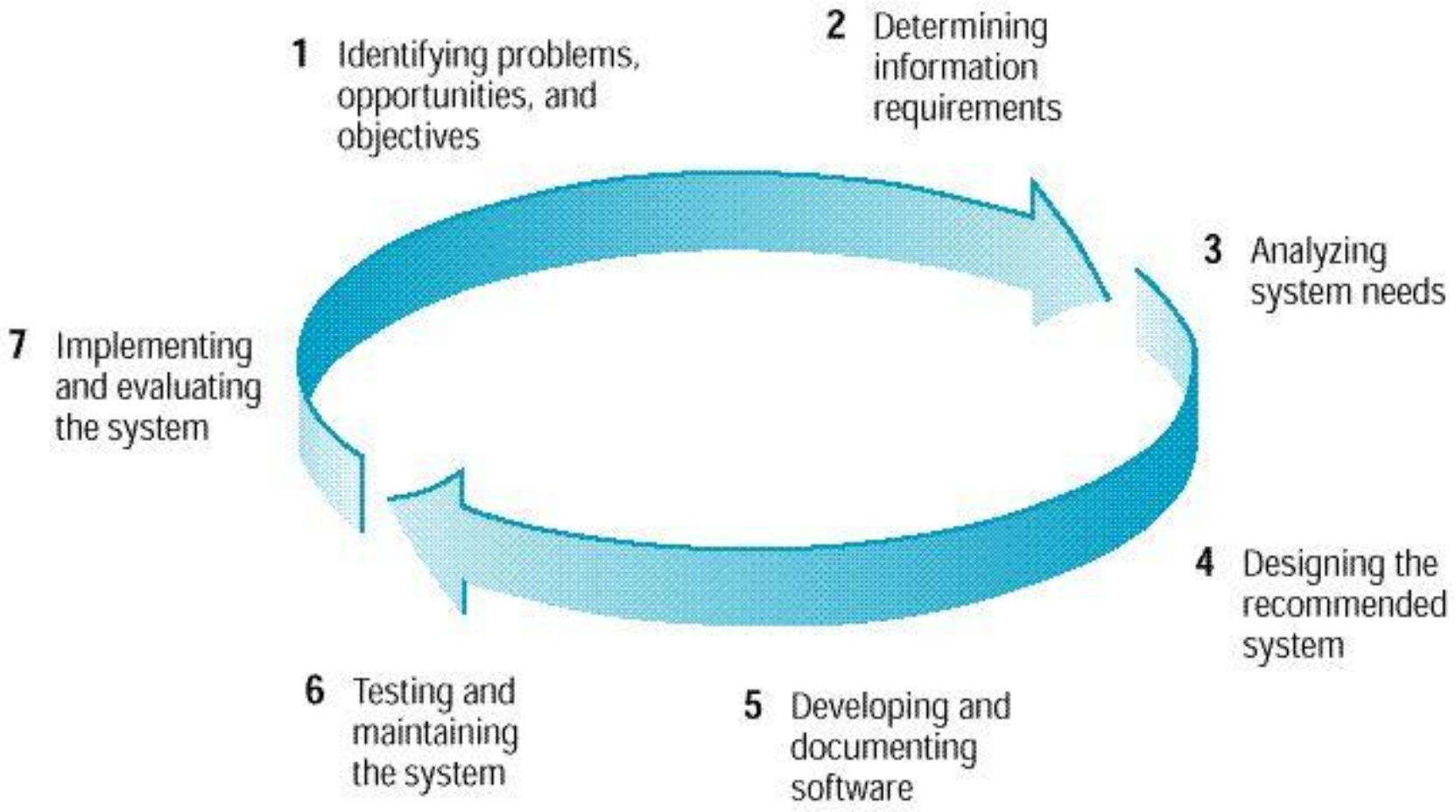
Framework, Methods and Approaches of System Development

ROJO, J/ALIWATE, D/EDO, A EDO/UY, ULRICH/ODARVE, C

SDLC

Figure 1.2 The Seven Phases of the Systems Development Life Cycle



SYSTEM DEVELOPMENT LIFECYCLE

- Refers to the overall process of developing software from conception to deployment and maintenance.

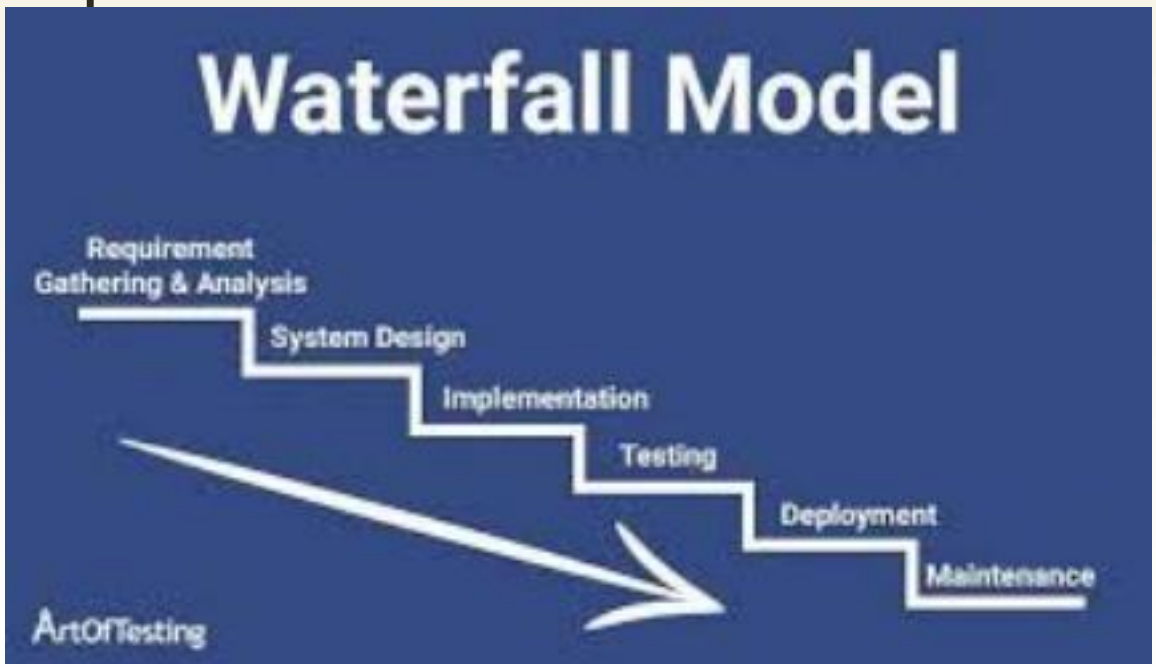
- It is a **framework** that defines the stages (e.g., planning, analysis, design, implementation, testing, deployment, and maintenance) and can follow various models like Waterfall, V-Model, Agile, etc.

TEN SYSTEM DEVELOPMENT METHODS

1. WATER FALL
2. AGILE SYSTEM DEVELOPMENT
3. SPIRAL MODEL
4. V-MODEL (Verification and Validation MODEL)
5. PROTOTYPING MODEL
6. RAD (Rapid Application Development)
7. INCREMENTAL Model
8. DevOps Methodology
9. LEAN Development
10. OBJECT ORIENTED DEVELOPMENT

WATER FALL

1. A linear, sequential approach.
2. Each phase must be completed before the next begins.
3. Best suited for projects with well-defined requirements.



AGILE

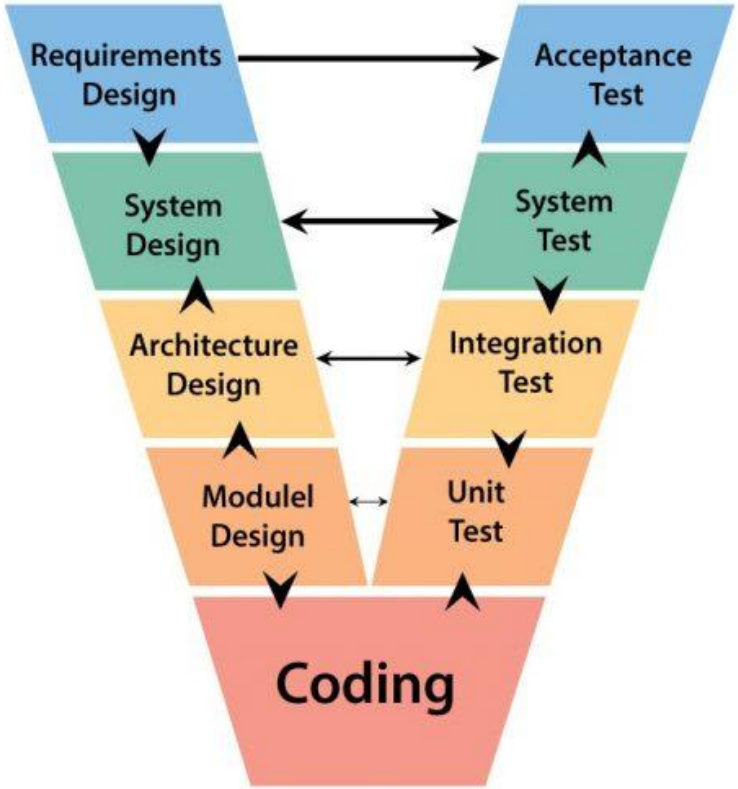
1. An iterative and incremental approach.
2. Focuses on flexibility, continuous feedback, and delivering working software in small increments.
3. Ideal for dynamic projects with evolving requirements.

SPIRAL

1. Combines iterative development with risk management.
2. Focuses on cycles (or "spirals") where each loop involves planning, risk analysis, development, and evaluation.
3. Suitable for large, high-risk projects.

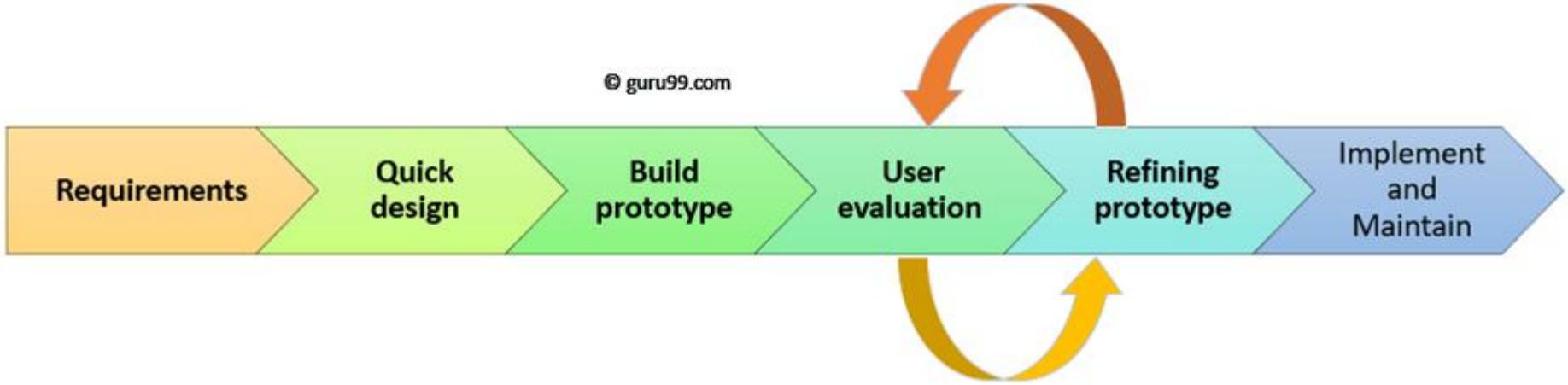
V-MODEL (Verification and Validation)

1. A structured, linear approach where each development phase is directly associated with a corresponding testing phase.
2. Verification (design and development) and validation (testing) activities run in parallel.
3. Relation to SDLC: It is a variant of the Waterfall model, emphasizing early and systematic testing within the SDLC framework.



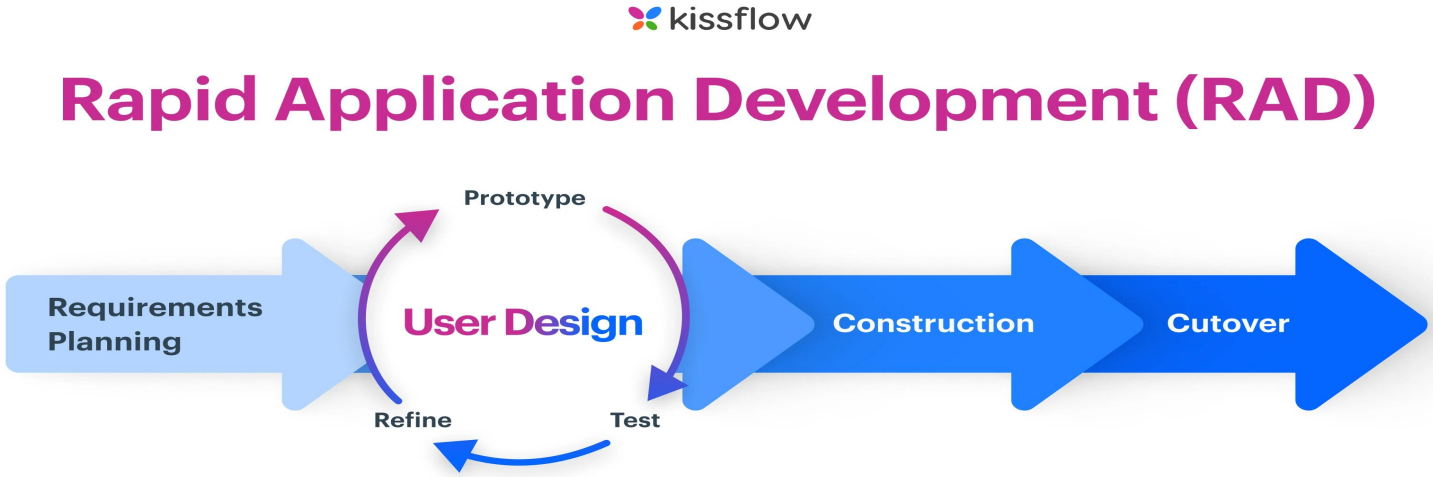
PROTOTYPING -MODEL

1. Focuses on building a prototype early in the development process to understand user requirements better.
2. Feedback from the prototype is used to refine requirements and develop the final product.
3. Relation to SDLC: It emphasizes iterative design and user involvement during the SDLC phases, particularly in requirements gathering and design.



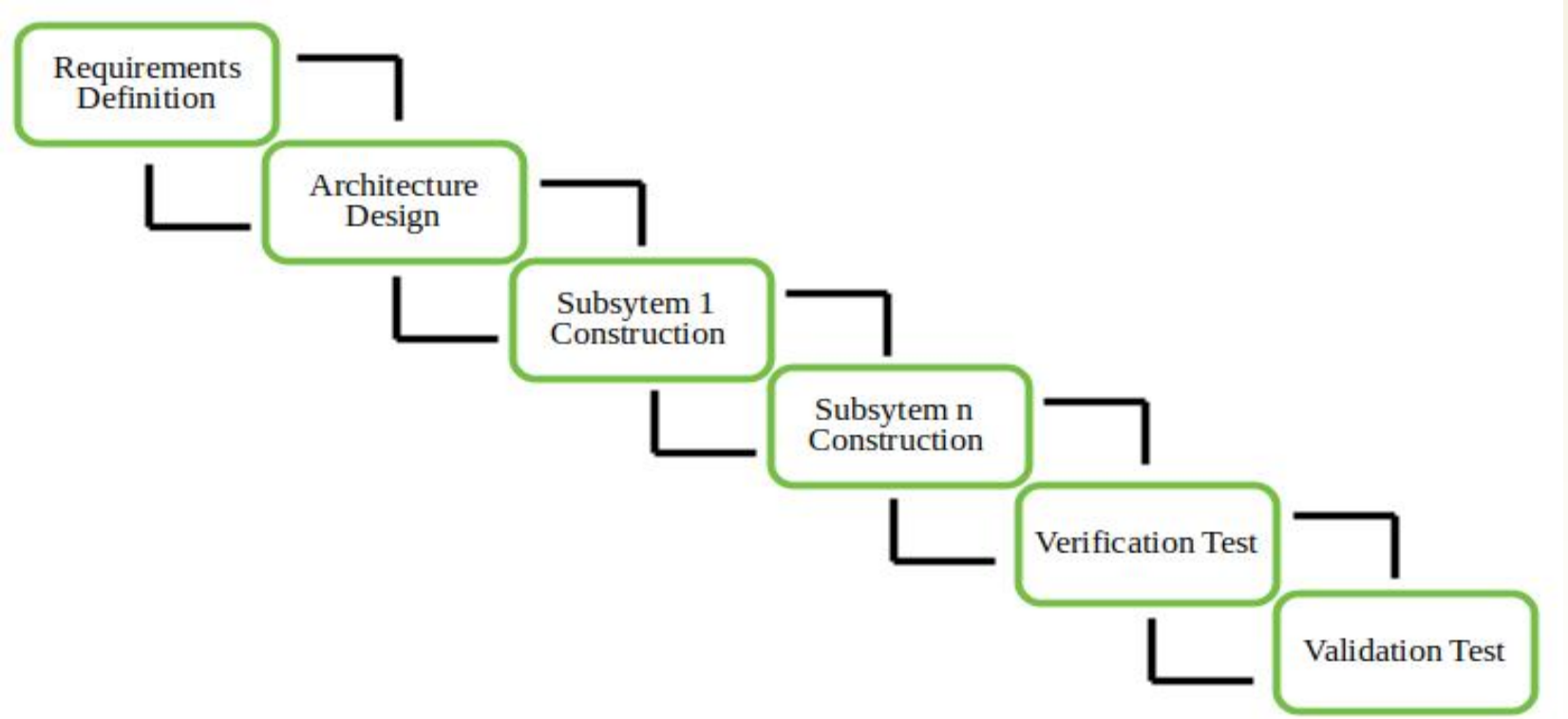
RAD (Rapid application development)

1. A user-centered, iterative approach that emphasizes speed and rapid prototyping.
2. Encourages the use of reusable components and collaborative development.
3. Relation to SDLC: It aligns with SDLC by compressing stages like design, development, and testing into iterative cycles for faster delivery.



INCREMENTAL DEVELOPMENT

1. Divides the project into smaller, manageable parts (increments) that are developed and delivered sequentially.
2. Each increment builds upon the previous one until the final product is completed.
3. Relation to SDLC: It executes the SDLC phases incrementally, providing working software at the end of each cycl



DEVOPS Methodology

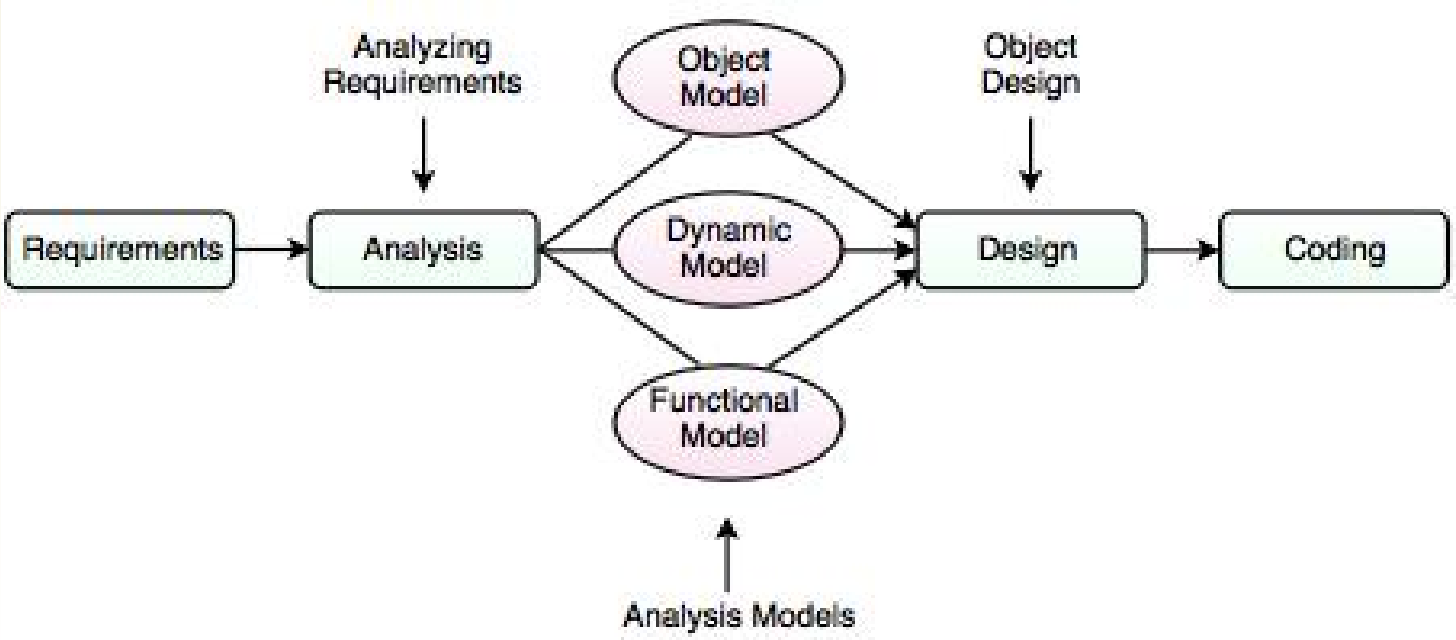
1. Focuses on collaboration between development and operations teams to automate and streamline software delivery.
2. Incorporates CI/CD (Continuous Integration and Continuous Deployment) pipelines and emphasizes monitoring and feedback.
3. Relation to SDLC: It enhances the implementation, deployment, and maintenance stages of SDLC by integrating automation and collaboration throughout the lifecycle

LEAN Methodology

1. Inspired by lean manufacturing principles, it aims to minimize waste and maximize value.
2. Encourages iterative development, continuous improvement, and customer-focused solutions.
3. Relation to SDLC: It optimizes SDLC processes by focusing on efficiency and removing unnecessary steps.

OBJECT-ORIENTED Methodology

1. A design approach based on the principles of object-oriented programming, emphasizing encapsulation, inheritance, and polymorphism.
2. Models the system using objects and their interactions to represent real-world entities.
3. Relation to SDLC: Primarily influences the design and implementation phases of SDLC, but the overall process follows the SDLC structure.



Modern approaches

- All the ten mentioned methods/models are approaches or methodologies that operate within the **SDLC framework**. They provide different ways to manage and execute the SDLC phases, tailoring the process to meet specific project needs, risks, and goals. Some are traditional (like Waterfall, Spiral, V-Model, Prototyping, RAD and Incremental), while others are modern (like **Agile**, **DevOps, OOP** and **Lean** Development). Though RAD is considered as Transitional approach between traditional and modern approach.

Modern approaches

- **Agile** methodology is unequivocally a modern approach, designed to address the dynamic and unpredictable nature of contemporary software development. It is well-suited for projects requiring flexibility, rapid iteration, and close collaboration with stakeholders.

Traditional approaches

- **RAD** is a traditional methodology due to its origins and limitations, its iterative nature and emphasis on user feedback make it a precursor to modern approaches like Agile. RAD serves as a bridge between the rigid traditional methods (e.g., Waterfall) and the adaptive, iterative methods of today.
- **Prototyping** is fundamentally a traditional methodology with iterative principles. While it is not fully modern, its core ideas have significantly influenced the development of modern, user-focused methodologies like Agile, Lean UX, and Design Thinking.

Traditional approaches

- **RAD** is a traditional methodology due to its origins and limitations, its iterative nature and emphasis on user feedback make it a precursor to modern approaches like Agile. RAD serves as a bridge between the rigid traditional methods (e.g., Waterfall) and the adaptive, iterative methods of today.
- **Prototyping** is fundamentally a traditional methodology with iterative principles. While it is not fully modern, its core ideas have significantly influenced the development of modern, user-focused methodologies like Agile, Lean UX, and Design Thinking.

Additional phases uses

Different methodologies and frameworks often add unique phases or expand upon the core SDLC (7) phases (Planning, Analysis, Design, Implementation, Testing, Deployment, and Maintenance) to address specific aspects of software development. Here are additional phases introduced by various methodologies:

1. Feasibility Study
2. Risk Management
3. Prototyping and Validation
4. CI/CD
5. Customer Feedback and Collaboration
6. Monitoring and Optimization
7. User Experience (UX) Design
8. Knowledge Transfer and Training
9. Retirement and Decommissioning
10. Compliance and Security Auditing

Matrics of Phase Methods focus

	Phase	Methodologies	Key Focus
1	Feasibility Study	Spiral, PMBOK	Project viability and risk assessment.
2	Risk Management	Spiral	Proactive identification and mitigation of risks.
3	Prototyping and Validation	Prototyping Model, RAD	Early requirement refinement and validation.
4	CI/CD	DevOps	Automation of integration and deployment.
5	Customer Feedback and Collaboration	Agile, Scrum, Kanban	Incorporating continuous feedback.
6	Monitoring and Optimization	DevOps, ITIL	Ongoing performance tracking and improvement.
7	User Experience (UX) Design	Design Thinking, UX-Centered	Creating intuitive and user-friendly designs.
8	Knowledge Transfer and Training	PMBOK, ITIL	Ensuring smooth adoption and operational continuity.
9	Retirement and Decommissioning	ITIL, DevOps	Safe phasing out of software systems.
10	Compliance and Security Auditing	ITIL, Agile (regulated industries)	Adherence to standards and legal requirements.

Relation with SDLC

Direct Relationship

Frameworks and methodologies explicitly designed for software development align closely with SDLC because they provide structured ways to manage its phases, such as planning, design, development, testing, and maintenance. For example:

Agile, Waterfall, V-Model, Spiral, Incremental, RAD: These are models within the SDLC framework, defining how to execute its stages.

Scrum, Kanban, SAFe: These are specific Agile frameworks that manage and enhance the SDLC's iterative execution.

Relation with SDLC

Indirect Relationship

Other frameworks, while not explicitly part of SDLC, focus on improving areas that overlap with SDLC stages or complement its processes. Examples include:

Project Management Frameworks:

PMBOK, PRINCE2: Help organize and manage the overall execution of SDLC, focusing on project timelines, budgets, and resources.

DevOps Frameworks:

CALMS, ITIL: Enhance the deployment, operations, and maintenance phases of SDLC by emphasizing automation, collaboration, and continuous delivery.

Relation with SDLC

Quality Assurance and Improvement Frameworks:

CMMI, Six Sigma: Improve the quality of the SDLC output by optimizing processes, reducing defects, and ensuring that each phase delivers high-quality results.

Design and Analysis Frameworks:

Domain-Driven Design (DDD), Unified Process (UP): Focus on the design and analysis phases, ensuring a strong foundation for the rest of the SDLC.

Suggested Activities

Review of pre-requisite topics (platform tech)

Activity 6) Title: Implement User Authentication in the CRUD Web Application

Activity 7) Title: Integrate CSS and Basic Front-End Design to the CRUD Application

Activity 8) Title: Connect Prototype Screens to CRUD Application Workflow

Suggested Activities

System Integration and Architecture (Activities)

Activity 9) Title: Deploy CRUD Application in a VirtualBox Environment with Apache Server

Activity 10) Title: Maintenance and Optimization of the CRUD Application

Reference books

- Sage A.P. and Rouse, W.B. Handbook of Systems Engineering and management, John Wiley & Sons, 1999.
- Kendall And Kendall, System Analysis and Design