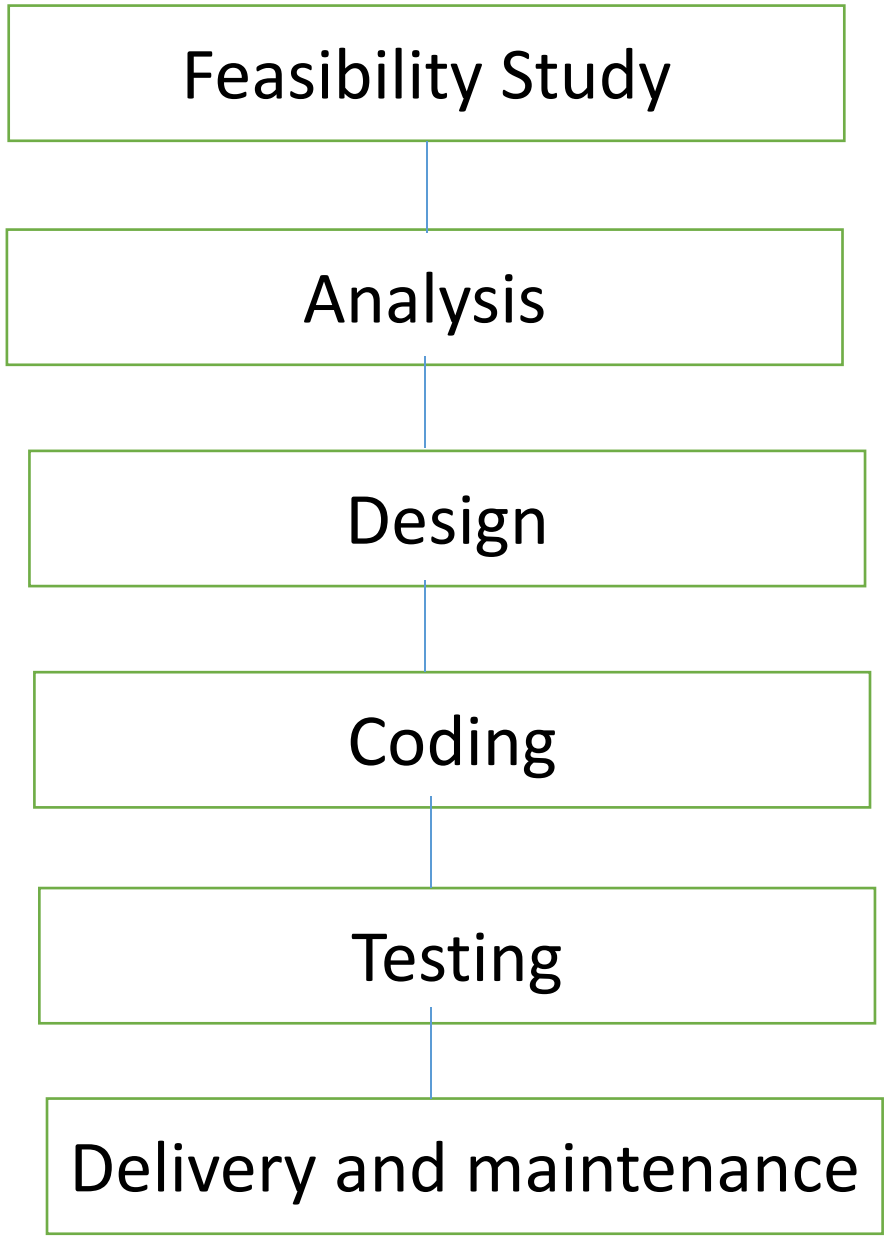
SDLC models

Software development life cycle



Software development life cycle (SDLC)

- Feasibility Study: The business analyst team gather the business requirements from the client environment. The requirements were gathered based on technical feasibility, economical, legal, operational, schedule feasibility.
- Technical feasibility: Checking the technical requirement of the company like software, hardware and staff requirements are existing.

- Economical feasibility: possibility of completing the project within the budget.
- Legal feasibility: rules and regulations to be verified.

- Operational feasibility: checking the possibility of developing those applications or operations.

- Schedule feasibility: possibility of delivering the project within the time.

Software development life cycle

- Business design document (BDD): The gathered requirements are written in a separate document. BDD is taken as input to the next phase. All the business requirements were decided here.

- Analysis phase: system analyst involved. Convert the business requirements to technical requirements. Software Requirement specification (SRS) is done at this phase. SRS shows the business requirements with technical requirements. SRS act as input to design phase.

- Design phase: consists of high level design (HLD), low level design (LLD). HLD has main modules and done at the project manager level, LLD has Sub modules and done at project leader levels. Output of LLD is a technical design document. This document is used for coding phase.
- Coding phase: Software development team is involved. This team involved in developing the source code at specific technology or environment. Ex: java, python

Software development life cycle

- White-box testing: coding is tested here. Tested by the software developer itself. build is released

- Testing phase: named as black-box testing. functional testing is done here by the testing team. Consists of six steps: test planning, test designing, test execution, result analysis, bug tracking, bug reporting.
  - Before delivery it leads to alpha and beta testing called as user acceptance testing.

  - Alpha testing: at company environment functionality is verified at the client level.

  - Beta testing: testing is done at client environment. Done by their testers or third party testers.

  - Maintenance: based on the rules and standards provided by the client maintenance is done.

BRD template

- A summary statement
- Project objectives
- Needs statement
- Project scope
- Financial statements
- Functional requirements
- Personal needs
- Schedule, timeline & deadlines
- Assumptions
- Cost & Benefit

Software Requirement specification-Sample

- Introduction

Purpose, scope, References, intended audience and suggestions, document overview

- Description

Product Perspective, Product Functions, Operating Environment, Design and Implementation Constraints, User documentation, Assumptions and Dependencies

External Interface Requirements:

Functional Requirement Specifications:

System Features, Functional Requirements, Front end (Storefront) Requirements, Back end (Administrative Tools) Requirements, Use Cases

Non-Functional Requirements:

Usability Requirements, Performance Requirements, Compatibility Requirements

Other Requirements

Capability Maturity Model (CMM)

- Capability Maturity Model for Software describes the principles and practices underlying software process maturity.

- Method used to develop an organizations software development process.

- CMM Level1: Initial: The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.

- CMM Level2: Repeatable: Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

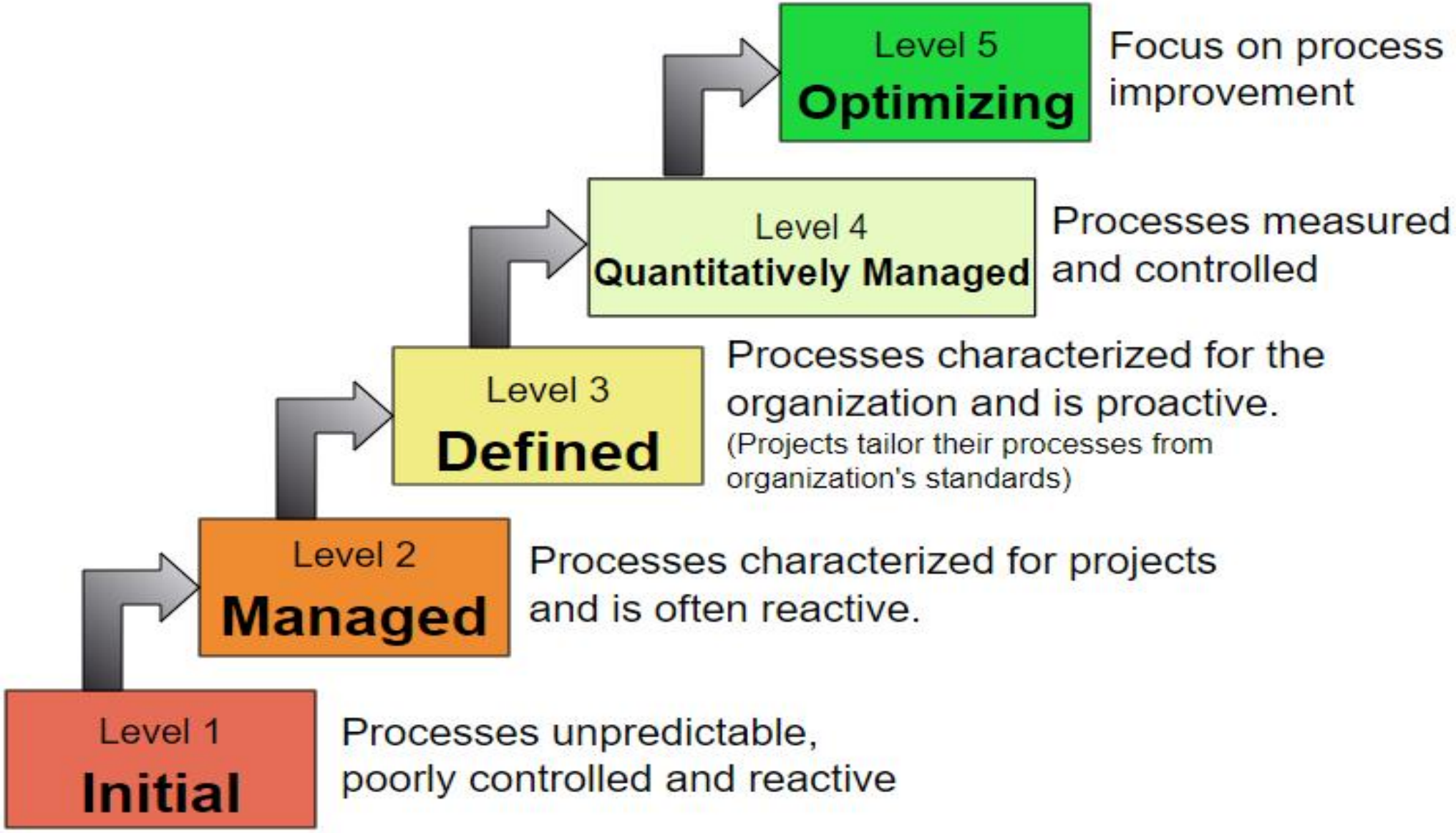
Capability Maturity Model (CMM)

- CMM Level 3: Defined: The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

- CMM Level 4: Managed: Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

- CMM Level 5: Optimizing Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

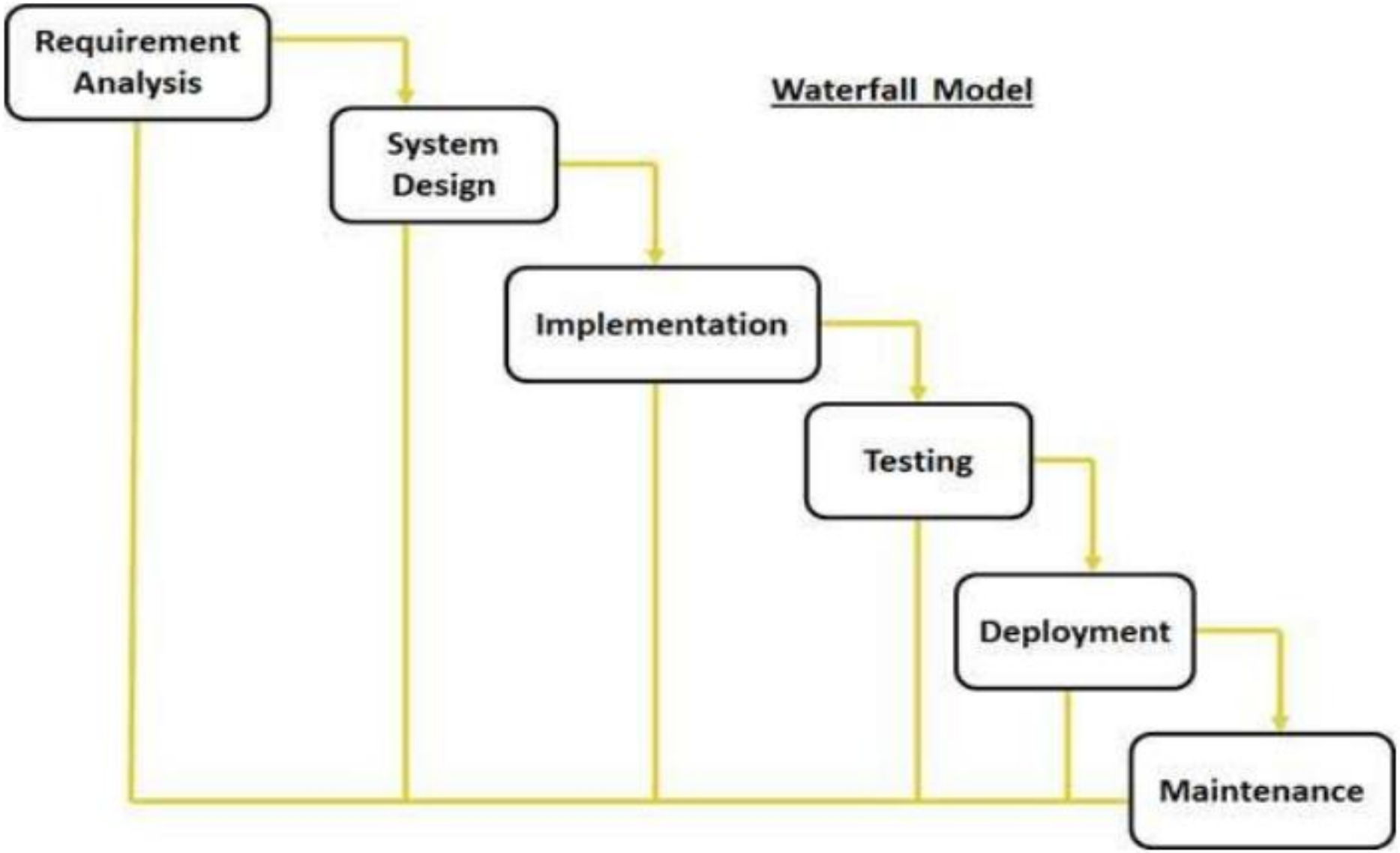
Capability Maturity Model (CMM)



SDLC models

- SDLC model should be adjusted to the features of the product, project, and company.
- The most used, popular and important SDLC models
  - Waterfall model
  - Iterative model
  - Spiral model
  - V-shaped model
  - Agile model

Waterfall SDLC Model



Waterfall SDLC Model

- Initial or earliest process. Linear sequential life cycle model.
- Easy to understand and apply.
- No overlapping of phases. Each phase to be completed before next phase.
- Outcome from previous phase used. No reverse process.

- Requirement Gathering and analysis: All possible requirements of the system are captured in this phase and documented.
- System Design: Specifying hardware and system requirements and also helps in defining overall system architecture.

- Implementation: System is developed in small programs called units.
- Integration and Testing: integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

Waterfall SDLC Model

- Deployment of system: Once the functional and non functional testing is done, the product is deployed in the customer environment or released to market.

- Maintenance: compatibility or issues in the client environment. patches are released to fix issues. Version developments. Deliver the changes in the customer environment.
- **Applications:**
- Construction, space
- traditional organizational environments
- Smaller projects
- Customer Relationship Management (CRM) systems

Waterfall SDLC Model

Advantages:

- Control and departmentalization
- Cost effectiveness
- Simple and easy. Well known by developers.
- Requirements are clear and not changing frequently.
- Minimum client intervention.

Disadvantages:

no reflection or revision.

High risk and uncertainty

Poor model for long and ongoing projects

Inflexible