Week1: Introduction to Software Engineering

Dr. Rasha Kashef

Agenda

- The Software…
- Software Engineering
  - Definition and Scope
- The software problem
  - Cost, Schedule, and Quality
- Software Development Process
  - Software Life Cycle
- Software Process Models
  - Waterloo Fall, Prototyping, Iterative, Time-Boxing, and Agile

Software?

- **Software** is considered to be collection of executable programming code, associated libraries and documentations
- ‘Software’ implies: large, robust, reusable, evolving
- Why we study software?
  - Number #1 Reason:
    - Teaches you **HOW to DESIGN** !!
    - How to think about design, systematic problem solving

Software systems?

System software:

such as compilers, editors, file management utilities

Application software:

stand-alone programs for specific needs

Engineering/scientific software:

“Scientific-related” algorithms

Embedded software:

Software that resides within a product or system

Product-line software:

Group of software that focus on a limited marketplace to address mass consumer market

Web & network centric software:

Software that are built in sophisticated environments integrated with remote database and business applications

Intelligent systems:

Software that uses non-numerical algorithm to solve complex problem.

Robotics, expert systems, pattern recognition

What is Software Engineering (SE)?

- The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software (product), that is, the application of engineering to software.

  — IEEE Standard 610.12

- Canadian Standards Association

  "The systematic activities involved in the design, implementation and testing of software to optimize its production and support"

Software Engineering Scope

- Software Engineering is part of a much larger *system* design activities
  - Telephone switching systems, banking systems, hospital admin systems, aircraft
- Performing Software Engineering in the right procedure would requires a much larger look at *system* engineering issues
  - Entities and activities within the *system*, its interface with other *systems* and users
- Understanding the application and user needs is a key attribute in software development
  - Decide what activity should be supported by the *system* and how
  - Many domains, where very different software *systems* must be developed, with emphasis on different priorities:
    - time-to-market (telecom), safety (aerospace, NASA Shuttle), maintainability (telecom, banking)

The Software Problem

- **A Software** means ***industrial strength software*** that solves a specific problem

- Three basic forces at play when developing industrial strength software:
  - Cost
  - Schedule
  - Quality

1. Cost: Software Is Expensive

- Let us look at costs involved
  - LOC: lines of code, PM: person-month
  - Appx 1000 LOC/PM
  - Cost per LOC (line of code) = $3 to $10
  - Cost = $3K to $10K/PM
  - i.e., each line of delivered code costs many $$

- A simple application for a business may have 20KLOC to 50KLOC
  - Cost = $100K to $2.25Million
  - Can easily run on $10K-$20K hardware
  - So, Hardware costs in an IT solution are small as compared to Softare costs.

2. Schedule: Software Requires tight Schedule

- Business requirements today demand short delivery times for software
- In the past, software products have often failed to be completed in time
- Along with cost, cycle time is a fundamental driving force

Productivity – for Cost and Schedule

- Both can be modeled by productivity.

- Productivity = output/input resources
- In Software systems, output is considered as LOC (lines of code), KLOC (thousands of lines of code)
  - Input resources is effort - person months; overhead cost modeled in rate for person month
  - **Higher productivity leads to lower cost**
  - **Higher productivity leads to lower cycle time**

- Hence, for projects (to deliver software), quality and productivity are basic drivers

3. Software Quality

Along with productivity, quality is the other major driving factor

Developing high Quality software is a basic goal

Quality of software is harder to define

Quality – ISO standard: 6 Attributes

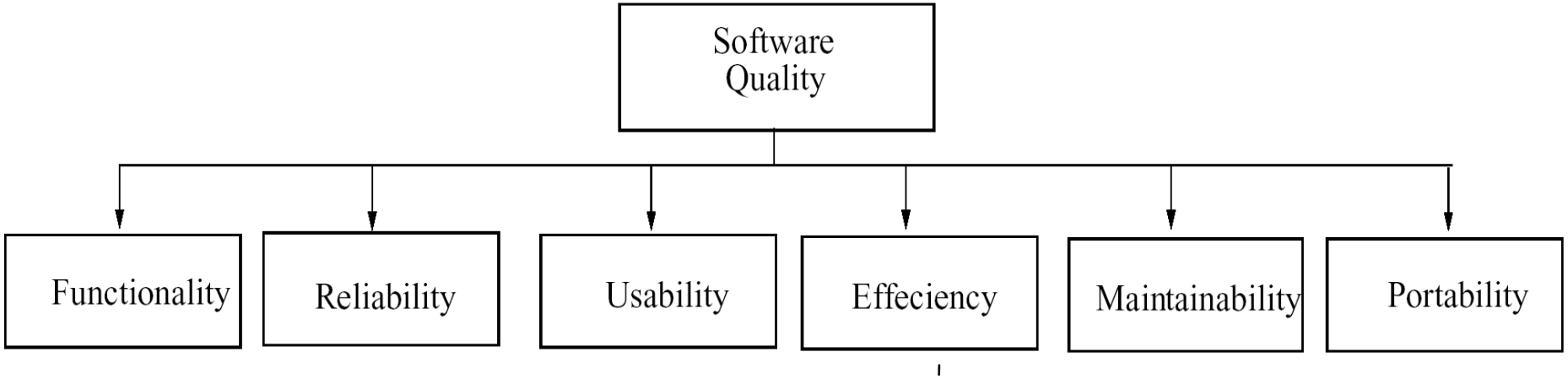
Multiple dimensions mean that it is not easy to map Quality to a single number

Concept of Quality is project specific

For some projects, reliability is most important

For others, usability may be more important

Reliability is generally considered the main Quality criterion



The Software Problem

Cost

Measured by LOC

Schedule

Cycle time is a fundamental driving force

Productivity = output/input resources

Higher productivity leads to lower cost

Higher productivity leads to lower cycle time

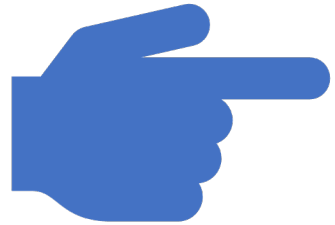
Quality

6 Attributes

Quality and Productivity

Quality and productivity (Q&P) are the basic drivers in a software project

The aim of most methodologies is to deliver software with a high Q&P



Besides the need to achieve high Q&P there are some other needs

Roles of people (Stakeholders) in software

- people involved in software production
  - **Customer** /client: wants software built
    - often does not know what he/she wants
  - **Managers** / designers: plan software
    - difficult to foresee all problems and issues in advance
  - **Developers**: write code to implement software
    - it is hard to write complex code for large systems
  - **Testers**: perform quality assurance (QA)
    - it is impossible to test every combination of actions
  - **Users**: purchase and use software product
    - users can misunderstand the product

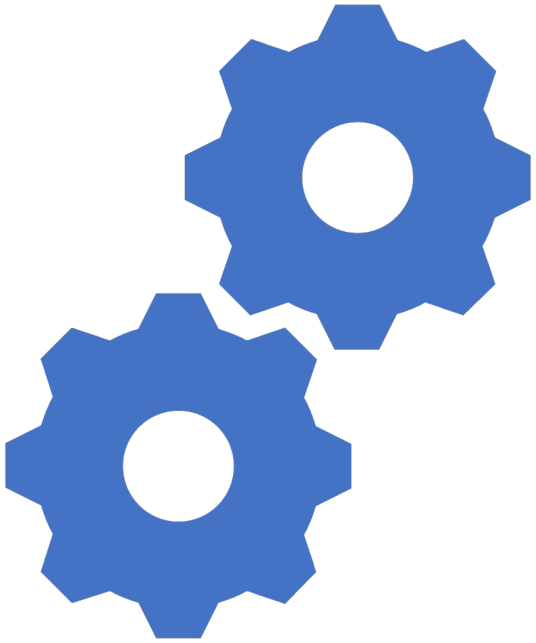
Software Development Fundamentals

- **Process:** A particular method, generally involving a number of steps. A process is a Collection of activities, actions and tasks performed when some product is to be created.

- **Software Process:** A set of steps or activities, along with ordering constraints on execution, to produce software with desired outcome

- Many types of activities performed by different people
- Software process is comprising of many component processes

Software Development Process



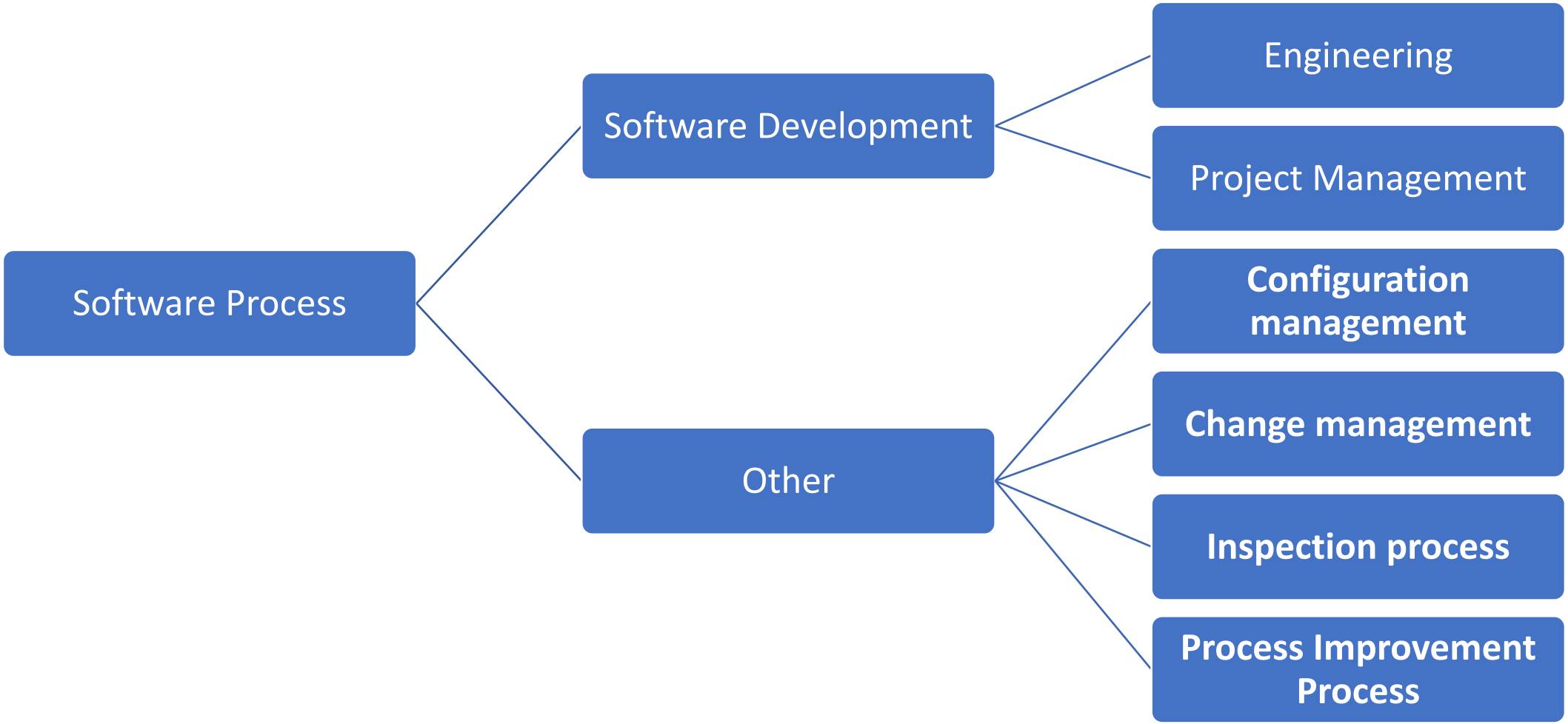
Process is distinct from product

Products are outcomes of executing a process on a project

Software Engineering focuses on process

Proper processes: Purpose is to deliver software product in timely manner, within project with sufficient quality to satisfy clients/end users (i.e. high QP)

Key Processes

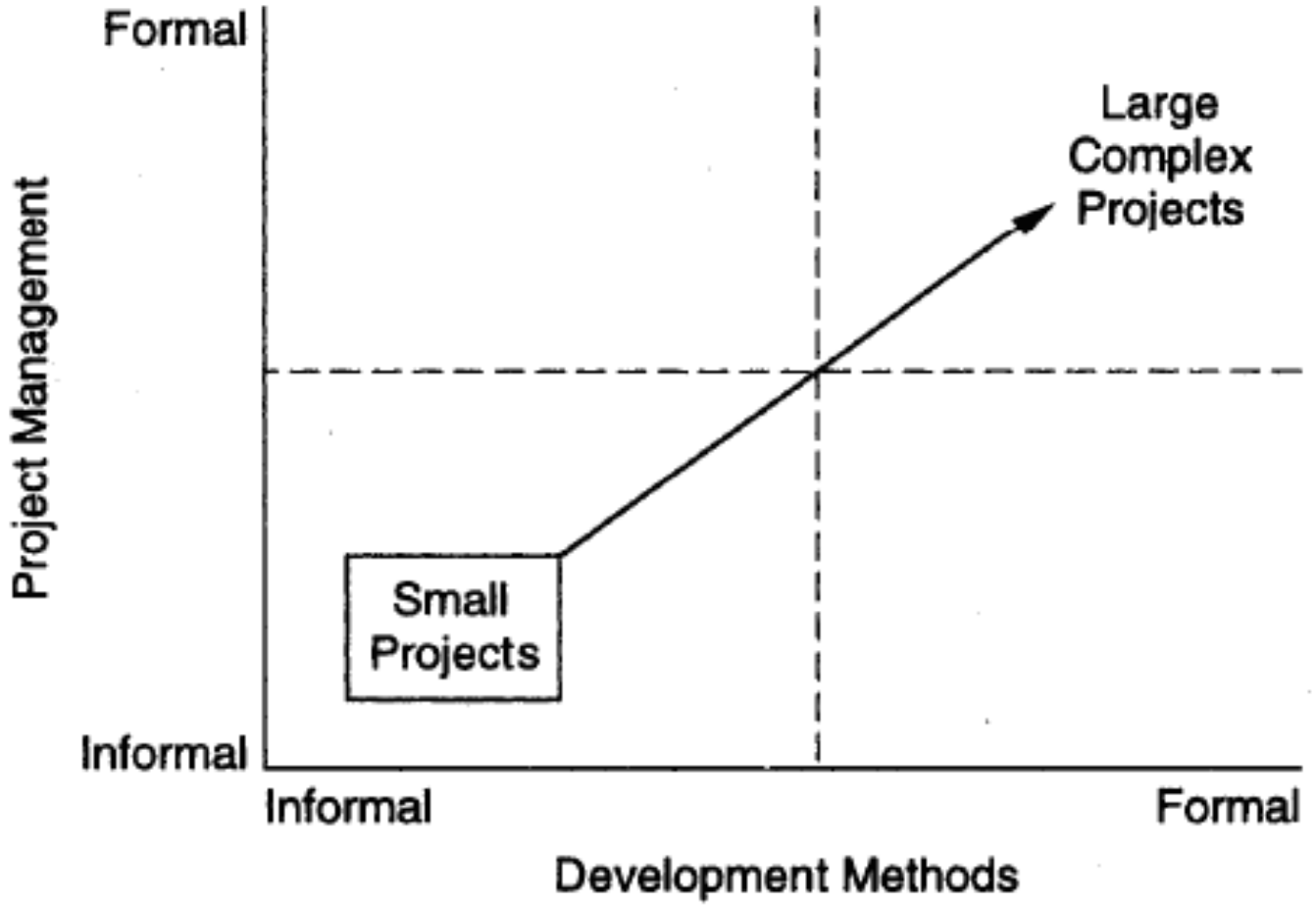


Software Development Process

- **Two major processes**
  - **Engineering**– “development and quality steps needed to engineer the software”
  - **Project management** – “planning and controlling the development process”

- Key Roles
  - Developers execute Engineering process
    - Software architects, lead developers, ...
  - Project manager(s) executes the management process

Software Development



Software Development Process

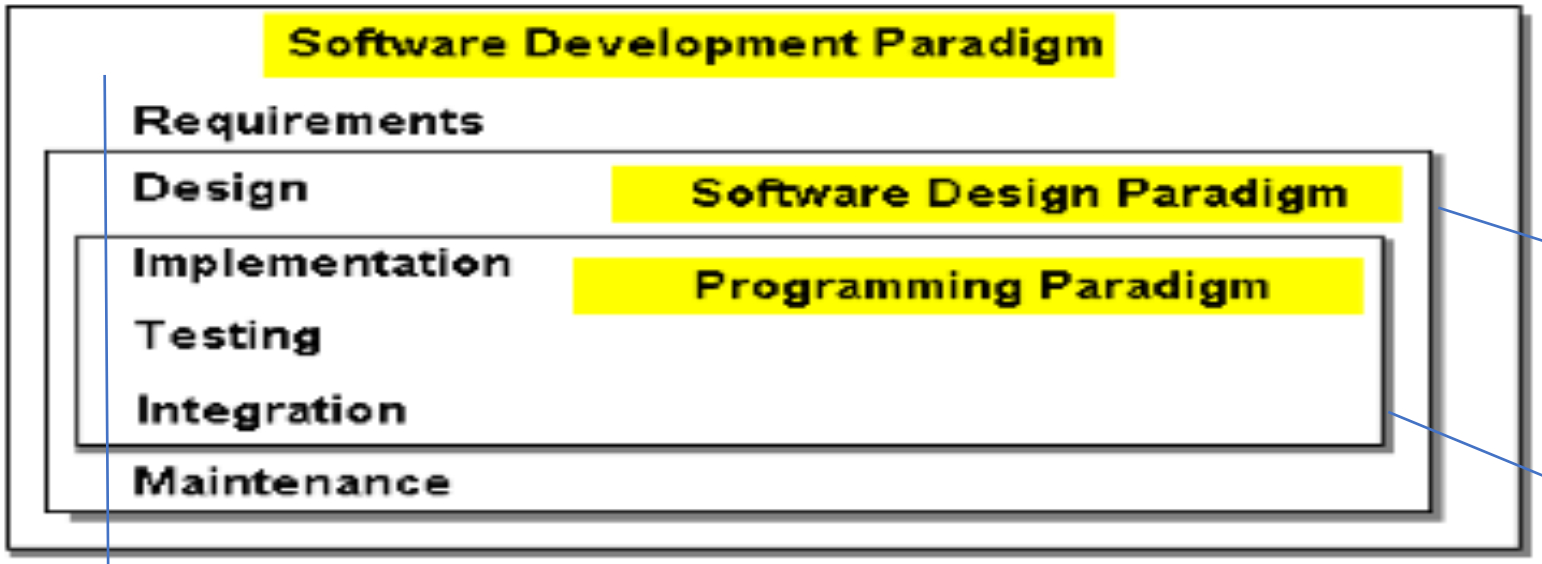
- A set of phases and each phase being a sequence of steps
- For each phase there are
  - A variety of methodologies
  - Corresponding to different sequence of steps for a phase

- Why have phases?
  - To employ divide and conquer
  - Each phase handles a different part of the problem
  - Helps in continuous validation

Software Development Process

- Commonly has these activities/phases:
  1. Requirements analysis
  2. Design
  3. Coding
  4. Testing
  5. Delivery

Basic Software Development Process



Planning/Design & Documentation

Software Design Models

1. **Procedural**
2. **Object Oriented**
3. Logic
4. Functional

Phase 1: Requirement Analysis

State the problem precisely!

Forms the basis of agreement between user and developer

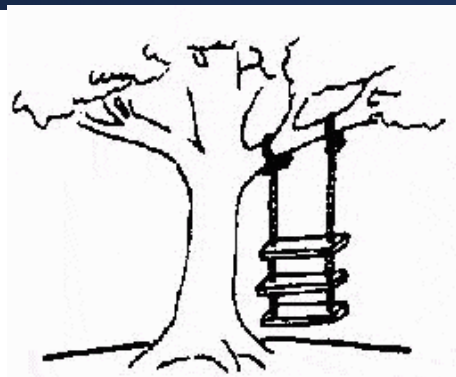
Specifies “**what**” not “**how**”

Hard task - needs often not understood well

Requirement specifications of even medium systems can be many hundreds of pages

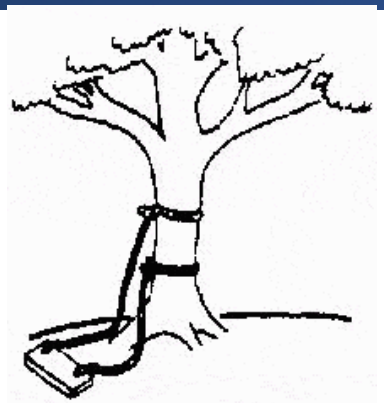
Output is the Software Requirements Spec (SRS) document

Requirements Analysis

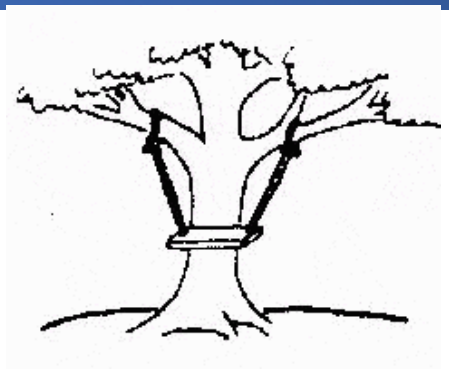


The requirements specification was defined like this

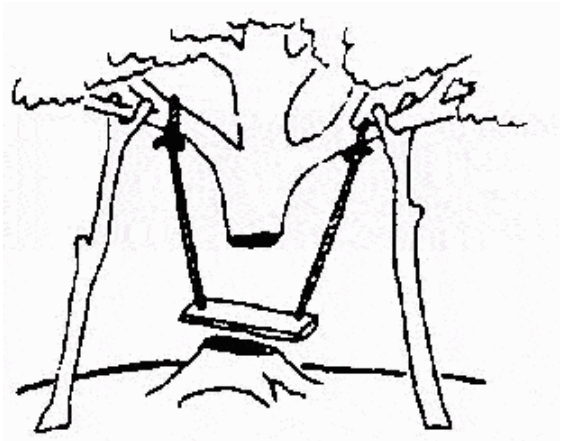
The developers understood it in that way



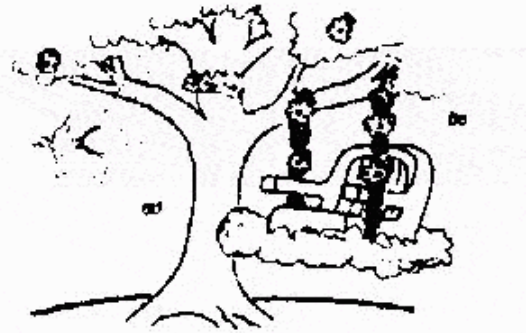
This is how the problem was solved before.



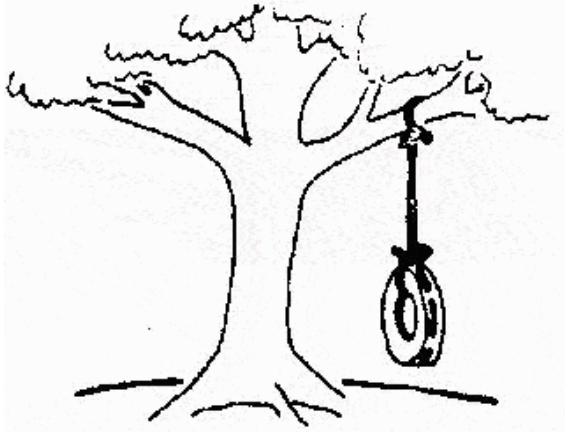
This is how the problem is solved now



That is the program after debugging



This is how the program is described by marketing department



This, in fact, is what the customer wanted … ;-)

Phase 2: Design

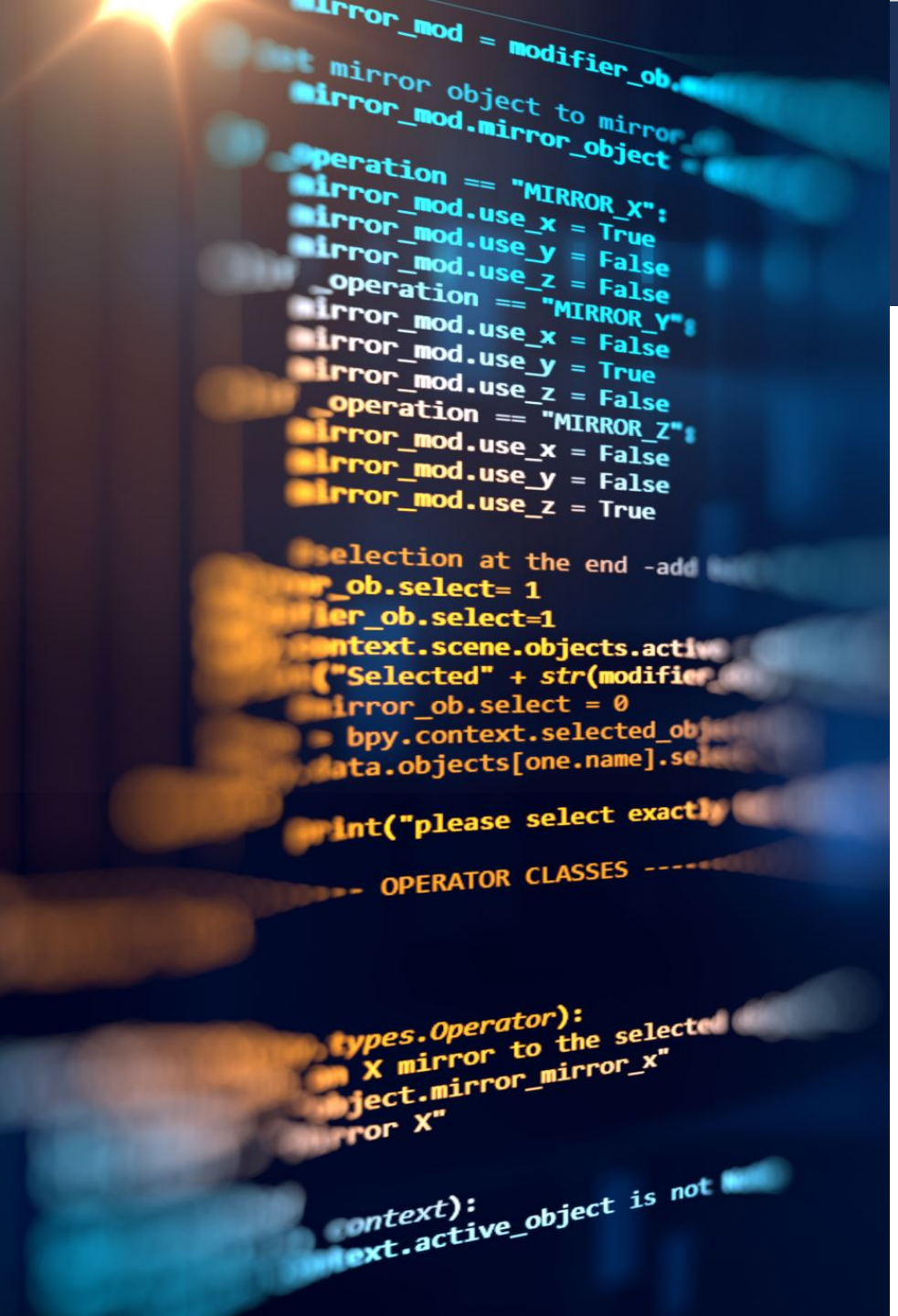
A major step in moving from problem domain to solution domain

Three main tasks

Architecture design – components and connectors that should be there in the system
High level design – modules and data structures needed to implement the architecture
Detailed design – logic of modules

Most methodologies focus on architecture or high-level design

Outputs are architecture/description/logic design docs



Phase 3: Coding

- Converts design into code in specific language
- **Goal:** Implement the design with simple and easy to understand code

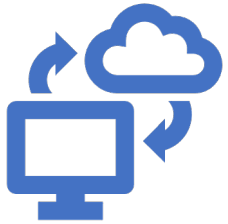
- Coding phase affects both testing and maintenance
  - Well written code reduces testing and maintenance effort
- Output is code

Phase 4: Testing & Quality Assurance

- Defects are introduced in each phase
- Must be found and removed to achieve high quality
- **Goal:** Identify most of defects
- Very expensive task; must be properly planned and executed
- Outputs are
  - Test plans/results, and
  - the final tested (hopefully reliable) code

Phase 5: Delivery

What the “Operations” group does.



Varies by distribution model

Shrink Wrapped Software

In house software

Web-based

Software As A Service (SaaS)

…

From a user’s perspective, it may be as important as design!

Software Development Lifecycle

Typical Effort Distribution

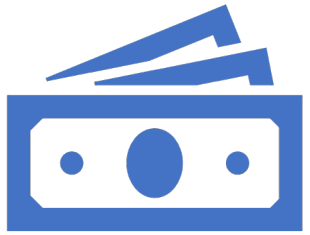
Distribution of effort :

Req. - 10-20%

Design - 10-20%

Coding - 20-30%

Testing - 30-50%



Coding is not only the most expensive!

When are defects introduced?

Distribution of error occurrences by phase is

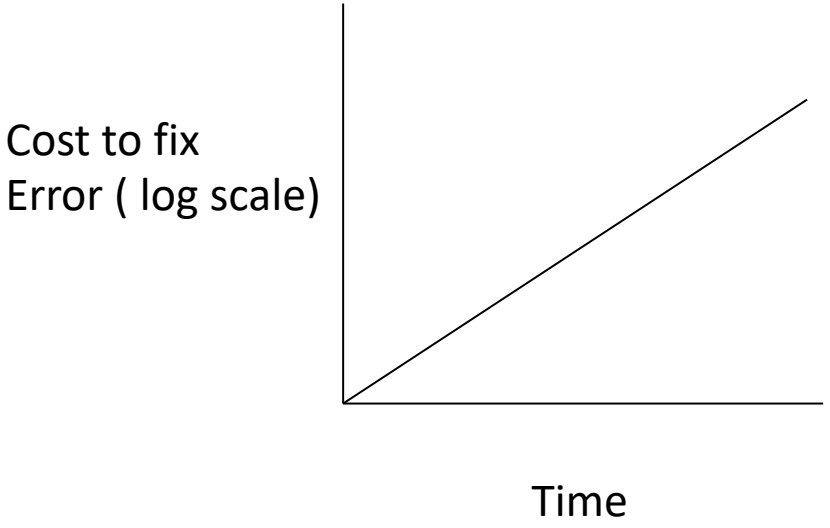
Req.	- 20%
Design	- 30%
Coding	- 50%

Defects can be injected at any of the major phases.

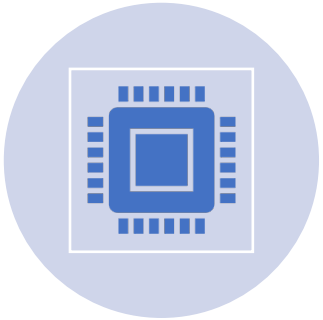
Cost of latency: Cost of defect removal increases exponentially with latency time.

Defects...

- Cheapest way to detect and remove defects close to where it is injected.
- Hence must check for defects after every phase.

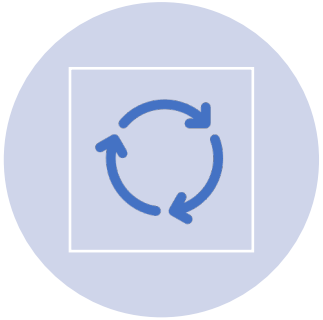


Software Project and Process Model

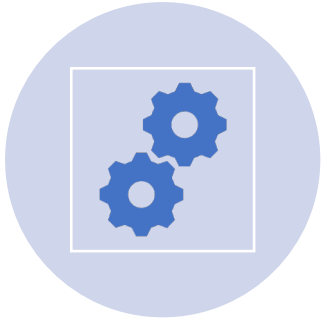


Project - to build a software system within cost and schedule and with high quality which satisfies the customer

A process model specifies a general process that is optimal for a class of problems. A process structure well suited for a class of projects. Best practices -> recipe for success.



Different process models perform the 5 phases of process development in different manner!



A project may select its process using one of the process models

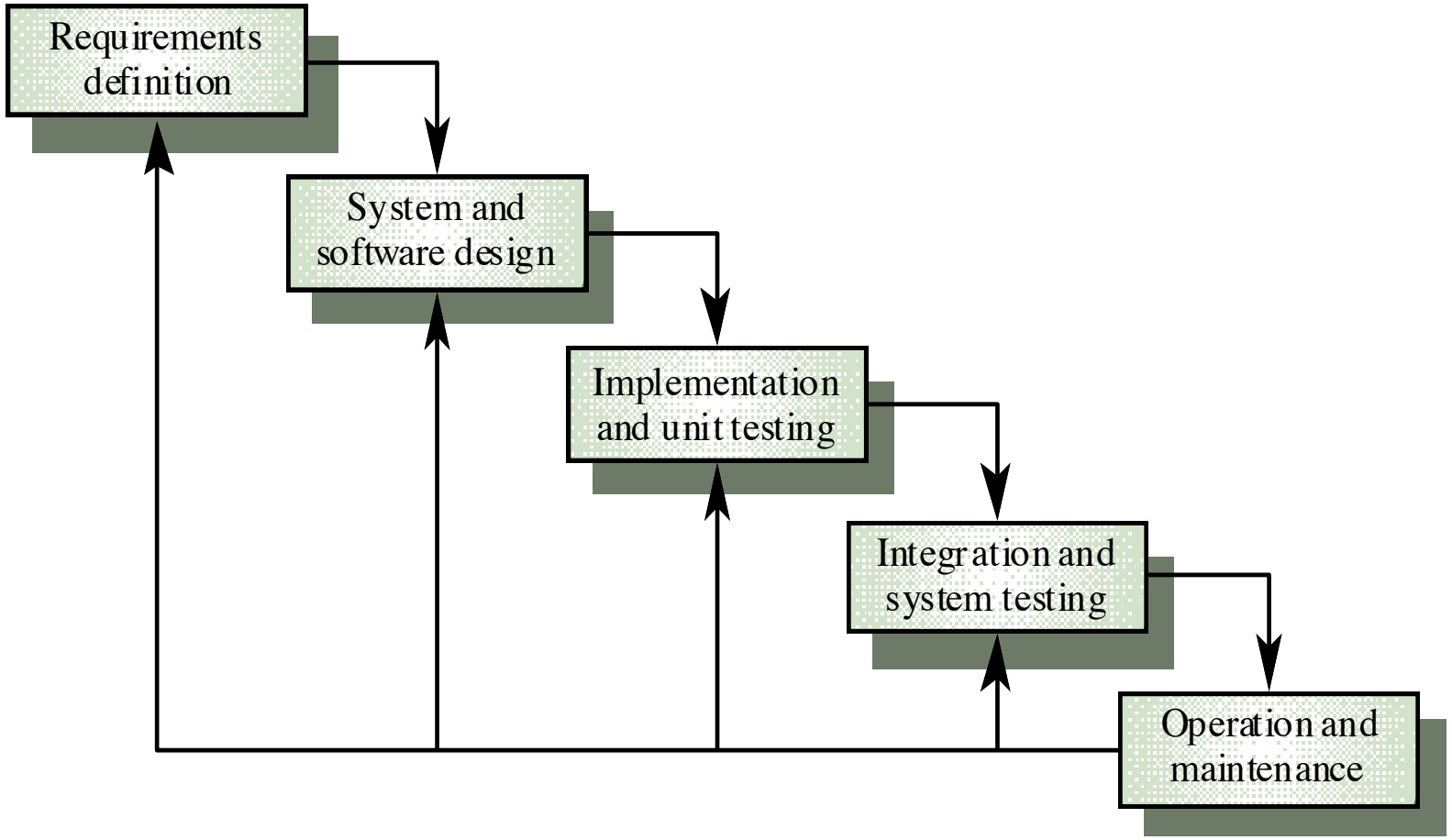
Process Models

- **A software process model** is an abstract representation of a process .It presents a description of a process from some particular perspective
  - **Traditional Modes**
    - **Waterfall** – the oldest and widely used
    - **Prototyping** – Prototype, followed by Waterfall
    - **Iterative** – used widely in product dev
    - **Timeboxing** – Iterative 2.0
  - **Modern Models**
    - **Agile** - Lightweight" methodologies (Speaker), Agile **model** is best suitable for mobile **applications**

1. Waterfall Model

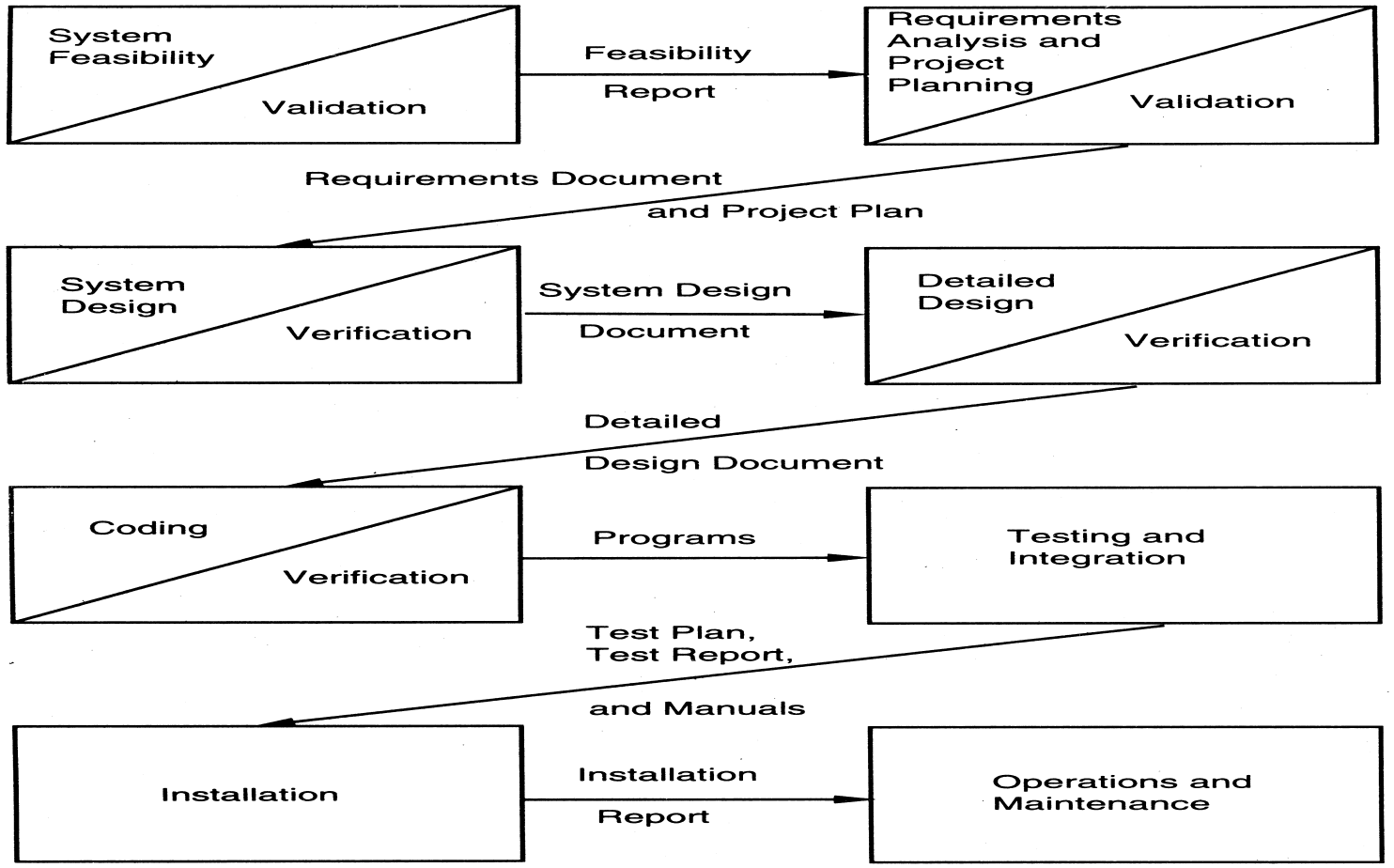
- Linear sequence of stages/phases
  - Requirements -> High Level Description –> Detailed Design –> Code –> Test –> Deploy
- A phase starts only when the previous has completed; no feedback!
- The phases partition the project, each addressing a separate concern

1. Waterfall Model



1. Waterfall Model

- Linear ordering implies each phase should have some output
- The output must be validated/certified
- Outputs of earlier phases: work products
- Common outputs of a waterfall: SRS, project plan, design docs, test plan and reports, final code, supporting docs



1. Waterfall Model Advantages and Disadvantages

Advantages

- Natural approach for problem solving
- Conceptually simple, divides the problem into distinct independent phases
- Easy to administer in a contractual setup – each phase is a milestone

Disadvantages

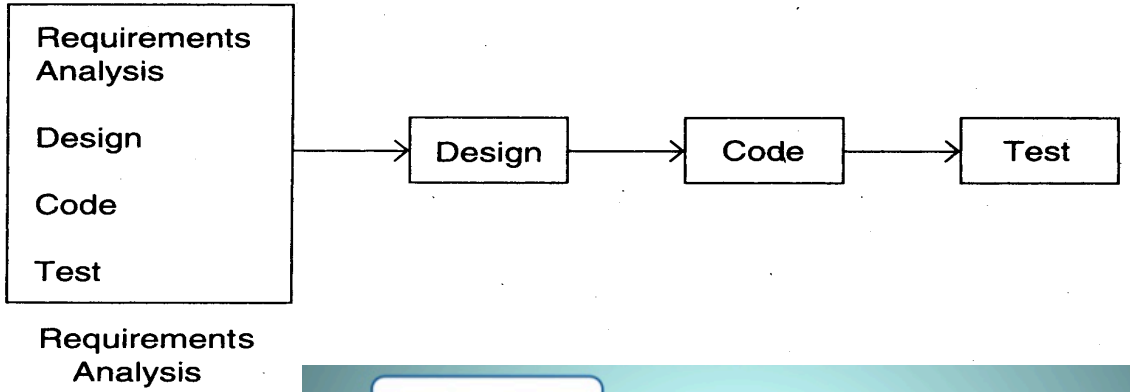
- **Inflexible partitioning** of the project into distinct stages
- This makes it difficult to respond to changing customer requirements
- Therefore, this model is only appropriate when the requirements are well-understood

Waterfall model describes a process of stepwise refinement

- Widely used in **military, healthcare** and **aerospace** industries

2. Prototyping Model

- Used for risky/unfamiliar projects
- Addresses the requirement specification limitation of waterfall
- Instead of freezing requirements only by discussions, a prototype is built to understand the requirements
- Helps alleviate the requirements risk, Expect to “throw away” first version
- A small waterfall model replaces the requirements stage



2. Prototyping Model: Development of a prototype

- Starts with initial requirements
- Only key features which need better understanding are included in the prototype
- No point in including those features that are well understood
- Feedback from users taken to improve the understanding of the requirements

- Cost can be kept low
  - Build only features needing clarification
  - Things like exception handling, recovery, standards are omitted
  - Cost can be a few % of the total
  - Learning in prototype building will help in building, besides improved requirements

2. Prototyping Model Advantages and Disadvantages

Advantages

- Requirement will be more stable and more likely to satisfy user needs
- Early opportunity to explore scale/performance issues
- Ability to modify or cancel the project early
- Enhanced user engagement

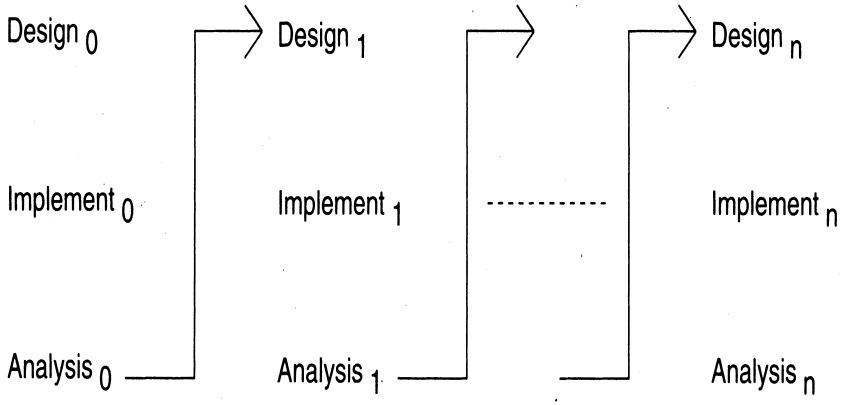
Disadvantages

- Potential hit on cost and schedule
- Potential false sense of security if prototype does not focus on key (high risk) issues

- Applicability:
  - When requirements are hard to elicit
  - When confidence in requirements is low
  - Where requirements are not well understood
  - When design is driven by user needs

3. The Iterative Model

- Counters the “all or nothing” drawback of the waterfall model
- Combines benefit of prototyping and waterfall
- Develop and deliver software in increments
- Each increment is complete in itself
- Can be viewed as a sequence of waterfalls
- Feedback from one iteration is used in the future iterations



- Most Software Products follow it
- Used commonly in customized development
  Newer approaches like XP, Agile,… all rely on iterative development

3. Iterative Model Advantages and Disadvantages

Advantages

- Get-as-you-pay
- Feedback for improvement

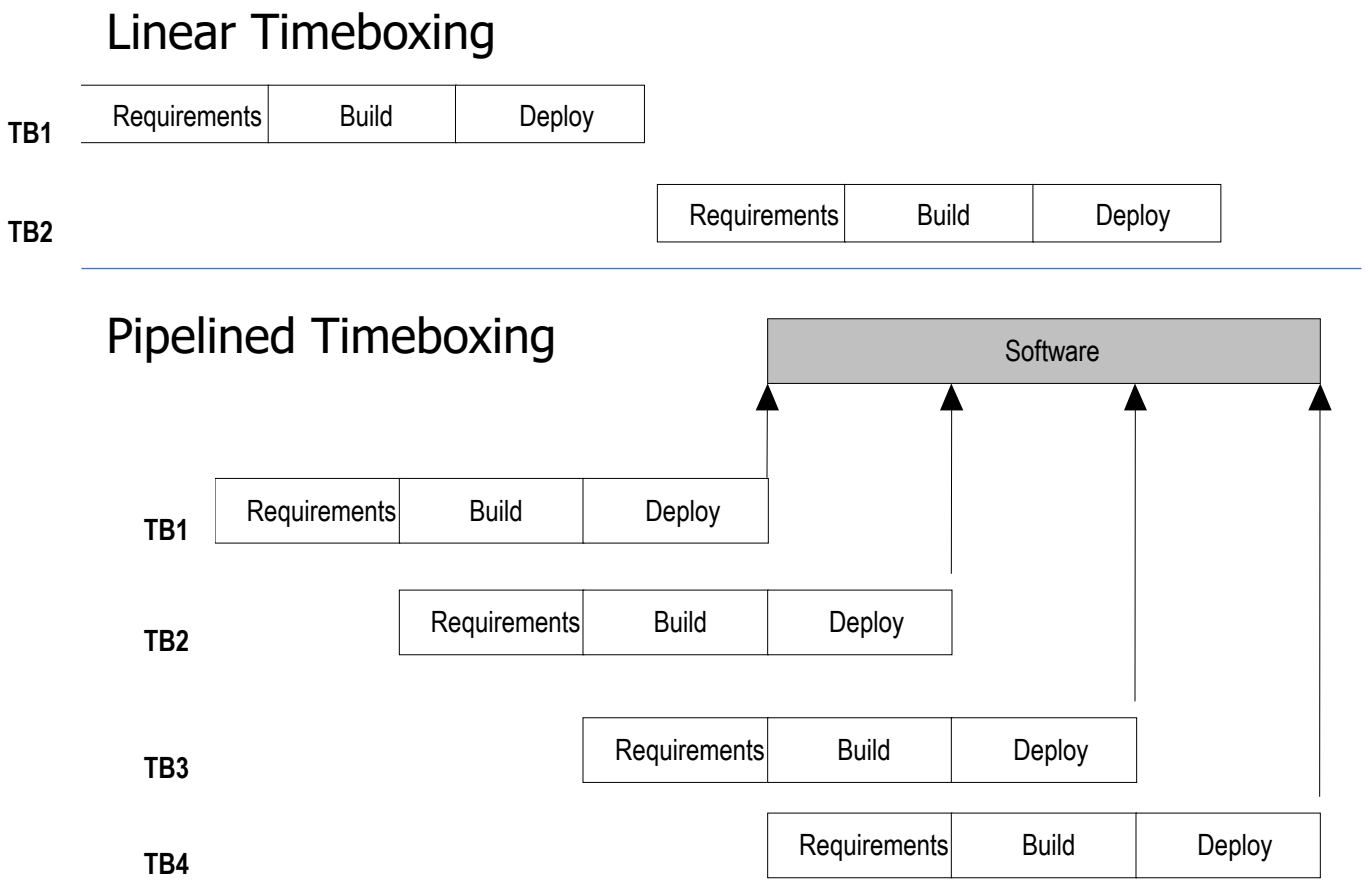
Disadvantages

- Architecture/design may not be optimal
- Amount of refactoring may increase
- Total cost may increase

- Applicability:
  - where response time is important
  - All requirements are not known
- Execution
  - Each iteration is a mini waterfall – decide the specs, then plan the iteration
  - Length of iteration driven by amount of new functionality to be added in an iteration

4. Time boxing

- Time boxing is like Iterative development but
  - fix an iteration duration, then determine the specs
- Divide iteration in a few equal stages
- Use pipelining concepts to execute iterations in parallel



4. Time boxing

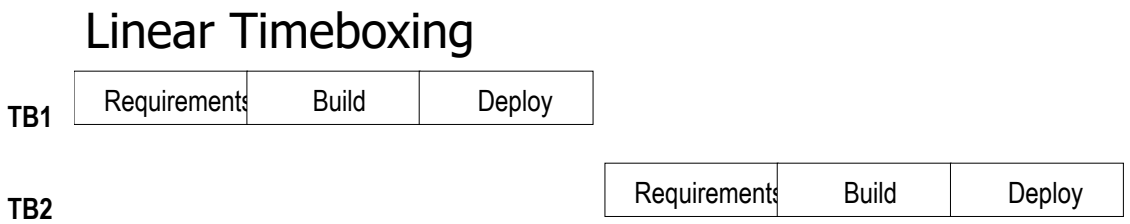
- General iterative development – fix the functionality for each iteration, then plan and execute it
- In time boxed iterations – fix the duration of iteration and adjust the functionality to fit it
- Completion time is fixed, the functionality to be delivered is flexible

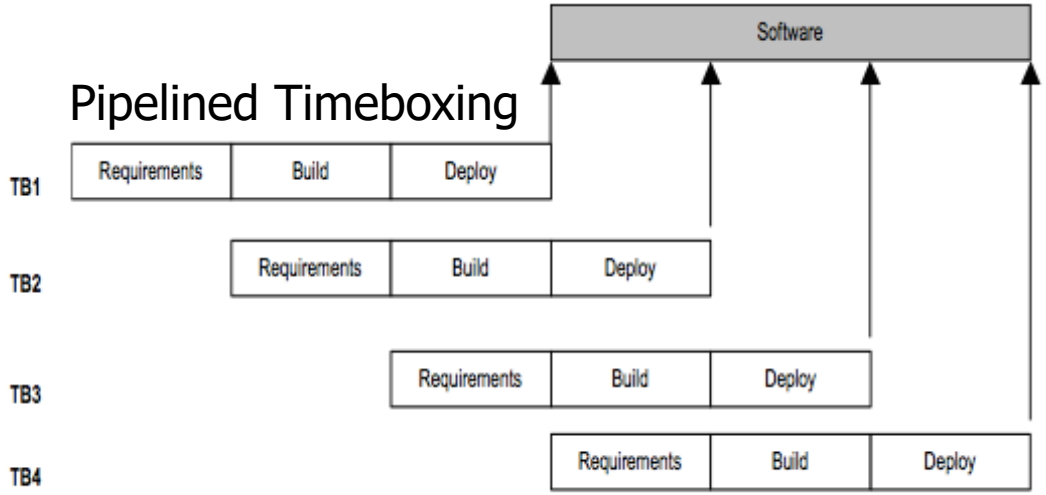
4. Time boxing: Example

An iteration with three stages – Requirements (R), Build (B), Deploy (D)

- These stages are appx equal in many situations
- Can adjust durations by determining the boundaries suitably
- Can adjust duration by adjusting the team size for each stage
- Have separate teams for R, B, and D

- Each iteration takes time T, Assume T is 3 weeks,
- In Linear Timeboxing
  - Delivery times will be 3 wks, 6 wks, 9 wks,…
- In Pipelined Timeboxing
  - First iteration finishes at time T
  - Second finishes at T+T/3; third at T+2 T/3, and so on
  - In steady state, delivery every T/3 time
  - first delivery after 3 wks, 2nd after 4 wks,3rd after 5 wks,…





4. Time boxing Execution and team Sizes

- Duration of each iteration still the same
- Total work done in a time box is also the same
- Productivity of a time box is same
- Yet, average cycle time or delivery time has reduced to a third

Team Size

- In linear execution of iterations, the same team performs all stages
- If each stage has a team of size S, in linear execution the total team size is S
- In pipelined execution, the total team size is three times (one for each stage)
- Total team size in timeboxing is larger; and this what reduces cycle time

- Timeboxing allows structured way to add manpower to reduce cycle time
- Note that we cannot change the time of an iteration – Brook's law still holds
- Work allocation different to allow larger team to function properly

4. Time boxing Model Advantages and Disadvantages

Advantages

- Shortened delivery times by use of additional manpower, Increases flexibility

Disadvantages

- Larger teams,
- project management is harder
- high synchronization needed

- Applicability:
  - When short delivery times is VERY important; architecture is stable; flexibility in feature grouping

5. The Agile Model

- Key Assumptions
  - Difficult to predict software requirements
  - Difficult to predict analysis, design, construction, and testing
  - Design and construction should be interleaved
- How can we design a process that can manage unpredictability?
  - Process adaptability.
- Example: Extreme Programming (XP)
  - 4 phases: Planning (stories), Design (prototype solutions), Coding (pair programming, re-factoring), Test
  - The tests are the specification
  - Communication paramount (small team, knowledgeable programmers)

Summary

- Process is a means to achieve project objectives of high Q&P
- Process models define generic process, which can form basis of project process
- Process typically has stages, each stage focusing on an identifiable task
- Many models for development process have been proposed
- A project should select a process model that is best suited for it (and tailor it to meet its requirements)

Next Class

- Software Modeling Tools
- Introduction to Requirements Analysis and Specifications