Software Process Models

Lecture 3

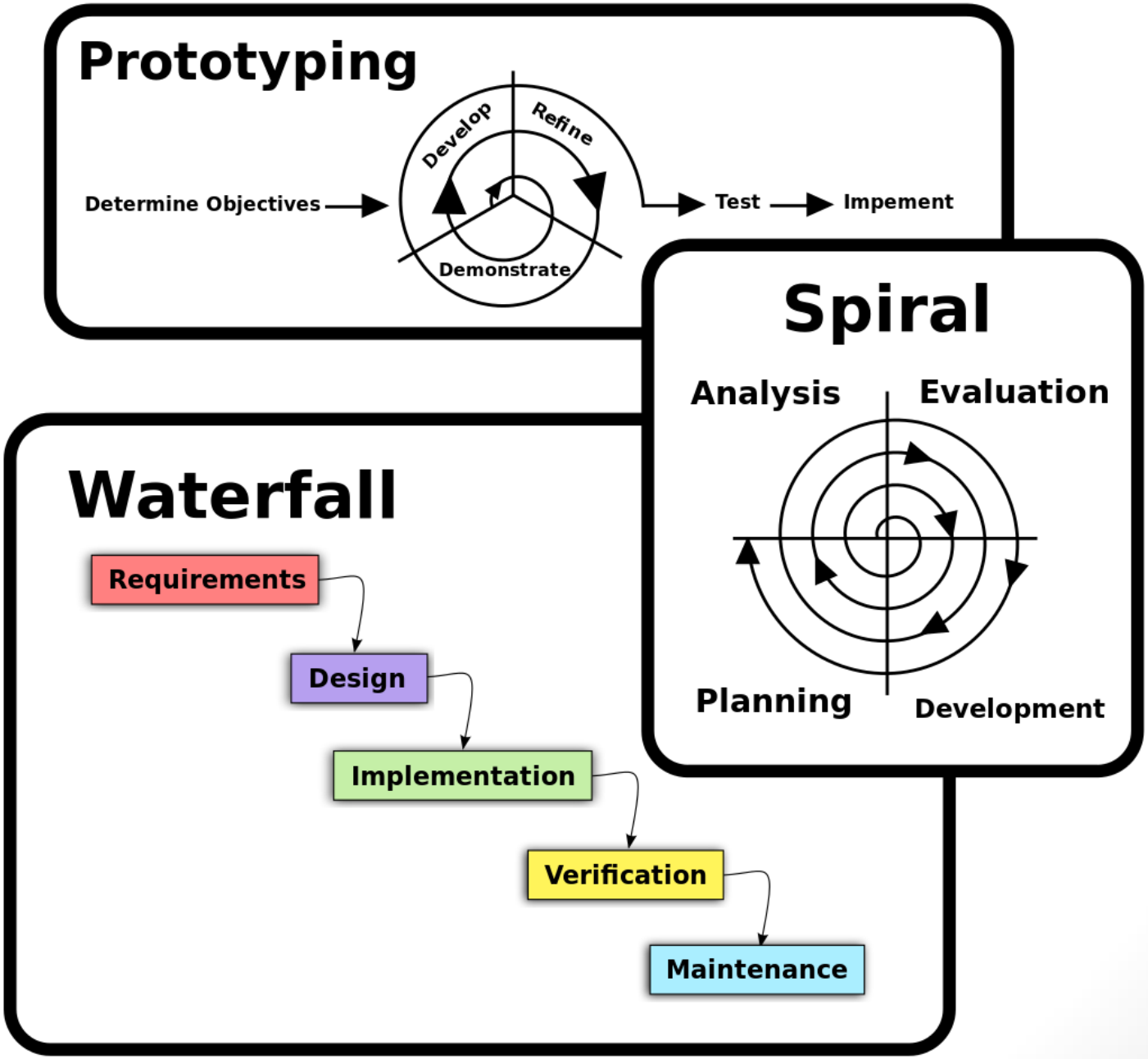
Overview

- Different process models
  - Build-and-fix model
  - Waterfall model
  - Incremental model
  - Evolutionary process models
    - Rapid prototyping model
    - Spiral model

Software Process Models

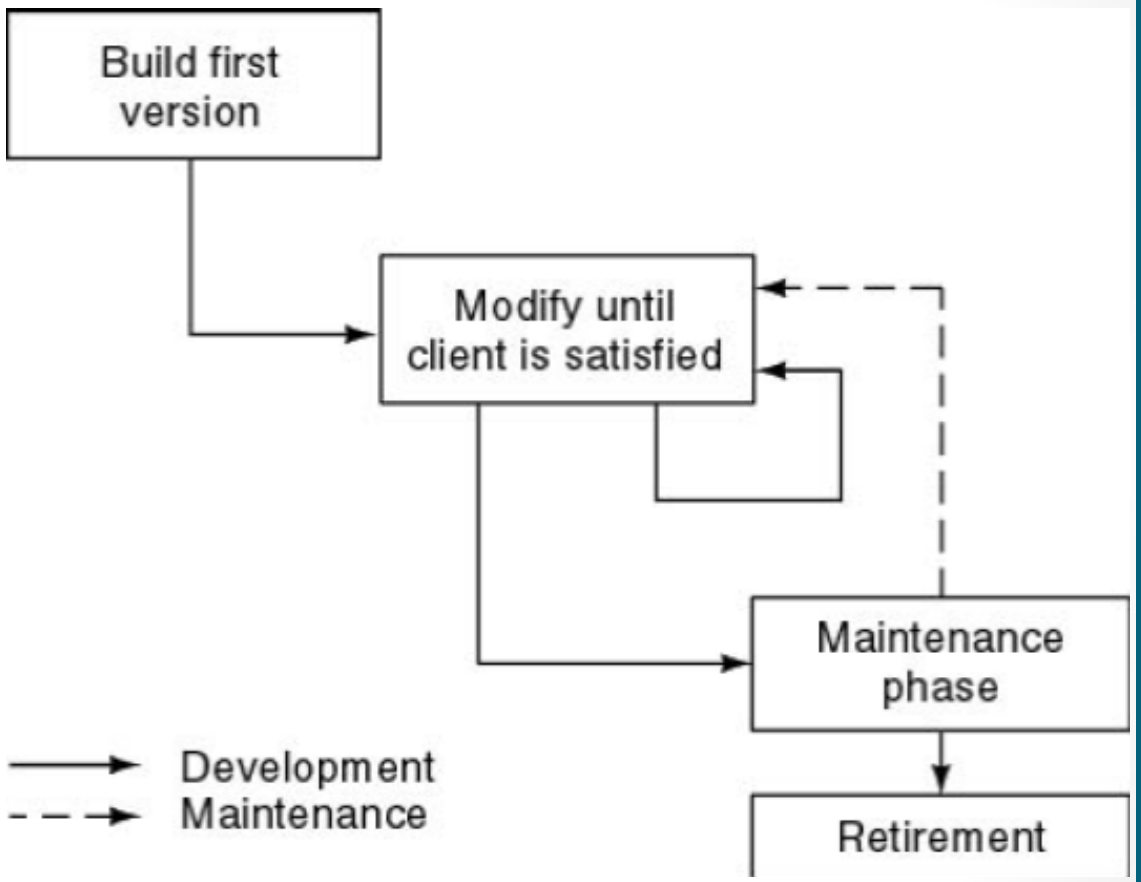
- Process model (Life-cycle model) -steps through which the product progresses
  - Requirements phase
  - Specification phase
  - Design phase
  - Implementation phase
  - Integration phase
  - Maintenance phase
  - Retirement

Software Process Models



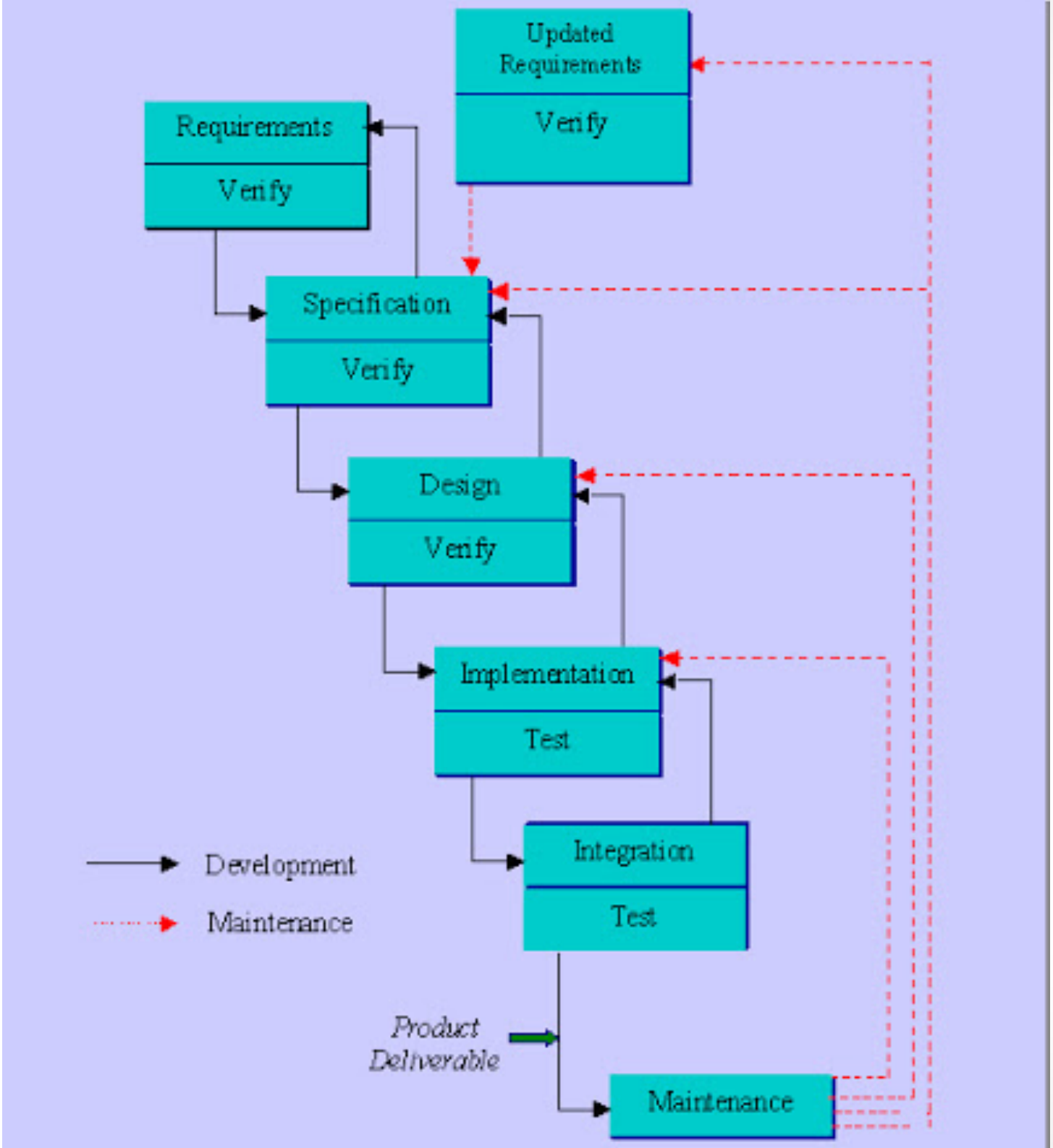
Build-and-Fix Model

- Problems
  - No specifications
  - No design
- Totally unsatisfactory
  - High cost
  - Difficult maintenance



Waterfall Model

- The **waterfall model** is a sequential design **process** in which progress is seen as flowing steadily downwards (like a **waterfall**) through the phases of SDLC.
- **Waterfall model** is an **example** of a Sequential **model**. In this **model**, the software **development** activity is divided into different phases and each phase consists of a series of tasks and has different objectives. (Refer to Lecture 2)
- **Waterfall model** is the pioneer of the SDLC processes.
- **Characterized by:**
  - Feedback loops
  - Documentation-driven

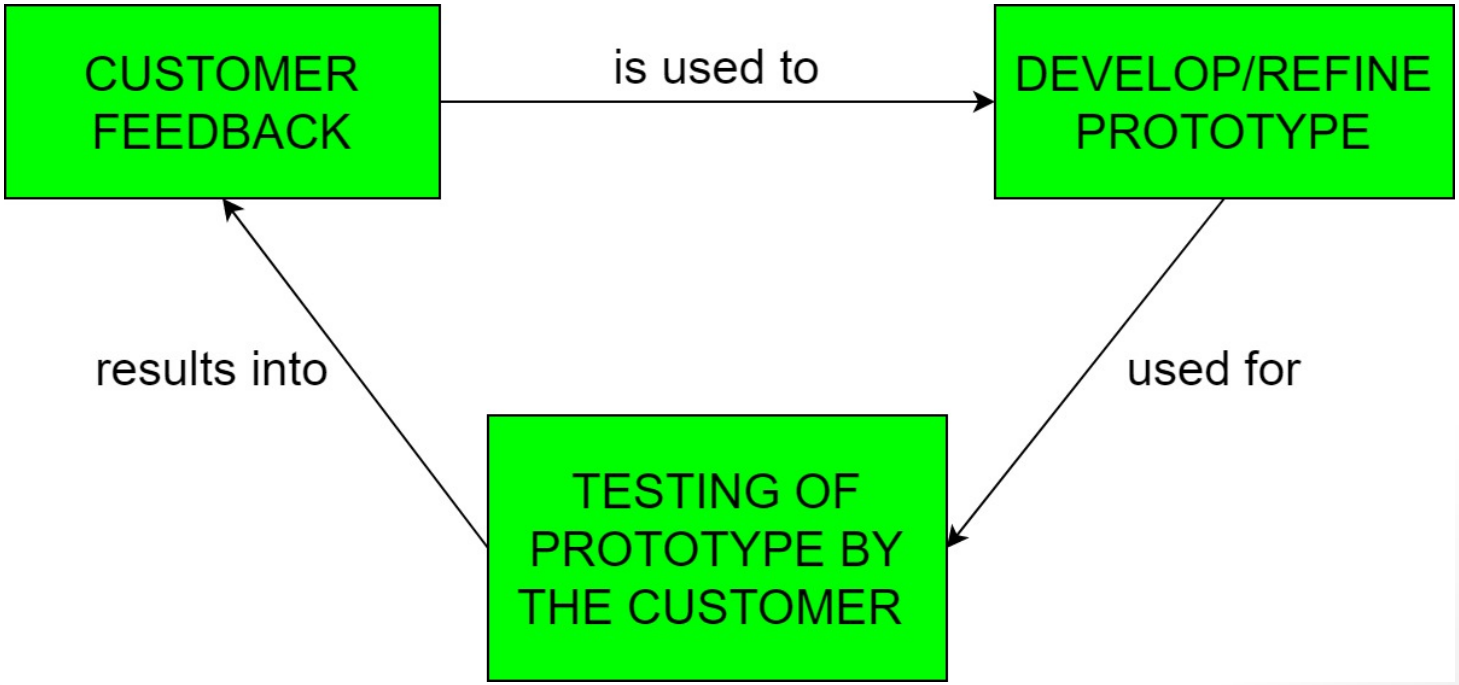

Updated Requirements
Verify
Requirements
Verify
Specification
Verify
Design
Verify
Implementation
Test
Integration
Test
Development
Maintenance
Product Deliverable
Maintenance

Waterfall Model (contd.)

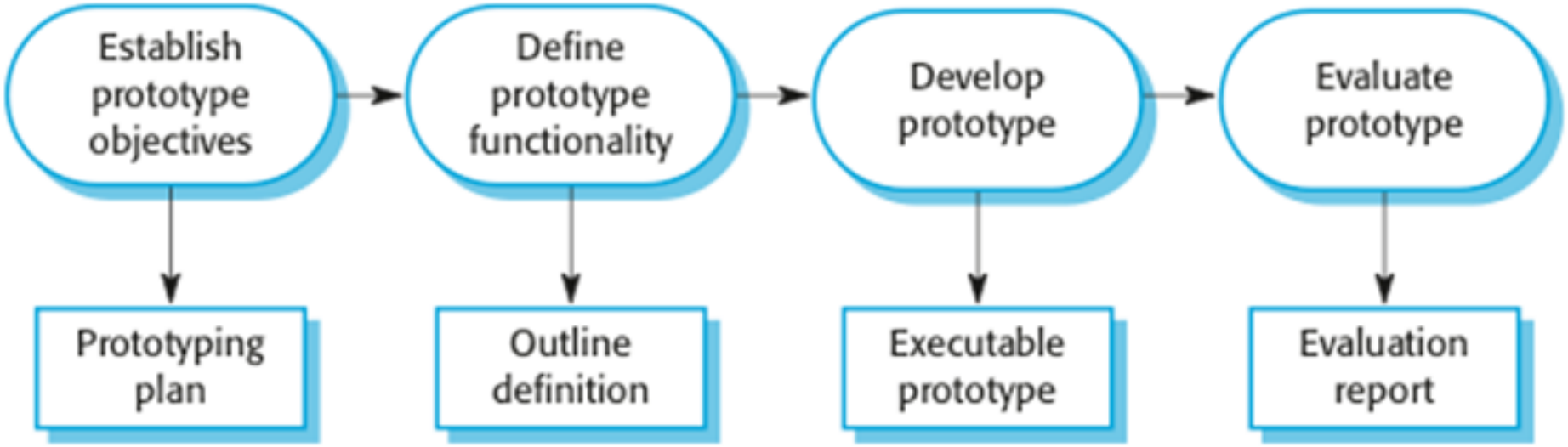
- Advantages
  - Enforces disciplined approach
    - Documentation for each phase
    - Products of each phase checked by SQA group
  - Maintenance is easier
    - Every change reflected in the relevant documentation
- Disadvantages
  - Working version of the software will not be available until late in the project time-span
  - Specifications are long, detailed, written in a style unfamiliar to the client
  - “Blocking states” –some project team members must wait for other team members to complete dependent tasks

Rapid Prototyping Model

- Prototyping is defined as the process of developing a working replication of a product or system that has to be engineered.
- It offers a small scale replica of the end product and is used for obtaining customer feedback as described below:



Rapid Prototyping Model (contd.)



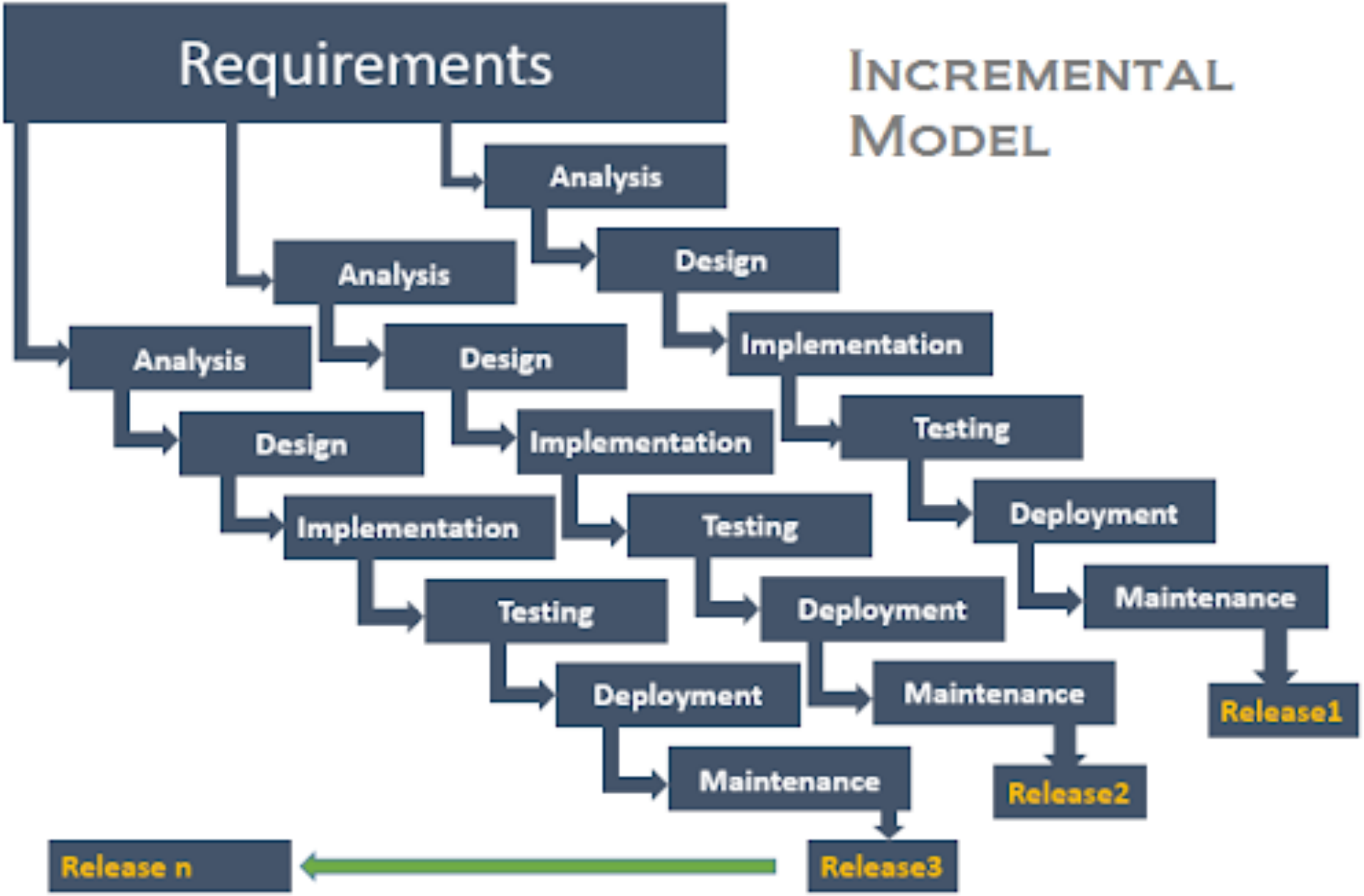
Rapid Prototyping Model (contd.)

- Rapid prototype characteristics:
  - Used in the requirements phase
  - Evaluated by the customer/user
  - Then, it is discarded -do not turn into product
- Rapid prototyping model is not proven and has its own problems
  - Possible solution
    - Rapid prototyping for defining requirements
    - Waterfall model for rest of life cycle

Incremental Model

- **Incremental Model** is a process of **software development** where requirements are broken down into multiple standalone modules of **software development** cycle.
- Each iteration passes through the requirements, design, coding and **testing** phases.
- Typical product takes from 5 to 25 builds (iterations).

Incremental Model (contd.)



Incremental Model (contd.)

- Waterfall and rapid prototyping models
  - Deliver complete product at the end
- Incremental model
  - Deliver portion of the product at each stage
- Advantages
  - The software will be generated quickly during the software life cycle
  - It is flexible and less expensive to change requirements and scope
  - Throughout the development stages changes can be done
  - This model is less costly compared to others
  - A customer can respond to each building
  - Errors are easy to be identified

Incremental Model (contd.)

- Disadvantages:
  - It requires a good planning designing
  - Problems might arise due to system architecture as not all requirements collected up front for the entire software lifecycle
  - Each iteration phase is rigid and does not overlap each other
  - Correcting a problem in one unit requires correction in all the units and consumes a lot of time

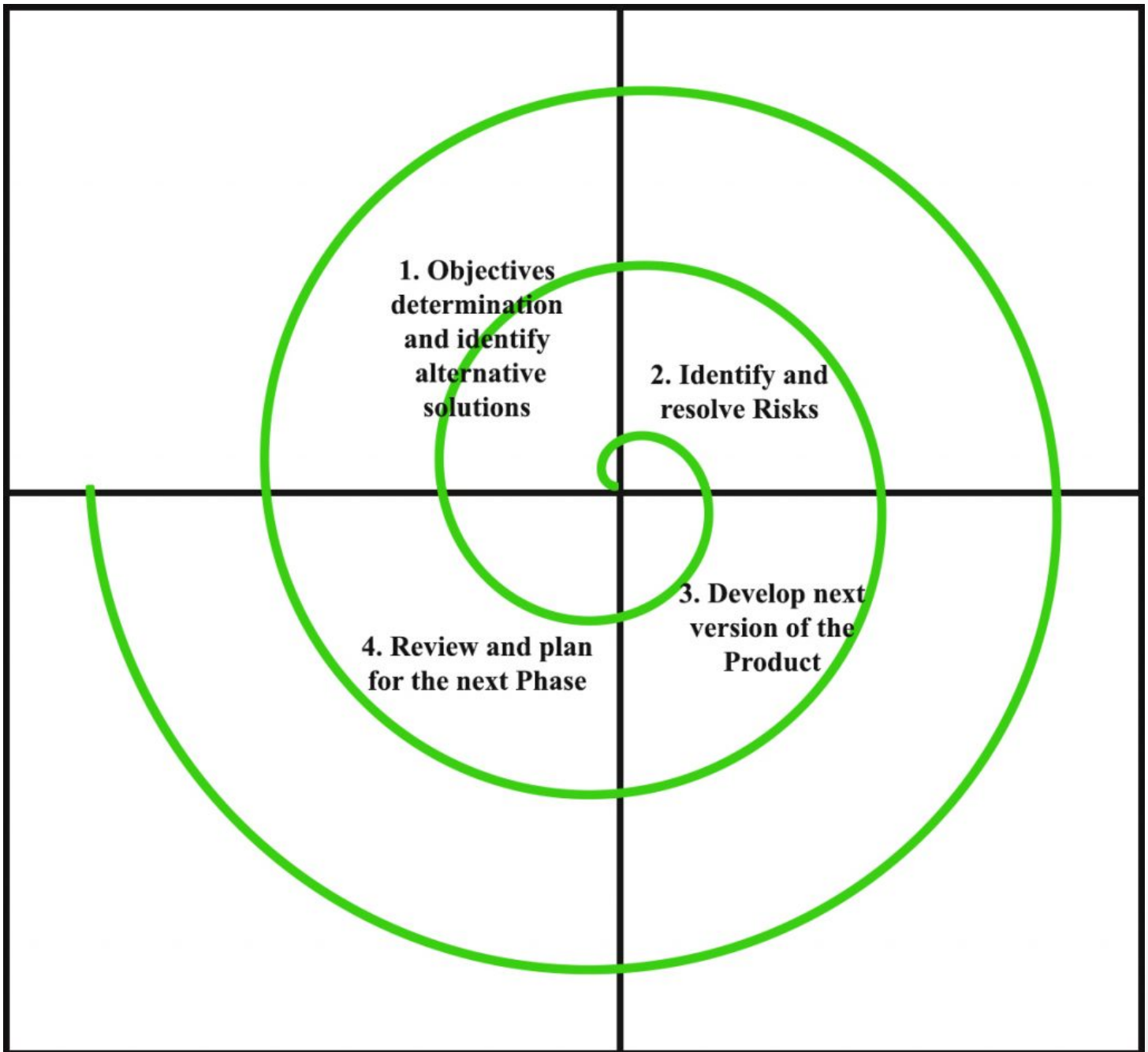
When to use Incremental models?

- Requirements of the system are clearly understood
- When **demand for an early release** of a product arises
- When software engineering **team are not very well skilled** or trained
- When high-risk features and goals are involved
- **Such methodology is more in use for web application and product based companies**

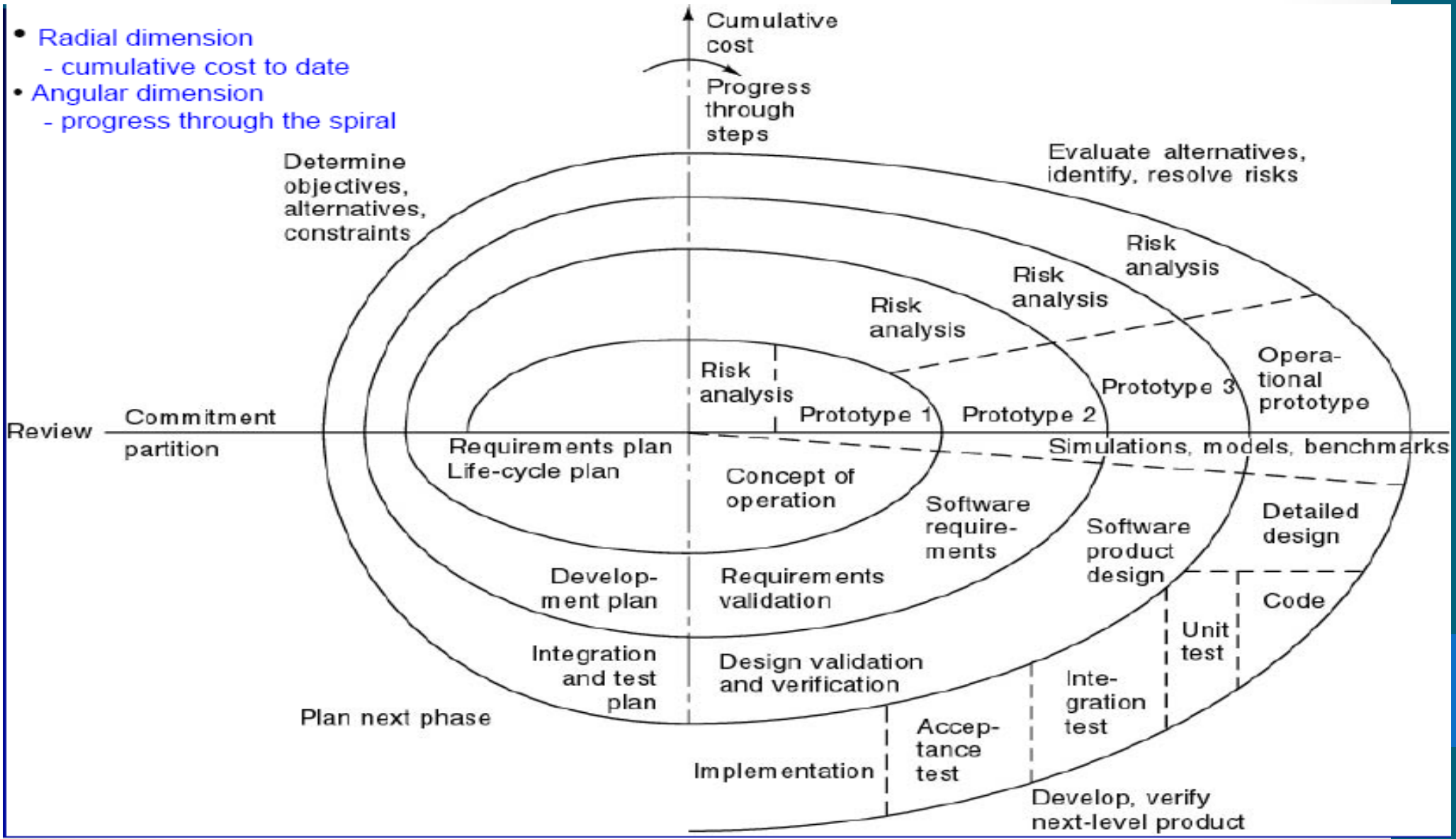
Spiral Model

- The **spiral model** is a risk-driven **software development** process **model**.
- Based on the unique risk patterns of a given project, the **spiral model** guides a team to adopt elements of one or more process **models**, such as incremental, waterfall, or evolutionary prototyping.
- **Risk Analysis:** Identification of potential risk is done while risk mitigation strategy is planned and finalized
- Precede each phase by
  - Alternatives
  - Risk analysis
- Follow each phase by
  - Evaluation
  - Planning of next phase

Simplified Spiral Model



Full Spiral Model



When to use Spiral Methodology?

- When project is large
- When releases are required to be frequent
- When creation of a prototype is applicable
- When risk and costs evaluation is important
- For medium to high-risk projects
- When **requirements are unclear** and complex
- When **changes may require at any time**
- When long term project commitment is not feasible due to changes in economic priorities

Advantages of Spiral Model

- Additional functionality or changes can be done at a later stage
- Cost estimation becomes easy as the prototype building is done in small fragments
- Continuous or repeated development helps in risk management
- Development is fast and features are added in a systematic way
- There is always a space for customer feedback

Disadvantages of Spiral Model

- Risk of not meeting the schedule or budget
- It works best for large projects only also demands risk assessment expertise
- For its smooth operation spiral model protocol needs to be followed strictly
- Documentation is more as it has intermediate phases
- It is not advisable for smaller project, it might cost them a lot