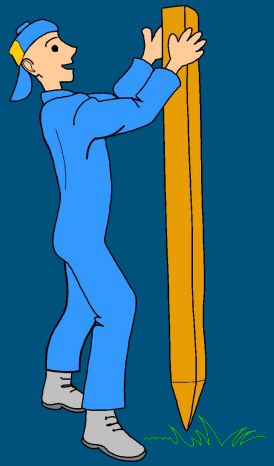
Lifecycle and Requirements-I

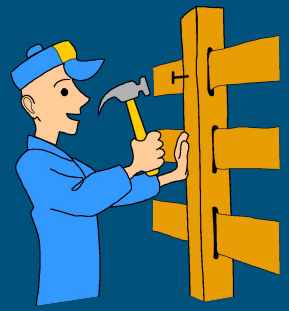
Manas Jyoti Das, PhD
Computer Science

Calculate the completion time



Setting posts
[3 time units]

Shaping wood
[2 time units]



Nailing
[2 time units for unpainted;
3 time units otherwise]

Painting
[5 time units for unshaped wood;
4 time units otherwise]

...shortest possible completion time = ? Trade off

Software is complex

- Complex ≠ complicated

- Complex = composed of many simple parts

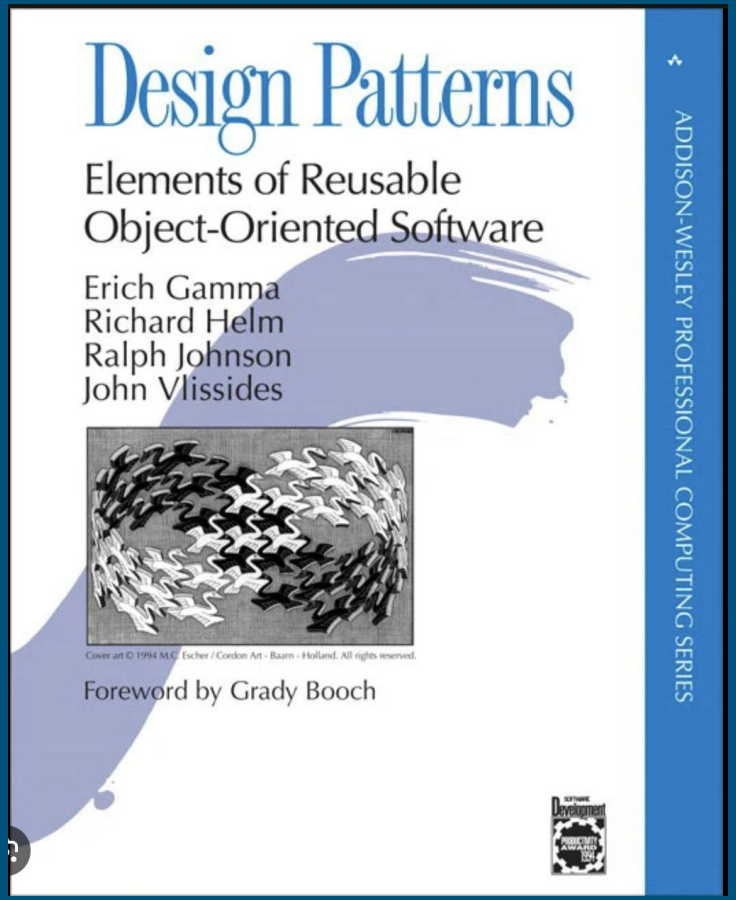
  related to one another

- Complicated = not well understood, or explained

Engineering process

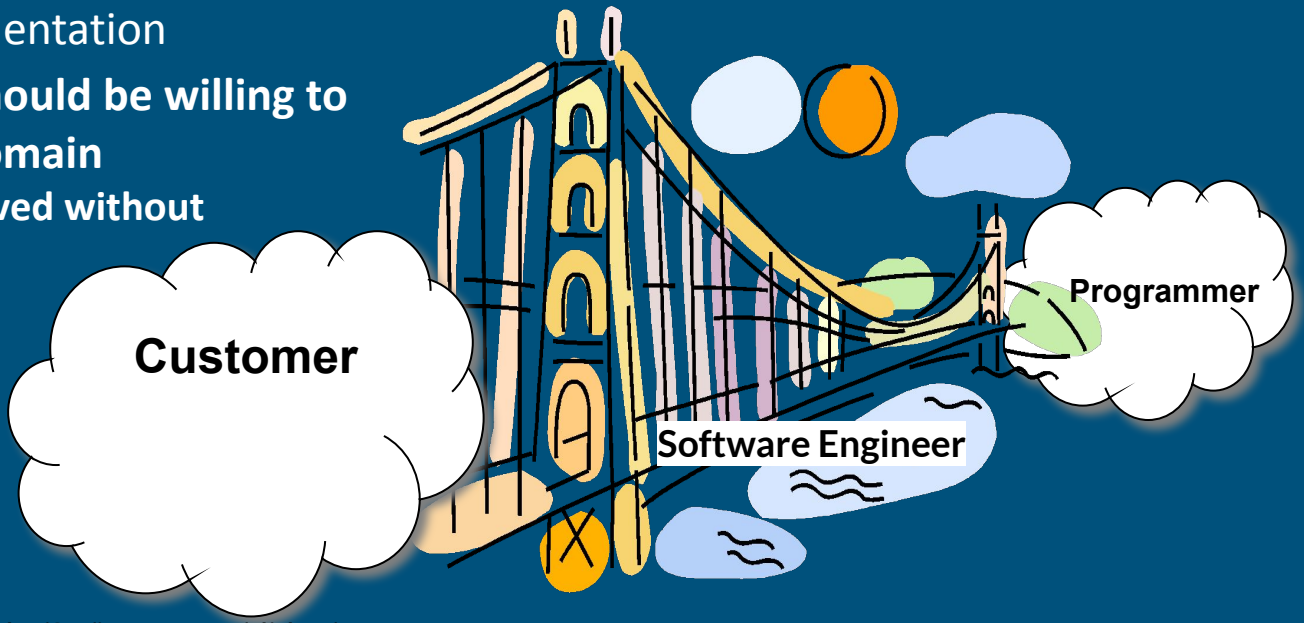
- Understanding the process is very important
- Building a bridge (road, car, airplane, ...) requires
  - Resources (land, mud, wood, brick, steel, …)
  - Labor
  - Skills
- Engineering makes planning possible
  - Can predict needed resources and costs
  - Can predict completion schedule
- Think back to previous projects
- Software engineering is about understanding the process of developing a software

Gang of 4 (GoF)


Design Patterns
Elements of Reusable
Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

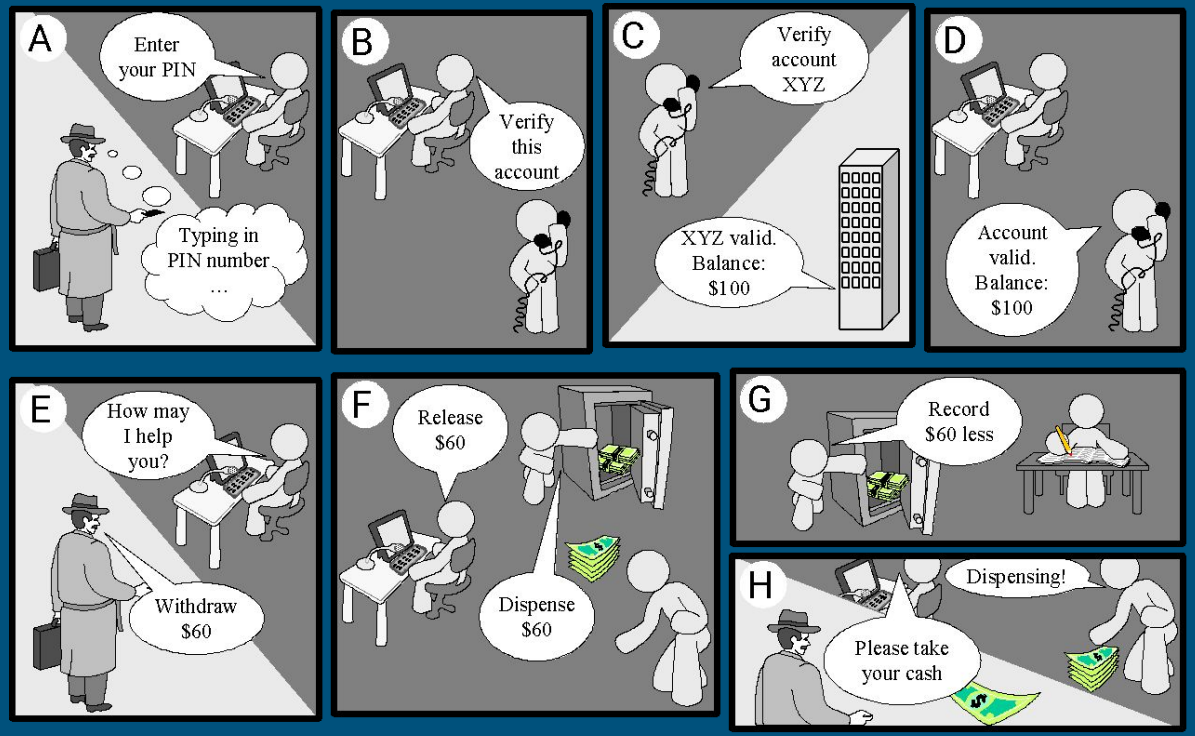
The Role of Software Engineer

- A bridge from customer needs to programming implementation
- **Software engineer should be willing to learn the problem domain (problem cannot be solved without understanding it first)**



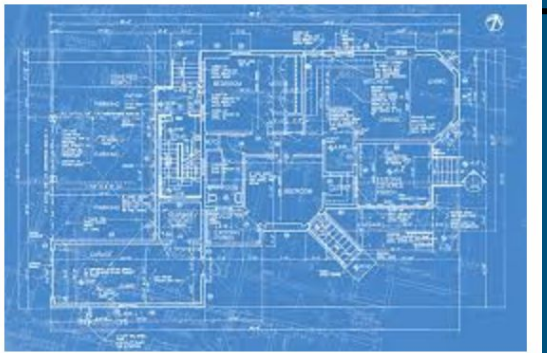
https://www.ece.rutgers.edu/~marsic/books/SE/instructor/slides/

How ATM machine works



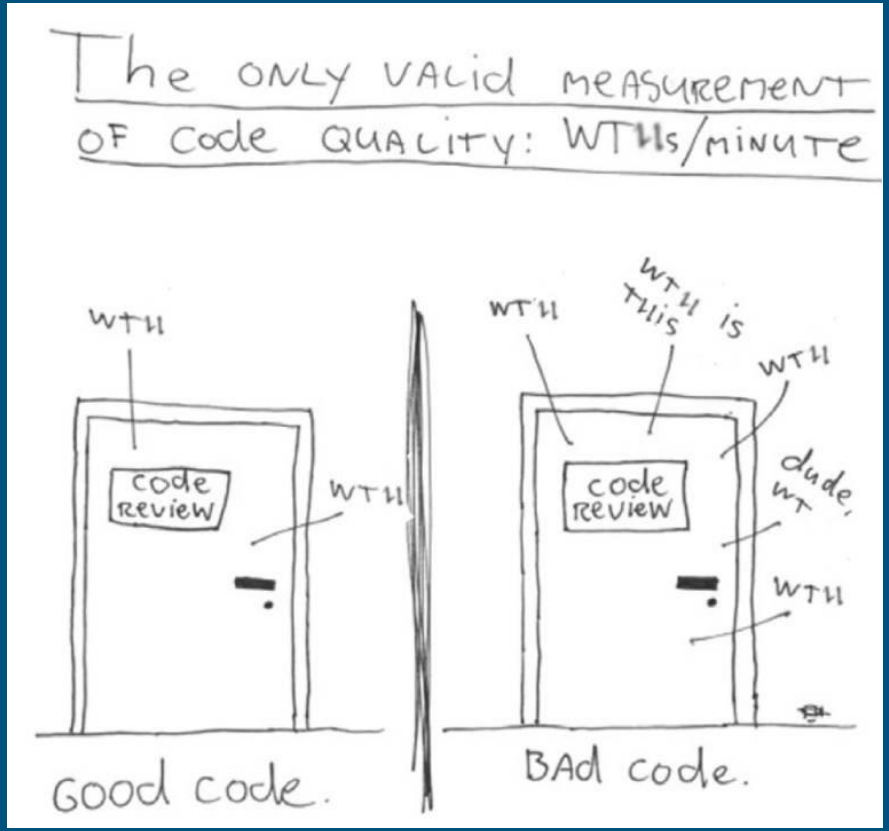
Software engineering blueprint

- Specifying software problems and solutions in cartoon strip is not the solution.
- Unfortunately, most of us are not artists, so we will use something less exciting: UML symbols and others

Architect design	Engineer design	Craftsman build
		

Code Quality: What the hell!!

- Which door represent your code



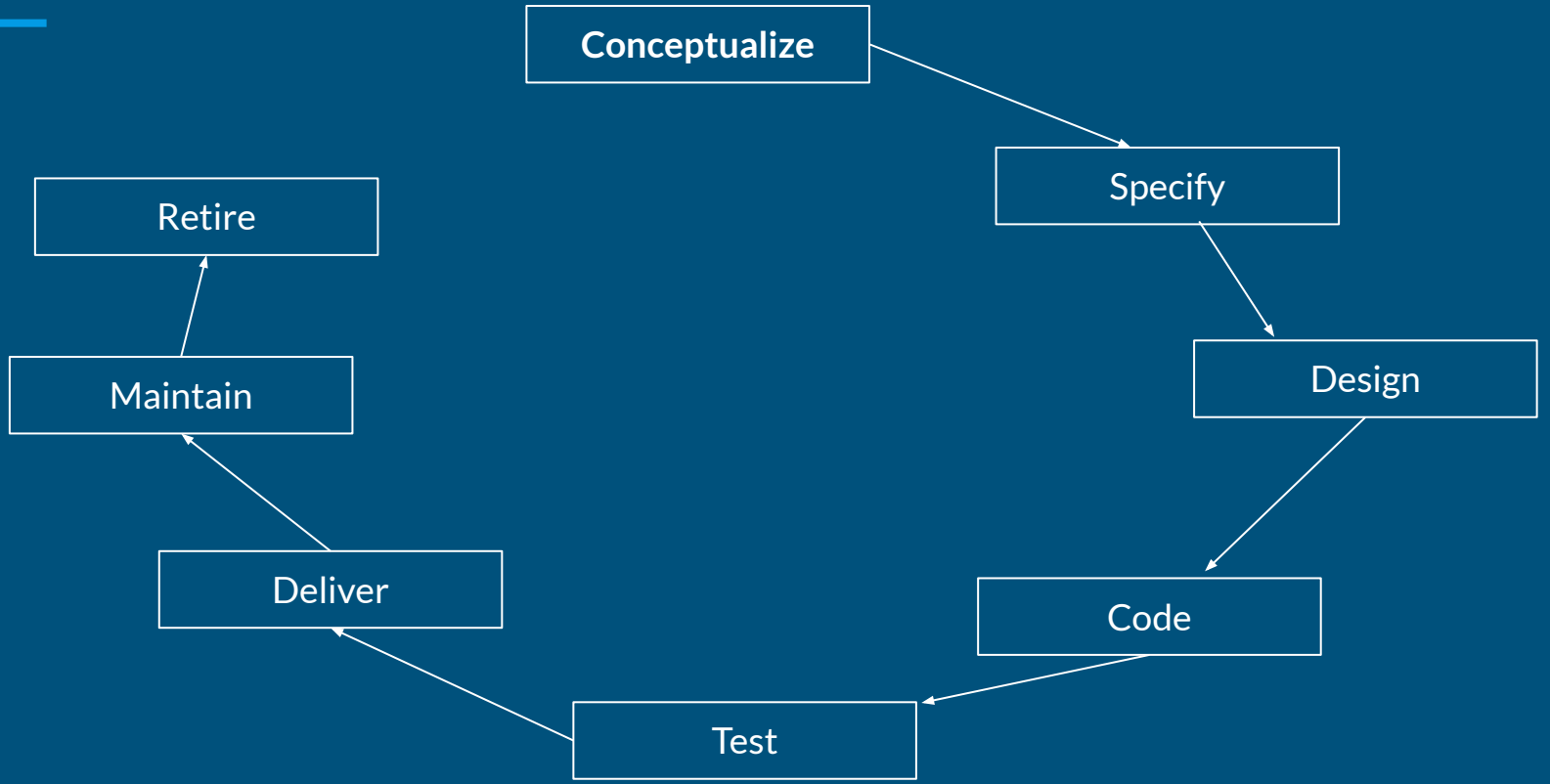
Software Development Methods

- Waterfall
  - Unidirectional, finish this step before moving to the next
- Iterative + Incremental
  - Develop increment of functionality, repeat in a feedback loop
- Agile
  - Continuous user feedback essential; feedback loops on several levels of granularity

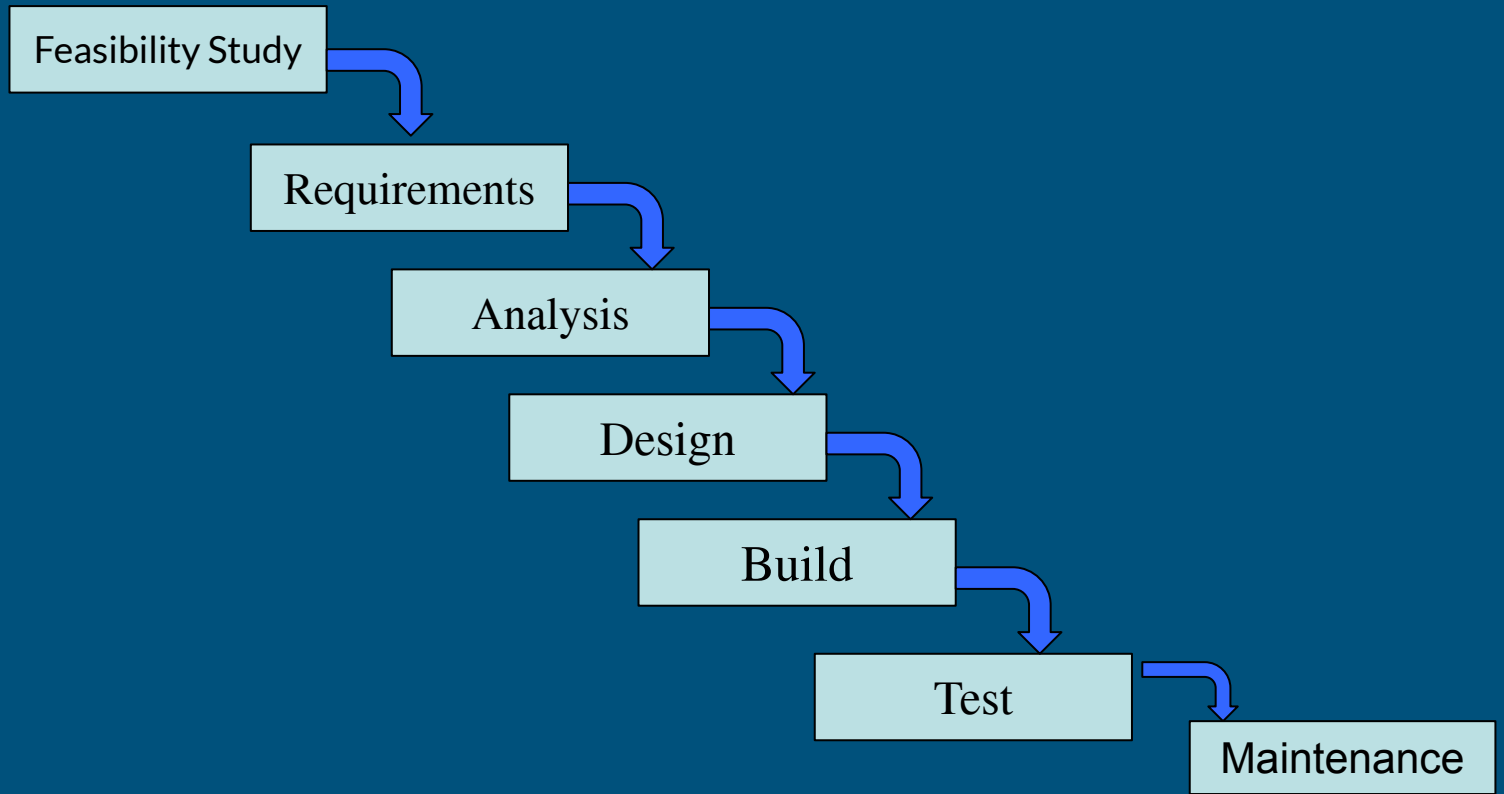
Life cycle model

- Also known as SDLC (Software Development Life Cycle)
- It is a diagrammatic model of software life cycle
- Establishes a precedence ordering among different activities
- Divide life cycle into phases

Software lifecycle (general outlook)



Waterfall model



Feasibility study

- Technical feasibility (some technology required by the project not known by the organization)
- Economic feasibility (cost/benefit)
- Operational feasibility (who will input the data? Is the infracture there, who will maintain it?)
- Scheduling feasibility (are the customer's requests feasible?)
- Legal feasibility
- Market survey
- Formulate different solution strategies
- Evaluate alternative strategies in terms of:
  - Resource required
  - Cost of development
  - Development time

Requirement analysis and specification

- Understand the exact requirements of the customer
- Document the requirement properly
- This phase is a important phase, the success and failure of a project depends on the understanding of this phase
- In requirement analysis:
  - Gather the data of what customer wants (interviews and/or discussions)
  - Remove inconsistencies (one part of the requirement contradict with another part of the req.)
  - Remove incompleteness (some features are missed)
  - Remove ambiguity (should have a clear understanding, not vague)
- Create a SRS (Software Requirement Specification) document

Design phase

- Requirement specifications are transformed into a suitable form that can be implemented using a programming language, two approaches:
  - Traditional approach
  - Object oriented approach
- Traditional approach
  - Structured analysis (using DFD, Data Flow Diagram)
  - Structure design (decompose the system into modules and draw the relationship between modules)
- Object oriented approach
  - Identify the objects
  - Identify the relation between objects
  - Advantages:
    - Lower developmental effort
    - Lower developmental time
    - Better maintainability

Coding and testing

- Each module identified by the design phase is coded
- Each module is unit tested
  - Test independently as a stand alone unit
- Each module is documented, documentation is important
- Modules are tested on its own, without integrating with other modules of the software

Testing

- In testing different modules are integrated
- The way to integrate the modules is one by one in a planned manner
- Modules are integrated logically, depending on the design document
- During each integration step, testing is done for the partially integrate system
- All modules will be integrated and a system testing will carried out
- It should not be one big bang at the end any error in one module may stall the whole process
- Fully integrated system will be tested and will ensure that the developed system follow all the requirements specified by a SRS document
- Software will be delivered to the customer

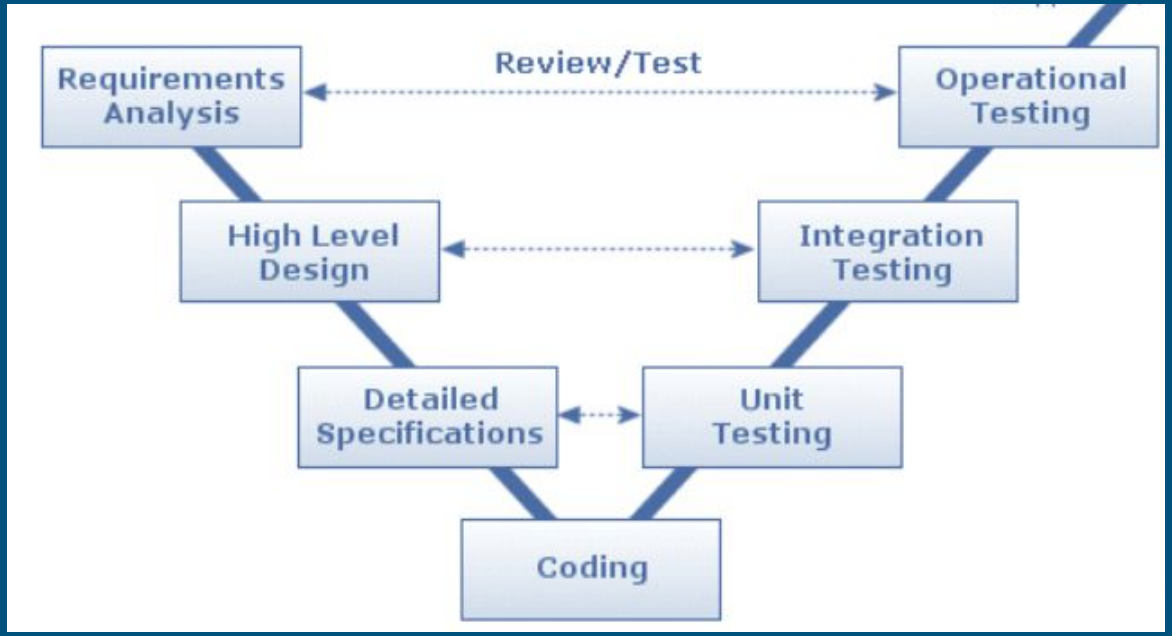
Maintenance

- Bug report and fixes
- Adding new feature
- Enhance the software (speedup etc.)
- The efforts in maintenance phase is longer than the developmental phase
- Corrective maintenance:
  - Bug correction
- Perfective maintenance:
  - Enhance functionality (speedup)
- Adaptive maintenance:
  - Posting the software to a new environment (e.g. new Operating system, etc.)

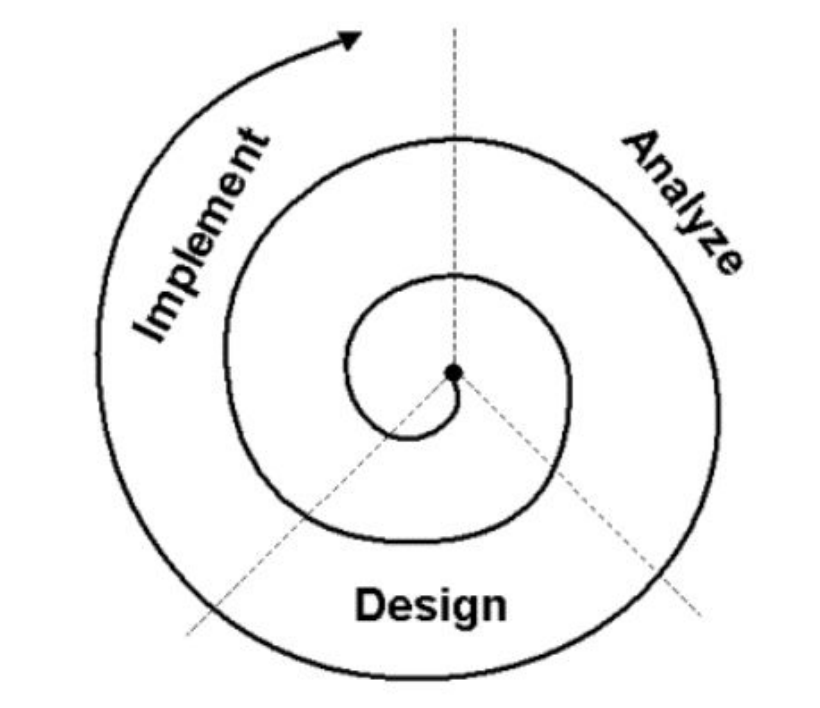
Drawbacks of waterfall model

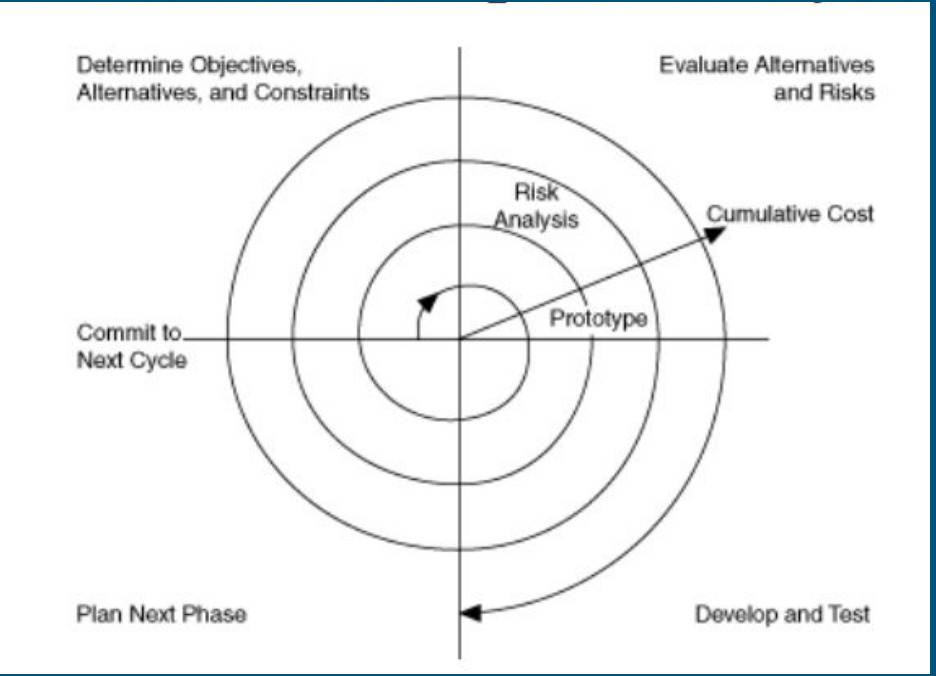
- Very idealistic, assumes all the previous phases has no defects in it
- It is not true in practice
- Defects are usually detected later in the life cycle
- The later the phase in which the defect gets detected, the more expensive is its removal
- We need feedback to previous stages (V model, spiral model)
- Advantages:
  - Easy to understand
  - Different phases are well understood
  - Requirements are known and stable
  - Technology is understood
  - Experienced development team

V model

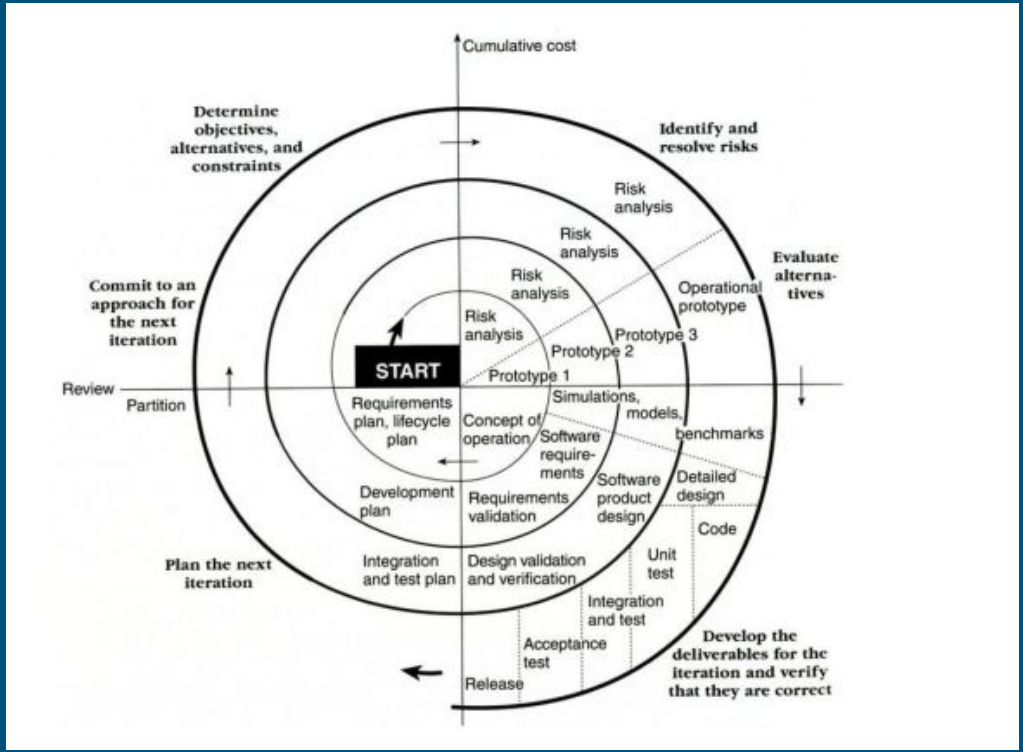

Requirements Analysis
Review/Test
Operational Testing
High Level Design
Integration Testing
Detailed Specifications
Unit Testing
Coding

Spiral Model

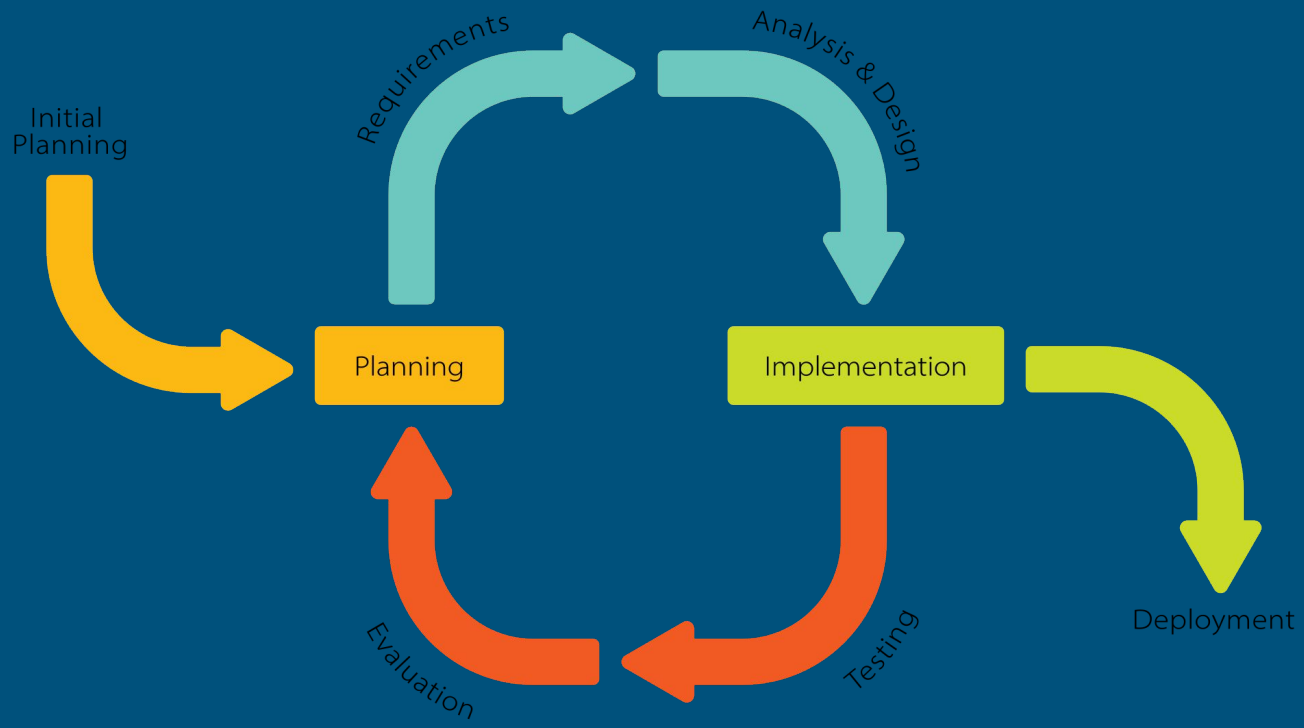




Spiral model



Iterative and incremental



Agile

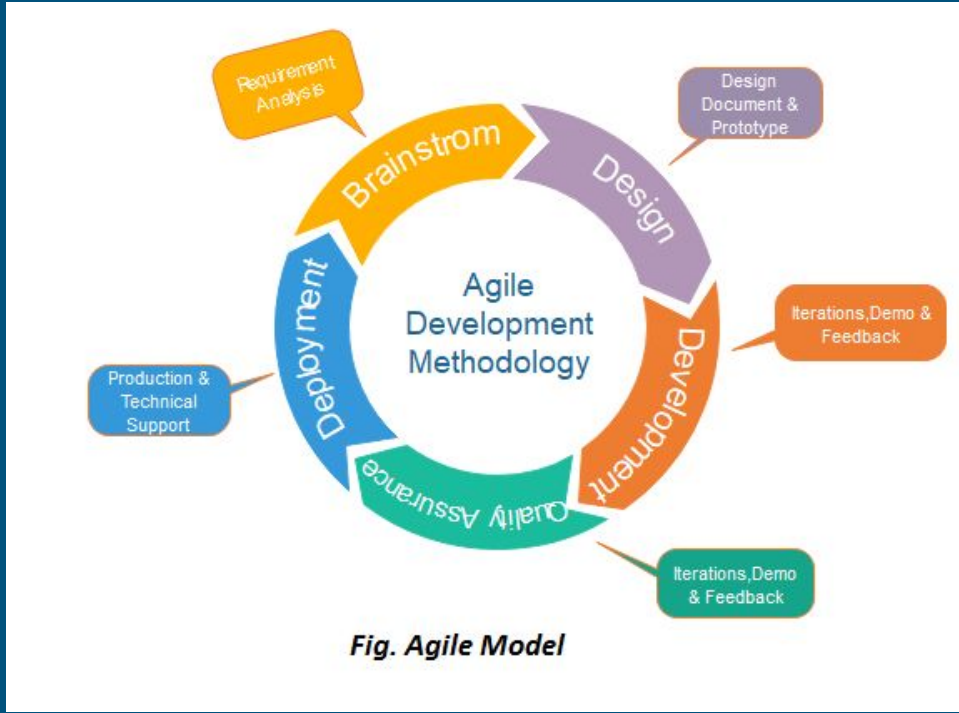


Fig. Agile Model

https://www.javatpoint.com/software-engineering-agile-model

Thank you

See you all in the next class