Mohsin Iban Hossain

AIUB, Software Engineering Notes

SOFTWARE ENGINEERING

Table of Contents

CH	Topic	Pages
1	Software & Software engineering	03-15
2	Software Development process model	16-26
3	Agile software development	27-36
4	Extreme programming (XP)	37-45
5	SCRUM	46-54
6	The Dynamic Systems Development Method (DSDM)	55-68
7	Feature Driven Development (FDD)	69-78
8	Requirement's engineering	79-89
#	Possible Sample Questions	90-96

SOFTWARE & SOFTWARE ENGINEERING

Ch: 1

Software engineering

➔ Software engineering is the art and science of building **powerful, reliable,** and **innovative** software solutions. It combines creativity with engineering principles to **design**, **develop**, and **optimize software** that enhances user experiences, drives technology forward, and solves real-world problems efficiently.

System

➔ A system is a **collection of components** that work together to achieve a specific goal. In software, a system refers to an integrated set of programs and processes designed to perform tasks efficiently. However, if a system is **poorly designed** or **maintained**, it can fail, leading to high costs, inefficiency, and user rejection.

Why system fails?

➔ Key Failure Points:

- **Doesn't meet business needs** → Leads to abandonment or costly maintenance.
- **Performance issues** → Fails user needs, requiring extra costs for fixes.
- **Errors & unexpected problems** → Needs costly patches.
- **User rejection** → Due to lack of involvement or commitment.
- **Becomes unmaintainable** → Initially works but later falls into disuse.

Software Crisis & Software Engineering

➔ **Software crisis** refers to challenges in software development, leading to:

- **Late delivery** → Projects take longer than expected.
- **Over budget** → Costs exceed initial estimates.
- **Low quality** → Software has many faults and issues.

Despite advancements, the software crisis persists after 35+ years. Software engineering aims to address these problems by improving development processes.

Software characteristics

- **Intangible** → Logical, not physical.
- **Engineered, not manufactured** → Developed through design, not production.
- **Cost in engineering** → Focus on development, not materials.
- **Deteriorates over time** → Needs updates, not replacement.
- **Differentiator** → Varies across systems (e.g., cashier workstation).
- **No spare parts** → No unnecessary features.
- **Mostly custom-built** → Tailored to specific requirements.

Computer science vs. Software engineering

- **Computer Science (CS)** → Explores various software development methods, both good and bad.
- **Software Engineering (SE)** → Focuses only on cost-effective and efficient techniques.

Software development life cycle (SDLC)

- **Software Development Process** → Structured activities to create software.
- **Key Phases:**
  - **Requirements Analysis** → Understanding needs.
  - **Design/Modeling** → Planning structure.
  - **Coding/Development** → Writing the software.
  - **Testing** → Ensuring quality.
  - **Implementation/Integration** → Deploying the system.
  - **Operation/Maintenance** → Ongoing support and updates.
  - **Documentation** → Recording details for future use.

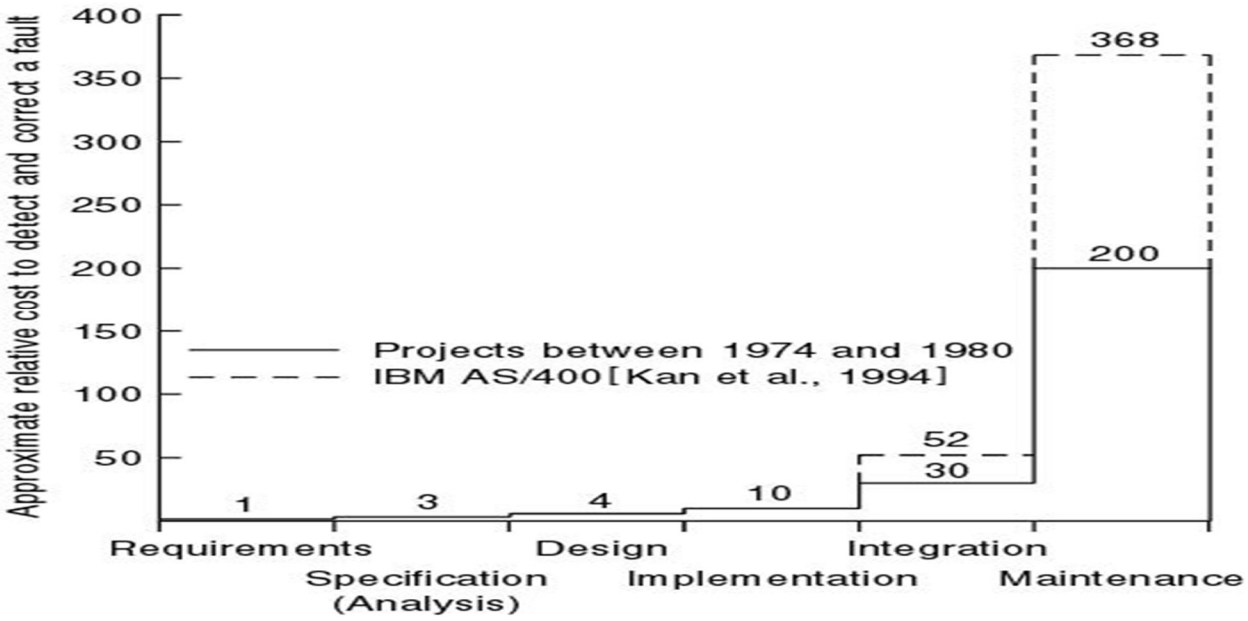
Good & bad software

- **Good software is maintained, bad software is discarded.**
- **Types of Maintenance:**
  - **Corrective (20%)** → Fixes issues.
  - **Enhancement (80%)** → Improves software.
    - **Perfective (60%)** → Enhances usability.
    - **Adaptive (20%)** → Updates for new environments.
    - **Preventive (20%)** → Prevents future issues.

Faults in software development phases

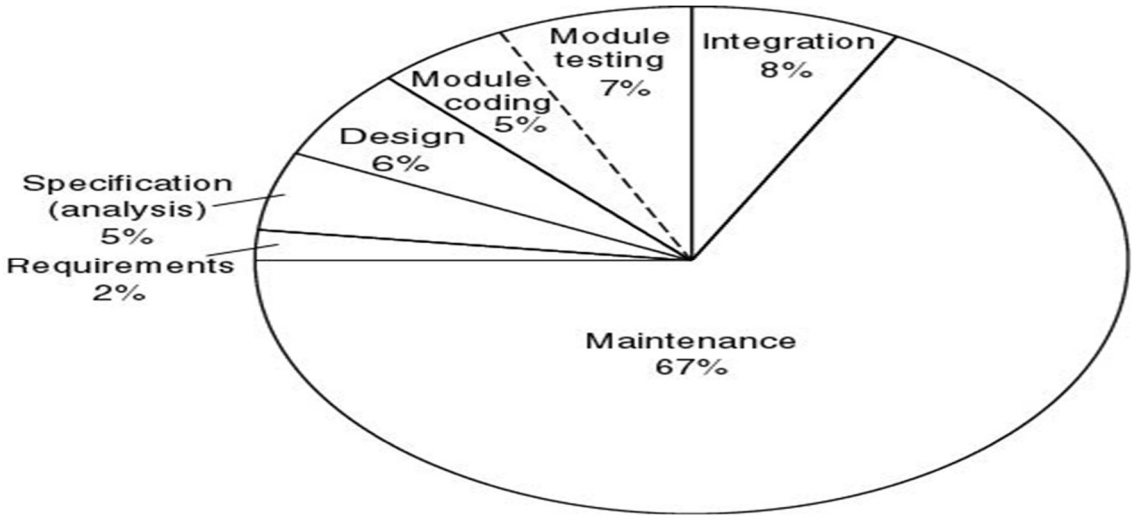
1. **Fault Distribution (Kelly, Sherif, and Hops, 1992)**:
   - 1.9 faults per page of specification.
   - 0.9 faults per page of design.
   - 0.3 faults per page of code.
2. **Fault Distribution (Bhandari et al., 1994)**:
   - 13% of faults from the previous version.
   - 16% of faults in new specifications.
   - 71% of faults in new design.
3. **Faults by Phase**:
   - 60-70% of faults are specification and design faults.

Cost of detection & correction of a fault

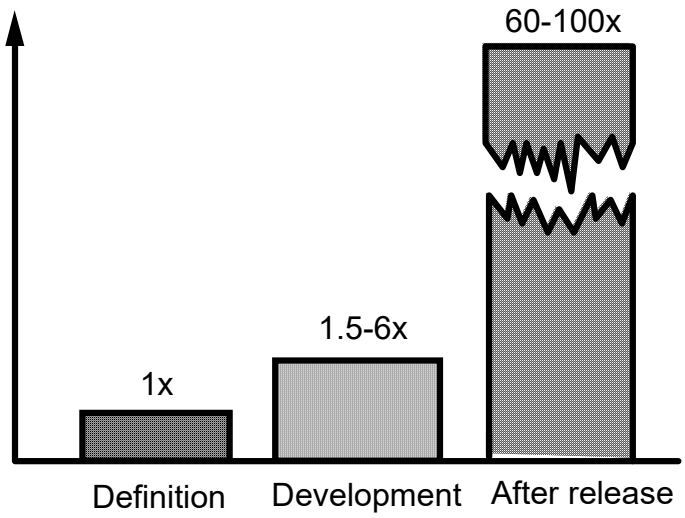


Main Points:

1. **Fixing Later Costs More**: The later a mistake is found, the more expensive it is to fix.
2. **Fix Early, Save Money**: Errors in the Requirements and Design stages cost much less to fix.
3. **Big Cost Jump**: Fixing mistakes in Integration and Maintenance is very expensive.
4. **Lesson**: Finding and fixing mistakes early is important to save time and money.

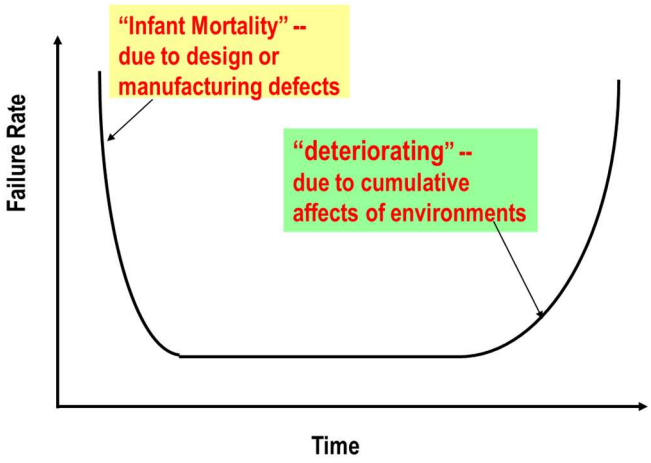


Cost of change



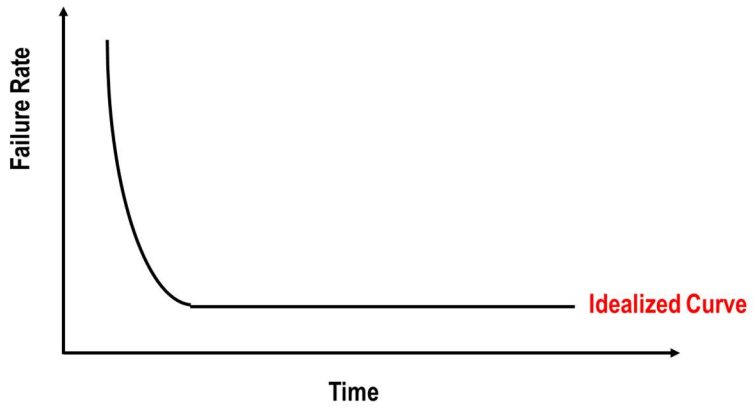
- **Three Bars Represent Different Phases**:
  - Smallest bar: **Design phase (lowest cost)**
  - Medium bar: **Coding phase (higher cost)**
  - Largest bar: **Maintenance phase (highest cost, with a break indicating extreme cost increase)**
- **Cost Escalation**: Fixing defects becomes much more expensive as the software progresses.
- **Lesson**: Detecting and fixing issues early saves significant time and money.

Product bathtub curve model



- **Bathtub Curve** represents failure rate over time.
- **Early Failures / Infant Mortality Phase (Yellow Box)**:
  - Happen due to design or factory defects.
  - Can be fixed through testing or early repairs.
- **Normal Life / Useful Life Phase (Middle Flat Section)**:
  - Failures happen randomly but not often.
  - The product works as expected.
- **Wearing Out Phase (Green Box)**:
  - Failures increase as parts get old and worn out.
  - The product is reaching the end of its life.
- Helps in planning repairs and making products last longer.

Software idealized curve



➔ The **idealized curve** for software reliability differs from hardware because software **does not physically degrade** over time. Instead, failures are mostly due to **bugs, design flaws, or external factors**.

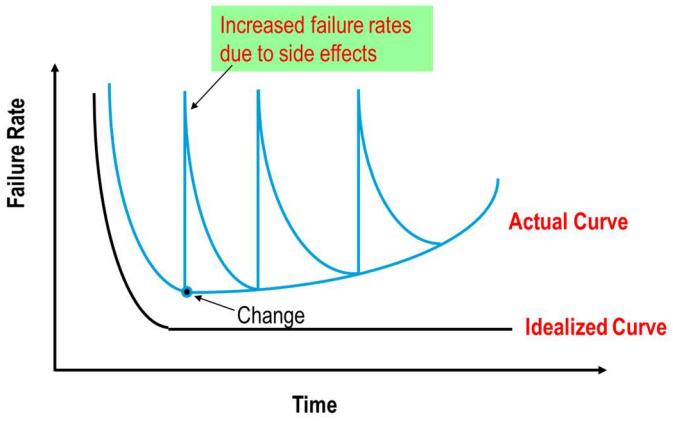
Key Points:

- **No Wear-Out Phase**:
  - Unlike hardware, software doesn’t "age" or wear out.
  - Failures don’t increase due to aging components.

- **Early Failures (High Initial Defects)**:
  - Software often has bugs at launch.
  - Frequent patches, updates, and debugging reduce failure rates.
- **Stable Failure Rate (Ideal Phase)**:
  - After debugging, failures become rare.
  - Well-maintained software remains reliable indefinitely.
- **External Factors**:
  - Failures may occur due to changes in hardware, updates, or cyber threats.
  - Regular maintenance helps maintain reliability.

The **software idealized curve** represents a stable, well-maintained system where failures are minimized and do not increase over time.

Software actual failure curve



➔ The **actual failure curve** for software is different from the idealized curve due to real-world challenges like **bugs, updates, and side effects**.

Key Points:

- **Initial High Failure Rate (Early Bugs)**:
  - Software has many bugs when first released.
  - Testing, debugging, and patches reduce failures over time.
  -

- **Failure Spikes After Updates**:
  - Each update or new feature introduces new bugs.
  - These bugs cause temporary increases in failure rates.
- **Side Effects of Fixes (Green Box in Image)**:
  - Fixing one bug may create new issues.
  - Regression testing is needed to avoid unintended failures.
- **Long-Term Trends**:
  - Unlike hardware, software does not "wear out."
  - Failures increase if updates introduce more issues than they fix.
- **Comparison to Idealized Curve**:
  - The **idealized curve** assumes continuous improvement.
  - The **actual curve** shows real-world challenges with software maintenance.

This explains why software requires **constant testing, monitoring, and maintenance** to stay reliable.

What is Software engineering?

1. **Efficient Technologies** – Enable faster, easier, and cost-effective software development.
2. **Software Engineering** – Ensures fault-free, timely, and budget-friendly software.
3. **Structured Process** – Follows engineering principles in software production.
4. **Crisis Solution** – Addresses software development challenges using engineering methods.

Types of Software Applications

1. **System Software** – Controls hardware (e.g., OS).
2. **Business Software** – For commercial use (e.g., SAP, ERP).
3. **Engineering & Scientific Software** – For analysis and computation (e.g., MATLAB, SPSS).
4. **Embedded Software** – Built into devices (e.g., autopilot, biometric systems).
5. **Personal Computer Software** – General use (e.g., Microsoft Office).
6. **Web-Based Software** – Runs on browsers (e.g., Gmail).
7. **AI Software** – Enables intelligent interaction (e.g., HCI, games).

Software Myths (Management)

Common Myths in Software Development

1. **Myth1: Standards & Procedures Are Enough** – Books alone don't ensure success; practical experience and adaptation are key.
2. **Myth2: Tools Solve Everything** – Advanced tools help, but skilled developers and proper methodologies are essential.
3. **Myth3: Adding More Programmers Speeds Up Work** – More people can slow progress due to communication overhead (Brooks' Law).
4. **Myth4: Outsourcing Means No Involvement** – Continuous oversight is necessary to ensure project success.

Software Myths (Customer)

More Software Development Myths

1. **Myth1: General Objectives Are Enough** – Vague goals lead to unclear requirements, causing costly rework later.
2. **Myth2: Changing Requirements Are Easy to Handle** – While software is flexible, frequent changes can increase complexity, cost, and delays.

Software Myths (Developer)

1. **Myth1: Once the Program Works, the Job is Done** – **Fact:** Maintenance and improvements continue after deployment.
2. **Myth2: Quality Can Only Be Assessed When Running** – Reviews, testing, and analysis help ensure quality before execution.
3. **Myth3: Only the Final Program Matters** – Documentation, testing, and design are crucial for long-term success.
4. **Myth4: Software Engineering Slows Development** – Proper engineering prevents costly mistakes and ensures efficiency.

MCQs Practice

1. **The primary aim of Software Engineering?**
   A) To create software quickly
   B) To solve the Software Crisis
   C) To reduce costs
   D) To enhance user experience

2. **Which phase is NOT part of the Software Development Life Cycle (SDLC)?**
   A) Requirements Analysis
   B) Testing
   C) Marketing
   D) Implementation

3. **What percentage of faults are typically found in specification and design phases?**
   A) 20-30%
   B) 40-50%
   C) 60-70%
   D) 80-90%

4. **Which type of maintenance accounts for the majority of software maintenance efforts?**
   A) Corrective
   B) Preventive
   C) Adaptive
   D) Enhancement

5. **What is a characteristic of software?**
   A) It is a physical product
   B) It deteriorates over time
   C) It is manufactured
   D) It wears out like hardware

6. **In software engineering, what does the term "software crisis" refer to?**
   A) High demand for software
   B) Issues with software quality and delivery
   C) The rise of software companies
   D) The need for software updates

7. **The cost of change after software release compared to during development?**
   A) 1x
   B) 1.5-6x
   C) 60-100x
   D) 10x

8. **Which software application type controls computer hardware?**
   A) Business software
   B) System software
   C) Embedded software
   D) Web-based software

9. **The idealized failure curve in software engineering?**
    A) A curve showing constant failure rates
    B) A curve showing decreasing failure rates over time
    C) A curve showing increasing failure rates
    D) A curve that represents all software failures

10. **Which myth suggests that having the latest tools guarantees successful software development?**
    A) Myth of standards
    B) Myth of outsourcing
    C) Myth of tools
    D) Myth of flexibility

11. **Main focus of Computer Science (CS) compared to Software Engineering (SE)?**
    A) CS focuses on economic techniques
    B) SE investigates various software production methods
    C) CS investigates good and bad software production methods
    D) SE focuses on hardware development

12. **What type of software is used for statistical analysis?**
    A) System software
    B) Business software
    C) Engineering and scientific software
    D) Personal computer software

13. **The term "differentiator" refer to in software?**
    A) Unique features of software
    B) Software that is easy to maintain
    C) Software that is manufactured
    D) Software that is obsolete

14. **Primary goal of software maintenance?**
    A) To discard bad software
    B) To enhance and fix existing software
    C) To develop new software
    D) To reduce costs

15. **Cost ratio of fixing a fault during the development phase compared to after release?**
    A) 1:1
    B) 1:6
    C) 1:10
    D) 1:100

16. **The purpose of the Software Development Life Cycle (SDLC)?**
    A) To analyze user requirements
    B) To structure software development activities
    C) To market software
    D) To maintain software

17. **A common misconception about software engineering?**
    A) It requires extensive documentation
    B) It guarantees fault-free software
    C) It is only about coding
    D) It is a one-time process

18. **Which maintenance type is aimed at avoiding future errors?**
    A) Corrective
    B) Enhancement
    C) Preventive
    D) Adaptive

19. **Main reason for system failure according to the presentation?**
    A) Lack of user involvement
    B) High costs
    C) Poor marketing
    D) Outdated technology

20. **What does the term "infant mortality" refer to in the context of software?**
    A) Early failures due to design defects
    B) Software that is never used
    C) Software that is constantly updated
    D) Software that is easy to maintain

Truth/False Practice

1. Software does not wear out but deteriorates over time.
2. All software is manufactured like physical products.
3. Good software is maintained, while bad software is discarded.
4. The cost of fixing a fault increases significantly after software release.
5. Software engineering focuses solely on coding.
6. The majority of faults occur during the coding phase.
7. Software engineering aims to produce fault-free software delivered on time.
8. Users often reject systems due to lack of involvement in development.
9. The Software Development Life Cycle (SDLC) includes documentation as a phase.
10. Software applications can only be web-based.

SOFTWARE DEVELOPMENT PROCESS MODEL

Ch: 2

Software development process model

Software Process & Models

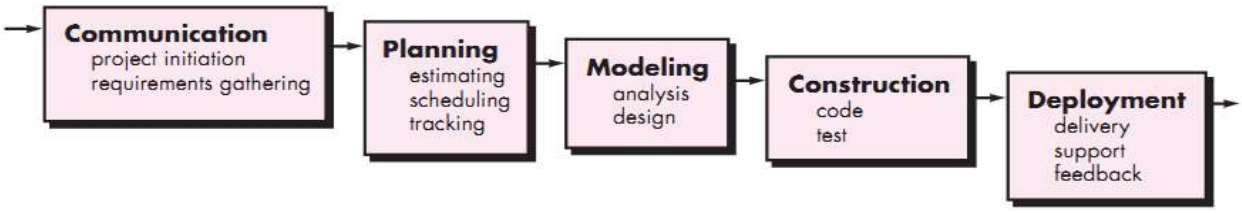
1. **Software Process** – A structured set of activities for software development.
2. **Software Process Model** – An abstract representation describing the process from a specific perspective.

Software Process Models

1. **Waterfall Model** – Sequential phases with minimal iteration.
2. **V-Model (Validation & Verification)** Emphasizes testing at each stage.
3. **Prototyping model** – Quickly builds a prototype for user feedback, which is then refined iteratively.
4. **Evolutionary Development Model** – Software is developed in iterative cycles with regular user feedback and improvements.
5. **Incremental Model** – Develops software in small, manageable parts.
6. **RAD (Rapid Application Development) Model** – Prioritizes quick prototyping and user feedback.
7. **Component-Based Development (CBD)** – Assembles pre-existing, reusable components to build a system efficiently and saleably.
8. **Rational Unified Process (RUP)** – Iterative and incremental, with four phases (Inception, Elaboration, Construction, and Transition) and a focus on use cases and risk management.
9. **Spiral Model** – Focuses on risk assessment and iterative development.
10. **Agile Model** – Emphasizes flexibility, collaboration, and rapid iterations.

Waterfall Model

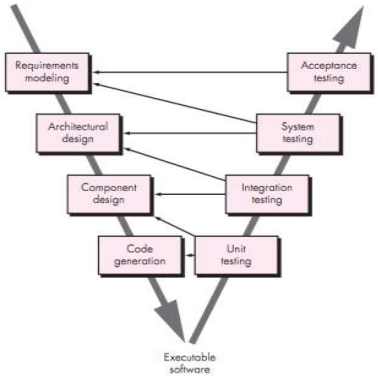
- **The waterfall or linear sequential model**



➔ Waterfall Model Problems

- Inflexible partitioning into distinct phases.
- **No backtracking:** next phase starts after the previous one.
- Difficult to accommodate changing customer requirements.
- Suitable only when requirements are well-understood.

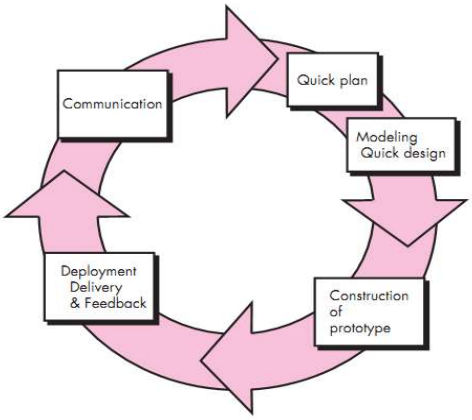
V - model



V-Model (Verification and Validation Model)

- Execution follows a **sequential, V-shaped** process.
- An extension of the **Waterfall model**, with **testing phases** corresponding to each **development phase**.
- Each **development phase** has a directly associated **testing phase**.
- **Highly disciplined**; the next phase starts only after the previous one is completed.

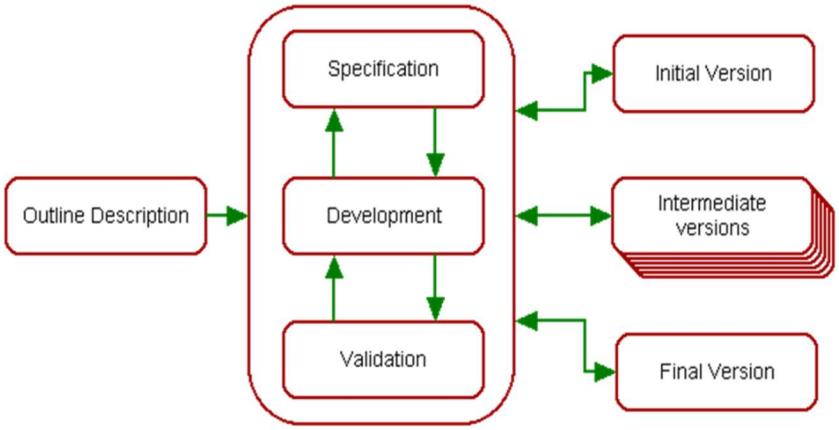
Prototyping model



Prototyping Model

- Requirements are unclear initially, and the prototype helps identify them.
- **Iteration** occurs as the prototype is refined to meet customer needs.
- **System requirements evolve**, making process iteration essential for large systems, where earlier stages may need to be reworked.

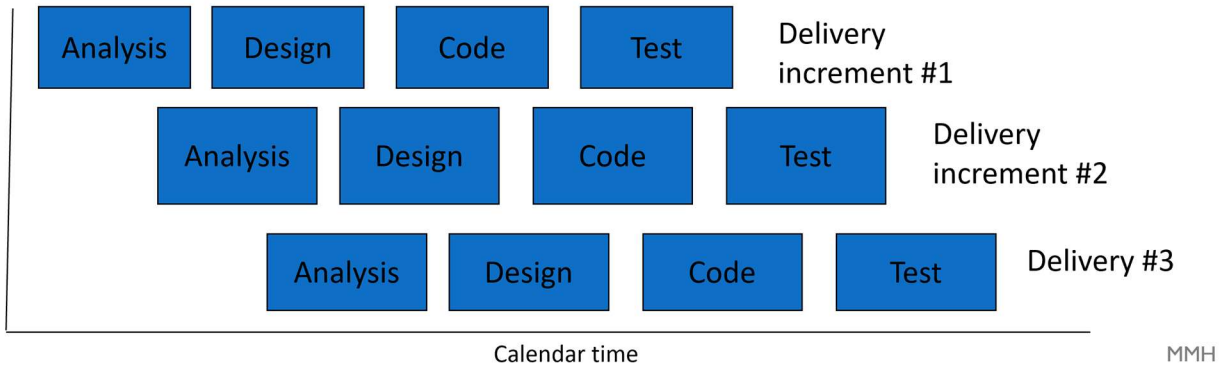
Evolutionary development



Types of Prototyping

- **Exploratory Development**:
  - **Objective**: Evolve a final system from an initial outline specification by working with customers.
  - Should start with **well-understood requirements**.
- **Throw-away Prototyping**:
  - **Objective**: Understand system requirements through prototype development.
  - Should start with **poorly understood requirements**.

Incremental development



Spiral Model

- Development and delivery are broken down into **increments**, each delivering part of the required functionality.
- **Requirements are relatively certain**, but frequent changes occur due to complexities.
- **User requirements are prioritized**, with high-priority ones included in early increments.
- Once an increment starts, its **requirements are frozen**, though later increments' requirements may continue to evolve.

➔ Advantages of Incremental Development

- **Customer value delivered early** with each increment, making system functionality available sooner.
- **Core product delivered first**, allowing essential features to be prioritized.
- **Early increments act as prototypes**, helping to refine requirements for later stages.
- **Lower risk of project failure** due to early delivery and iterative feedback.
- **Highest priority system services** receive the most testing, ensuring critical features are robust.

Rapid Application Development (RAD)

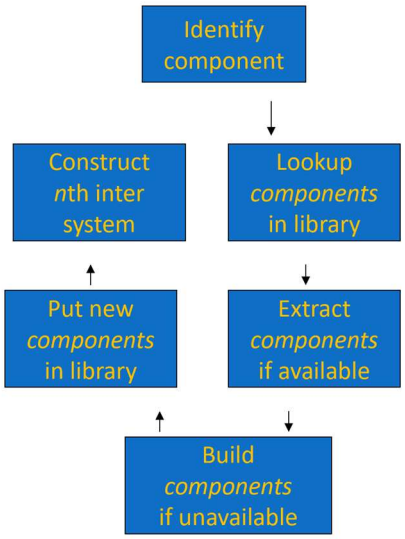
➔ RAD Model (Rapid Application Development)

- **Incremental** model with **time-boxed** developments, delivered as a working prototype.
- Components developed **in parallel** as mini projects with frozen requirements per increment.
- Quick customer feedback by delivering something to **see and use**.
- Delivers a fully functional system in **90 days** (± 30 days).

➔ RAD Phases:

1. **Requirements Planning** – Define scope and gather requirements.
2. **User Design** – Users work with analysts to design the system.
3. **Construction** – Development of applications.
4. **Cutover** – Testing, system changeover, and user training.

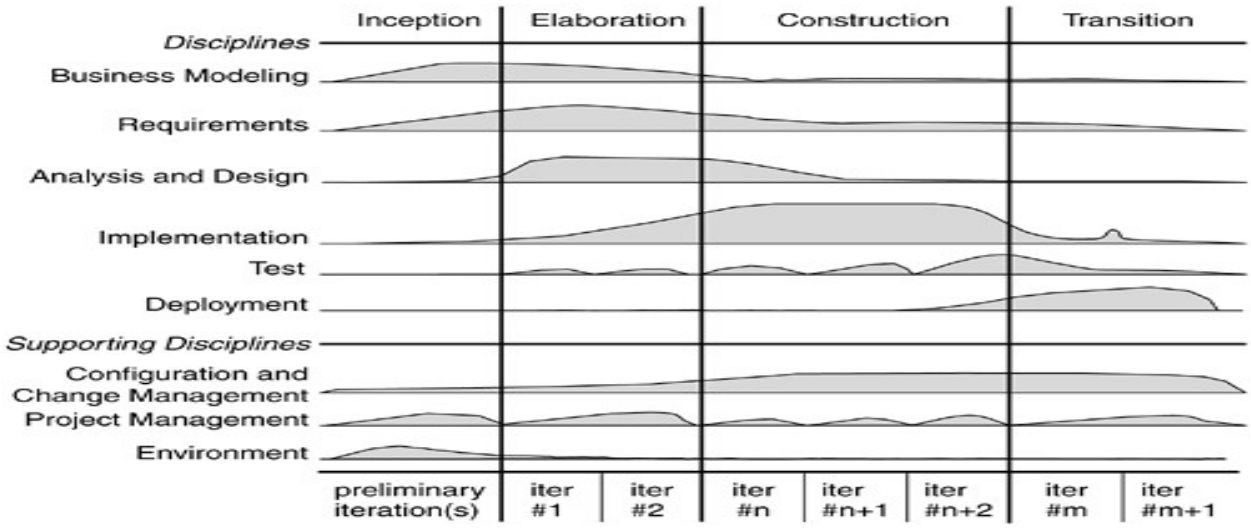
Component based development model



➔ Component-Based Development (CBD) Model

- Focuses on assembling **pre-existing, reusable components** to build software systems.
- Reduces development time and cost by reusing **modular components**.
- Components interact via **well-defined interfaces** for compatibility.
- Enables **scalability** as components can be added or replaced independently.
- Improves **maintainability** and **flexibility** through loosely coupled components.

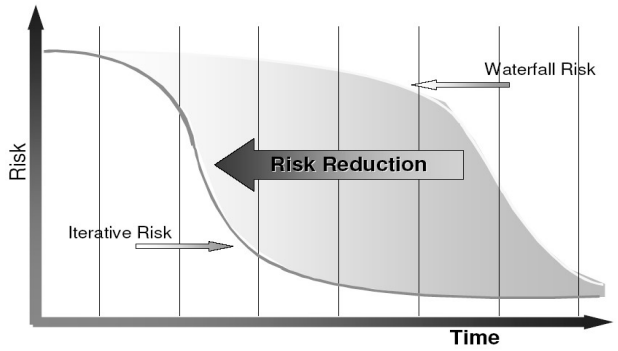
Rational Unified Process (RUP)



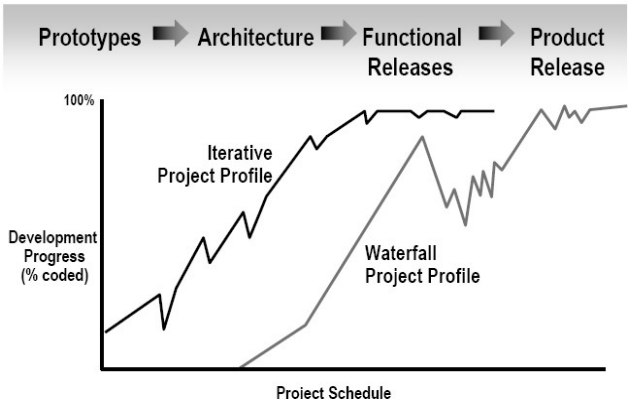
➔ Phases of the Process

1. **Inception**: Refine the idea to a well-founded concept for development.
2. **Elaboration**: Define and prioritize **requirements** and **architecture**.
3. **Construction**: Develop the system from baseline to readiness for user transition.
4. **Transition**: Deliver the software to users, while continuously improving it.
5. **Iteration**: A set of tasks resulting in an **executable system** for testing and evaluation.

Risk profile



Reduce Scrap/Rework: Use an Iterative Process



Iterative Development

- **Products are visible early** in the development process, allowing for early feedback.
- **Low probability of rework** as defects are identified and addressed in each iteration.

MCQs Practice

1. **A software process model mean.**
   A) A physical representation of software
   B) An abstract representation of a process
   C) A coding standard
   D) A project management tool

2. **Which model is characterized by inflexible partitioning into distinct stages?**
   A) V-Model
   B) Prototyping Model
   C) Waterfall Model
   D) Incremental Model

3. **V-Model emphasize?**
   A) Rapid development
   B) Verification and validation
   C) User feedback
   D) Component reuse

4. **In which model are prototypes used to clarify requirements?**
   A) Waterfall Model
   B) V-Model
   C) Prototyping Model
   D) RAD Model

5. **Main goal of evolutionary development?**
   A) To deliver a complete system at once
   B) To evolve a system from an initial outline
   C) To minimize costs
   D) To eliminate user involvement

6. **Which model delivers functionality in increments?**
   A) Waterfall Model
   B) Incremental Development
   C) V-Model
   D) RAD Model

7. **Key advantage of Incremental Development?**
   A) Higher risk of failure
   B) Customer value delivered early
   C) No testing involved
   D) Fixed requirements

8. **Which model focuses on time-boxed development?**
   A) Waterfall Model
   B) RAD Model
   C) V-Model
   D) Prototyping Model

9. **The characteristic of Component-Based Development?**
   A) Focus on coding from scratch
   B) Reusable object-oriented classes
   C) No testing phases
   D) Single delivery of the system

10. **Iterative development help reduce?**
   A) Time to market
   B) Scrap and rework
   C) User involvement
   D) Project costs

11. **The primary focus of the Rational Unified Process (RUP)?**
   A) Rapid prototyping
   B) Iterative and incremental development
   C) Waterfall approach
   D) Component reuse

12. **Which phase follows requirements planning in RAD?**
   A) Implementation
   B) User Design
   C) Testing
   D) Deployment

13. **Disadvantage of the Waterfall Model?**
   A) Flexibility in requirements
   B) Easy to manage
   C) Difficult to respond to changes
   D) High customer involvement

14. **In the V-Model, what is associated with each development phase?**
   A) A prototype
   B) A testing phase
   C) A design document
   D) A user feedback session

15. **What does the RAD model aim to provide quickly?**
   A) Detailed documentation
   B) User training
   C) A working prototype
   D) Comprehensive testing

16. **Which model is best for projects with poorly understood requirements?**
   A) Waterfall Model
   B) Prototyping Model
   C) Incremental Model
   D) V-Model

17. **Key feature of the Incremental Development model?**
   A) All requirements must be defined upfront
   B) Development is done in a single phase
   C) User requirements can evolve over time
   D) No testing is involved

18. **Which model is known for its structured approach to software development?**
    A) RAD Model
    B) Waterfall Model
    C) Prototyping Model
    D) Incremental Model

19. **Main purpose of the Rational Unified Process?**
    A) To minimize costs
    B) To provide a framework for iterative development
    C) To eliminate user involvement
    D) To deliver a complete system at once

20. **Which model is characterized by parallel development of components?**
    A) Waterfall Model
    B) V-Model
    C) RAD Model
    D) Prototyping Model

True/False Practice

1. The Waterfall Model allows for backtracking.
2. The V-Model is an extension of the Waterfall Model.
3. Prototyping is used when requirements are well understood.
4. Incremental Development delivers parts of functionality over time.
5. RAD models are time-boxed and focus on quick delivery.
6. Component-Based Development focuses on building from scratch.
7. Iterative development reduces the probability of rework.
8. The main goal of evolutionary development is to minimize user involvement.
9. In RAD, user design involves interaction with system analysts.
10. The Rational Unified Process does not involve iterations.

AGILE SOFTWARE DEVELOPMENT

Ch: 3

Agile Development

Key Points

- **Plan-driven methods** work best when **requirements** are well-defined and stable (1% change per month). – Barry Boehm
- **Agility** enables companies to **adapt to change** in a **turbulent business environment**.

Companies Need to:

- Innovate and improve **operations** faster.
- Respond quickly to
  - **competition**,
  - **new technology**, and
  - **customer requirements**.

Agile Model

Iterative and Evolutionary Methods

- Each iteration is a **mini project** with design, implementation, and testing.
- Results in an **iteration release** that adds to the final system.
- **Short iterations** allow feedback to improve and adjust requirements for the next cycle.

Agile methods vs. Past Iterative methods

- **Agile methods** have **shorter iterations** (1-4 weeks, max 30 days).
- Past iterative methods had iterations of **3-6 months**.
- **Shorter iterations** result in **lower complexity**, **better feedback**, and **higher productivity** and **success rates**

Timebox & scope

- **Iteration length** acts as a **timebox** for the team.
- The **scope (tasks)** for each iteration is chosen to fit within the iteration length.
- If needed, the **scope is reduced** to fit the iteration length, rather than increasing the iteration length.

Agile proponents believe

- **Current software processes** are **too heavyweight** and **cumbersome**.
- Too many **unnecessary tasks** are done that don't directly contribute to the software.
- Existing processes are **too rigid**, struggling with **incomplete or changing requirements**.
- **Shorter development cycles** are needed.
- **More active customer involvement** is required.

What is an agile method?

- **Agile methods are considered**:
  - **Lightweight** (do not concentrate on the whole software development at once)
  - **People-based** rather than **Plan-based**
- **Several agile methods**:
  - **No single agile method**
  - **Different agile methods** can be combined in software development (**Hybrid**)
- **No single definition**:
  - **Agile Manifesto** is the closest to a definition
  - **Set of principles**
  - Developed by **Agile Alliance**

Agile value statement

Agile Manifesto (2001)

In 2001, **Kent Beck** and 16 other software developers, writers, and consultants (known as **Agile Alliance**) signed a manifesto stating:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:"

- **Individuals and interactions** over **processes and tools**
- **Working software** over **comprehensive documentation**
- **Customer collaboration** over **contract negotiation**
- **Responding to change** over **following a plan**

Agile vs. Plan driven process

Aspect	Agile Process	Plan-Driven Process
Team Size	Small teams, limited scalability	Large teams, harder to scale down
Suitability for Critical Products	Not ideal for safety-critical products	Works well for safety-critical products
Best for Environment	Good for dynamic, but expensive for stable	Good for stable, but expensive for dynamic
Personnel Expertise Needed	Requires experienced Agile personnel	Requires experienced personnel mainly at the start
Working Style	Success thrives on freedom and chaos	Success thrives on structure and order

Agile assumption

- **Uncertainty in Requirements and Priorities**
  - Hard to predict which requirements will change.
  - Customer priorities may change during the project.
- **Overlapping Design and Construction**
  - Design and construction often occur together.
  - Difficult to determine how much design is needed before construction.
- **Unpredictability in Process Stages**
  - Analysis, design, construction, and testing are not easily predictable.
  - Planning these stages is challenging.

Agile manifesto (Policy)

- **Customer Satisfaction**
  - Prioritize satisfying the customer with early and continuous software delivery.
- **Welcoming Changes**
  - Embrace changing requirements, even late in development, for the customer's advantage.
- **Frequent Software Delivery**
  - Deliver working software frequently, with a preference for shorter timescales.
- **Collaboration**
  - Business people and developers must work together daily throughout the project.
- **Motivated Teams**
  - Build projects around motivated individuals, providing them with the environment and support they need, and trusting them to get the job done.
- **Effective Communication**
  - Face-to-face conversation is the most efficient way to convey information within a development team.
- **Measure of Progress**
  - Working software is the primary measure of progress.

- **Sustainable Development**
  - Promote sustainable development with a constant, manageable pace for sponsors, developers, and users.
- **Focus on Technical Excellence**
  - Continuous attention to technical excellence and good design enhances agility.
- **Simplicity**
  - Use simple approaches to make changes easier.
- **Self-Organizing Teams**
  - The best architectures, requirements, and designs emerge from self-organizing teams through iterative development.
- **Continuous Improvement**
  - Teams regularly reflect on how to improve effectiveness and adjust their behavior accordingly.

Human factors in agile development

- **Competence/Skill/Capability**
  - Team members must have the necessary expertise and abilities.
- **Common Focus**
  - Everyone should have a shared understanding of goals and objectives.
- **Collaboration**
  - Effective teamwork and communication are essential for success.
- **Decision-Making Ability**
  - Teams should be empowered to make decisions at all levels.
- **Fuzzy Problem-Solving Ability**
  - Teams need to navigate and solve vague or unclear problems.
- **Mutual Trust and Respect**
  - Building a foundation of trust and respect is critical for team cohesion.
- **Self-Organization**
  - Teams should be capable of organizing themselves and managing their own work.

Agile methods

- **Extreme Programming (XP)**
  - Focuses on technical excellence, continuous feedback, and close collaboration between developers and customers.
- **Scrum**
  - A framework that uses iterative cycles (sprints) to deliver small increments of working software, emphasizing team collaboration and frequent reviews.
- **Dynamic Systems Development Method (DSDM)**
  - A rapid application development approach focused on delivering functional software quickly and adapting to changing requirements.
- **Feature-Driven Development (FDD)**
  - Emphasizes building and delivering software based on well-defined, client-valued features in short iterations.
- **Crystal Methods**
  - A family of methodologies that focus on people, interactions, and flexibility, adapting to the needs of the project and team.
- **Lean Development (LD)**
  - Focuses on eliminating waste, improving efficiency, and delivering value to the customer by optimizing processes.
- **Adaptive Software Development (ASD)**
  - Emphasizes adaptive planning, collaboration, and frequent feedback to manage uncertainty and change in software projects.

MCQs Practice

1. **Main focus of Agile development?**
   A) Predictability
   B) Flexibility
   C) Documentation
   D) Planning

2. **How long do Agile iterations typically last?**
   A) 1-4 weeks
   B) 3-6 months
   C) 1 year
   D) 2-3 weeks

3. **What is a key principle of the Agile Manifesto?**
   A) Processes over people
   B) Comprehensive documentation over working software
   C) Customer collaboration over contract negotiation
   D) Following a plan over responding to change

4. **A timebox in Agile development mean?**
   A) A set of tasks for the project
   B) A predetermined iteration length
   C) A tool for documentation
   D) A method for testing

5. **Which Agile method focuses on customer feedback and iterative development?**
   A) Waterfall
   B) Scrum
   C) V-Model
   D) Spiral

6. **The primary measure of progress in Agile?**
   A) Documentation
   B) Working software
   C) Customer satisfaction
   D) Team velocity

7. **Agile methods are considered:**
   A) Heavyweight
   B) Lightweight
   C) Plan-based
   D) Time-consuming

8. **What do Agile proponents believe about current software development processes?**
   A) They are too flexible
   B) They are too rigid and heavyweight
   C) They are perfect as they are
   D) They require more documentation

9. **Which of the following is NOT an Agile method?**
   A) Extreme Programming (XP)
   B) Scrum
   C) Lean Development
   D) Waterfall

10. **Agile assumption regarding requirements?**
   A) They are always stable
   B) They can be predicted accurately
   C) They will change over time
   D) They are irrelevant

11. **In Agile development, how is the scope determined for each iteration?**
   A) It is fixed regardless of time
   B) It is reduced to fit the iteration length
   C) It is expanded to fill the timebox
   D) It is based on customer demands only

12. **Role of customer involvement in Agile?**
   A) Minimal involvement
   B) Active and continuous involvement
   C) Only at the beginning
   D) Only during testing

13. **Which Agile method emphasizes face-to-face communication?**
   A) Waterfall
   B) Scrum
   C) Extreme Programming (XP)
   D) Lean Development

14. **Agile approach to change?**
   A) Resist change
   B) Welcome change
   C) Ignore change
   D) Delay change

15. **Which of the following is a characteristic of Agile teams?**
   A) Hierarchical structure
   B) Self-organization
   C) Rigid roles
   D) Fixed schedules

16. **A common challenge in Agile development?**
    A) Too much documentation
    B) Lack of customer involvement
    C) Predicting requirements
    D) Long development cycles

17. **Which Agile method is known for its iterative approach and frequent releases?**
    A) Waterfall
    B) Scrum
    C) V-Model
    D) Spiral

18. **The Agile value statement regarding individuals?**
    A) Processes over individuals
    B) Individuals and interactions over processes and tools
    C) Individuals are irrelevant
    D) Individuals should follow strict guidelines

19. **In Agile, what is the preferred method of communication?**
    A) Email
    B) Documentation
    C) Face-to-face conversation
    D) Phone calls

20. **Goal of Agile development?**
    A) To minimize costs
    B) To deliver software quickly and efficiently
    C) To create comprehensive documentation
    D) To follow a strict plan

Truth/False Practice

1. **Agile development is primarily focused on extensive documentation.**
2. **In Agile, customer feedback is essential for each iteration.**
3. **Agile methods allow for longer iteration cycles compared to traditional methods.**
4. **The Agile Manifesto was created in 2001.**
5. **Agile processes are rigid and do not accommodate changes.**
6. **Self-organization is a key characteristic of Agile teams.**
7. **Agile development requires less customer involvement than traditional methods.**
8. **The primary measure of progress in Agile is working software.**
9. **Agile methods are considered heavyweight and cumbersome.**
10. **Agile teams thrive on structure and order.**

EXTREME PROGRAMMING (XP)

Ch: 4

Extreme programming (XP)

- **Origins**

  Evolved from problems with long development cycles in traditional models (Beck 1999a).
- **Initial Purpose**

  Started as a way to "**get the job done**" using effective practices from previous software development (Haungs 2001, Beck 1999b).
- **Methodology**

  Built around common-sense principles and simple practices.
- **Tailoring to Projects**

  No single process fits all projects; practices should be adapted to each project's needs.

XP values

- **Communication:** XP has a culture of oral communication and its practices are designed to encourage interaction.

  "Problems with projects can invariably be traced back to somebody not talking to somebody else about something important."
- **Simplicity**: Design only what's needed to meet current requirements, avoiding anticipation of unstated needs.
- **Feedback**: Gather customer feedback after each iteration and external release to drive the next iteration.
- **Courage**: Empower the team to make decisions and resist unrealistic commitments.
- **Respect**: Team members should care for each other and the project.

XP process

1. **Exploration**: Define requirements and feasibility.
2. **Planning**: Create plans and prioritize features.
3. **Iterations to Release**: Develop in short cycles, releasing incrementally.
4. **Productionizing**: Ensure stability and readiness for deployment.
5. **Maintenance and Death**: Maintain and eventually phase out the product.

Xp process – Exploration phase

- **Customer Story Cards**: Customers write the story cards for the first release.
- **Team Familiarization**: The project team learns the tools, technology, and practices for the project.
- **Duration**: The exploration phase lasts from a few weeks to months, depending on the team's familiarity with the technology.

Xp process – planning phase

- **User Stories**: Write user stories for the project.
- **Effort Estimation**: Estimate the effort required for each user story.
- **Prioritization**: Prioritize user stories for implementation.
- **Release Planning**: Create the release schedule based on priorities.

Xp process – iterations to release phase

- **Multiple Iterations**: Several iterations before the first release, each lasting 1-4 weeks.
- **First Iteration**: Build the system's architecture by selecting stories that shape the overall structure.
- **Customer Selection**: The customer chooses which stories to include in each iteration.
- **Final Iteration**: At the end of the last iteration, the system is ready for production.

Xp process – productionizing phase

- **Extra Testing**: Perform additional testing and performance checks before release.
- **New Changes**: Evaluate if new changes should be included in the current release.
- **Iteration Speed**: May shorten iterations from 3 weeks to 1 week.
- **Postponed Ideas**: Document postponed ideas and suggestions for later implementation.

Xp process – maintenance phase

- **System Maintenance**: Keep the system running while producing new iterations.
- **Customer Support**: Allocate effort for customer support tasks.
- **Development Velocity**: Velocity may slow down after the system is in production.
- **Team Changes**: May require adding new members or changing the team structure.

Xp process – death phase

- **No More Stories**: The customer has no more stories to implement.
- **Satisfaction**: The system meets customer needs, including performance and reliability.
- **Documentation**: Final documentation is written, as no more changes are made to architecture, design, or code.
- **Death**: The system may end if it fails to deliver desired outcomes or becomes too costly to develop further.

Xp - Roles and responsibility

- **Customer**: Writes stories, creates functional tests, and decides when requirements are satisfied. Sets implementation priorities.
- **Programmer**: Keeps the code simple and clear.
- **Tester**: Assists with writing functional tests, runs tests regularly, shares results, and maintains testing tools.
- **Tracker**: Provides feedback on estimates and progress, evaluating whether goals are achievable within constraints.
- **Coach**: Oversees the process, guiding team members to follow XP practices.
- **Consultant**: Provides specialized technical knowledge as an external expert.
- **Manager (Big Boss)**: Makes the final decisions.

Xp – practices

- **Simple Design**: Focus on the simplest implementable solution at the moment.
- **Testing**: Test-driven development with continuous unit testing.
- **Refactoring**: Improve the system by removing duplication, enhancing communication, simplifying, and adding flexibility.
- **Collective Ownership**: Anyone can modify any part of the code at any time.
- **Pair Programming**: Two programmers work together, one as the driver (coding) and the other as the observer (reviewing and strategizing).
- **Continuous Integration**: Integrate new code into the codebase as soon as it's ready.
- **40-Hour Week**: Limit work to a maximum of 40 hours per week.
- **On-Site Customer**: The customer must be present and available full-time for the team.
- **Coding Standards**: Follow established coding rules and ensure communication through code.
- **Open Workspace**: Prefer a large room with small cubicles.
- **Just Rules**: Team follows its own rules, which can be changed as needed, with impact assessments for changes.

Xp – artefacts

- **User Story Cards:**
  - Brief descriptions of features or non-functional requirements.
  - Serve as a commitment for further discussion between developer and customer.
  - Includes customer priority and developer resource estimate.
  - The resource estimate must fit within the iteration duration.
- **Task List:**
  - A list of tasks (1-3 days duration) for user stories in an iteration.
  - Tasks represent concrete aspects of a user story.
  - Programmers volunteer for tasks, rather than being assigned.
- **CRC Cards (Optional):**
  - Index cards used to record classes, their responsibilities, and collaborators.
  - Serve as a basis for software design.
  - Identified during brainstorming/role-playing sessions involving multiple developers.
- **Customer Acceptance Tests:**
  - Textual descriptions and automated test cases developed by the customer.
  - Development team demonstrates completion of a user story by passing these tests.
- **Visible Wall Graphs:**
  - Progress graphs posted in the team's work area to foster communication and accountability.
  - Show how many stories are completed and/or how many acceptance test cases are passing.

How xp solves some SE problems

Problem	Solution
1. **Slipped schedule**	**Short development cycles**
2. **Cost of changes**	**Ongoing testing, system always running**
3. **Defect rates**	**Unit tests, customer tests**
4. **Misunderstanding business**	**Customer is part of the team**
5. **Business changes**	**Changes are welcome**
6. **Staff turnover (Replacement)**	**Intensive teamwork**

MCQs Practice

1. **The primary goal of Extreme Programming (XP)?**
   A) Long development cycles
   B) Rapid delivery of high-quality software
   C) Extensive documentation
   D) Individual programming

2. **Which phase comes first in the XP life cycle?**
   A) Planning
   B) Iterations to Release
   C) Exploration
   D) Maintenance

3. **What value emphasizes the importance of communication in XP?**
   A) Courage
   B) Simplicity
   C) Respect
   D) Communication

4. **In XP, what is meant by 'Simplicity'?**
   A) Complex designs
   B) Designing for future needs
   C) Creating the simplest solution for current requirements
   D) Extensive feature lists

5. **The purpose of user story cards in XP?**
   A) To document all requirements
   B) To outline project budgets
   C) To facilitate conversation between developers and customers
   D) To track time spent on tasks

6. **How long do iterations typically last in XP?**
   A) 1-2 weeks
   B) 2-4 weeks
   C) 1-4 weeks
   D) 4-6 weeks

7. **Key practice in XP that involves two programmers working together?**
   A) Pair programming
   B) Solo programming
   C) Group coding
   D) Code reviews

8. **'Continuous Integration' refer to in XP?**
   A) Integrating code at the end of the project
   B) Regularly integrating new code into the codebase
   C) Delaying integration until all features are complete
   D) Integrating only after extensive testing

9. **Which role is responsible for writing user stories in XP?**
   A) Programmer
   B) Tester
   C) Customer
   D) Coach

10. **The focus of the Productionizing phase in XP?**
    A) Initial planning
    B) Final testing and performance checks
    C) User story writing
    D) Team restructuring

11. **Which XP value encourages team members to support each other?**
    A) Courage
    B) Respect
    C) Simplicity
    D) Communication

12. **Purpose of feedback in XP?**
    A) To finalize the project
    B) To drive the next iteration
    C) To document changes
    D) To assess team performance

13. **'Refactoring' mean in XP?**
    A) Adding new features
    B) Restructuring code for improvement
    C) Writing new documentation
    D) Testing existing code

14. **Maximum working week recommended in XP?**
    A) 30 hours
    B) 40 hours
    C) 50 hours
    D) 60 hours

15. **What is a key limitation of XP?**
    A) It requires large teams
    B) It is not suitable for small projects
    C) Team members must be co-located
    D) It focuses on documentation

16. **In XP, who decides the priority of user stories?**
    A) The programmer
    B) The tester
    C) The customer
    D) The coach

17. **The purpose of Customer Acceptance Tests in XP?**
    A) To evaluate team performance
    B) To validate user story completion
    C) To track project budget
    D) To assess coding standards

18. **'Death' phase in XP signify?**
    A) The end of the project
    B) The start of maintenance
    C) No more user stories to implement
    D) Project failure

19. **Key characteristic of XP's development approach?**
    A) Extensive planning
    B) Focus on individual work
    C) Emphasis on teamwork and collaboration
    D) Long feedback cycles

20. **The role of the Tracker in XP?**
    A) To write user stories
    B) To provide feedback on estimates and progress
    C) To manage customer relations
    D) To lead the development team

21. **The main benefit of short development cycles in XP?**
    A) Increased documentation
    B) Faster delivery of features
    C) More time for testing
    D) Reduced team communication

22. **'Collective Ownership' mean in XP?**
    A) Only one person can change the code
    B) All team members can change any part of the code
    C) Code changes are restricted
    D) Only the lead programmer can modify code

23. **Which XP practice emphasizes the importance of a shared understanding of the system?**
    A) Simple design
    B) Metaphor
    C) Continuous integration
    D) Pair programming

24. **The main focus of the Maintenance phase in XP?**
    A) Finalizing project documentation
    B) Keeping the system running and producing new iterations
    C) Planning for new projects
    D) Conducting user interviews

25. **What does XP encourage regarding changes in business requirements?**
    A) Changes are discouraged
    B) Changes are welcome
    C) Changes must be documented
    D) Changes are only accepted after testing

Truth/False Practice

1. **Extreme Programming (XP) is designed for long development cycles.**
2. **In XP, communication is considered a key value.**
3. **The Exploration phase in XP involves writing user stories.**
4. **Pair programming involves two programmers working together at one computer.**
5. **The Productionizing phase focuses on initial planning and requirements gathering.**
6. **Feedback from customers is obtained at the end of each iteration in XP.**
7. **In XP, the customer is not involved in the development process.**
8. **Continuous Integration is a practice where new code is integrated into the codebase as soon as it is ready.**
9. **The maximum recommended working week in XP is 50 hours.**
10. **XP encourages extensive documentation for every aspect of the project.**


SCRUM
Ch: 5

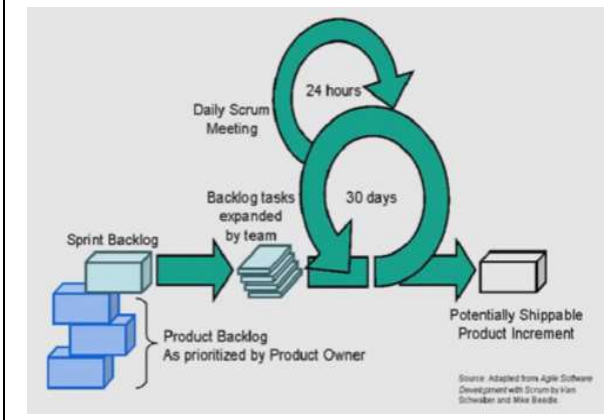
Scrum

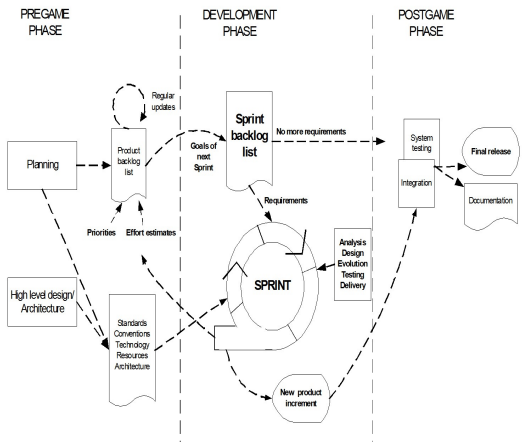
⇨ SCRUM Process Overview:

≫**SCRUM** comes from "rugby", meaning **teamwork to bring the ball back into play.**

Phases of SCRUM:

1. **Pre-game** – Planning and setting up the project.
2. **Development (Game phase)** – Actual work, sprint cycles, and product building.
3. **Post-game** – Final testing, review, and project closure.





Pre-game phase

⇨ The pre-game phase has two main parts:

1. **Planning:**
   - Define the system and create a **Product Backlog** (list of requirements).
   - Prioritize tasks and estimate effort.
   - Continuously update the backlog.
   - Set up the team, tools, risks, and approvals.
2. **Architecture:**
   - Plan the system's high-level design.
   - Identify changes if updating an existing system.
   - Hold a **design review meeting** for final decisions.

Development phase (Game phase)

- This phase is unpredictable, like a **"black box."**
- Work is done in **Sprints** (short development cycles).
- Each **Sprint** includes:
  - Requirements
  - Analysis
  - Design
  - Development
  - Delivery
- Sprints last **1 week to 1 month.**

Post-game phase

- **Begins** when all requirements are completed.
- No new changes or issues are added.
- Focus on **release preparation**, including:
  - **Integration**
  - **System testing**
  - **Documentation**

Roles and Responsibilities

1. **Scrum Master** – Ensures the project follows Scrum rules, interacts with the team, customer, and management.
2. **Product Owner** – Manages the **Product Backlog**, makes final decisions on backlog tasks. He is selected by the Scrum Master, the customer, and the management.
3. **Scrum Team** – Plans and executes sprints, estimates effort, and organizes work.
4. **Customer** – Provides input on product backlog items.
5. **Management** – Makes final decisions, sets goals, and ensures standards.

Scrum practices (Key Concepts)

➢ Product Backlog:

- A **prioritized** list of features, fixes, and updates.
- Continuously updated throughout the project.

➢ Effort Estimation:

- An **iterative process** to estimate workload more accurately as details emerge.
- Done by **Product Owner** and **Scrum Team**.

➢ Sprint:

- Short development cycle to adapt to changes.
- Key tools: **Sprint Backlog, Sprint Planning, Daily Scrum Meetings**.

➢ Sprint Backlog:

- A **fixed list** of tasks selected for a Sprint.
- Created in **Sprint Planning**, remains stable until Sprint completion.

➢ Sprint Planning Meeting:

- **Phase 1:** Decide Sprint goals and features (**Scrum Master, Management, Product Owner, Scrum Team**).
- **Phase 2:** Plan how to implement the product increment (**Scrum Master, Scrum Team**).

➢ Daily Scrum Meeting:

- A **15-minute** daily check-in to track progress, plan, and resolve issues.
- Led by **Scrum Master**, attended by **Scrum Team** (Management may join).

➢ Sprint Review Meeting:

- **Final Sprint presentation** to assess progress and decide next steps.
- Attended by **Scrum Team, Scrum Master, Management, Customers, Users, Product Owner**.
- May result in **new Backlog items or project direction changes**.

MCQs Practices

1. **SCRUM primarily used for?**

   A) Documentation B) Project Management

   C) Hardware Development D) Marketing Strategies

2. **Which phase of SCRUM includes planning and defining the system?**

   A) Development Phase B) Pre-game Phase

   C) Post-game Phase D) Review Phase

3. **What is a Product Backlog?**

   A) A list of completed tasks B) A prioritized list of requirements

   C) A schedule of meetings D) A financial report

4. **Who is responsible for managing the Product Backlog?**

   A) Scrum Master B) Product Owner

   C) Development Team D) Stakeholders

5. **The duration of a typical Sprint?**

   A) 1 day B) 1 week to 1 month

   C) 3 months D) 6 months

6. **What does the Scrum Master do?**

   A) Defines project requirements B) Ensures adherence to SCRUM practices

   C) Manages the budget D) Designs the system architecture

7. **Purpose of the Daily Scrum meeting?**

   A) To review the project budget B) To track progress and plan for the day

   C) To finalize project requirements D) To conduct performance reviews

8. **Which phase is treated as a "black box" in SCRUM?**

A) Pre-game Phase
B) Development Phase
C) Post-game Phase
D) Planning Phase

9. **Outcome of a Sprint Review meeting?**

A) Final project approval
B) Assessment of the product increment
C) Budget allocation
D) Team performance evaluation

10. **Role of the Customer in SCRUM?**

A) To manage the team
B) To define project timelines
C) To participate in Product Backlog tasks
D) To approve budgets

11. **What is a Sprint Backlog?**

A) A list of completed tasks
B) A list of items selected for the current Sprint
C) A historical record of Sprints
D) A financial report

12. **First step in the Pre-game phase?**

A) Architecture planning
B) Requirement gathering
C) Sprint planning
D) Testing

13. **Which of the following is NOT a role in SCRUM?**

A) Scrum Master
B) Product Owner
C) Project Manager
D) Development Team

14. **Purpose of effort estimation in SCRUM?**

A) To allocate budget
B) To predict project completion time
C) To assess the effort needed for Backlog items
D) To evaluate team performance

15. **What happens during the Post-game phase?**

A) Requirements are defined
B) System testing and documentation
C) Sprint planning
D) Team performance review

16. **The term 'Scrum' originate from?**

A) A software tool
B) A rugby strategy
C) A project management methodology
D) A financial term

17. **Who selects the items for the Sprint Backlog?**

A) Scrum Master
B) Development Team and Product Owner
C) Management
D) Customers

18. **Main focus during a Sprint?**

A) Documentation
B) Delivering a product increment
C) Budget management
D) Team building

19. **The role of Management in SCRUM?**

A) To define project tasks
B) To make final decisions and set goals
C) To conduct daily meetings
D) To manage the development team

20. **What is a key benefit of using SCRUM?**

A) Increased documentation
B) Enhanced flexibility and adaptability
C) Longer project timelines
D) Reduced team collaboration

21. **The purpose of a design review meeting?**

A) To finalize the project budget
B) To review implementation proposals
C) To assess team performance
D) To conduct customer interviews

22. **What is the main characteristic of a Sprint?**

A) It is a one-time event
B) It is iterative and time-boxed
C) It lasts indefinitely
D) It requires extensive documentation

23. **The term 'increment' refer to in SCRUM?**

A) A budget increase
B) A completed product feature
C) A team member's performance
D) A project delay

24. **What is the focus of the architecture phase in SCRUM?**

A) Budget allocation
B) High-level system design
C) Team performance
D) Customer feedback

25. **What is the significance of prioritizing the Product Backlog?**

A) To reduce project costs B) To ensure the most important features are developed first
C) To increase team workload
D) To minimize customer involvement

26. **In the context of software engineering, how would you apply the principles of DSDM to improve project management in a real-world scenario?**

A) By strictly adhering to a predefined project timeline without flexibility.
B) By incorporating user feedback at various stages of development to ensure alignment with user needs.
C) By minimizing communication among team members to streamline processes.
D) By focusing solely on documentation to guide the project without iterative reviews.

27. **In what way does the term 'Scrum' relate to its origins in rugby, and how can this analogy be applied to team dynamics in software development?**

A) Scrum emphasizes individual performance over teamwork, similar to a rugby player's focus on scoring.
B) Scrum represents a strategy for reintroducing a product into development, akin to a rugby scrum where the ball is brought back into play through collaboration.
C) Scrum is primarily concerned with the speed of development, much like a rugby match where speed is the only factor for success.
D) Scrum discourages collaboration, focusing instead on competition among team members, similar to opposing teams in rugby.

28. **Considering the three phases of the Scrum process, which phase would be most critical for ensuring that the team is aligned on goals before development begins?**
    A) The pre-game phase is essential for establishing team alignment and setting clear objectives before the development phase.
    B) The development phase is where alignment is tested and adjusted based on ongoing feedback from stakeholders.
    C) The post-game phase focuses on reflection and improvement, which is important but comes after the goals have been set.
    D) The pre-game phase is less important than the development phase, as it does not directly impact product delivery.

29. **How can the adaptive nature of Scrum, as described in the literature, be utilized to enhance product development in a rapidly changing market?**
    A) By adhering strictly to the initial plan without deviations, ensuring that all team members follow the same path.
    B) By allowing teams to self-organize and adapt their processes based on real-time feedback and changing requirements.
    C) By focusing solely on completing tasks quickly, regardless of the quality of the output.
    D) By minimizing communication among team members to avoid confusion and maintain focus on individual tasks.

True/False Practice

1. The term 'Scrum' originates from a strategy used in rugby.
2. The Scrum process includes four phases: Pre-game, Development, Post-game, and Review.
   Answer: *(It includes Pre-game, Development, and Post-game.)*
3. The Product Owner is responsible for prioritizing the Product Backlog.
4. Daily Scrum meetings are held to assess team performance and allocate tasks.
   Answer: *(They are held to track progress and identify impediments.)*
5. Sprints in Scrum typically last between one week and one month.
6. The Scrum Master is responsible for writing code for the project.
   Answer: *(The Scrum Master facilitates the Scrum process and supports the team.)*
7. The Post-game phase involves preparing the product for release, including testing and documentation.
8. The Sprint Backlog is a dynamic list that can change throughout the Sprint.
   Answer: *(The Sprint Backlog is stable until the Sprint is completed.)*
9. User feedback is not considered important in the Scrum framework.
   Answer: *(User feedback is crucial for improving product quality.)*
10. Scrum promotes continuous improvement through regular feedback and adaptation.

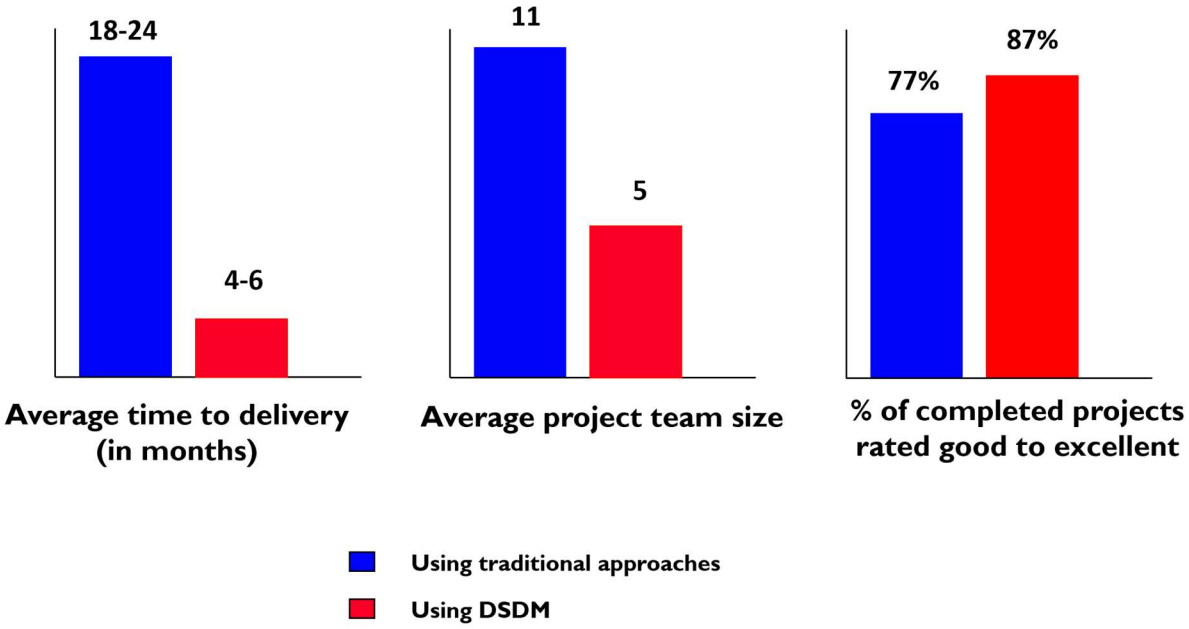
THE DYNAMIC SYSTEMS DEVELOPMENT METHOD (DSDM)

Ch: 6

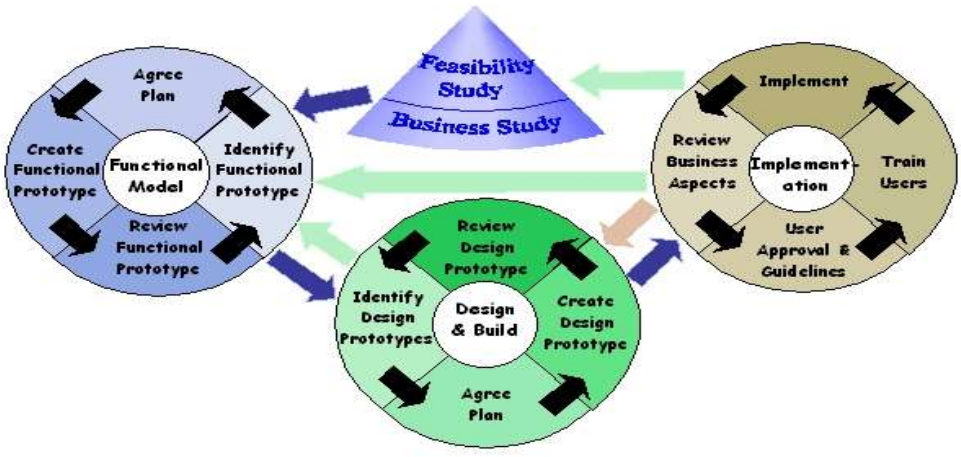
The Dynamic Systems Development Method (DSDM)

- **A public domain** Rapid Application Development (RAD) method.
- Developed from **vendor and user experience**, the UK's **de-facto standard** for RAD.
- **Key Goal:** Deliver what the business **needs, when it needs it**.
- **Approach:**
  - Uses **various techniques** within the framework.
  - **Flexible requirements** to adapt to business priorities.
  - Focuses on **current and immediate business needs**, not all possible future needs.

Traditional method vs. DSDM



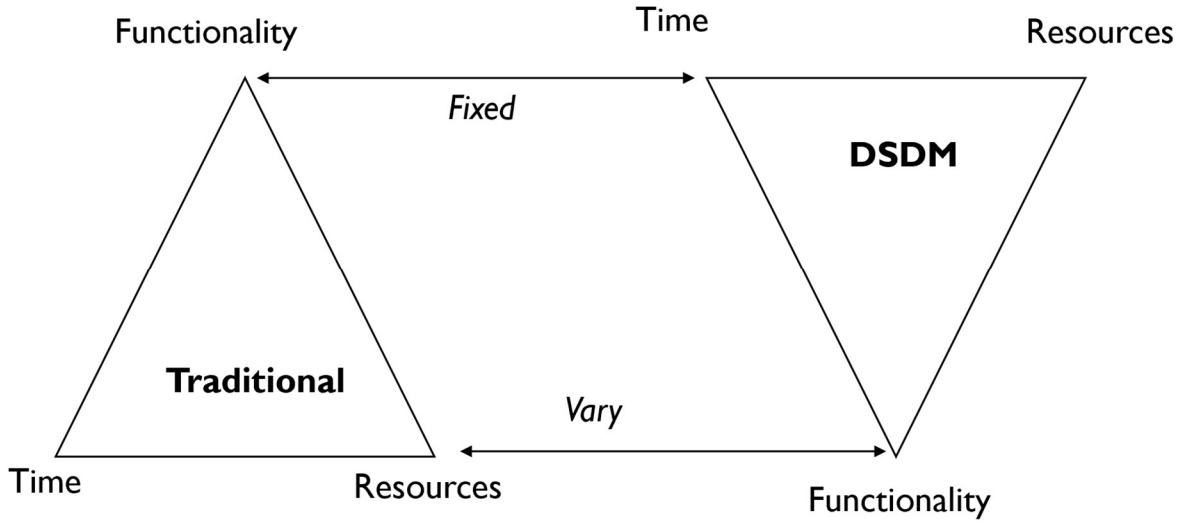
DSDM Process view



DSDM Process

Activity	Sub-Activity	Main Points
Study	Feasibility Study	- Assess suitability of DSDM - Decision to use DSDM or not - Generates: Feasibility Report, Feasibility Prototype, Global Outline Plan (Development Plan, Risk Log)
	Business Study	- Analyze business & technology - Conduct workshops with customer experts - Develop: Prioritized Requirements List, Business Area Definition, System Architecture Definition, Outline Prototyping Plan
Functional Model Iteration	Identify Functional Prototype	- Select features for the prototype. - Develop Functional Model.
	Agree Schedule	- Decide timeline for development.
	Create Functional Prototype	- Build the prototype.
	Review Functional Prototype	- Test and review; Create Prototyping Review Document.
Design and Build Iteration	Identify Design Prototype	- Define system needs. - Plan Implementation Strategy.
	Agree Schedule	- Set timeline for implementation.
	Create Design Prototype	- Develop a working system for testing.
	Review Design Prototype	- Test and review. - Create User Documentation & Test Record.
Implementation	User Approval & Guidelines	- Get approval. - Create usage guidelines.
	Train Users	- Train end users.
	Implement	- Deploy the system.
	Review Business	- Evaluate business impact. - Create Project Review Document.

Difference between Traditional development vs. DSDM



Method	Functional/Requirements	Time & Resources
Traditional Method	Fixed	Can vary
DSDM/Agile Method	Can vary	Fixed

Techniques to consider in DSDM

Concept	Description
Flexibility	Adapt to changes in requirements and priorities.
Timeboxing	**Set fixed time limits** for tasks to ensure progress.
MoSCoW Rules	Prioritize requirements as **Must-have, Should-have, Could-have, Won't-have**.
Prototyping	Develop early versions of the system for feedback and improvement.
Facilitated Workshops	Gather key stakeholders to discuss and agree on project aspects.

DSDM Techniques: Flexibility

- **DSDM assumption: Nothing is built perfectly the first time**
- **80:20 Rule:** 80% of the system can be built in 20% of the total time.
- **Traditional method issues:** Too much time spent perfecting, leads to delays and budget overruns.
- **DSDM approach**: Steps can be revisited later, complete only enough to move forward.

DSDM Techniques: Timeboxing

Concept	Main Points
Importance of Timeboxing	• Keeps teams focused and under control.
How It Works	• Focuses on achieving business objectives, not just tasks.
Timeboxing Basics	• Fixed start & end time; Uses nested timeboxes (2-4 weeks).
Key Objective	• Complete the easiest 80%, carry forward the remaining 20%.
Benefits	• Focus on essentials, improves estimation & resource allocation.

DSDM Techniques: MoSCoW Rules

- **MoSCoW rules formalized in DSDM v3.**
- **Must Have** – Essential for project success.
- **Should Have** – Important but not critical.
- **Could Have** – Can be left out without major impact.
- **Won't Have** – Deferred for future development (Waiting List).

DSDM Techniques: Prototyping

Concept	Main Points
Importance of Prototypes	• Essential in DSDM.
Requirement Validation	• Helps users confirm detailed requirements.
Workshops & Strategy	• Define high-level requirements and strategy.
User Awareness & Feedback	• Demonstrations help users understand possibilities and provide feedback.
Development Speed & Accuracy	• Speeds up development and ensures the right solution is delivered.

DSDM Techniques: Facilitated Workshops

➢ **Purpose:**

- Produce clear outcomes by consensus

➢ **Participants:**

- Workshop sponsor
- Development team (Participants)
- Scribes (record)
- Observers
- Prototypers
- Facilitator (help a group of people understand their common objectives and assists them to plan how to achieve these objectives)

➢ **Advantages:**

- Speed
- Involvement/ownership
- Productivity
- Consensus
- Quality decisions
- Overall perspective/synergy

MCQ Practice

1. **What is DSDM?**

   A) A programming language
   B) A Rapid Application Development method
   C) A project management tool
   D) A database system

2. **DSDM is considered the de-facto standard for which country?**

   A) USA
   B) Canada
   C) UK
   D) Australia

3. **What is the primary goal of DSDM?**

   A) To reduce costs
   B) To deliver what business needs when it needs
   C) To increase project duration
   D) To create complex systems

4. **Which of the following is NOT a technique used in DSDM?**

   A) Flexibility
   B) Timeboxing
   C) Waterfall
   D) MoSCoW Rules

5. **What does MoSCoW stand for?**

   A) Must, Should, Could, Won't
   B) Must, Should, Can, Won't
   C) Must, Should, Could, Will
   D) Must, Should, Can, Will

6. **In DSDM, what is the purpose of timeboxing?**

   A) To extend project timelines
   B) To concentrate on when a business objective will be met
   C) To reduce team size
   D) To eliminate prototypes

7. **Which stage assesses the suitability of DSDM for a project?**

   A) Business Study
   B) Feasibility Study
   C) Implementation
   D) Design and Build

   Answer: B

8. **What is produced during the Feasibility Study?**

   A) A functional prototype B) A feasibility report

   C) A project review document D) A user documentation

9. **What does the Business Study stage focus on?**

   A) Analyzing project costs B) Analyzing business and technology characteristics

   C) Finalizing user documentation D) Conducting user training

10. **What is the primary deliverable of the Functional Model Iteration?**

    A) Design Prototype B) Functional Prototype

    C) Feasibility Report D) Project Review Document

11. **Which of the following is a key assumption of DSDM?**

    A) All steps must be completed perfectly the first time

    B) Previous steps can be revisited

    C) Prototypes are unnecessary

    D) User involvement is not needed

12. **What is the duration of a typical timebox in DSDM?**

    A) 1-2 weeks B) 2-4 weeks

    C) 4-6 weeks D) 6-8 weeks

13. **What does the term "80:20 Rule" refer to in DSDM?**

    A) 80% of the project should be completed in 20% of the time

    B) 80% of the requirements are fixed

    C) 80% of the budget is spent on 20% of the work

    D) 80% of users will use 20% of the features

14. **What is produced during the Implementation stage?**

A) Feasibility Report
B) User Approval Document
C) Business Area Definition
D) Functional Prototyping Review Document

15. **What is the main purpose of facilitated workshops in DSDM?**

A) To develop prototypes
B) To produce clear outcomes by consensus
C) To conduct user training
D) To finalize project budgets

16. **Which of the following is a deliverable of the Design and Build Iteration?**

A) Feasibility Prototype
B) Functional Prototype
C) Design Prototype
D) Project Review Document

17. **What is the role of a facilitator in a workshop?**

A) To develop the system
B) To help the group understand objectives
C) To record meeting minutes
D) To approve the project

18. **What does the term "flexibility" imply in DSDM?**

A) Fixed requirements
B) Ability to adapt to changes
C) Longer project timelines
D) No prototypes

19. **What is the significance of prototypes in DSDM?**

A) They slow down the development process
B) They help in defining high-level requirements
C) They are not necessary
D) They are used only for testing

20. **What is the outcome of the Review Business stage?**

A) Approval for implementation
B) Feasibility Report
C) Functional Prototype
D) Design Prototype

21. **Which of the following is a key feature of traditional methods compared to DSDM?**

A) Fixed functionality
B) Flexible timelines
C) Iterative development
D) User involvement

22. **What is the average project team size in DSDM?**

A) 5
B) 10
C) 15
D) 20

23. **What percentage of completed projects using DSDM are rated good to excellent?**

A) 50%
B) 70%
C) 87%
D) 90%

24. **What is the purpose of a Risk Log in DSDM?**

A) To document user feedback
B) To track project costs
C) To identify and manage risks
D) To finalize project timelines

25. **Which of the following is a sub-activity of the Functional Model Iteration?**

A) Create Design Prototype
B) Identify Functional Prototype
C) Review Business
D) Train Users

26. **What is a key difference between DSDM and traditional methods?**

A) DSDM has fixed requirements
B) DSDM allows for revisiting previous steps
C) Traditional methods are faster
D) DSDM does not involve users

27. **What does the term "delivered system" refer to in DSDM?**

A) The final prototype
B) The system implemented at user locations
C) The feasibility study report
D) The design prototype

28. **What is the outcome of the Review Functional Prototype stage?**

A) Approval for implementation B) Functional Prototyping Review Document

C) Feasibility Report D) Design Prototype

29. **What is the main benefit of using DSDM?**

A) Increased project costs B) Reduced user involvement

C) Faster delivery of business needs D) Fixed project timelines

30. **What is the focus of the Design Prototype?**

A) To test functionality B) To create a system for daily use

C) To finalize project budgets D) To document user feedback

31. **What does the term "user documentation" refer to in DSDM?**

A) Documentation for developers B) Instructions for end-users

C) Project budget documentation D) Feasibility study report

32. **What is the purpose of the Global Outline Plan in DSDM?**

A) To finalize project budgets B) To outline the entire project scope

C) To document user feedback D) To create a detailed design

33. **Which document is created during the Business Study stage?**

A) Feasibility Report B) Prioritized Requirements List

C) Functional Prototype D) User Documentation

34. **What is the main objective of the Implementation stage?**

A) To develop prototypes B) To review business needs

C) To implement the tested system D) To conduct user training

35. **What is the role of the project sponsor in a facilitated workshop?**

A) To develop the system
B) To record meeting minutes
C) To provide project funding
D) To help achieve consensus

36. **What is the main focus of the Design and Build Iteration?**

A) To conduct user training
B) To create and test prototypes
C) To finalize project budgets
D) To review business needs

37. **What is the purpose of the Approval stage in DSDM?**

A) To finalize project budgets
B) To get user approval for implementation
C) To create prototypes
D) To document user feedback

38. **Which of the following is a key feature of DSDM?**

A) Fixed requirements
B) Iterative development
C) No user involvement
D) Longer project timelines

39. **What is the primary focus of the Review Business stage?**

A) To implement the system
B) To assess the impact of the system on business
C) To create prototypes
D) To finalize project budgets

40. **What does the term "functional prototype" refer to?**

A) A design document
B) A system that meets user requirements
C) A feasibility report
D) A project budget

41. **What is the purpose of the Risk Log in DSDM?**

A) To track project costs
B) To identify and manage risks
C) To document user feedback
D) To finalize project timelines

42. **What is the average time to delivery using DSDM?**

A) 2-4 months
B) 4-6 months
C) 6-8 months
D) 8-10 months

43. **What is the main advantage of using prototypes in DSDM?**

A) They slow down the development process
B) They help in defining high-level requirements
C) They are not necessary
D) They are used only for testing

44. **What is the role of the development team in a facilitated workshop?**

A) To record meeting minutes
B) To develop the system
C) To provide project funding
D) To help achieve consensus

45. **What is the significance of user training in DSDM?**

A) It is not necessary
B) It helps users understand the system
C) It slows down the development process
D) It is only for developers

46. **What is the primary deliverable of the Design and Build Iteration?**

A) Feasibility Prototype
B) Functional Prototype
C) Design Prototype
D) Project Review Document

47. **What is the focus of the Review Business stage?**

A) To implement the system
B) To assess the impact of the system on business
C) To create prototypes
D) To finalize project budgets

True/False Practice

1. DSDM is a Rapid Application Development method.
2. The primary goal of DSDM is to deliver what the business needs when it needs it.
3. In DSDM, requirements are fixed and cannot change during the project.
4. MoSCoW rules are used to prioritize requirements in DSDM.
5. Timeboxing in DSDM refers to extending project timelines indefinitely.
6. The Feasibility Study stage assesses whether DSDM is suitable for a project.
7. Prototypes are unnecessary in the DSDM process.
8. The average project team size in DSDM is typically around 5 members.
9. DSDM allows for revisiting previous steps as part of its iterative approach.
10. The Review Business stage focuses on finalizing project budgets.

FEATURE DRIVEN DEVELOPMENT (FDD)

Ch: 7

Feature Driven Development (FDD)

⇨ **Feature-Driven Development (FDD)** is an **agile software development methodology** that focuses on building and delivering software features in **short, iterative cycles**. It emphasizes **modeling, planning, and incremental progress** through five key activities:

1. **Develop an Overall Model**
2. **Build a Feature List**
3. **Plan by Feature**
4. **Design by Feature**
5. **Build by Feature**

FDD is best suited for **large-scale projects** and promotes **efficiency, collaboration, and continuous progress tracking**.

History of FDD

- **Creator:** Jeff De Luca
- **Year & Place:** 1997, Singapore
- **Origin:** Evolved from **Bank Loan Automation** project
- **Role:** Jeff De Luca was the **Project Manager**
- **Team Size:** 50 developers
- **Focus:** Incremental, feature-based software development

What is FDD?

- **Agile software development process**
- **Short-iteration model** for continuous progress
- **Combines advantages of other agile methods** and industry best practices
- **Designed for scalability** in large projects and teams

What is a Feature?

- **Delivers the system feature by feature**
- **Feature** = Small function expressed in **client-valued terms**
- **Iteration:** Features should be **completed within 2 weeks** (typically **1-5 days**)
- **Larger features** must be **broken into smaller ones**
- **Feature Naming Template:**
  - `<action> the <result> <by|for|of|to> a(n) <object>`
  - **Examples:**
    - **Calculate the total of a sale**
    - **Validate the password of a user**
    - **Authorize the sales transaction of a customer**

Class Ownership

- **Class (feature) assigned to a specific developer**
- **Class owner** responsible for all changes in implementing new features
- **Collective Ownership:**
  - Any developer can modify any artifact at any time
- **Advantages of Class Ownership:**
  - Ensures **integrity** of each class
  - Provides a **dedicated expert** for each class
  - **Faster** implementation of changes
  - Supports the **concept of code ownership** (similar to XP)

FDD Roles

Role Type	Roles	Responsibilities
Primary Roles	**Project Manager**	Oversees the project, ensures timelines & deliverables
	Chief Architect	Defines system architecture & high-level design
	Class Owners	Owns specific classes, responsible for their integrity
	Domain Experts	Provides business/domain knowledge
	Chief Programmers	Leads development teams, ensures feature implementation
Supporting Roles	**Language Guru**	Maintains shared coding vocabulary & standards
	Toolsmith	Develops tools for automation & efficiency
	Tester	Conducts testing to ensure software quality
	Technical Writer	Creates documentation for users & developers

FDD Process

1. **Develop an Overall Model** – Create a high-level system design
2. **Build a Features List** – Identify and define all required features
3. **Plan By Feature** – Prioritize and schedule feature implementation
4. **Design By Feature** – Create detailed design for each feature
5. **Build By Feature** – Implement, test, and integrate each feature

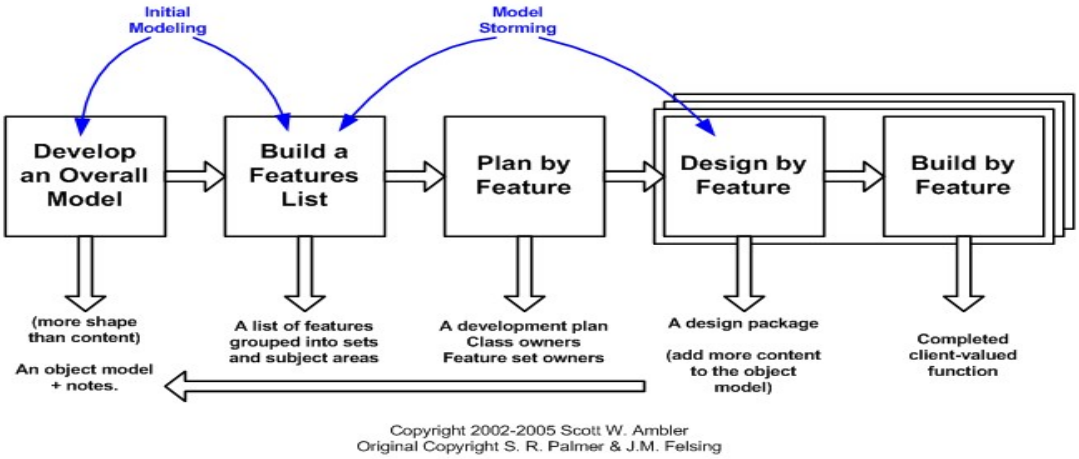
FDD Process Breakdown

⇨ Project-Wide Upfront Design Activities:

- **Process #1: Develop an Overall Model**
- **Process #2: Build a Features List**
- **Process #3: Plan By Feature**
- **Goal:** Do minimal initial design to enable iterative development

⇨ Feature-Based System Delivery:

- **Process #4: Design By Feature**
- **Process #5: Build By Feature**
- **Goal:** Deliver real, client-valued functionality frequently



FDD Process Details

Process #1: Develop an Overall Model

- **Form a modeling team**
- **Domain walk-through** to understand the business context
- **Build a high-level object model** to represent key concepts
- **Record notes** to document insights and decisions
- **Goal:** Ensure team members have a shared understanding of the problem domain and establish a foundation for development

Process #2: Build a Features List

- **Features are organized into a three-level hierarchy**

Process #3: Plan by Feature

- **Construct initial schedule**
  - Based on individual feature levels
  - Prioritize features by **business value**
  - Consider **dependencies, difficulty, and risks**
- **Assign responsibilities to team members**
  - Determine **Class Owners**
  - Assign feature sets to **Chief Programmers**

Process #4: Design by Feature

- **Form Feature Teams**
  - Teams collaborate on detailed analysis and design
  - May involve **domain experts** for specific features
  - Teams update the **model artifact** as they make changes
- **Feature Team Setup:**
  - **Chief Programmers** form teams based on the feature in development
  - Teams are typically **3 to 5 people**
  - Teams **disband** after completing the feature
  - Teams work on **independent iterations**
  - It is possible to be on **multiple teams**

Process #5: Build by Feature

- **Implement and Test the Feature**
  - Perform **unit-level testing**
  - Conduct **feature-level testing**
  - **Code Inspections** (formal review with checklist)
  - Integrate the feature into the **regular build**

FDD Key Points

➢ **Mandated Code Inspections:**

- **Reasons for Inspections:**
  - **More effective at finding bugs** and a broader range of bug types than other testing methods
  - **Great learning experience** for team members
- **Reporting:**
  - FDD emphasizes providing **accurate, meaningful, and timely progress updates** to all stakeholders
- **Feature Milestones:**
  - Milestones track progress at the feature level, ensuring **timely delivery** and **client value**

MCQs Practice

1. **Original creator of Feature Driven Development (FDD)?**
   A) Jeff De Luca
   B) Scott Ambler
   C) Martin Fowler
   D) Kent Beck

2. **In which year was FDD created?**
   A) 1995
   B) 1997
   C) 2000
   D) 2005

3. **Primary focus of FDD?**
   A) Documentation
   B) Feature delivery
   C) Code quality
   D) Team collaboration

4. **How long should each feature take to complete in FDD?**
   A) 1-2 days
   B) 1-5 days
   C) 1-2 weeks
   D) 1 month

5. **What is a key advantage of FDD?**
   A) It is easy to implement
   B) It scales well to larger projects
   C) It requires minimal documentation
   D) It focuses solely on coding

6. **Which of the following is NOT a primary role in FDD?**
   A) Project Manager
   B) Class Owner
   C) Tester
   D) Marketing Specialist

7. **FDD process start with?**
   A) Build a Features List
   B) Develop an Overall Model
   C) Plan by Feature
   D) Design by Feature

8. **The purpose of 'Class Ownership' in FDD?**
   A) To assign all tasks to one developer
   B) To ensure integrity of each class
   C) To limit modifications to specific classes
   D) To enhance team collaboration

9. **Which process involves identifying feature owners?**
   A) Build a Features List
   B) Plan by Feature
   C) Design by Feature
   D) Build by Feature

10. **What is emphasized in FDD reporting?**
    A) Code quality
    B) Timely progress information
    C) Team meetings
    D) Client feedback

11. **What is the maximum time for completing a feature in FDD?**
    A) 1 day
    B) 2 days
    C) 1 week
    D) 2 weeks

12. **Which role is responsible for making tools for the application in FDD?**
    A) Class Owner
    B) Chief Programmer
    C) Toolsmith
    D) Tester

13. **What is the goal of the 'Design by Feature' process?**
    A) To finalize the project
    B) To create a design package
    C) To test the features
    D) To gather client feedback

14. **Which of the following is a feature naming template in FDD?**
    A) Calculate the total of a sale
    B) Develop a new application
    C) Write documentation
    D) Test the software

15. **The purpose of 'Mandated Code Inspections' in FDD?**
    A) To speed up development
    B) To find bugs and improve learning
    C) To reduce team size
    D) To finalize the project

16. **FDD is primarily classified as which type of development process?**
    A) Waterfall
    B) Agile
    C) V-Model
    D) Spiral

17. **What is the maximum number of features a team can work on simultaneously in FDD?**
    A) One
    B) Two
    C) Three
    D) Multiple

18. **Which of the following is a supporting role in FDD?**
    A) Project Manager
    B) Chief Architect
    C) Tester
    D) Class Owner

19. **What is the primary goal of FDD?**
    A) Complete documentation
    B) Deliver client-valued functions
    C) Minimize team size
    D) Maximize code complexity

20. **The first step in the FDD process?**
    A) Build a Features List
    B) Design by Feature
    C) Develop an Overall Model
    D) Plan by Feature

True/False Practice

1. FDD was created to focus on small teams only.
2. Features in FDD should take no more than two weeks to complete.
3. FDD is a linear development process.
4. Class ownership in FDD allows any developer to modify any artifact at any time.
5. The FDD process includes a step for building a features list.
6. FDD emphasizes individual contribution over team collaboration.
7. FDD can be scaled to larger projects and teams.
8. The primary goal of FDD is to deliver features based on client value.
9. FDD requires extensive documentation before starting development.
10. Mandated code inspections are a part of the FDD process.

REQUIREMENT'S ENGINEERING

Ch: 8

Requirement's engineering (RE)

⇨ Requirements Engineering (RE) is a **crucial phase** in software and system development that ensures the final product meets stakeholder needs. It involves:

1. **Elicitation** – Gathering requirements from users and stakeholders.
2. **Analysis** – Evaluating feasibility, conflicts, and priorities.
3. **Specification** – Clearly documenting requirements.
4. **Validation** – Ensuring correctness and completeness.
5. **Management** – Handling requirement changes throughout development.

A well-executed RE process helps prevent misunderstandings, reduces development costs, and ensures project success.

Requirements engineering phases

1. **Inception**: Define project goals and scope.
2. **Elicitation**: Gather requirements from stakeholders.
3. **Analysis and Elaboration**: Refine and clarify requirements.
4. **Negotiation**: Resolve conflicts and prioritize requirements.
5. **Specification**: Document requirements clearly.
6. **Validation**: Ensure requirements are correct and complete.
7. **Requirements Management**: Track and manage requirements changes.

Inception

Inception – Key questions to establish:

- **Basic understanding of the problem**: What is the core issue to solve?
- **Stakeholders**: Who are the people or organizations involved?
- **Nature of the solution**: What type of solution is desired (technical, functional)?
- **Communication and collaboration**: How effective is the interaction between the customer and developer?
- **Economic benefit**: What are the expected financial or operational gains from the solution?

Requirements Elicitation

- **Interviewing stakeholders**: Conduct interviews using a pre-determined questionnaire.
- **Meetings**: Organize sessions involving both software engineers and customers.
- **Observation & ethnography**: Observe users in their natural environment to gather insights.
- **Definition mechanism**: Use tools like worksheets, flip charts, or virtual forums to collect requirements.
- **Goal**:
    - Identify the problem.
    - Propose elements of the solution.
    - Specify a preliminary set of solution requirements.

Requirements Elaboration

⇨ **Building the Analysis Model:**

1. **Scenario-based elements**:
    - **Functional**: Narratives describing software functions.
    - **Use-case**: Descriptions of interactions between an actor and the system.
2. **Class-based elements**:
    - Derived from scenarios to identify system classes and their relationships.
3. **Behavioral elements**:
    - **State diagram**: Represents system states and transitions.
4. **Flow-oriented elements**:
    - **Data flow diagram (DFD)**: Visualizes data movement through the system.
    - **Sequence diagram**: Shows the interaction order between system components.
    - **Activity diagram**: Illustrates workflow and activities within the system.

Requirements Analysis

➢ **Requirements Analysis:**

- Specifies **software's operational characteristics**.
- Identifies **software's interface** with other system elements.
- Establishes **constraints** the software must meet.
- Enables the software engineer to:
    - Elaborate on **basic requirements** from earlier tasks.
    - Build **models** for user scenarios, functional activities, problem classes, system behavior, and data flow transformations.

➢ **Requirements Analysis Modeling:**

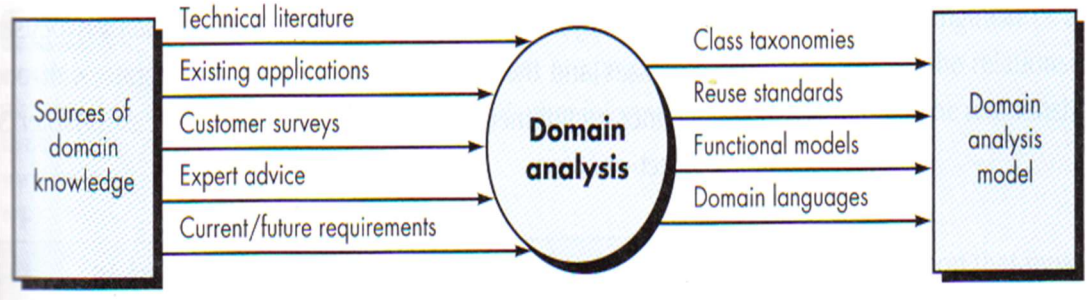
- **Build models** using requirements elicited from the customer.
- **First technical representation** of the system.
- **Provides quality access** for both the developer and the customer after software is built.
- Focus on **WHAT** the system should do, not **HOW** it will be implemented.

➢ **Requirements Analysis Modeling Objectives**:

- **Describe** what the customer requires.
- **Establish** a foundation for creating software design.
- **Define** requirements that can be validated after the software is built.

Domain Analysis

1. **Define the domain** to be investigated.
2. **Collect a representative sample** of applications within the domain.
3. **Analyze each application** in the sample to identify patterns and requirements.
4. **Develop an analysis model** for the identified objects and their relationships.

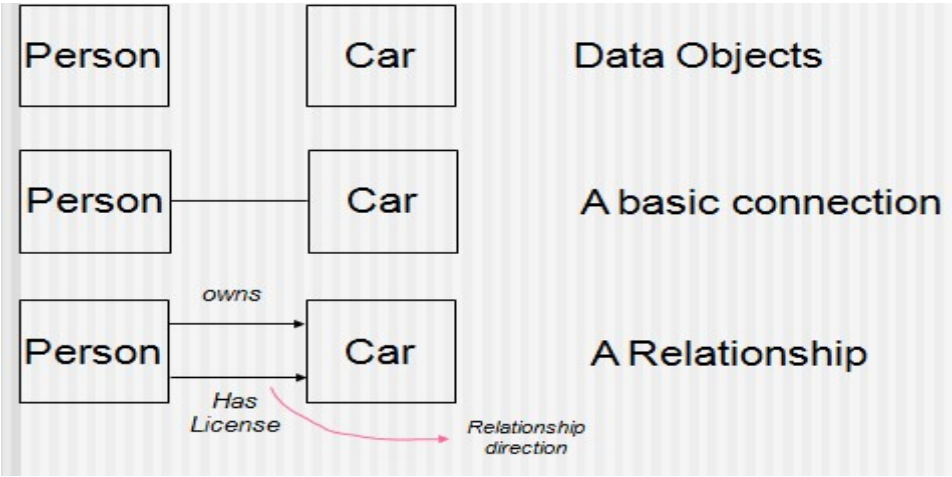


Data Modelling

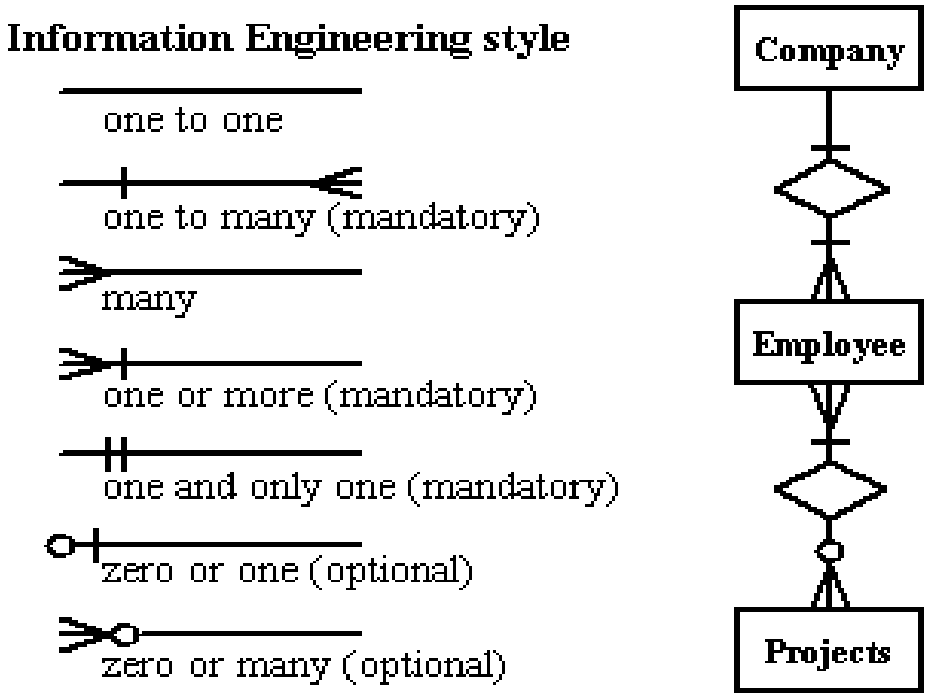
- **Defines relationships** between data objects.
- **Data object**: Represents composite information with multiple attributes (e.g., **Dimension** instead of just Length or Breadth).
- Examples of **Data Objects**:
  - **External entities**: Printer, user, sensor.
  - **Things**: Reports, displays, signals.
  - **Occurrences/events**: Interrupt, alarm.
  - **Roles**: Manager, engineer, salesperson.
  - **Organizational units**: Division, team.
  - **Places**: Manufacturing floor.
  - **Structures**: Employee record.

Data objects & relationships

⇨ Data objects are connected to one another in different ways.

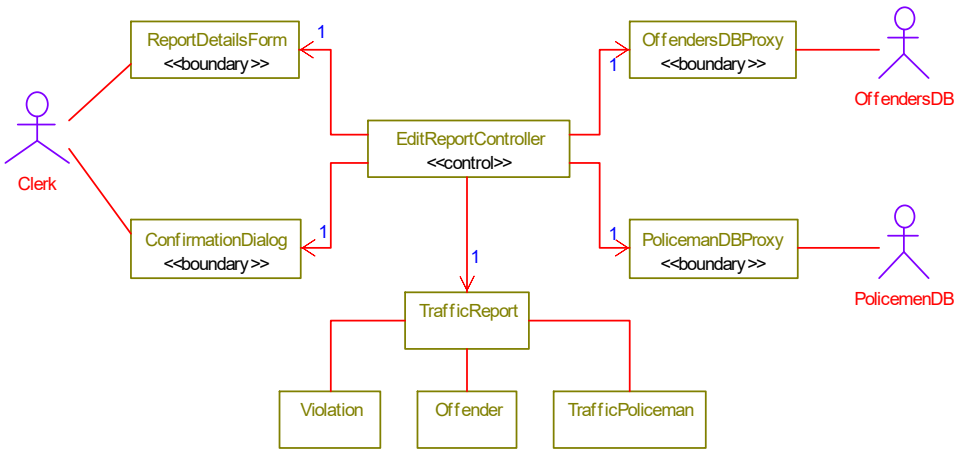


Cardinality – ERD Notation



Classes Categorization

1. **Boundary Classes (UI)**:
   - Model interactions between the system and external environment.
   - **User Interface Classes**: Focus on **what** information is presented, not UI details.
   - **System/Device Interface Classes**: Define protocols, not how they are implemented.
2. **Entity Classes**:
   - Represent **key system concepts**.
   - Store **persistent information**.
   - Contain **logic** for solving system problems.
   - Can be reused across multiple behaviors.
3. **Control Classes**:
   - **Control and coordinate** system behavior.
   - Only **delegate tasks** to other classes, not perform them.
   - Act as an **intermediary** between boundary and entity classes.



CRC Card

- Class Responsibility Collaboration
- CRC goals: provide the simplest possible conceptual introduction to OO design

class name	
subclasses:	
superclasses:	
Responsibilities	Collaborators

Figure 2-2 *A CRC card sample*

Class: FloorPlan

Description:

Responsibility:	Collaborator:
defines floor plan name/type	
manages floor plan positioning	
scales floor plan for display	
scales floor plan for display	
incorporates walls, doors and windows	Wall
shows position of video cameras	Camera

- **Physical index cards** (3×5" or 4×6") used for class modeling.
- **Encourages responsibility division** across objects.
- **Limits class size and complexity** due to physical card constraints.
- **Not a UML tool**, but helps **identify class details** for UML diagrams.
- **Card structure**:
  - **Left side**: Lists **class responsibilities**.
  - **Right side**: Lists **collaborators** (other objects that assist in fulfilling responsibilities).

Requirements Negotiation

1. **Identify Key Stakeholders**: Determine who is involved in the negotiation process.
2. **Determine Stakeholders' "Win Conditions"**: Understand their needs, goals, and expectations (not always obvious).
3. **Negotiate & Prioritize**: Resolve conflicts and prioritize requirements.
4. **Achieve a "Win-Win" Outcome**: Ensure all stakeholders agree on a feasible, balanced set of requirements.

Requirements Validation

➢ **Requirements Validation Checklist:**

✔ **Consistency** – Does each requirement align with the system's objectives?
✔ **Abstraction Level** – Are requirements specified at the right level of detail?
✔ **Necessity** – Is the requirement essential or just an optional add-on?
✔ **Clarity** – Is each requirement unambiguous?
✔ **Conflict Check** – Do any requirements contradict each other?
✔ **Feasibility** – Can each requirement be implemented in the given technical environment?
✔ **Testability** – Can each requirement be verified once implemented?
✔ **Model Accuracy** – Does the requirements model correctly represent system information, function, and behavior?
✔ **Requirements Patterns** – Have patterns been used to simplify the model?
✔ **Pattern Validation** – Are all patterns validated and aligned with customer needs?

The Requirements Baseline

⇨ **Definition**: A reviewed and agreed-upon set of requirements that serves as the foundation for development.

⇨ **Stakeholder Confidence from Baselining:**

✔ **Customer Management/Marketing** – Ensures project scope remains controlled.
✔ **User Representatives** – Assures that the right solution will be delivered, even if all requirements weren't initially identified.
✔ **Development Management** – Provides a structured approach to balancing schedule, cost, functionality, and quality.
✔ **Business Analysts & Project Managers** – Helps manage changes effectively and minimize chaos.
✔ **QA & Test Teams** – Enables preparation of accurate test scripts and testing strategies.

MCQs Practice

1) **The first phase of Requirements Engineering?**
   a. Elicitation
   b. Inception
   c. Analysis
   d. Validation

2) **Which technique is used for gathering requirements from stakeholders?**
   a. Code Review
   b. Elicitation
   c. Testing
   d. Deployment

3) **What does requirements analysis specify?**
   a. User interface design
   b. Software's operational characteristics
   c. Hardware requirements
   d. Marketing strategies

4) **What is the goal of requirements validation?**
   a. To gather more requirements
   b. To ensure requirements are achievable and testable
   c. To finalize the project budget
   d. To begin coding

5) **Which of the following is NOT a phase in Requirements Engineering?**
   a. Specification
   b. Negotiation
   c. Implementation
   d. Elicitation

6) **What is a requirements baseline?**
   a. A set of requirements that has been reviewed and agreed upon
   b. A draft of initial requirements
   c. A list of rejected requirements
   d. A final software product

7) **Which of the following is a method for requirements elicitation?**
   a. Code Inspection
   b. Interviews
   c. Debugging
   d. Deployment

8) **What do boundary classes model?**
   a. Internal system logic
   b. User interactions with the system
   c. Database connections
   d. Network protocols

9) **The purpose of data modeling?**
   a. To design user interfaces
   b. To indicate how data objects relate to one another
   c. To write code
   d. To conduct tests

10) **Which of the following is a behavioral element in requirements elaboration?**
   a. Use-case diagram
   b. Data flow diagram
   c. State diagram
   d. All of the above

11) **Main focus during requirements analysis modeling?**
   a. How to implement the software
   b. What the customer requires
   c. Testing the software
   d. Marketing the product

12) **Which of the following is a true statement about requirements negotiation?**
   a. It involves only developers.
   b. It aims for a win-win solution.
   c. It is done after implementation.
   d. It is not necessary in software development.

13) **What does a CRC card represent?**
   a. Class Responsibility Collaboration
   b. Code Review Checklist
   c. Customer Requirement Criteria
   d. Class Reference Code

14) **Primary outcome of the requirements elicitation phase?**
   a. Final software product
   b. Preliminary set of solution requirements
   c. User interface design
   d. Testing plan

15) **Which of the following is NOT a characteristic of a good requirement?**
   a. Unambiguous
   b. Vague
   b. Testable
   d. Necessary

16) **Purpose of domain analysis?**
   a. To define the project scope
   b. To collect and analyze applications in a specific domain
   c. To design the user interface
   d. To write code

17) **What is the role of control classes?**
   a. To manage user interactions
   b. To store data
   c. To coordinate system behavior
   d. To define system protocols

18) **The significance of stakeholder identification in the inception phase?**
   a. To finalize the budget
   b. To understand who will use the system
   c. To begin coding
   d. To conduct tests

19) **What does requirements management involve?**
   a. Gathering new requirements
   b. Tracking and controlling changes to requirements
   c. Writing code
   d. Conducting user tests

20) **The Main benefit of a well-defined requirements baseline?**
   a. It reduces project costs.
   b. It ensures all stakeholders agree on project scope.
   c. It eliminates the need for testing.
   d. It guarantees project success.

True/False Practice

1. Requirement's engineering is only concerned with gathering requirements.
2. Elicitation involves gathering requirements from only the end-users.
3. The analysis phase focuses on specifying software's operational characteristics.
4. A requirements baseline is a set of requirements that has not been reviewed.
5. Boundary classes model the interaction between the system and its environment.
6. Data modeling indicates how data objects do not relate to one another.
7. Requirement's negotiation aims for a win-win situation among stakeholders.
8. Control classes should perform the actual work of the system.
9. The requirements management process is only necessary during the implementation phase.
10. Requirement's validation ensures that all requirements are achievable and testable.

Possible Sample Questions

Part – A True/False

1. The cost of change after release of software is comparatively less than before release.
2. In Inception, we ask a set of questions that establish technical structure of the problem.
3. In plan-driven process personnel succeed on structure and order.
4. Software does not "deteriorate" but "wearing out",
5. Most software continuing to be custom built.
6. Microsoft word in an example of system software.
7. The Developers can change, add, or remove the Sprint Backlog Items after the Sprint Planning meeting has ended, often in consultation with the Product Owner.
8. If we get behind schedule in a software development, we can add more programmers and catch up.
9. Until we get the program "running," we have no way of assessing its quality. the later stages of software development.
10. Waterfall model addresses the maximum risk at the later stages of software development.
11. In iterative development, the products are visible at the early stage of development.
12. XP includes the development team to be located in remote place.
13. In DSDM, functionalities are fixed in respect of varied time and recourses.
14. Sprint review meeting takes place at the beginning of the sprint process.
15. In Incremental development, once the development of an increment starts, the relevant requirements are frozen

Part – B Multiple Choice Questions

16. Which model doesn't allow defining requirements early in the cycle?

    a. Waterfall b. Prototyping c. Iterative Waterfall d. Scrum

17. Waterfall model is not suitable for?

    a. Small Project b. Complex Project c. Accommodating changes d. None

18. Which of the following is the most important phase of the SDLC?

    a. Requirement's analysis b. Coding c. Testing d. Designing

19. Which of the following is not a named phase in the software development life cycle?

a. Assessment b. Maintenance c. Development d. Testing

Matching Questions:

(A) Architectural Design (B) Component Design (C) Code Generations (D) Requirement Modelling

20. Acceptance Testing

21. Integration Testing

22. System Testing

23. Unit Testing

24. Which prioritization decides the urgency of the requirements development?

a. Must have b. Should have c. Could have d. Want to have but Won't have

25. The functionality is "important, but project does not rely on it" sets the priority of

a. Must have b. Should have c. Could have d. Want to have but Won't have

26. Which of the following is mandated in FDD?

a. Workshop b. Code Inspection c. User interface prototyping d. MoSCoW rules

Matching Questions:

(A) Scrum Master (B) Product Owner (C) Scrum Team (D) Management

27. Makes the final decisions of the tasks related to product Backlog

28. Sets the goals

29. Ensures that the project is carried through according to the practices, values and rules of Scrum.

30. Has the authority to decide on the necessary actions and to organize itself in order to achieve the goals of Sprint

Part – C Descriptive Questions:

31. In FDD, what is a Feature? Write the template of a Feature with examples. How is the feature team selected and organized in Feature-Driven-Development?

⇨ Feature in Feature-Driven Development (FDD)

- **Feature**: FDD Delivers the system feature by feature. Feature is a small function expressed in client-valued terms which presents the customer requirements to be developed in software using small iteration. Features should be completed within 2 weeks (typically 1-5 days). Larger features must be broken into smaller ones

- **Feature Naming Template**:

<action> the <result> <by|for|of|to> a(n) <object>

- **Examples**:
  - Calculate the total of a sale
  - Validate the password of a user
  - Authorize the sales transaction of a customer

⇨ Feature Team Selection and Organization in FDD

- **Feature Teams** are formed by **Chief Programmers** based on the feature currently being developed.
- Teams typically consist of **3 to 5 members**.
- Teams collaborate on detailed analysis and design for the feature.
- Teams may involve domain experts for specific features.
- Teams update the model artifacts as they work.
- Teams disband after the feature is completed.
- Developers can be part of multiple feature teams simultaneously.

This approach promotes focused, iterative development with clear ownership and collaboration for each feature.

32. Explain Baselining with importance in Requirement's specification and management? How do you motivate your development team who are new to use Scrum process model? Explain the answer in accordance with scrum principle.

⇨ Baselining in Requirements Specification and Management

- **Baselining**: The process of formally reviewing and agreeing on a set of requirements or documents at a specific point in time, creating a "baseline" that serves as a reference for future development and changes.

- **Importance**:
  - Provides a stable foundation for development.
  - Helps manage changes systematically by comparing new requests against the baseline.
  - Ensures all stakeholders have a common understanding of agreed requirements.
  - Facilitates traceability and impact analysis of changes.
  - Supports project control and quality assurance.

⇨ Motivating a Development Team New to Scrum

- **Use Scrum Principles to Motivate:**

1. **Empowerment & Self-Organization**
   - Encourage the team to self-organize and make decisions, fostering ownership and accountability.

2. **Transparency & Communication**
   - Promote open communication through daily Scrum meetings to share progress, challenges, and plans.

3. **Frequent Delivery & Feedback**
   - Emphasize delivering small, working increments regularly (Sprints), allowing the team to see tangible progress and receive quick feedback.

4. **Collaboration with Stakeholders**
   - Involve the team in discussions with Product Owners and customers to understand the value of their work.

5. **Focus on Continuous Improvement**
   - Use Sprint Retrospectives to reflect on what went well and what can be improved, encouraging a growth mindset.

6. **Sustainable Pace**
   - Ensure a manageable workload to avoid burnout and maintain motivation.

By aligning motivation strategies with Scrum values—commitment, courage, focus, openness, and respect—you create an environment where the team feels supported and driven to succeed.

33. What are the key differences between Agile process and Plan Driven Process? What does Agile propose when Scope cannot be completed in given Timebox? And why? How does 80-20 rule can be applied in the whole project as well as in an iteration?

⇨ Key Differences Between Agile Process and Plan-Driven Process

Aspect	Agile	Plan-Driven
Team Size	Small, flexible teams	Large, structured teams
Environment	Adapts to change	Suited for stable, defined needs
Requirements	Evolving and flexible	Fixed upfront
Style	Iterative and adaptive	Linear and sequential
Customer Role	Continuous involvement	Limited involvement
Documentation	Minimal	Extensive
Change Handling	Welcomes change	Resists change
Progress	Working software	Milestones & documents

⇨ Agile Proposal When Scope Cannot Be Completed in Given Timebox

- **Reduce Scope** to fit the fixed iteration length (timebox) rather than extending the timebox.
- **Why?**
  - Agile values predictability and rhythm through fixed-length iterations (1–4 weeks).
  - Timeboxing enforces discipline, focus, and early feedback.
  - This keeps the team on schedule, avoids delays, and ensures incremental value delivery.
  - Focuses on delivering the most valuable features first.

⇨ Applying the 80-20 Rule in Projects and Iterations

- **80-20 Rule (Pareto Principle)**: 80% of the value can be delivered by 20% of the effort or features.
- **In the Whole Project**:
  - Prioritize the 20% of features that deliver 80% of business value.
  - Focus resources on most valuable functionalities first.
- **In an Iteration**:
  - Complete the easiest 80% of the tasks within the timebox.
  - Carry forward the remaining 20% to future iterations, avoiding delays.

This approach optimizes effort, improves focus, and accelerates delivery of valuable outcomes.

34. What is DSDM in software development, what are its advantages, what is the main focus of DSDM, how does it differ from traditional development methods, and what are the names of the core techniques used in DSDM?

⇨ DSDM in Software Development

It is an **Agile, iterative, and incremental software development framework** primarily used for **Rapid Application Development (RAD)**.

⇨ Goal:

Deliver what the business needs, **when it needs it**.

⇨ Advantages of DSDM

1. **Early and Frequent Deliveries** – Increments are delivered in short timeboxes.
2. **High User Involvement** – Stakeholders are consistently engaged through workshops and feedback.
3. **Predictable Timelines and Budget** – Focuses on **fixed time and resources**, with **flexible scope**.
4. **Improved Communication** – Thanks to facilitated workshops and daily collaboration.
5. **Reduced Risk** – Regular testing, feedback, and delivery reduce late-stage surprises.
6. **Business Focused** – Aligns development strictly with business goals.

⇨ Main Focus of DSDM

- **Meeting Business Needs**
- **On-time and in-budget delivery**
- **Continuous user involvement and feedback**
- **Incremental development using prioritized requirements**

⇨ Differences from Traditional Development Methods

Aspect	Traditional Development	DSDM
Requirements	Fixed upfront	Evolve and prioritized (MoSCoW rules)
Time & Resources	Flexible	Fixed
Delivery	At end of lifecycle	Incremental deliveries
User Involvement	Low	High
Flexibility	Low	High
Adaptability to Change	Difficult	Built-in through iterative development

⇨ Core Techniques Used in DSDM

1. **Flexibility**
    - Adapt to changes in requirements and priorities.

2. **MoSCoW Prioritization**
    - Must have, Should have, Could have, Won't have (for now)

3. **Timeboxing**
    - Fixed time slots (2–4 weeks) to deliver value

4. **Prototyping**
    - Building early versions of functionality for validation

5. **Facilitated Workshops**
    - Collaborative meetings to align on goals, clarify needs, and make decisions quickly