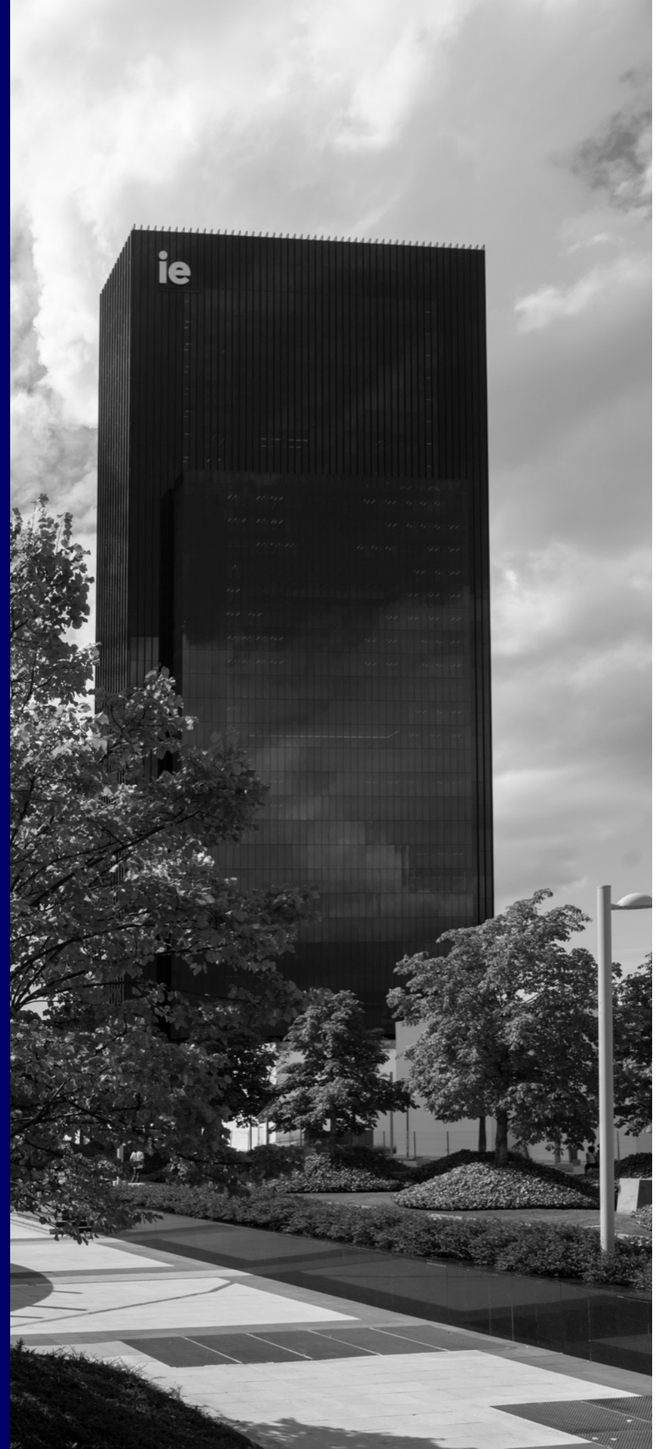


SOFTWARE DEVELOPMENT AND DEVOPS

Session 2

ie UNIVERSITY
SCHOOL OF SCIENCE & TECHNOLOGY

ie UNIVERSITY
SCHOOL OF SCIENCE & TECHNOLOGY

01
SOFTWARE DEVELOPMENT LIFE CYCLE

02
SDLC PHASES

03
SDLC MODELS

04
ROLES & RESPONSIBILITIES IN SDLC MODELS

05
INDIVIDUAL ASSIGNMENT 1

SOFTWARE DEVELOPMENT LIFE CYCLE

ie UNIVERSITY
SCHOOL OF SCIENCE & TECHNOLOGY

WHAT IS SDLC?

Software Development Life Cycle is the structured process for building and maintaining software from idea to retirement.

IMPORTANCE OF SDLC

Structured processes ensure software is delivered reliably, on time, and aligned with user needs.

SDLC PHASES

SDLC PHASES – PLANNING

Planning

aka Feasibility Study, Concept Phase, Project Initiation…

- Define objectives, scope, stakeholders, and high-level requirements
- Assess feasibility (technical, financial, operational, etc.)

SDLC PHASES – REQUIREMENTS ANALYSIS

Requirements Analysis

aka Business Analysis, Discovery, Requirement Engineering…

- Collect functional & non-functional requirements
- Document requirements in Software Requirements Specification (SRS)

SDLC PHASES – DESIGN

Design

aka System Design, Architecture & Modeling, Technical Design…

- Translate requirements into system architecture
- High-level design (HLD): architecture, modules, data flow
- Low-level design (LLD): detailed algorithms, database schemas

SDLC PHASES – DEVELOPMENT

Development

aka Implementation, Coding, Programming…

- Actual programming of the software
- Translate design and requirements into working code.
- Integrating modules, building features

SDLC PHASES – TESTING

Testing

aka Verification, Validation, Quality Assurance (QA)...

- Ensure the software meets requirements.
- Types of testing: unit, integration, system, acceptance, regression.

SDLC PHASES – DEPLOYMENT

Deployment

aka Release, Delivery…

- Delivering the product to the user environment
- Involves setting up servers, databases, and configurations.

SDLC PHASES – OPERATIONS & MAINTENANCE

Operations & Maintenance

aka Maintenance, Operations, Support…

- Bug fixing, patching, performance tuning
- Performance monitoring
- Feature upgrades, adapting to new requirements

SDLC PHASES – EXTENDED PHASES

Decommission

aka Retirement, Sunsetting, End-of-Life (EOL)…

- When software is obsolete and replaced
- Includes data migration, system shutdown

SDLC PHASES – EXTENDED PHASES

Proof of Concept (PoC)

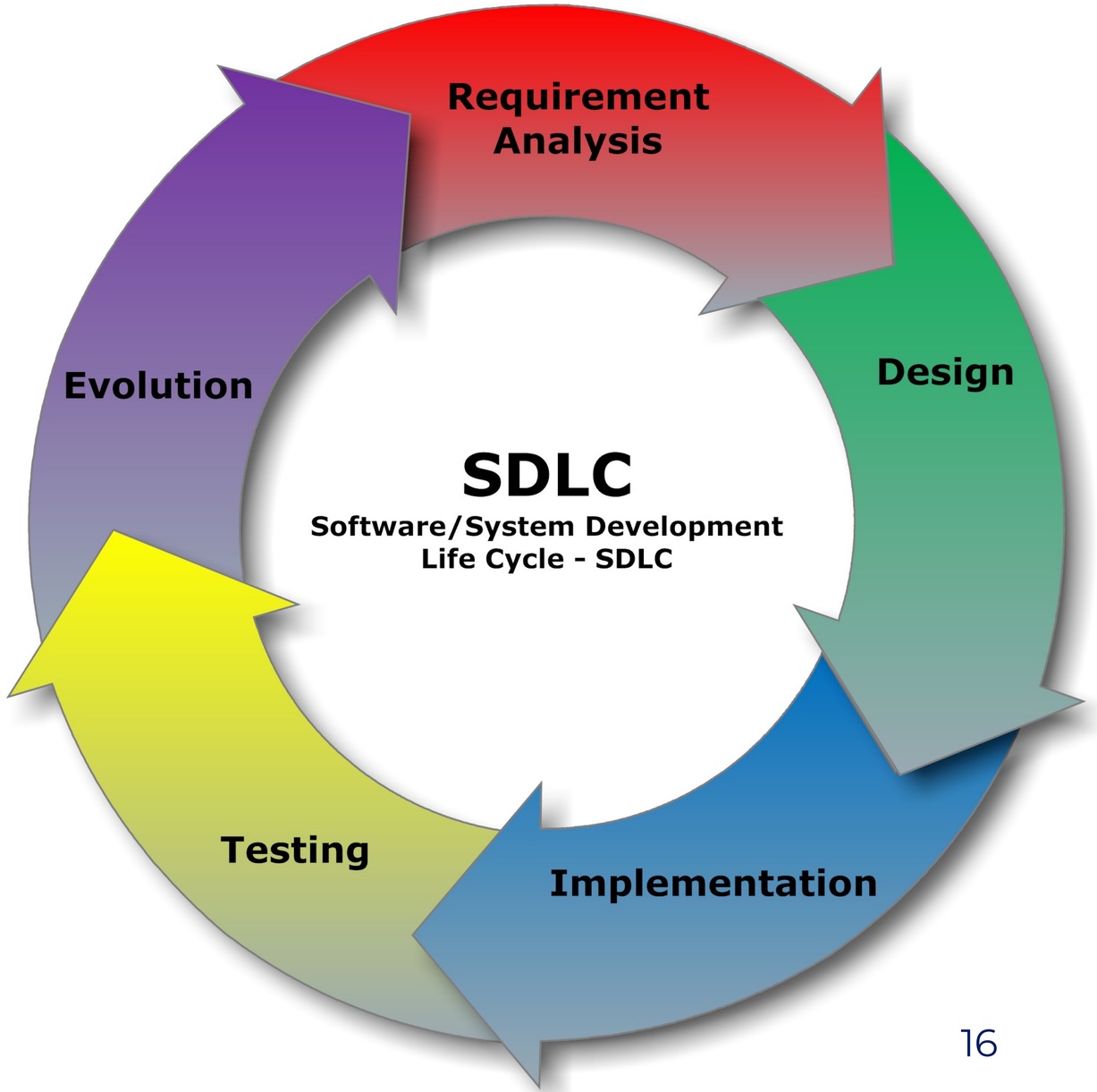
aka Prototyping…

- Building a mock-up or prototype before full development
- Common in Agile, Spiral, and Design Thinking

SDLC PHASES – EXTENDED PHASES

Continuous Improvement

- Agile/DevOps treat this as an ongoing cycle
- Integrates monitoring and user feedback after deployment.



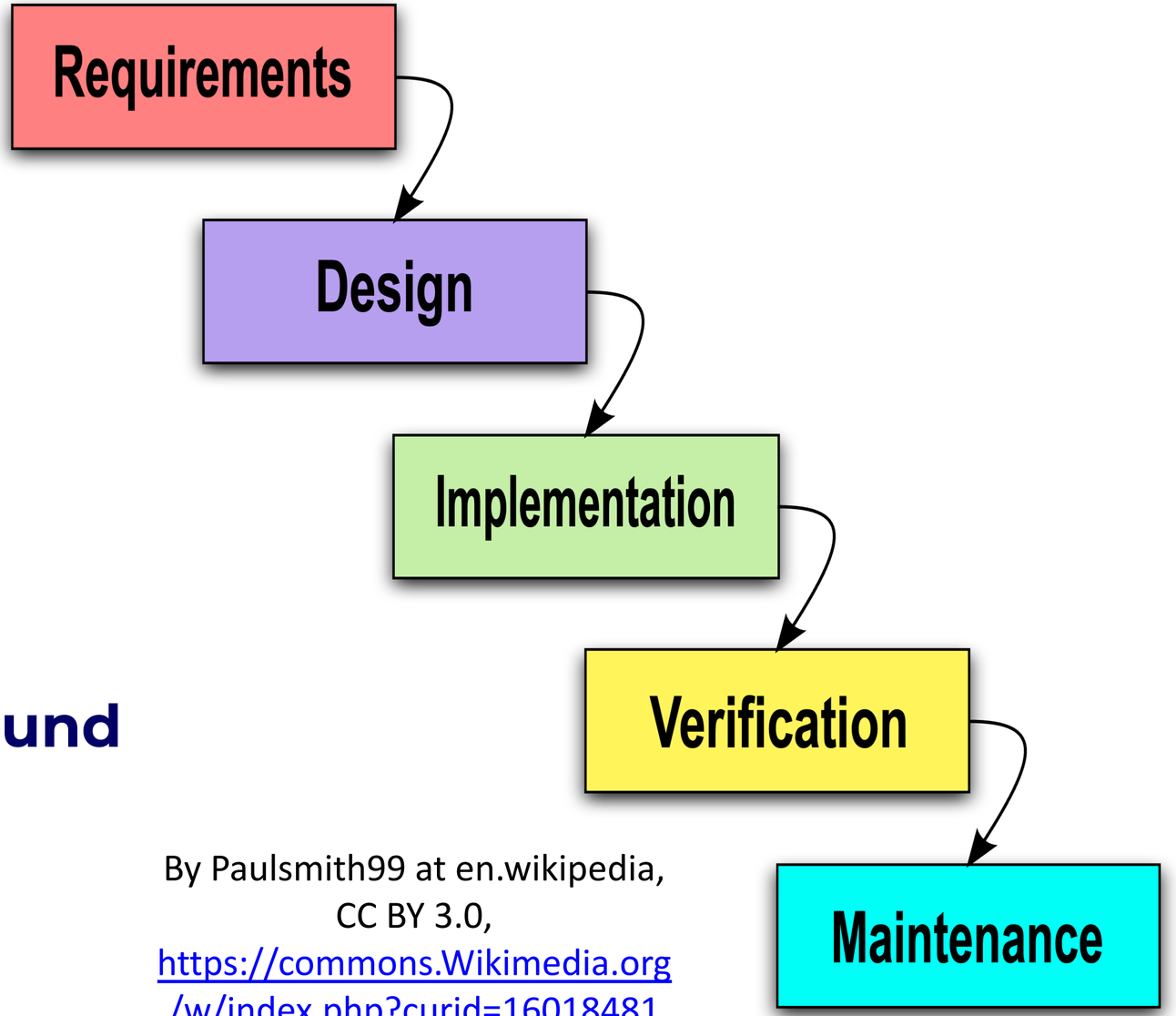
SDLC MODELS

SDLC MODELS

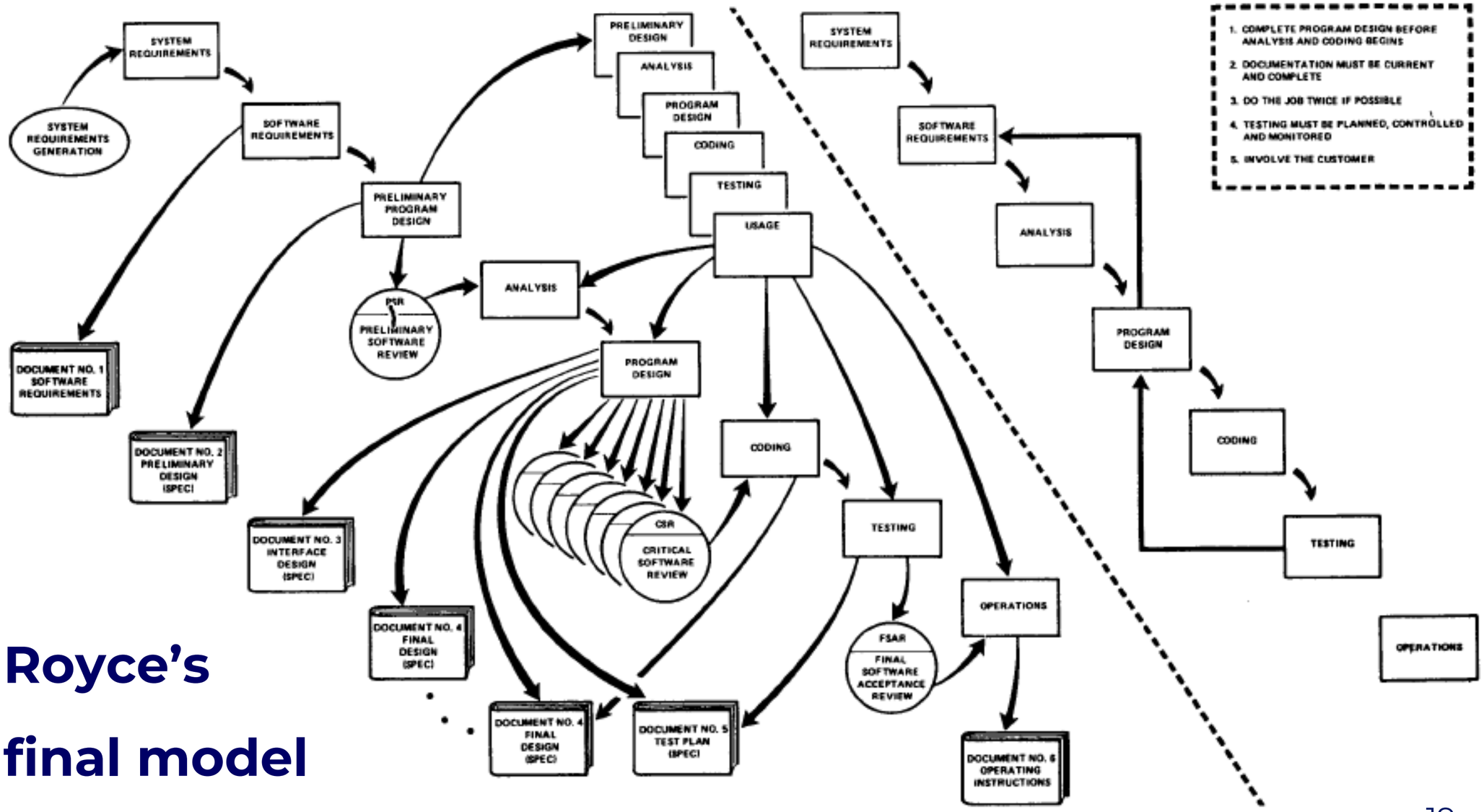
Waterfall

Royce, 1970

- Linear, **sequential** process
- Clear **documentation**
- Rigid structure, inflexible to change
- Weakness: design **issues found later** in testing



SDLC MODELS



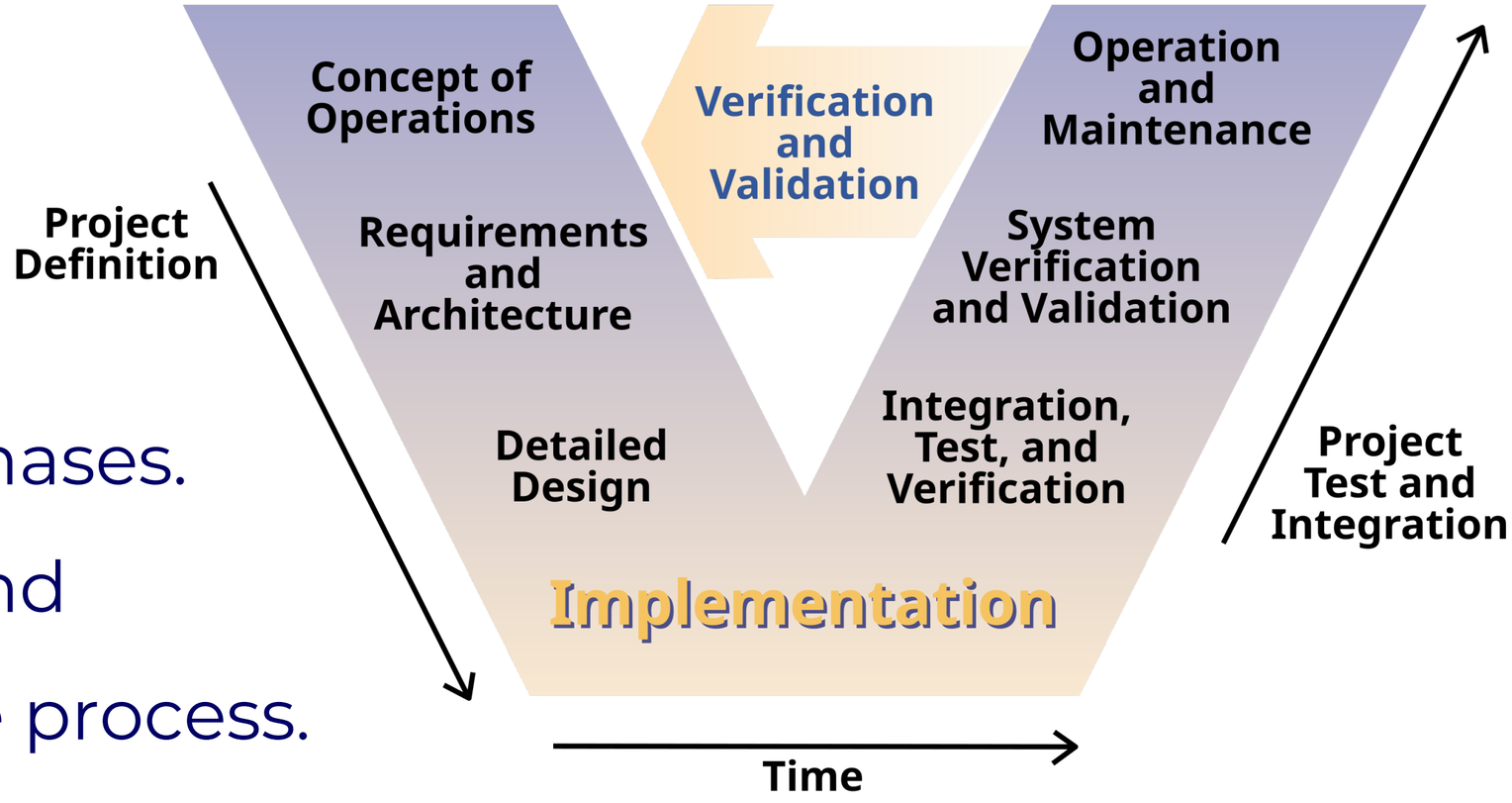
Royce's final model

SDLC MODELS

V-model

Late 80s early 90s

- **Matches** dev **phases** with test phases.
- Focus on **quality** and **testing early** in the process.
- Weakness: still rigid, better for **slow-changing** requirements.

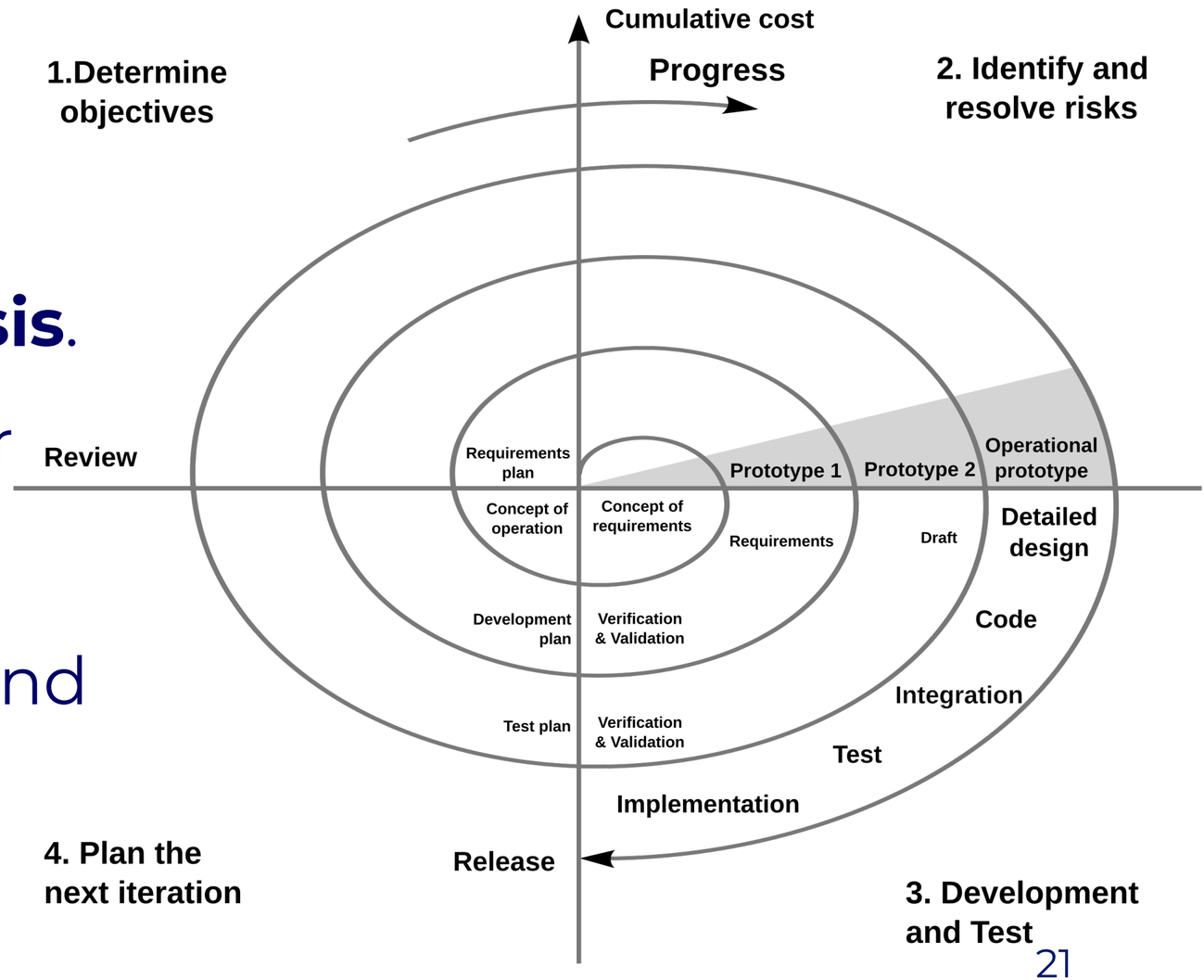


SDLC MODELS

Spiral

Boehm, 1988

- Iterative cycles combining development + **risk analysis**.
- Good for large, complex, or **high-risk** projects.
- Weakness: can be **costly** and require **strong risk management**.



By Connyderivative work: Spiral_model_(Boehm,_1988).png: Marctroyderivative work: Conan (talk) - File:Spiralmodel_nach_Boehm.png, Spiral_model_(Boehm,_1988).png, Public Domain, https://commons.wikimedia.org/w/index.php?curid=9000950

SDLC MODELS

Agile Software Development

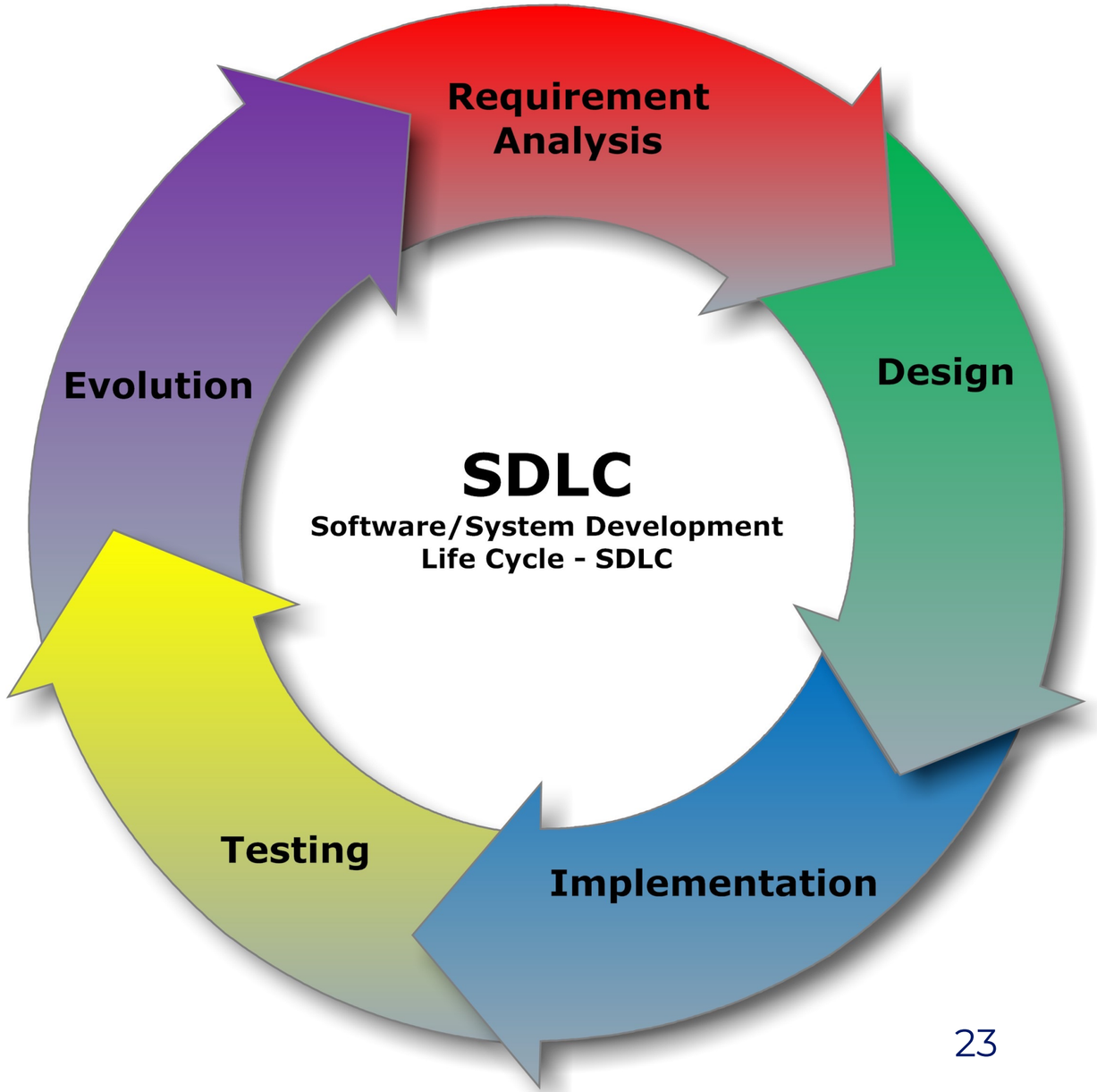
Agile Manifesto, 2001

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

SDLC MODELS

Agile Methodologies

- Iterative, incremental delivery in short cycles.
- Focus on customer collaboration, adaptability, and continuous feedback.
- Integrated with DevOps: automation, CI/CD, and continuous improvement.



ROLES & RESPONSIBILITIES IN SDLC MODELS

SDLC CORE ROLES

- **Business Analyst / Product Owner**
  - Defines requirements & priorities.
- **Software Architect / Designer**
  - Creates system architecture & design.
- **Developer / Programmer**
  - Builds features & integrates components.
- **Tester / QA Engineer**
  - Verifies quality through testing.

SDLC MODERN ROLES IN AGILE & DEVOPS

- **Agile Coach**
  - Facilitates team process & removes blockers.
- **DevOps Engineer**
  - Automates builds, testing, deployment, and monitoring.
- **UX/UI Designer**
  - Ensures usability and customer experience.
- **Site Reliability Engineer (SRE)**
  - Focuses on reliability, scalability, and performance.

INDIVIDUAL ASSIGNMENT 1

UNIVERSITY
SCHOOL OF SCIENCE & TECHNOLOGY

INDIVIDUAL ASSIGNMENT 1

Objective

- Design and implement a **minimal software application** that will later be used for DevOps pipeline design

Due date

- Sunday 2025-10-05 23:59

INDIVIDUAL ASSIGNMENT 1

Application Scope: Choose a simple use case, e.g.:

- To-do list manager
- URL shortener
- Weather dashboard (using an open API)
- Basic blog (CRUD posts)
- Other ideas may be proposed **(discuss with professor for approval)**

ie UNIVERSITY
SCHOOL OF SCIENCE & TECHNOLOGY

INDIVIDUAL ASSIGNMENT 1

Recommended tech stack

- Backend: Node.js, Python (Flask/Django/FastAPI), or similar
- Frontend: Optional, but encouraged (HTML/CSS/JS or minimal React)

Core Features (at least 2 of the following):

- CRUD functionality (Create, Read, Update, Delete)
- Persistent storage (local JSON file, SQLite, or simple DB)
- RESTful API endpoints or minimal UI
- Other ideas may be proposed **(discuss with professor for approval)**

INDIVIDUAL ASSIGNMENT 1

Documentation

- Software Development Life Cycle (SDLC) model chosen and justification
- UML/Class diagram or basic architecture diagram
- README with setup instructions

Version Control

- Repository setup on Git (individual repo)
- At least 3 commits with meaningful content and messages.

INDIVIDUAL ASSIGNMENT 1

Deliverables

- Git **repository** with code
- Short **report** (5-6 pages) including:
  - SDLC explanation & chosen model
  - Architecture overview diagram
  - Reflection: how this app could be scaled or adapted for DevOps practices

INDIVIDUAL ASSIGNMENT 1

Grading criteria

- 25% - First working feature
- 25% - Second working feature
- 10% - Third working feature or exceptional quality in the two required features
- 20% - Documentation & Report
- 20% - Code Quality & Version Control

THANK YOU!

ie UNIVERSITY
SCHOOL OF SCIENCE & TECHNOLOGY