CSC 420
Software Engineering

Introduction, SDLC, Software Processes

Amos O. Bajeh, PhD

Room 29, Ground Floor

Dept of Computer Science

University of Ilorin

Course Content

- Introduction: Principles of software engineering
- Software life cycle
- Project management
- Computer based system engineering
- Requirements and Specification: Analysis, definition, specification, algebraic specification and mode-based specification.
- Software Design architecture
- Object-oriented design, Function-oriented design
- Dependable systems.
- Reliability and usability
- Safety-critical consideration

Course Content

- Good programming practices
- Computer Aided Software Engineering (CASE)
- Verification and Validation: validation and testing
- Problems of assessing and quantifying the system reliability
- Test case and test data design
- Management: people and organization issues.
- Cost management
- Quality management
- Process improvement
- Configuration and re-engineering of software

Introduction

- Objectives:

  By the end of this topic, you will be able to:

  - defined software
  - describe the characteristics of a good software
  - describe the SDLC and process models
  - describe the concept of Agile software development
  - compare and contrast different software process models

Introduction

- What is software engineering?

  *The application of a **systematic**, **disciplined**, **quantifiable** approach to the development, operation and maintenance of software* (IEEE)

- Software engineering is a branch of computer science, which uses well-defined engineering concepts required to produce efficient, durable, scalable, in-budget and on-time software products

Introduction

- Why does software needs to be engineered?
  - Large software
  - Cost
  - Dynamic nature
  - Scalability
  - Quality management

Course Content

- Characteristics of good Software

Operational	Transitional	Maintenance
•budget	• portability	•modularity
•usability	• interoperability	•maintainability
•efficiency	• reusability	•flexibility
•correctness	• adaptability	•scalability
•functionality		
•dependability		
•security		
•safety		

Software Development Life Cycle and Process Model

- **SDLC** consist of the stages or activities through which a software product passes through from inception to retirement

- **Activities**:
  - Feasibility study
  - Requirement Analysis and Specification
  - Design
  - Coding and Unit testing
  - Integration and System testing
  - maintenance

- Specification;
- Design;
- Implementation;
- Validation;
- Evolution

Software Development Life Cycle and Process Model

- **Process model** is a descriptive (diagrammatic) representation of the activities, actions and tasks (software life cycle) for the development of high-quality s/w

- It maps the different activities performed on a software product from its inception to retirement .

Process Model

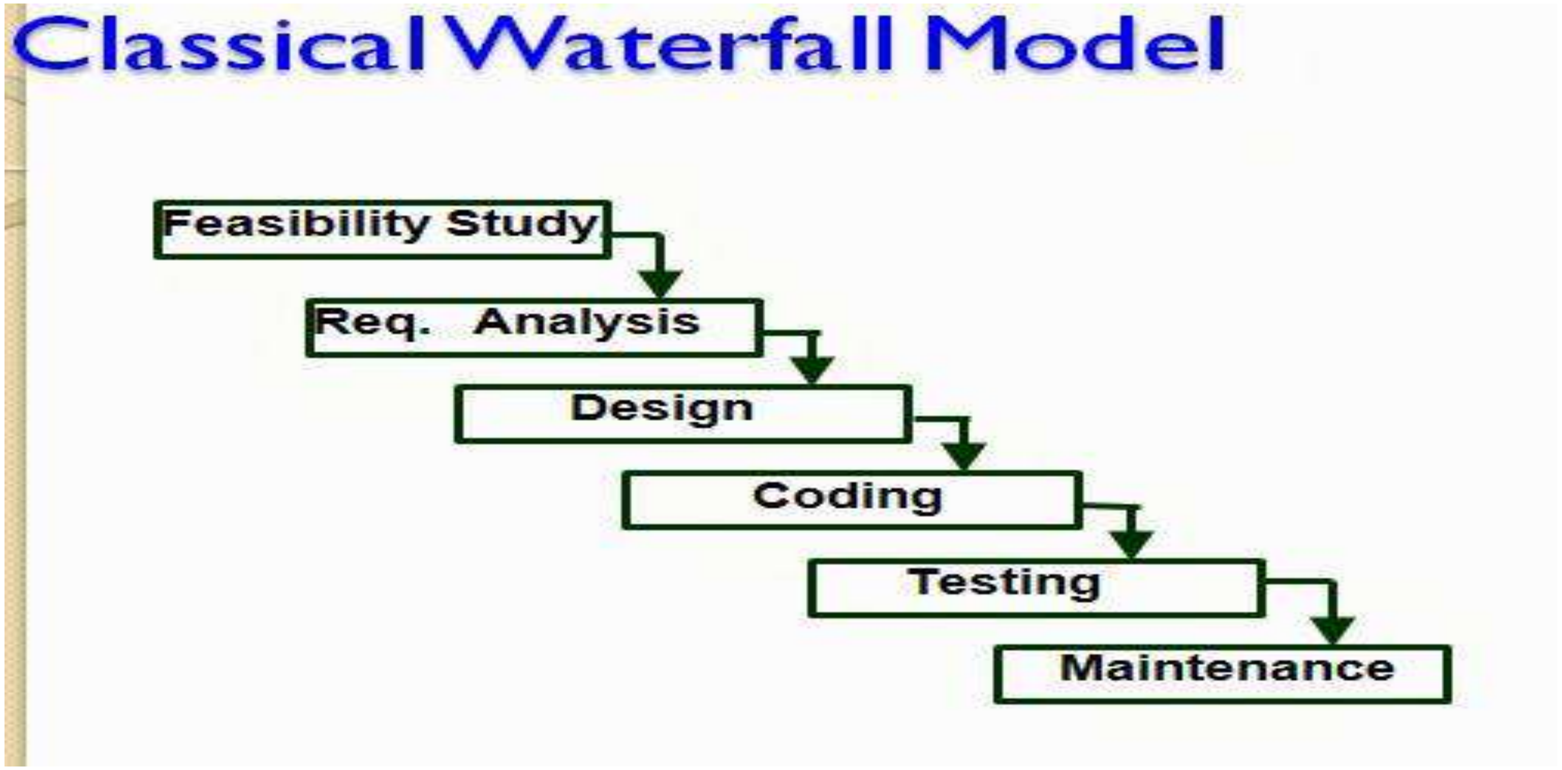
- Categories of process models:
    - Linear
    - Iterative
    - Evolutionary
    - Parallel

Process Model

- Types of process models:
  - Waterfall model
  - V model
  - Prototyping model
  - Evolutionary model
  - Spiral model

Process Model:

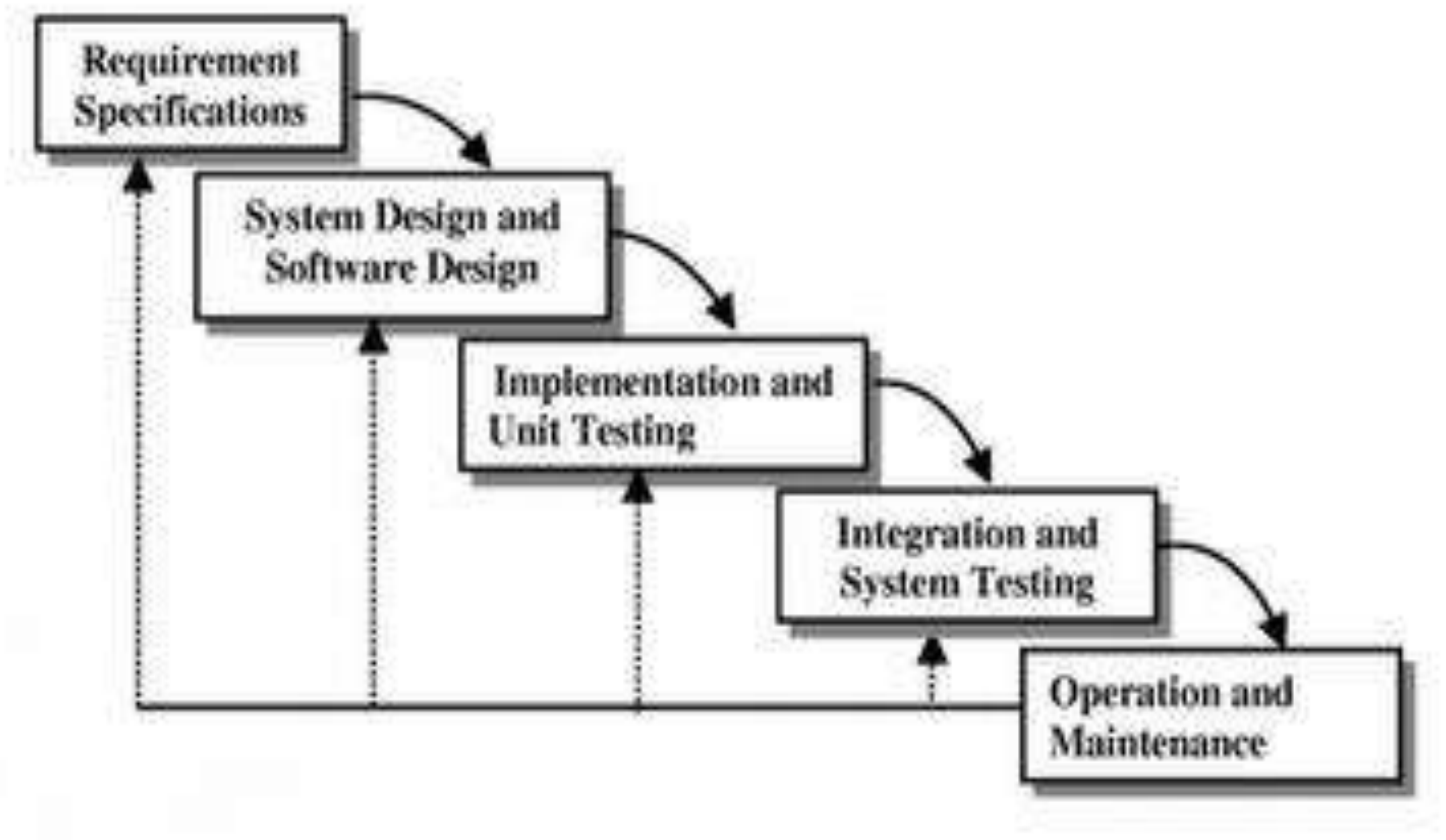
- Classical Waterfall Model



- Shortcoming of the classical waterfall model: it does not fit real life scenario

Process Model:

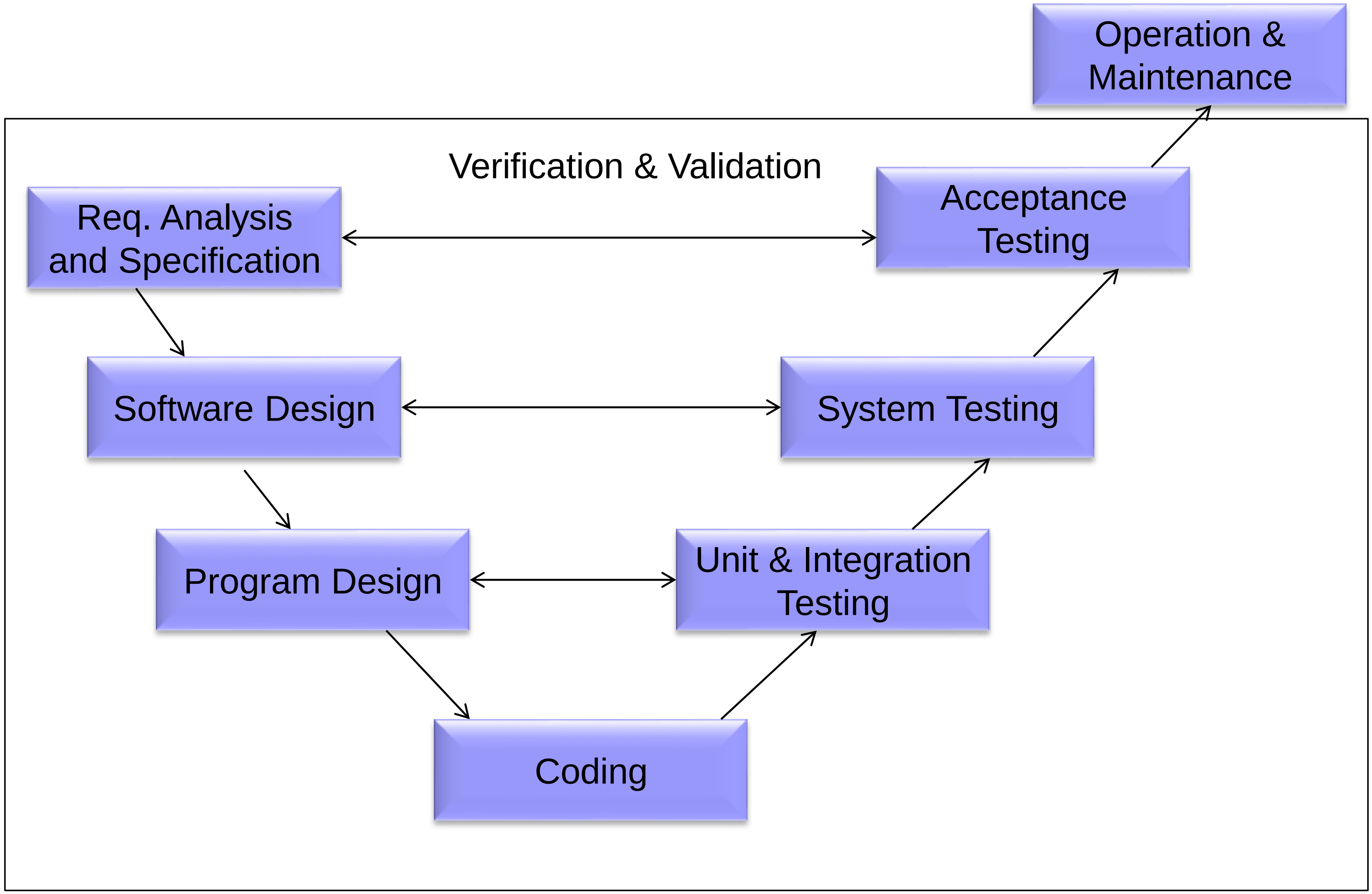
- Iterative Waterfall Model


Requirement Specifications
System Design and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

- Strength
  - ❑ Disciplined approach
  - ❑ Careful checking by the Software Quality Assurance Group at the end of each phase.
  - ❑ Testing in each phase.
  - ❑ Documentation available at the end of each phase.

- Weakness
  - ❑ It is difficult to respond to changing customer requirements - only appropriate when the requirements are well-understood and changes fairly limited
  - ❑ A working version of the program will not be available until late in the project time-span
  - ❑ Feedback from one phase to another might be too late and hence expensive.

Process Model: V model



Process Model: V model

Advantages of V-model:

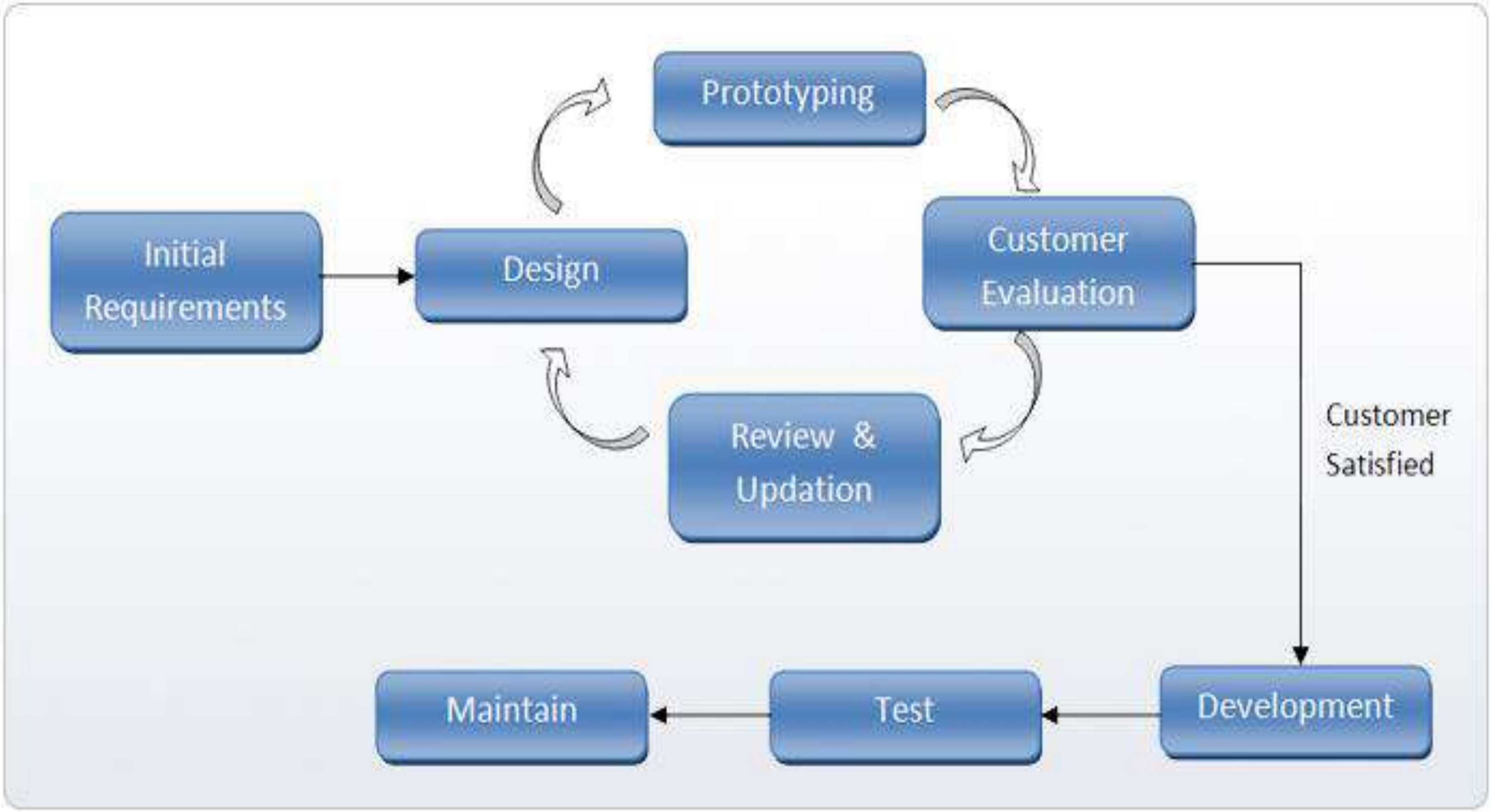
- Simple and easy to use.
- Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
- Proactive defect tracking – that is defects are found at early stage.
- Avoids the downward flow of the defects.
- Works well for small projects where requirements are easily understood.

Disadvantages of V-model:

- Very rigid and least flexible.
- Software is developed during the implementation phase, so no early prototypes of the software are produced.
- If any changes happen in midway, then the test documents along with requirement documents has to be updated.

Process Model

- Prototyping model:



Process Model

- Need for Prototype:
  - To illustrate to customer various aspect of s/w
    - how the user interface might look like
    - how the input data format might be designed
    - how the s/w will produce outputs e.g. messages, reports,
  - It is impossible to get the perfect s/w in the first attempt
  - For critical examination of technical issues by developers
- When is a prototype needed
  - User req. are not complete
  - Technical issues are not clear

Process Model

- Types of Prototype:

  - Throwaway Prototype

    A throwaway prototype is not part of the final product. Throwaway prototypes should:

    - be fast to build;
    - help to clarify requirements and prevent misunderstanding;
    - warn implementers of possible difficulties

  - Evolutionary Prototype

    - Evolutionary prototypes become part of the final product.
    - They are usually written in the final language of the application

  - Incremental Prototype

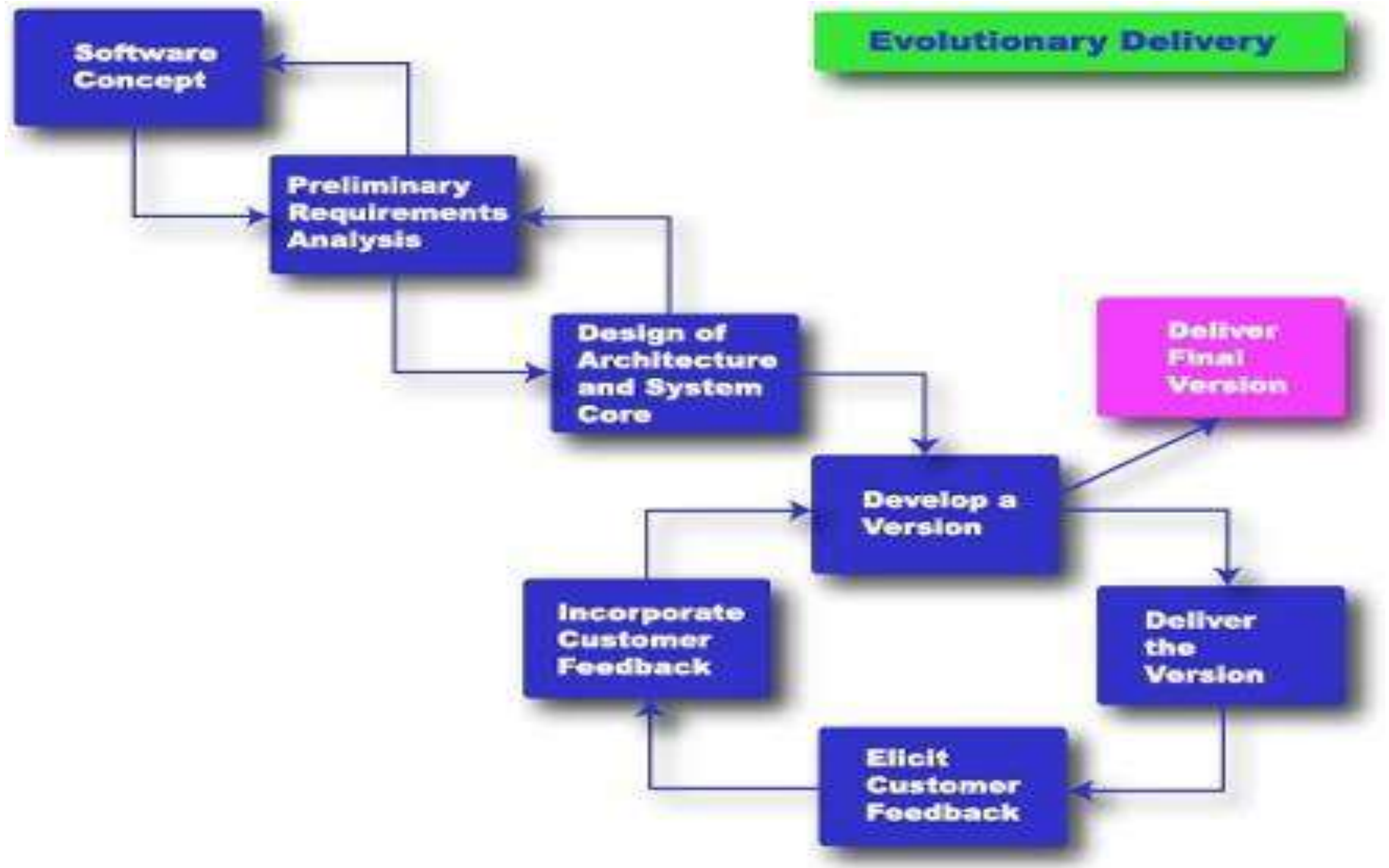
    - the product is developed as a sequence of working components

- Strength
  - ❑ Requirements can be set earlier and more reliably
  - ❑ Customer sees results very quickly.
  - ❑ Customer is educated in what is possible, helping to refine requirements.
  - ❑ Requirements can be communicated more clearly and completely

- Weakness
  - ❑ Requires a rapid prototyping tool and expertise in using it–a cost for the development organisation
  - ❑ Smoke and mirrors - looks like a working version, but it is not.

Process Model

- Evolutionary model:



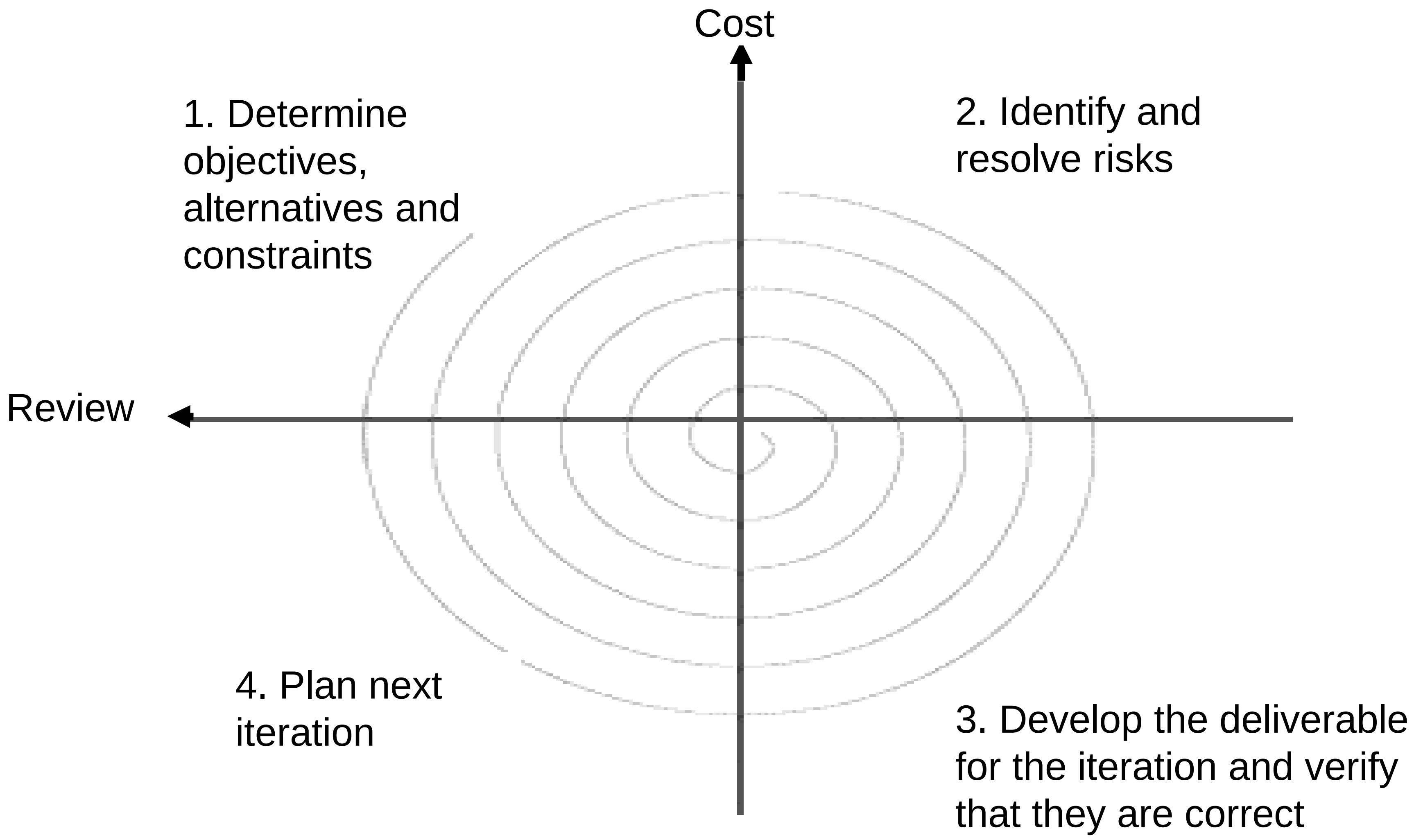
Process Model

- Advantages:
  - ❑User gets a chance to experiment partially developed system
  - ❑Reduce the error because the core modules get tested thoroughly.

- Disadvantages:
  - ❑It is difficult to divide the problem into several versions that would be acceptable to the customer which can be incrementally implemented & delivered.

Process Model

- Spiral model: proposed by Barry Boehm(1986)


Cost
1. Determine objectives, alternatives and constraints
2. Identify and resolve risks
Review
4. Plan next iteration
3. Develop the deliverable for the iteration and verify that they are correct

Process Model: Spiral model (cont’d)

- **First quadrant (Objective Setting)**
  - During the first quadrant, it is needed to identify the objectives of the phase.
  - Identify the risks associated with these objectives.

- **Second Quadrant (Risk Assessment and Reduction)**
  - A detailed analysis is carried out for each identified project risk.
  - Steps are taken to reduce the risks. For example, if there is a risk that the requirements are inappropriate, a prototype system may be developed.

Process Model: Spiral model (cont’d)

- **Third Quadrant (Development and Validation)**
  - Develop and validate the next level of the product after resolving the identified risks.

- **Fourth Quadrant (Review and Planning)**
  - Review the results achieved so far with the customer and plan the next iteration around the spiral.
  - Progressively more complete version of the software gets built with each iteration around the spiral.

Process Model: Spiral model (cont'd)

- When to use the Spiral model:
  - The spiral model is suitable for development of technically challenging software products that are prone to several kinds of risks.
  - However, this model is much more complex than the other models – this is probably a factor deterring its use in ordinary projects

Process Model: Spiral model (cont’d)

- Advantages:
  - Risk reduction
  - Software is produced early
  - Early feedback from customer
- Disadvantages:
  - Risk analysis requires high expertise
  - Highly dependent on risk analysis
  - Complex and costly to implement
- Assignment
  - *Compare and contrast the types of process models*

Agile methods

Manifesto for Agile Software Development
By Kent Beck and 16 others in 2001

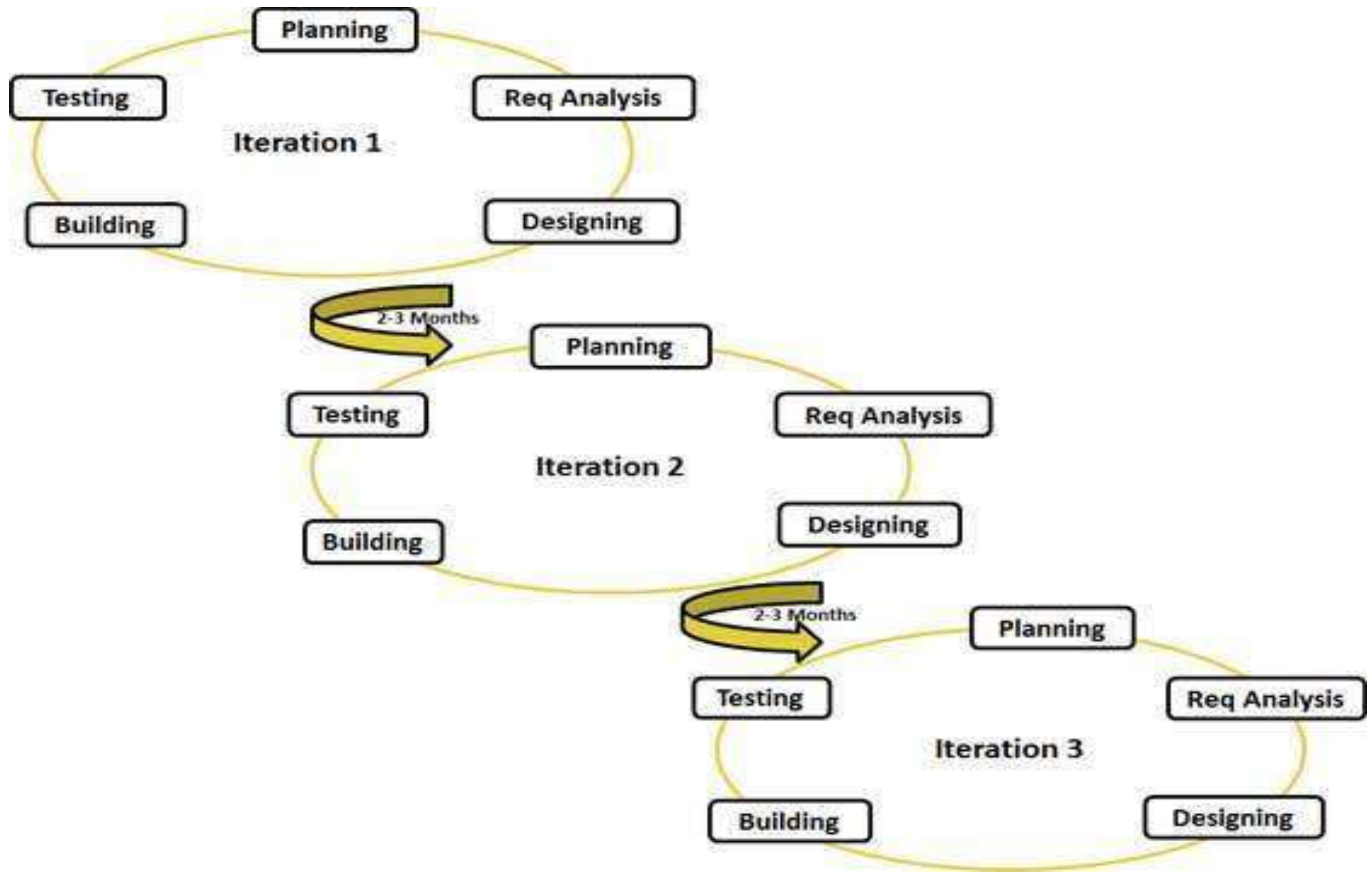
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Agile methods

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.



Agile methods

- What is Agility?

  It encompasses:

  - Quick and effective response to change
  - Team structure and attitude that makes communication facile
  - Rapid delivery of operational s/w
  - De-emphasis the importance of intermediate work product
  - Adopts customer as part of the development team
  - Eliminates “us and them” attitude
  - Flexibility in project plan

- Philosophy and Development guideline
  - Customer satisfaction
  - Early incremental delivery
  - Small and highly motivated project team
  - Informal methods
  - Minimal s/w engineering work products

Agile methods

- **Agile Principles**

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

Agile methods

- **Agile Principles (cont'd)**

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely

Agile methods

- **Agile Principles (cont'd)**

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity—the art of maximizing the amount of work not done—is essential.

11. The best architectures, requirements, and designs emerge from self–organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile methods

The Agile methods:

- Rational Unified Process (1994),
- Scrum (1995),
- Extreme Programming (1996),
- Adaptive Software Development,
- Feature Driven Development, and
- Dynamic Systems Development Method (DSDM) (1995).
- Lean Software Development (LSD)
- etc.