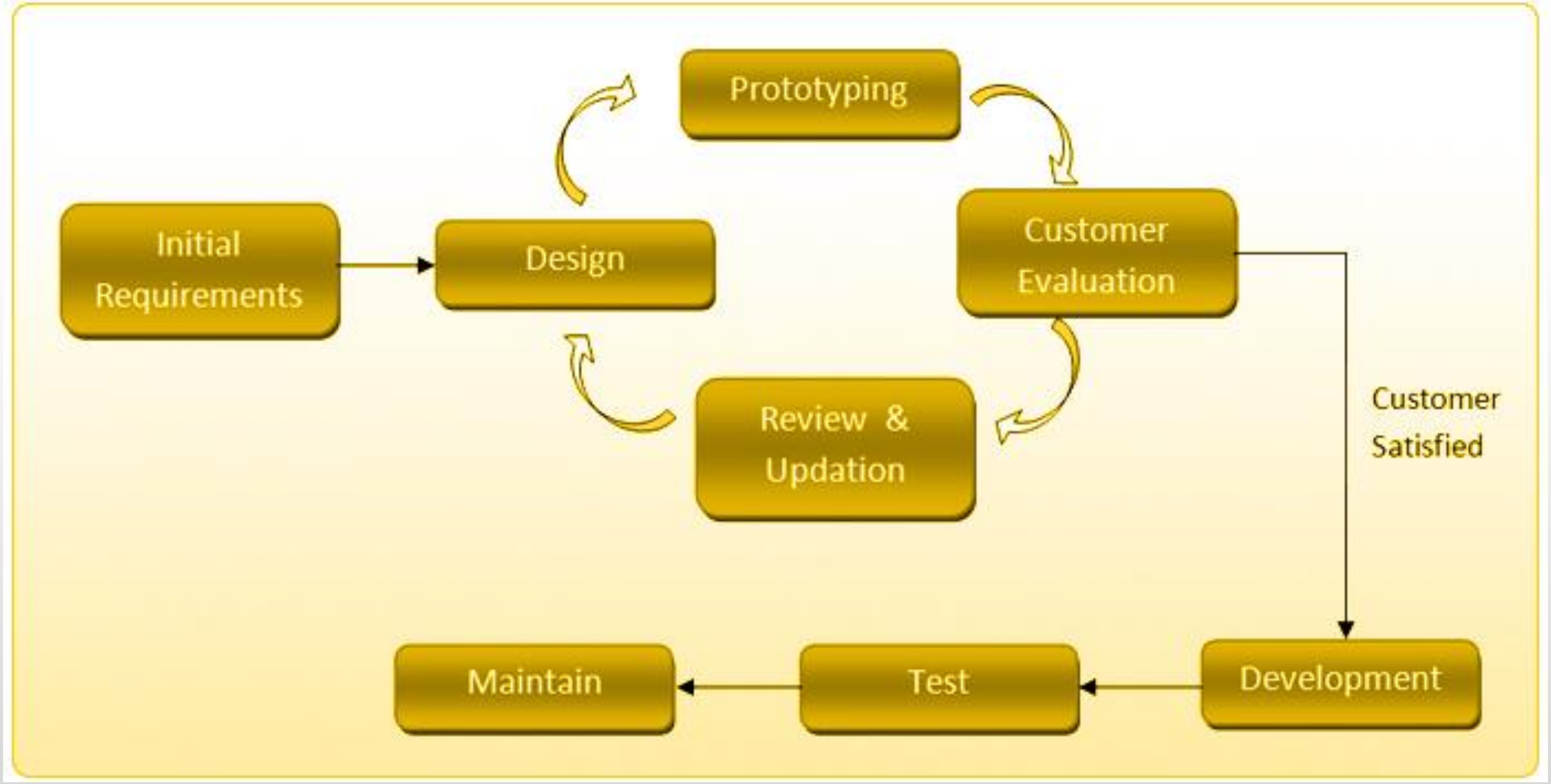
→ all group members can book with this link.

Prototype Model

- Instead of freezing the requirements before coding or design, a prototype is built to clearly understand the requirements.
- This prototype is built based on the current requirements.
- Through examining this prototype, the client gets a better understanding of the features of the final product.
- Requirements may be changed with the client feedback on the prototype.

SLIIT
Discover Your Future

Prototype Model



Prototyping Model - Strengths

- Ability to clarify user’s expectations for the system to be developed
- Prototype stimulates awareness of additional needed functionality
- Better user satisfaction
- Early user feedback

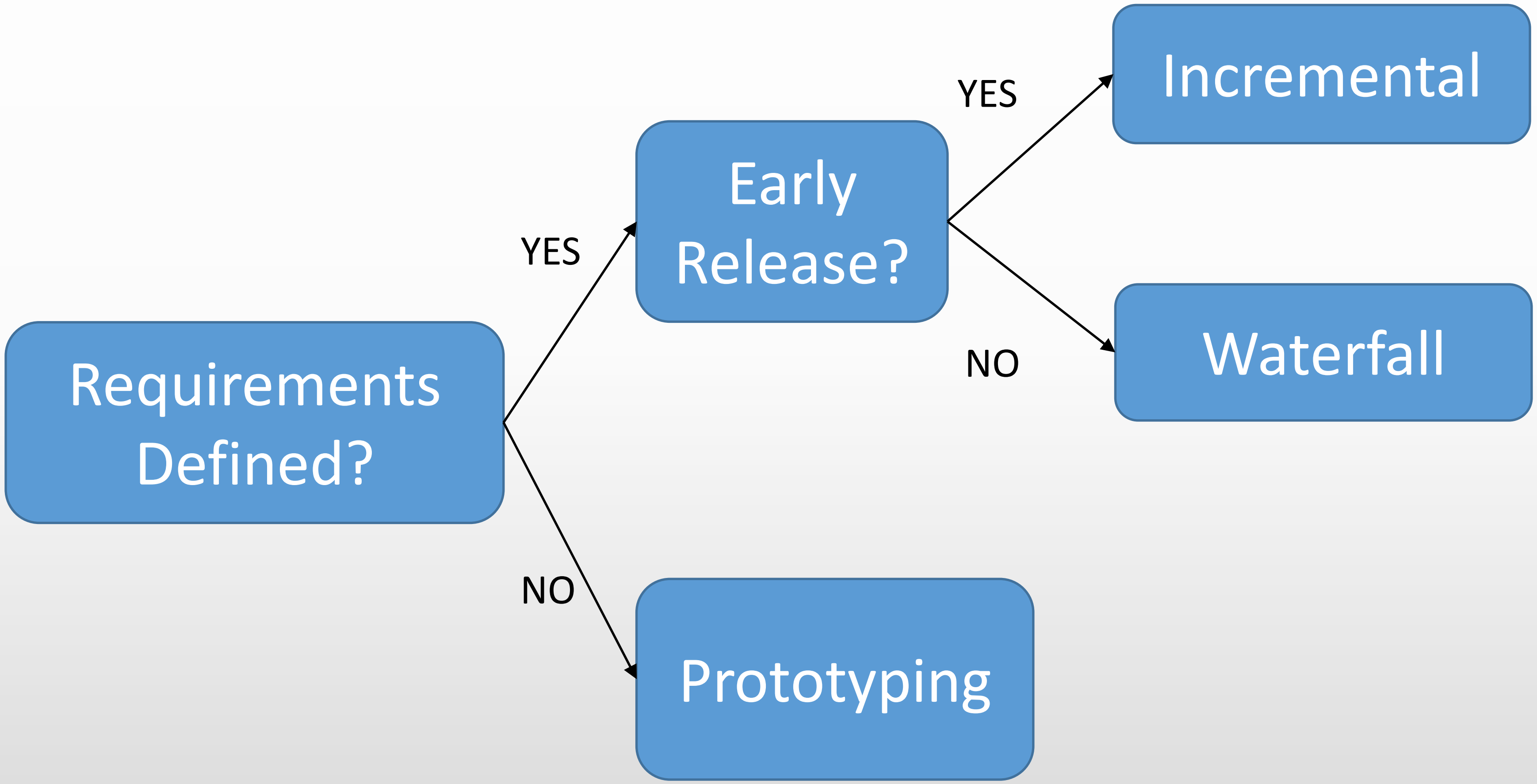
Prototyping Model - Weaknesses

- Scope Creep - The system scope may expand beyond original plans.
- Overall maintainability may be overlooked.
- The customer may want the prototype be delivered.
- Process may continue forever

When to Use Prototyping

- Requirements are unstable or have to be clarified
- Many user interfaces
- New technology
- New, original development
- Developers are not familiar with the technical and development tools

Selection of Approach



Question

- A company is developing an advanced version of their current software available in the market, what model approach would they prefer ?

a) Waterfall
b) Incremental
c) Prototyping

Question

Selection of a model is based on…?

a) Requirements
b) Development team
c) Users
d) Project type and associated risk
e) All of the mentioned

SDLC Model Selection

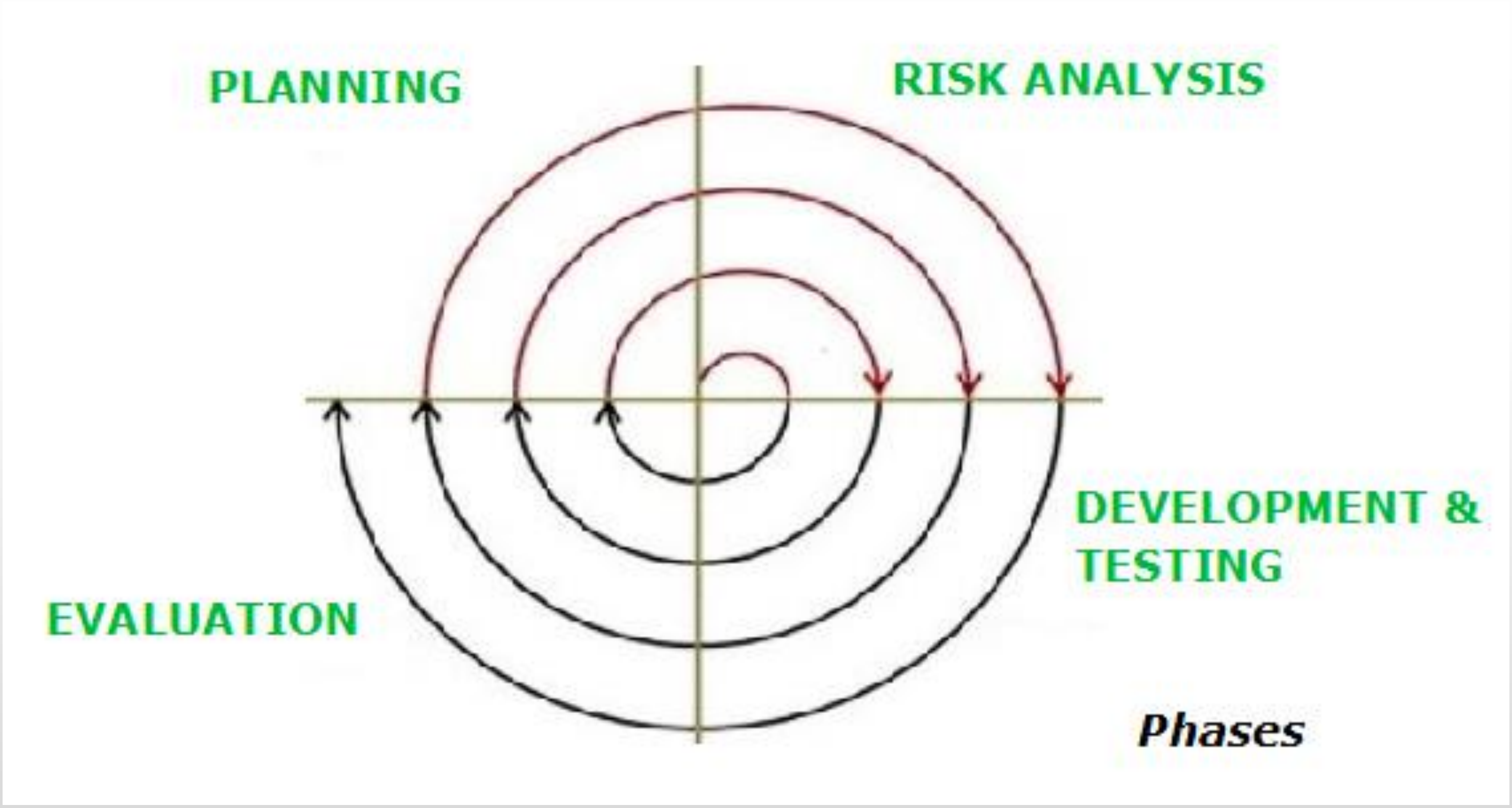
- Is based on
  1. Customer
     a) Budget
     b) Timeline
     c) Existing software/ OSS
     d) Client experience and technical knowledge
     e) Flexibility
  2. Product
     a) Complexity and risk level
     b) New / experienced technology
     c) Availability of tools
     d) Requirements Clarity
  3. Developer experience and technical knowledge ……..

5
Spiral Model

Spiral Model

- Meta Model - combines iterative and prototype development with the systematic, controlled aspects of the waterfall model.
- Allows for incremental releases
- Introduced by Barry Boehm in 1986.
- Allows elements of the product to be added in when they become available or known.
- Emphasises on risk management

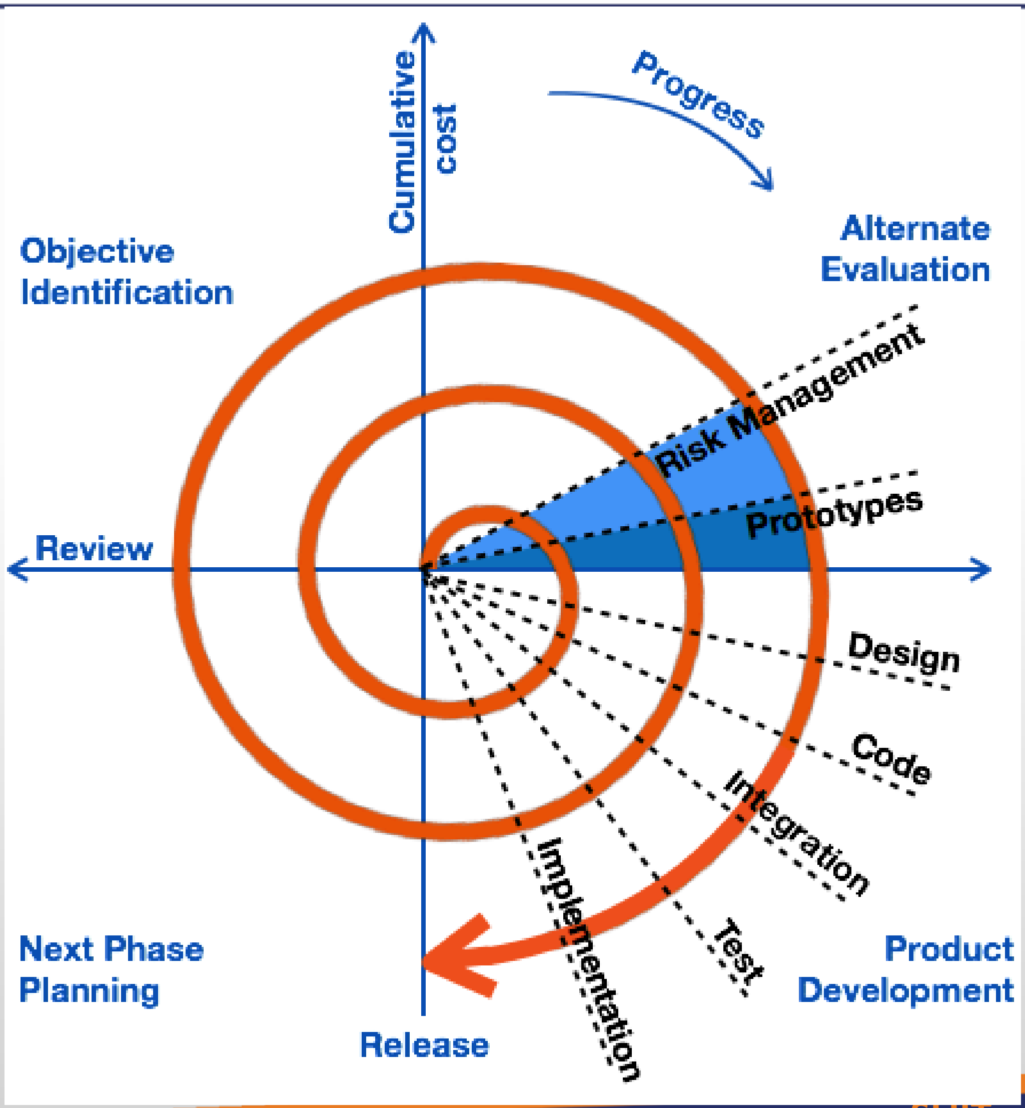
Spiral Model



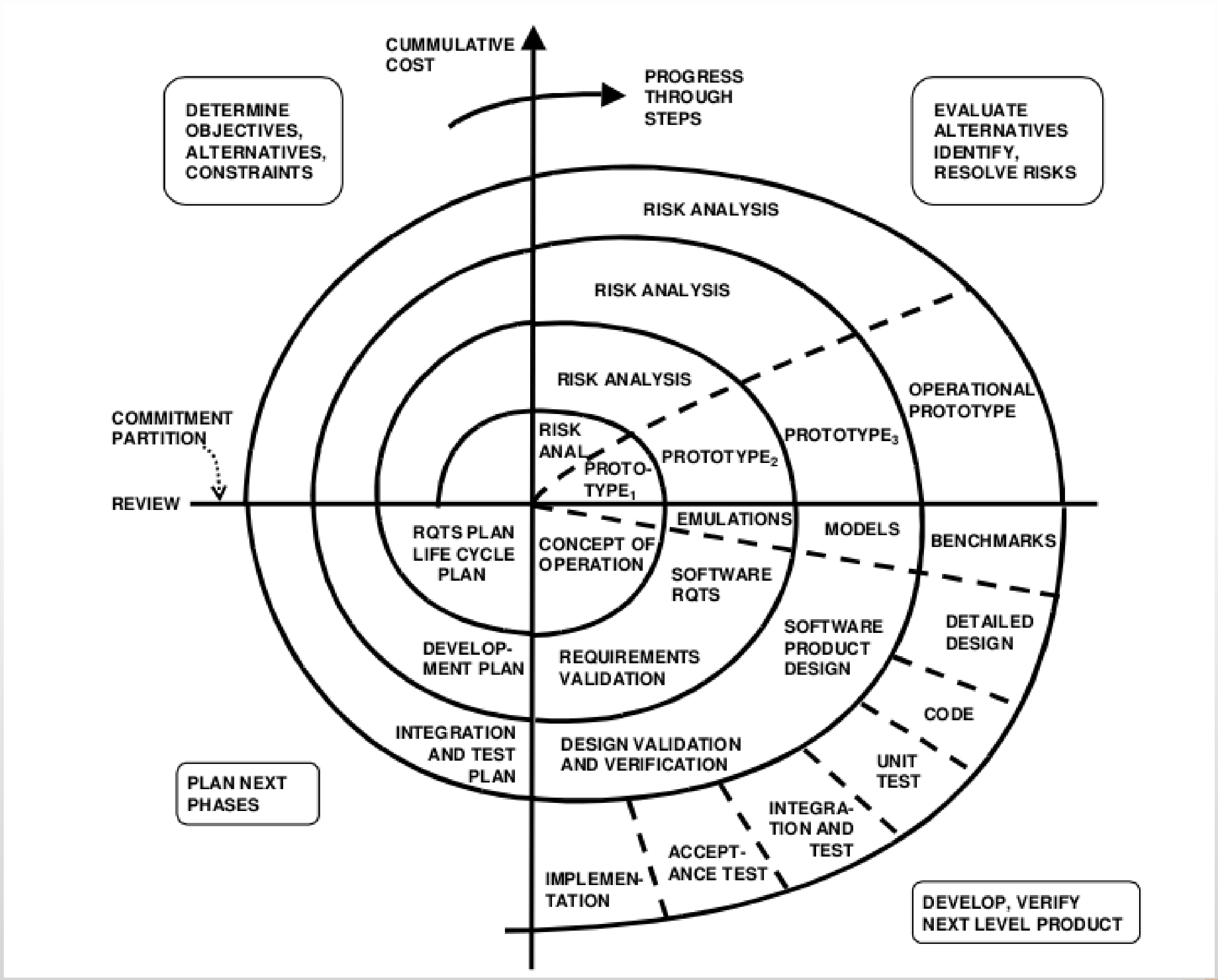
Spiral Model

- Quadrant 1: Planning
  - Determine objectives, alternatives, and constraints.
- Quadrant 2: Risk Analysis
  - Evaluate alternatives, identify, resolve risks.
- Quadrant 3: Development & Test
  - Develop, verify, next-level product.
- Quadrant 4: Evaluation
  - Analyse feedback and plan next phases.

Spiral Model



Detailed Spiral



Question

- Which of the following models will not be able to give the desired outcome if user's participation is not involved?

a) Waterfall
b) Spiral
c) Prototyping

Spiral Model - Strengths

- Focus on risk analysis.
- Good for large and mission critical projects
- A working software is produced early
- The design does not have to be perfect
- Early and frequent feedback from users
- Cumulative costs assessed frequently

SLIIT
Discover Your Future

Spiral Model - Weaknesses

- Can be a costly model to use.
- Risk analysis requires expertise.
- Success is highly dependent on the risk analysis phase.
- Doesn’t work well for smaller projects.

SLIIT
Discover Your Future

When to use Spiral Model

- For medium to high-risk projects
- New technology to be used
- Complex, constantly changing and continuous Requirements
- Significant changes are expected (research and exploration)
- Users are unsure of their needs

Question

Which of the following life cycle model can be chosen if the development team has less experience on similar projects?

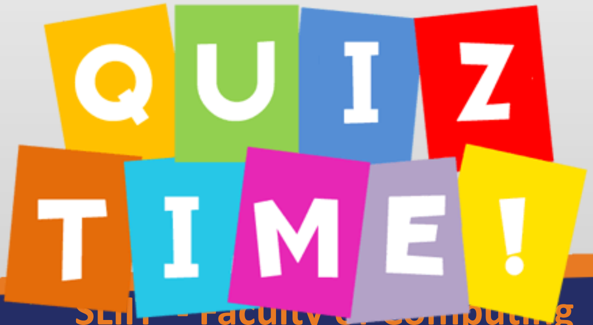
a) Spiral
b) Waterfall
d) Iterative

Activities

Which life cycle is suitable?

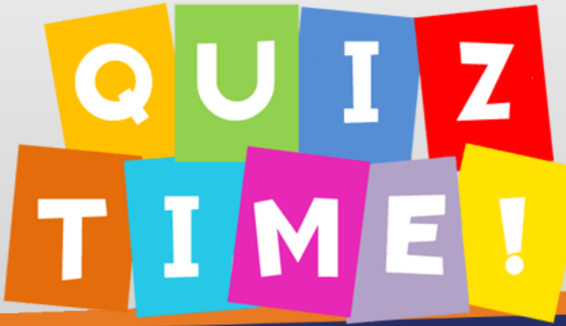
Mini-Case 2

- A library management system is required for a National Library with a number of branches in many cities. System should link to a number of other online libraries, databases, journals and university libraries through web and manage different user subscriptions. The project team has little experience in developing library systems before. However there is not much pressure on time.



Mini-Case 3

- A Project is to develop a complete system for a new bank. System will have many users, interrelationships, and functions. The project has few risks related to requirements definition. These risks needs to be reduced.



	Waterfall	V-Shaped	Evolutionary Prototyping	Spiral	Iterative and Incremental	Agile
Unclear User Requirement	Poor	Poor	Good	Excellent	Good	Excellent
Unfamiliar Technology	Poor	Poor	Excellent	Excellent	Good	Poor
Complex System	Good	Good	Excellent	Excellent	Good	Poor
Reliable system	Good	Good	Poor	Excellent	Good	Good
Short Time Schedule	Poor	Poor	Good	Poor	Excellent	Excellent
Strong Project Management	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent
Cost limitation	Poor	Poor	Poor	Poor	Excellent	Excellent
Skills limitation	Good	Good	Poor	Poor	Good	Excellent
Documentations	Excellent	Excellent	Good	Good	Excellent	Poor
Component reusability	Excellent	Excellent	Poor	Poor	Excellent	Poor

References

- I. Sommerville, Software Engineering, 10th ed., Addison-Wesley, 2011. – Chapter 2
- *Roger Pressman, Software Engineering: A Practitioner’s Approach – Chapter 2*

Next Lecture

Requirements Engineering