COE691: Software Requirements Analysis and Specifications

Course Outline

- ➔ Passing: Need 50% separately in theory and lab components
- ➔ Mandatory Lab Submissions during lab sessions
- ➔ Course Resources: Textbooks, Eclipse IDE (CDT), Violet UML Editor
- ➔ Course Objectives: Familiarize with requirement methods, elicitation, analysis, ethics
- ➔ Course Topics: Software engineering basics, requirements engineering, analysis, verification, validation, risk assessment
- ➔ Weekly Quizzes: Mondays at 10:10 am on D2L, 5-10 minutes
- ➔ Weekly Lectures: 3 hours, mix of Synchronous and Asynchronous, Mondays 10 am-1 pm
- ➔ Weekly In-person Labs: 2 hours each, starting week 2, check D2L
- ➔ Mandatory Lab Attendance
- ➔ Tentative Midterm: Week 8, March 4, 2024, 2 hours, 10:30 am - 12:30 pm
- ➔ Online Final Exam: During Exam Period, TBD
- ➔ Evaluation Breakdown:
  - ☐ Labs (5 Labs Total Each 5 Marks): 25%
  - ☐ Midterm (Online): 30%
  - ☐ Quizzes (Online Every Week): 10%
  - ☐ Final (Online TBA): 35%

Lecture 1

- ➔ (Week 1 - Jan 8, 2024)

Software: A collection of executable programming code, libraries, and documentation, encompassing large, robust, reusable, and evolving components.

- ➔ Why Study Software: To learn how to design and engage in systematic problem-solving.

Types of Software:

1. **System Software:** Tools like compilers, editors, and file management utilities.
2. **Application Software:** Stand-alone programs designed for specific needs.
3. **Engineering/Scientific Software:** Involves algorithms related to scientific or engineering tasks.
4. **Embedded Software:** Software residing within a product or system.
5. **Product-line Software:** A group of software targeting a specific marketplace.
6. **Web & Network-centric Software:** Developed in sophisticated environments, often integrated with remote databases and business applications.
7. **Intelligent Systems:** Software using non-numerical algorithms to solve complex problems, such as robotics and expert systems.

What Is Software Engineering (SE):

- The application of a systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of software, applying engineering principles.
  — IEEE Standard 610.12
- "The systematic activities involved in the design, implementation and testing of software to optimize its production and support" — Canadian Standards Association

SE Scope:

- Part of larger system design activities, considering system engineering issues and interface with other systems and users.
- **Understanding Application and User Needs:** Essential for decision-making about which activities should be supported by the system and how.
- **Different Domains:** Emphasize different priorities, like time-to-market, safety, or maintainability.

The Software Problem:

- Three Key Forces: Cost, Schedule, Quality.

1. Cost:

- Cost per LOC (Line of Code): Ranges from $3 to $10.
- Examples: A simple business application may have 20KLOC to 50KLOC.
  - Productivity: Output/input resources ratio, with output measured in LOC or KLOC (thousands of lines of code), and input resources measured in person-months.
  - (Cost = $100K to $2.25Million and Can easily run on $10K - $20K hardware)
- Significance: Software costs outweigh hardware costs in IT solutions.

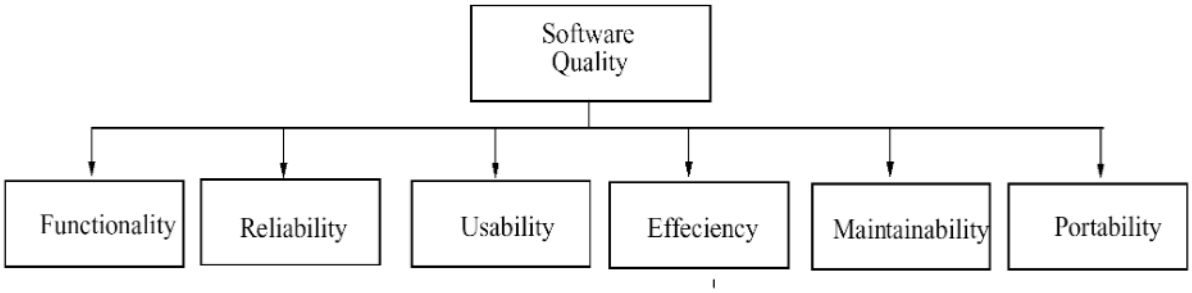
2. Schedule:

- Business Demands: Short delivery times for software.
- Historical Issue: In the past, software products often failed to meet deadlines.
- Productivity: Measured as output/input resources, where output can be LOC or KLOC, and input resources are person-months.
  - Cycle Time: The time taken to complete a software project.

3. Software Quality:

- Quality is Key: The other major driving factor besides productivity.
  - Refers to the overall excellence, reliability, and performance of the software.
- Objective Hard to Define: Quality is harder to define than cost and schedule.
- Goal: Developing high-quality software is a basic goal in software engineering.

Quality — ISO standard: 6 Attributes



Cost	Schedule	Productivity = output/input resources	Quality
Measured by LOC	Cycle time is a fundamental driving force	Higher productivity leads to lower cost Higher productivity leads to lower cycle time	6 Attributes

Quality and Productivity (Q&P):

- Q&P are the core drivers in software projects.
- Aim of most methodologies is to deliver high Q&P software.

Role of stakeholders:

➔ (People involved in software production)

1. **Customer**/client wants software but may not know what they want.
2. **Managers**/designers plan software but can't foresee all problems.
3. **Developers** write code for large systems, which is challenging.
4. **Testers** perform quality assurance but can't test every action.
5. **Users** purchase and use the software product, sometimes misunderstanding it.

Software Development Fundamentals:

- Process involves a set of steps or activities to create a product.
- Software process includes many component processes.
- Software engineering focuses on the process.

Software Development Process:

- Process is distinct from the product.
- Software engineering emphasizes proper processes to deliver timely, high-quality software.
- A set of steps with ordering constraints to produce desired software outcomes.

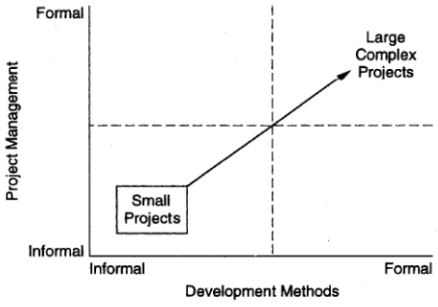
Process is distinct from product

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)

- **Two major processes:** Engineering (development and quality steps) and Project management (planning and control).
- **Key Roles:** Developers execute engineering process, project managers execute management process.



Key Processes:

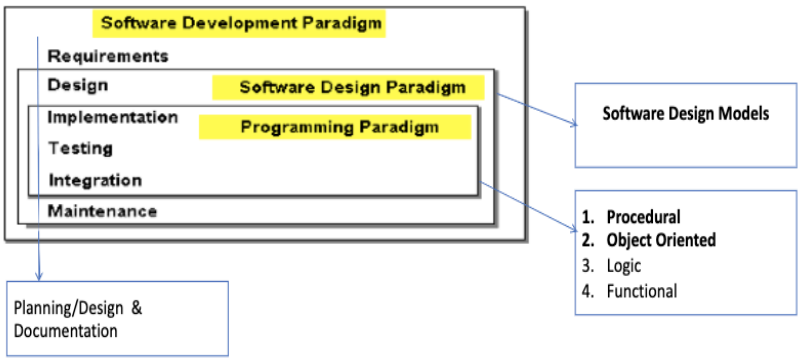
Phases of Software Development:

- A set of phases, each consisting of a sequence of steps.
- Phases are employed to divide and conquer the software development process, handling different parts of the problem.
- Helps in continuous validation.

Common Software Development Phases:

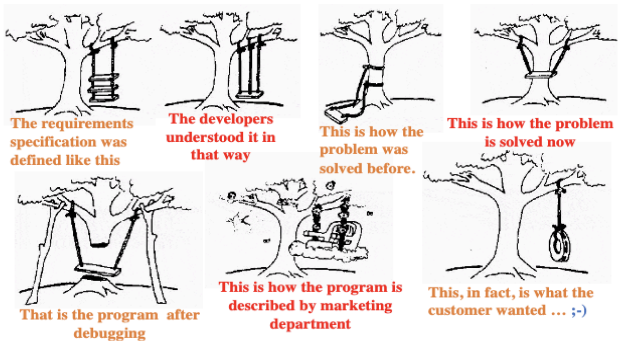
➔ 5 Phases to software development process

1. Requirements analysis
2. Design
3. Coding
4. Testing
5. Delivery



Phase 1: Requirements 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



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: Design

- 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: Coding

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!

Effort Distribution:

- Distribution of effort in the software development process:
  - Requirements: 10-20%
  - Design: 10-20%
  - Coding: 20-30%
  - Testing: 30-50%
- Coding is not only the most expensive phase.

Defect Introduction:

- Distribution of error occurrences by phase:
  - Requirements: 20%
  - Design: 30%
  - Coding: 50%
- Defects can be injected at any major phase.
- Cost of defect removal increases exponentially with latency time.

Cost-Efficient Defect Removal:

- The cheapest way to detect and remove defects is close to where they are injected.
- Therefore, checking for defects after every phase is essential.

Software Project and Process Model:

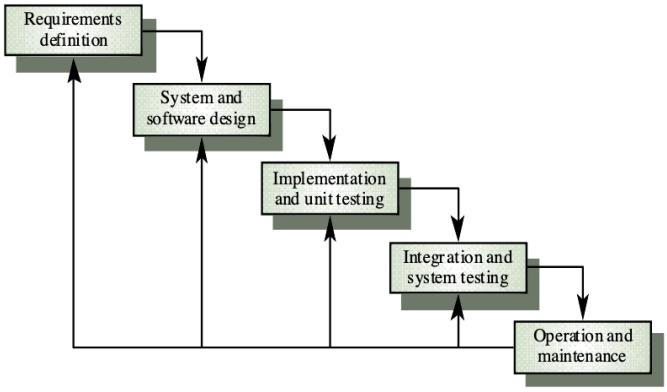
- Project's goal: To build a software system within cost, schedule, and high quality that satisfies the customer.
- A process model specifies a general process optimal for a class of problems.

Process Models:

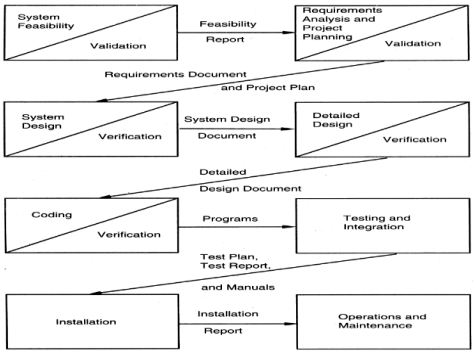
- Software process model: Abstract representation of a process.
- Traditional Models:
  1. Waterfall – the oldest and widely used
  2. Prototyping – Prototype, followed by Waterfall
  3. Iterative – used widely in product dev
  4. Timeboxing – Iterative 2.0
- Modern Models: Agile (best for mobile applications).

1. Waterfall Model:

- Linear sequence of stages/phases: Requirements, High-Level Description, Detailed Design, Code, Test, Deploy.
- Phases start only after the previous one is completed, with no feedback.
- Widely used in military, healthcare and aerospace industries
- Advantages: Natural approach, conceptually simple, easy to administer in a contractual setup.
- Disadvantages: Inflexible, challenging to adapt to changing requirements, suitable only when requirements are well-understood.

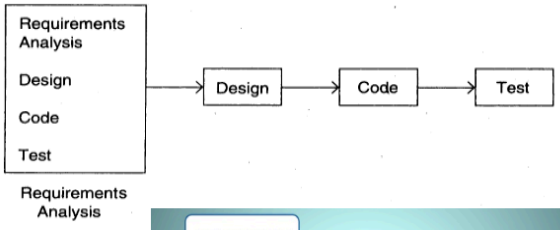


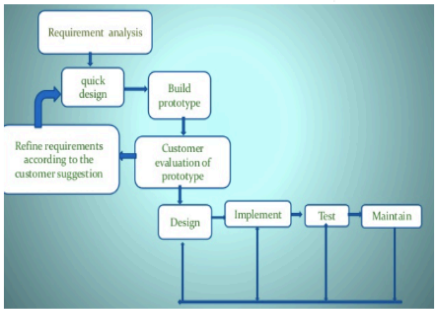
- Linear ordering implies each phase should have some output
- The output must be validated/certified
- Outputs of earlier phases: work products
- Common outputs of a waterfall: SRS, project plan, design docs, test plan and reports, final code, supporting docs



2. Prototyping Model:

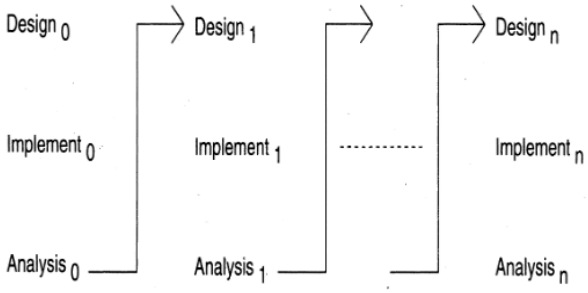
- Purpose: Used for risky or unclear projects.
- Approach: Builds a prototype to clarify and refine requirements.
- Expectation: Expects to "throw away" the initial version.
- Key Features: Focuses on including key features needing clarification.
- User Feedback: Essential for improving requirements understanding.
- Cost: Prototype cost is a small percentage of the total project.
- Advantages: Stable requirements, early exploration of issues, project modification, enhanced user engagement.
- Disadvantages: Potential cost/schedule impact, false security, applicable in challenging requirement scenarios.





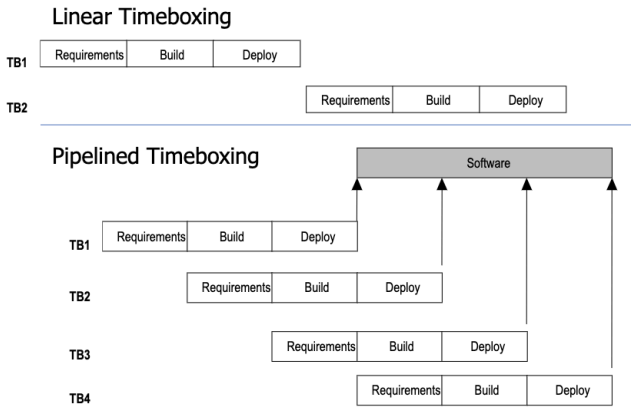
3. Iterative Model:

- Purpose: Addresses the "all or nothing" drawback of the waterfall model.
- Approach: Combines benefits of prototyping and waterfall.
- Development: Software is developed and delivered in increments.



- Incremental: Each increment is complete in itself.
- Feedback: Feedback from one iteration informs future iterations.
- Used commonly in customized development Newer approaches like XP, Agile,... all rely on iterative development
- Advantages: Get-as-you-pay, continuous feedback for improvement.
- Disadvantages: Suboptimal architecture/design, potential cost increase.
- Applicability: Suitable when response time is crucial, requirements are unclear, and execution involves mini-waterfalls for each iteration.

4. Time Boxing Model:



- Approach: Utilizes pipelining concepts for parallel execution of iterations.
- Iteration Division: Divides each iteration into a few equal stages.
- Comparison to Iterative Development:
    - In general iterative development, functionality is fixed for each iteration, and then planning and execution occur.
    - In time-boxed iterations, the duration of the iteration is fixed, and functionality is adjusted accordingly.
    - Completion time is fixed, but the functionality to be delivered is flexible.

➔ Key Points:

- Duration of each iteration remains the same.
- Total work done in a time box remains constant.
- Productivity within a time box remains consistent.
- Despite these, average cycle time or delivery time is significantly reduced to a third.

➔ Execution and Team Sizes:

- In linear execution of iterations, the same team handles all stages.
- In pipelined execution, the total team size is three times (one for each stage).
- Timeboxing involves a larger total team size, which reduces cycle time.
- Timeboxing allows structured addition of manpower to reduce cycle time.
- Brook's law still applies; you can't change the duration of an iteration.

➔ Advantages:

- Shortened delivery times through the use of additional manpower.
- Increased flexibility.

➔ Disadvantages:

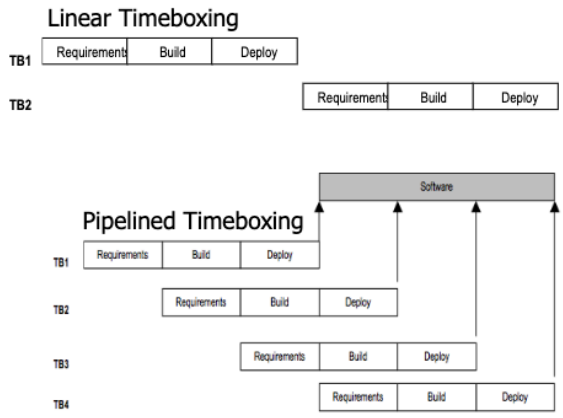
- Requires larger teams, making project management more challenging.
- High synchronization among teams is needed.
- Applicable when short delivery times are extremely important, the architecture is stable, and flexibility in feature grouping is required.

Time Boxing Model 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,…



5. Agile Model:

- Key Assumptions:
  - Difficult to predict software requirements.
  - Difficult to predict analysis, design, construction, and testing.
  - Design and construction should be interleaved.
- Process Approach: Agile focuses on process adaptability to manage unpredictability.
- Example Methodology: Extreme Programming (XP):
  - Phases:
    - Planning (stories).
    - Design (prototype solutions).
    - Coding (pair programming, re-factoring).
    - Test.
  - Specification: The tests serve as the specification in XP.
  - Communication: Paramount importance given to communication within a small team of knowledgeable programmers.

Summary:

- The process is a means to achieve high-quality and productive project outcomes.
- Process models define generic processes that serve as a foundation for project-specific processes.
- A process typically consists of stages, with each stage focusing on a specific task.
- Numerous development process models have been proposed, each with its own approach.
- It's essential for a project to select the process model that best suits its needs and customize it to meet its specific requirements