POWERING POTENTIAL

# Software Developer

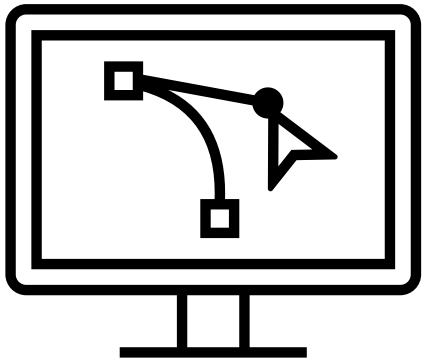
# Agile Software Development

# The SDLC

# What is systems development?

The process of taking a set of business requirements through a series of structured stages and translating them into an operational IT system.

**Let’s see what these structured stages are …**

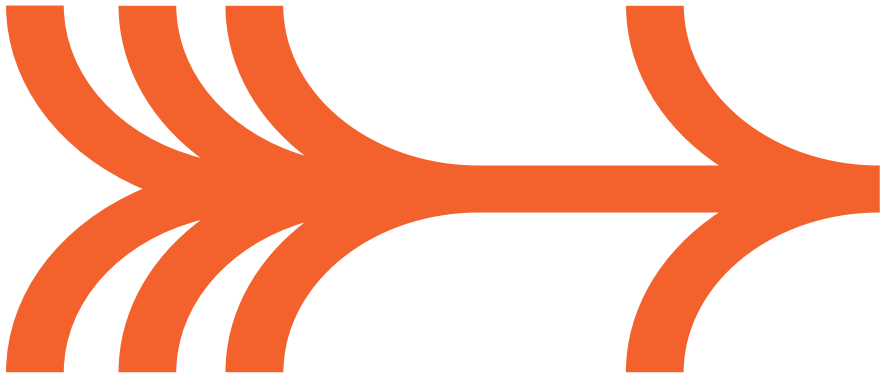


# What is a systems development life cycle?

A system development life cycle (SDLC) is a framework describing a process to:

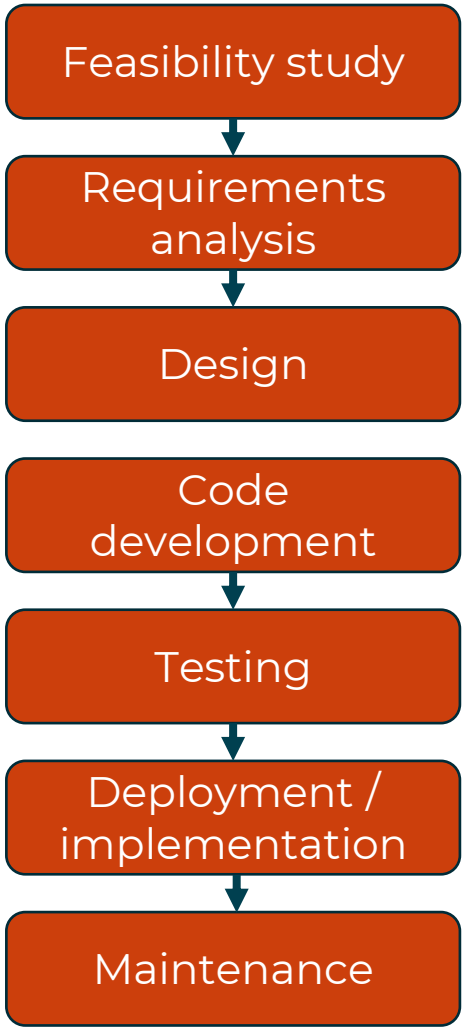
- understand
- plan
- build
- test
- deploy

an information system.



Can apply to:

- Hardware
- Software
- Both



# Life cycle phases

**Initiation**

Begins when a sponsor identifies a need or an opportunity. Concept Proposal is created

**System Concept Development**

Defines the scope or boundary of the concepts. Includes Systems Boundary Document. Cost Benefit Analysis. Risk Management Plan and Feasibility Study.

**Planning**

Develops a Project Management Plan and other planning documents. Provides the basis for acquiring the resources needed to achieve a soulution.

**Requirements Analysis**

Analyses user needs and develops user requirements. Create a detailed Functional Requirements Document.

**Design**

Transforms detailed requirements into complete, detailed Systems Design Document Focuses on how to deliver the required functionality

**Development**

Converts a design into a complete information system Includes acquiring and installing systems environment; creating and testing databases preparing test case procedures; preparing test files, coding, compiling, refining programs; performing test readiness review and procurement activities.

**Integration and Test**

Demonstrates that developed system conforms to requirements as specified in the Functional Requirements Document. Conducted by Quality Assurance staff and users. Produces Test Analysis Reports.

**Implementation**

Includes implementation preparation, implementation of the system into a production environment, and resolution of problems identified in the Integration and Test Phases

**Operations & Maintenance**

Describes tasks to operate and maintain information systems in a production environment. includes Post-Implementation and In-Process Reviews.

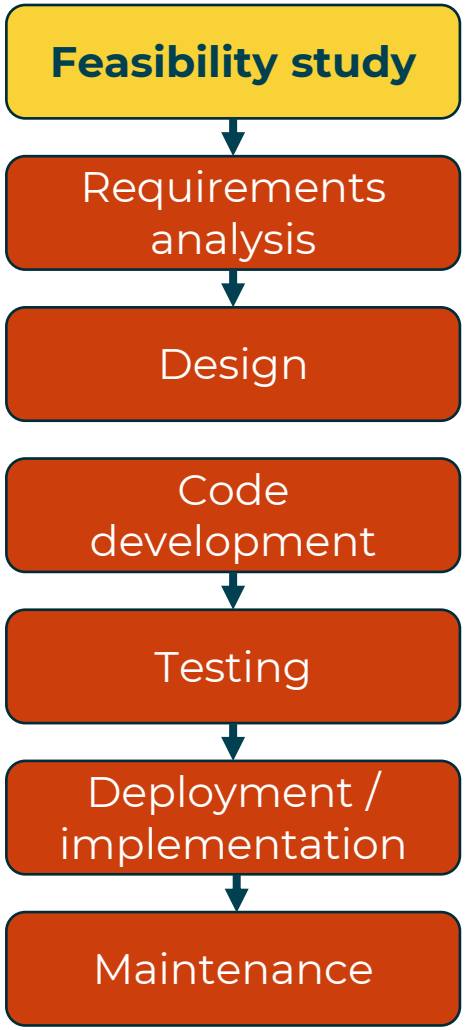
**Disposition**

Describes end-of-system activities, emphasis is given to proper preparation of data.

# **Feasibility (**initiation**) study**

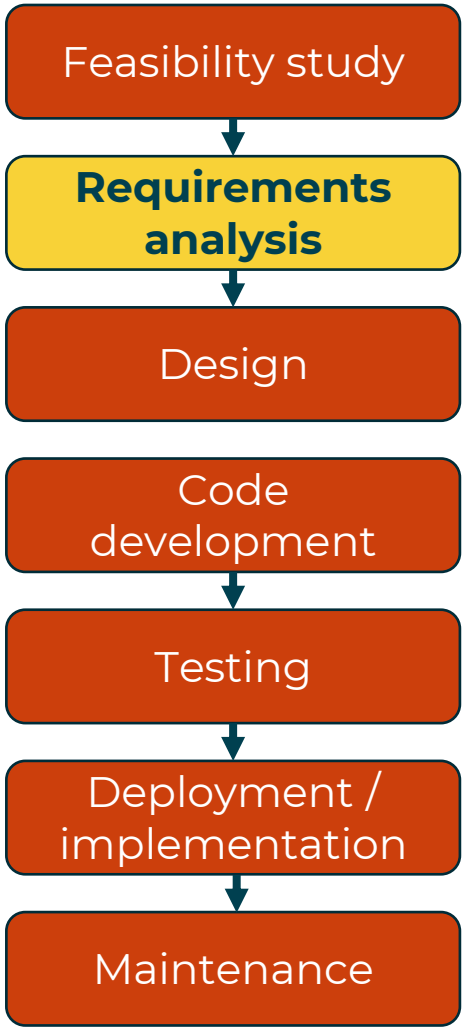
- **Funding** required before starting.
- Evaluate benefits and likelihood of success.
- Resources and costs versus value of completed system:
  - Business case
  - Return on investment (**ROI**)

- Time dependent on size and complexity of project.



# Requirement analysis

- What business needs proposed system to do.
  - Ask if unsure.
- Elicit, analyse, document, and validate requirements.
- Decide how to store, manage, access, and update requirements.
- Key input to design; requirements traced through SDLC.
- Level of detail influenced by SDLC.
- Also known as **requirements engineering**.



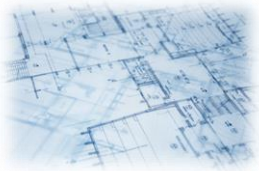
# The importance of good requirements

**What do you think is the consequence of badly/poorly designed system requirements?**

# Characteristics of good requirements

**Categorised** Grouped by type of requirement to make easier to review

**Relevant** Within scope of the project.



**Prioritised**

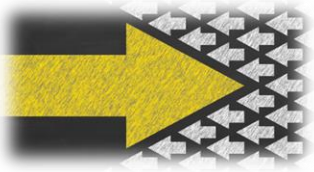
**Achievable** Within the technical or budget constraints

**Unambiguous** Unambiguous and clearly expressed

**Testable**

**Requirement not solution**

# Characteristics of good requirements

| | |
|---|---|
| **Consistent** | No conflict with other requirements  |
| **Owned** | By someone who can make decisions |
| **Unique and atomic** | About one thing |
| **Traceable** | Where requirement came, where to go for more info |
| **Conformant** | Meets standards of how requirements should be documented |

# Requirement Classification

## Functional

Define **what** the solution must do

For example to identify an invoice is overdue and issue alert

## Non-Functional

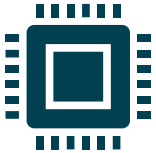
Define **how** the solution must operate (product qualities)

For example the result of a query must be displayed with two seconds of pressing enter

## General

High-level overarching requirements of a business policy nature

For example to comply with data protection



## Technical

High-level overarching requirements of a technical policy nature

For example to operate on a certain platform and use a particular programming language

# Functional Requirements - categories

Is **What** the system is expected to do

- Data entry – Gathering and recording data
- Data maintenance
  Changes to data, including data deletion
- Procedural – Implementation of business rules
- Retrieval – Reporting, responding to enquiries

# Non-functional requirements

Define constraints on **how** the functional requirements will be provided

→ Factors in the cost and are very important to the success of the product

→ Define a characteristic we must implement

# Non-functional requirements

**Availability** - System is available when required to fulfil its purpose

- E.g. Reprint the report on demand
  This is an availability constraint – When this functionality must be available

**Capacity** - Supports transaction and other data volumes as necessary

**Throughput** – Relates to data/transactional volumes within a specific timeframe

- E.g. Respond to a credit enquiry within 5 seconds. This is a performance constraint

**Scalability** - Scale up/down the number of transactions or users

**Reliable** – System is robust and available as defined by SLA.

**Maintainability** - How easy to fix/change – Designed well, flexible, well-documented

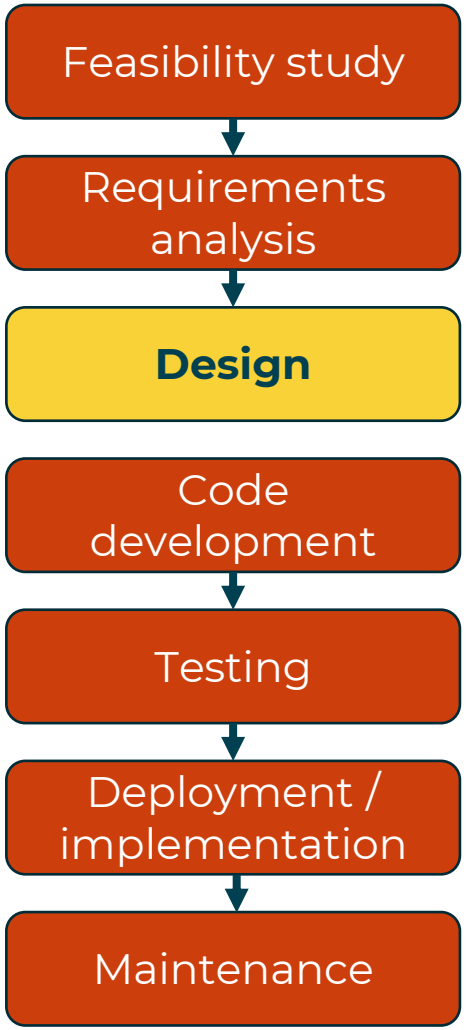
# Non-functional requirements types

- **User-friendly** Functionality delivered in a consistent, intuitive way.
- **Secure** Adequate controls to restrict access to the system and its data.
- **Accessible** Is accessible to all users regardless of disability, language…
- **Responsive** Provides a response to a particular request within a timeframe.
- **Flexible** Adapt to changing business situations.
- **Portable** Can ‘port’ the design to different technology platforms/devices.
- **Configurable** Can change the behaviour by setting configuration options

# Design

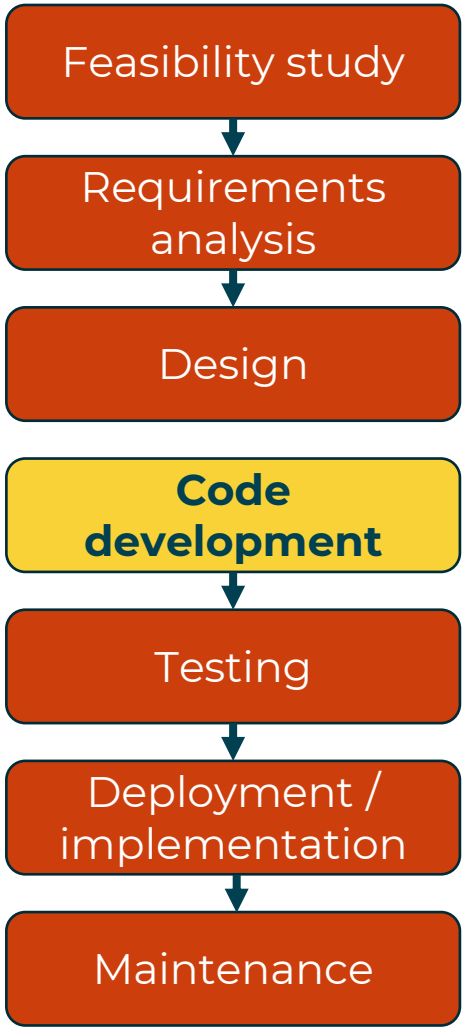
- Evaluate possible solutions that meet requirements.
- Chosen design elaborated to sufficient detail to start development.





# Code development

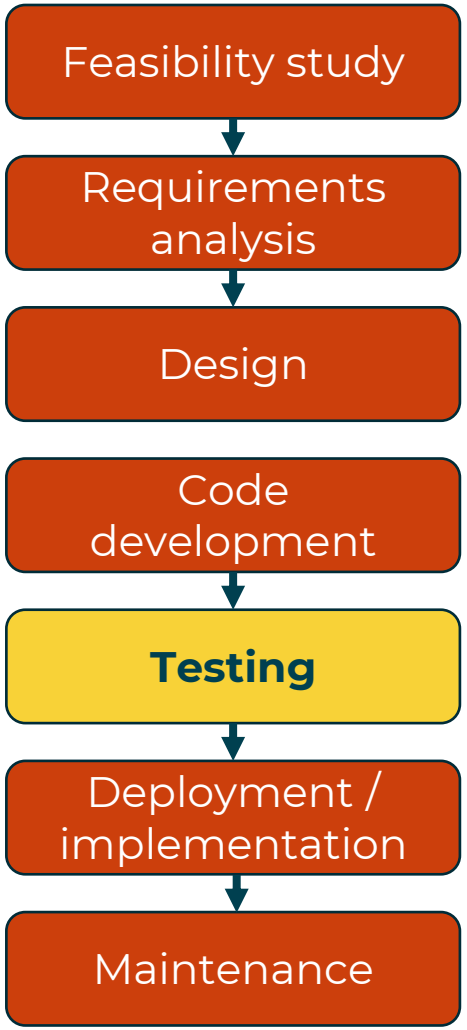
- Hardware and software technical components created, procured or configured
- Follow design to ensure system does what is required
- Also known as **programming** or **build**.



# Testing

- Components produced tested to ensure working properly and does what supposed to do.

- **Different levels of testing:**
  - Unit
  - Integration
  - System
  - User acceptance

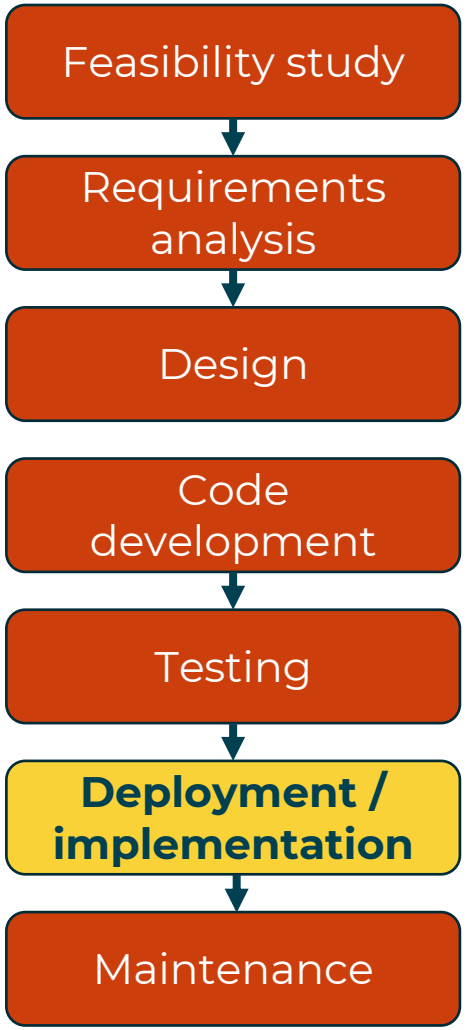


# Deployment / implementation

- Commissioning in a live environment:
  - Also known as **operation** or **production** environment.
  - Developed in test environment to protect live systems.

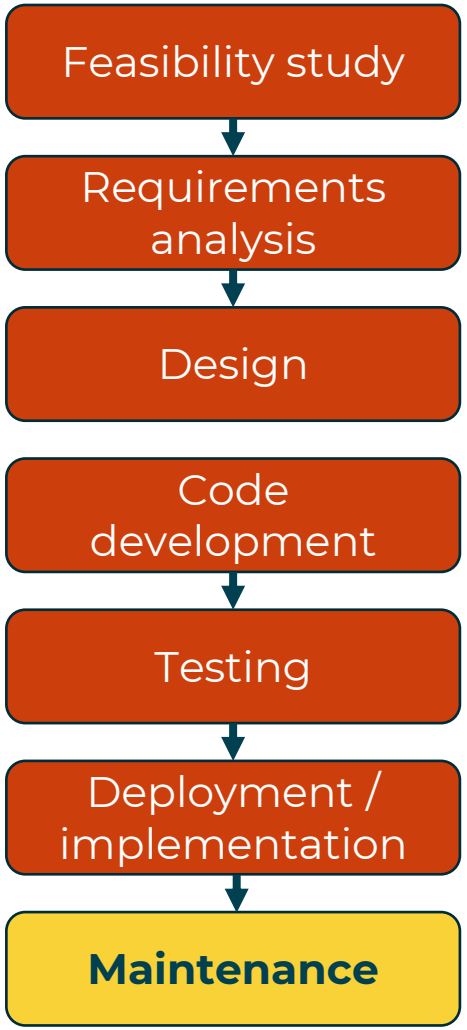
- Needs to be carefully planned, understood, and managed.





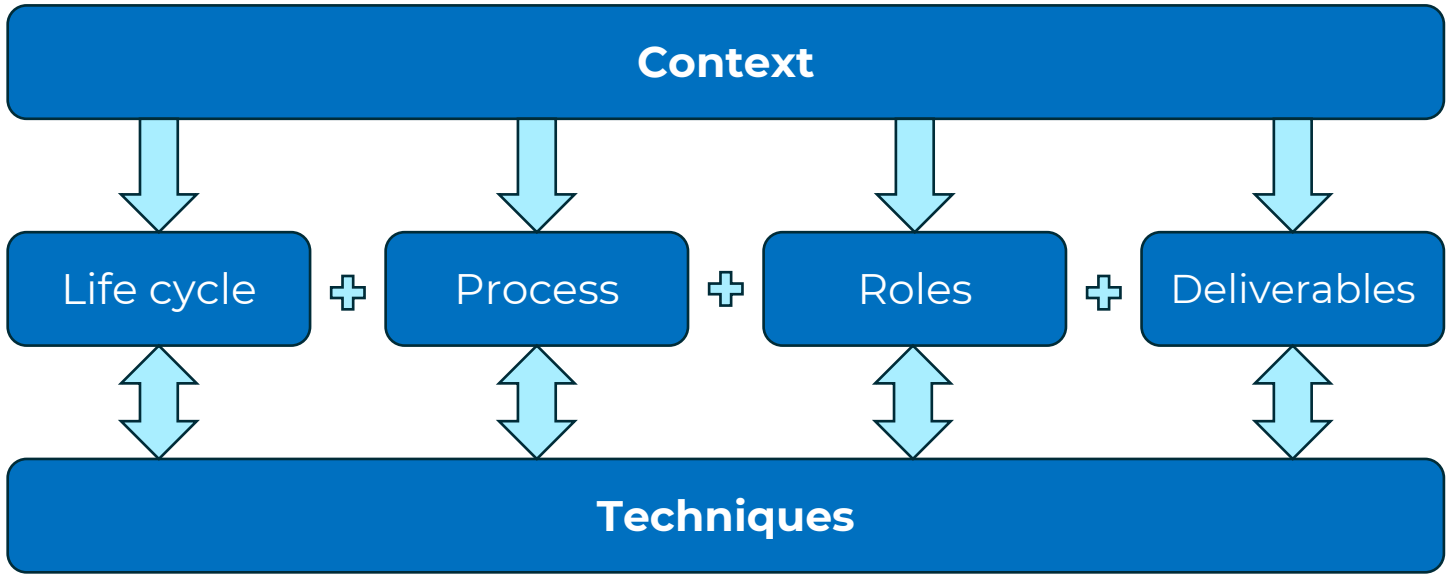
# Maintenance

- More than just correcting faults or keeping the system in good running order:
  - 20% corrective
  - 80% enhancements
- Depends on:
  - development strategy and lifetime of operation
- **Maintenance types:**
  - Corrective
  - Adaptive
  - Perfective
  - Preventative

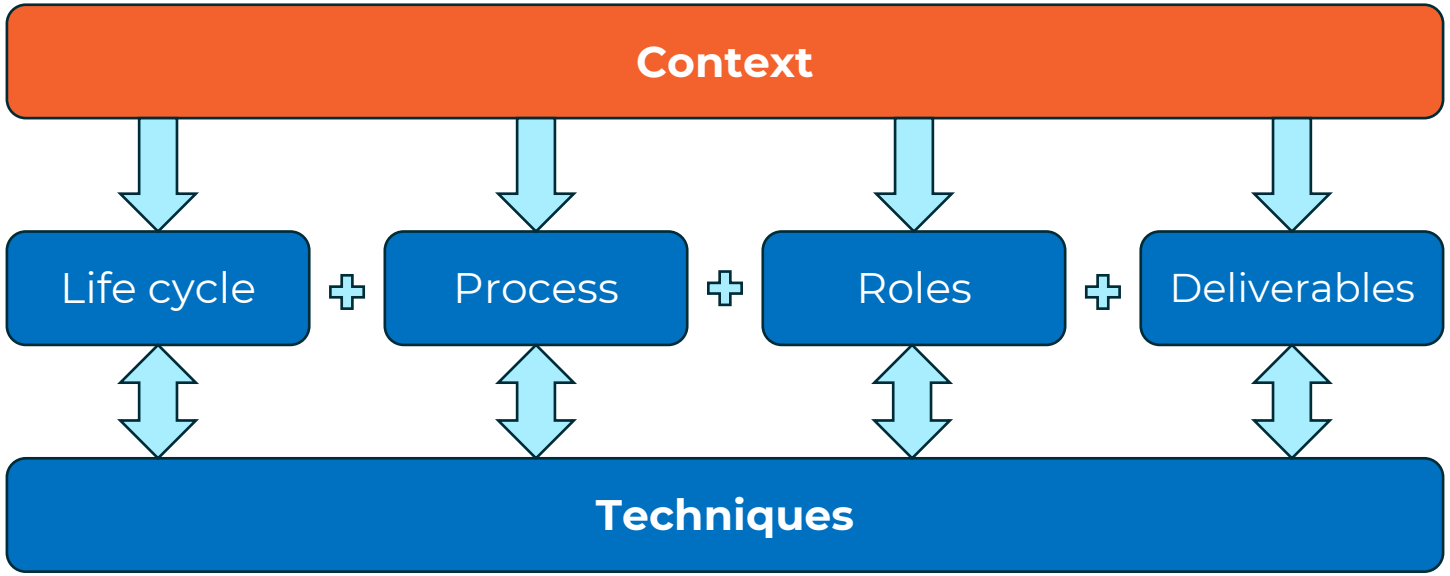


# Elements of the SDLC

In addition to SDLC main stages, you also need to consider other elements:



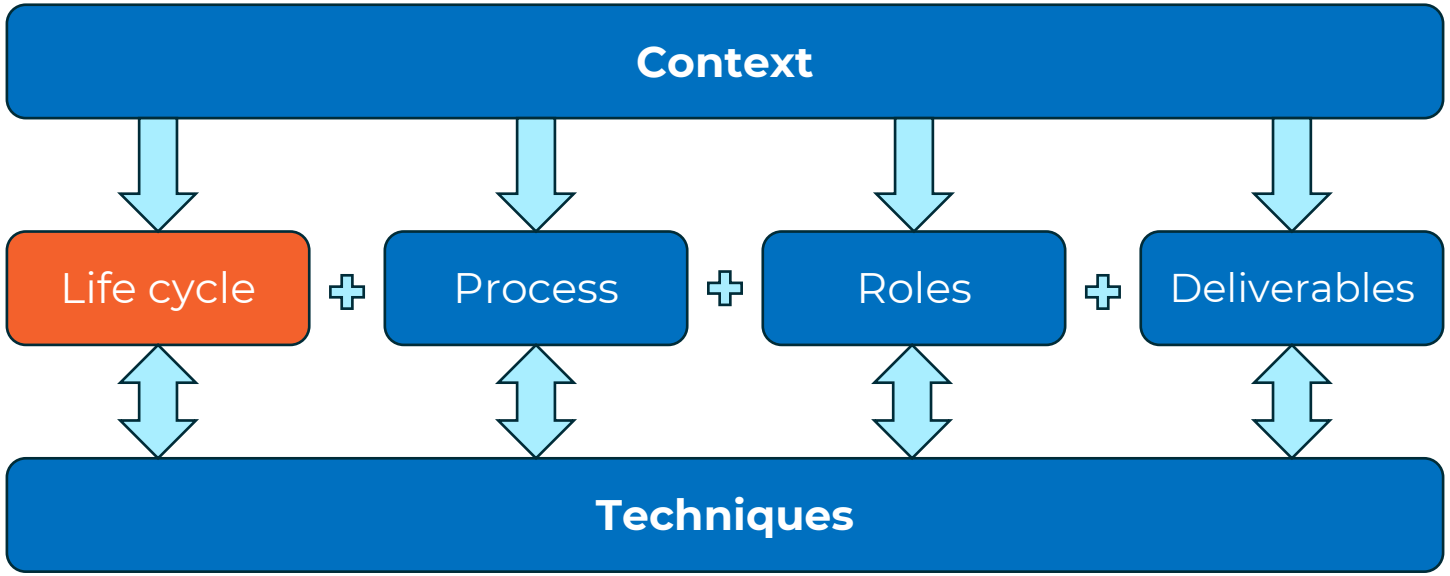
# Context




Needs to be considered before starting:

- **Release plan**: single or multiple
- Level of **skills** and expertise: preferred delivery style
- **Location of teams** and stakeholders: single site or dispersed
- **Requirements**: audit, quality, regulatory, complexity and stability
- **Technology** to be used: tried and tested or new

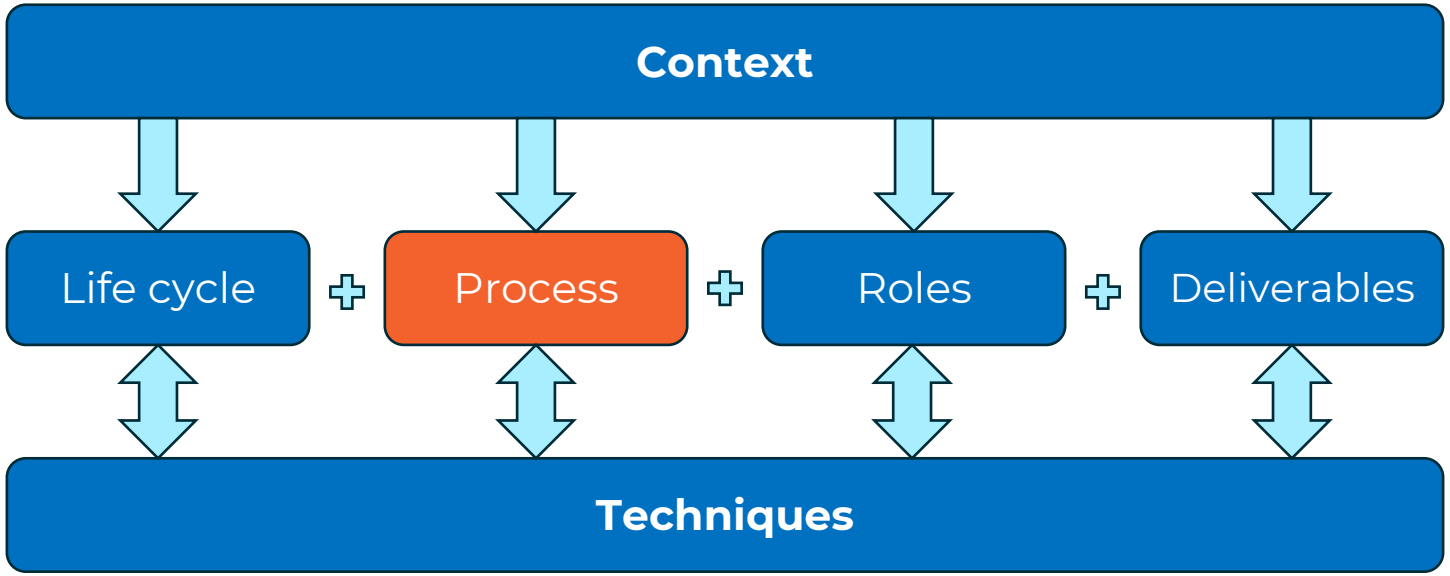
# Life cycle




- Describes stages to follow: plan, design, build, test, and deliver.

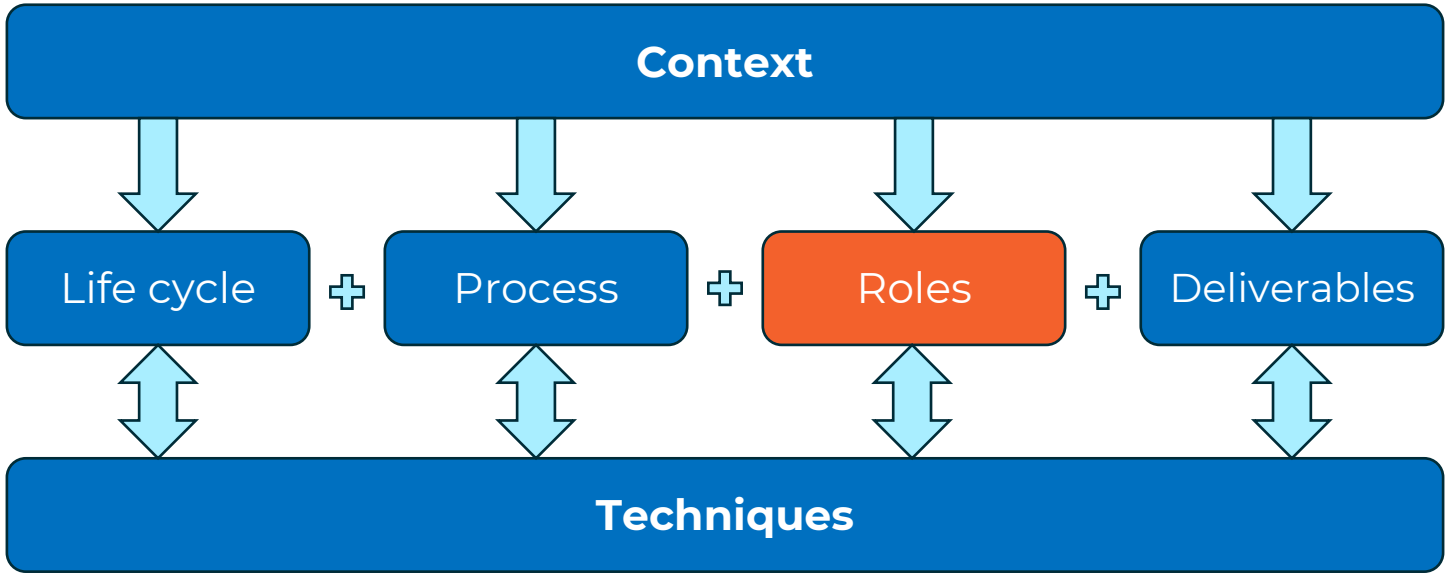
  - **Linear** sequential or step by step
  - **Evolutionary** iterative, evolving through progressive versions

# Process



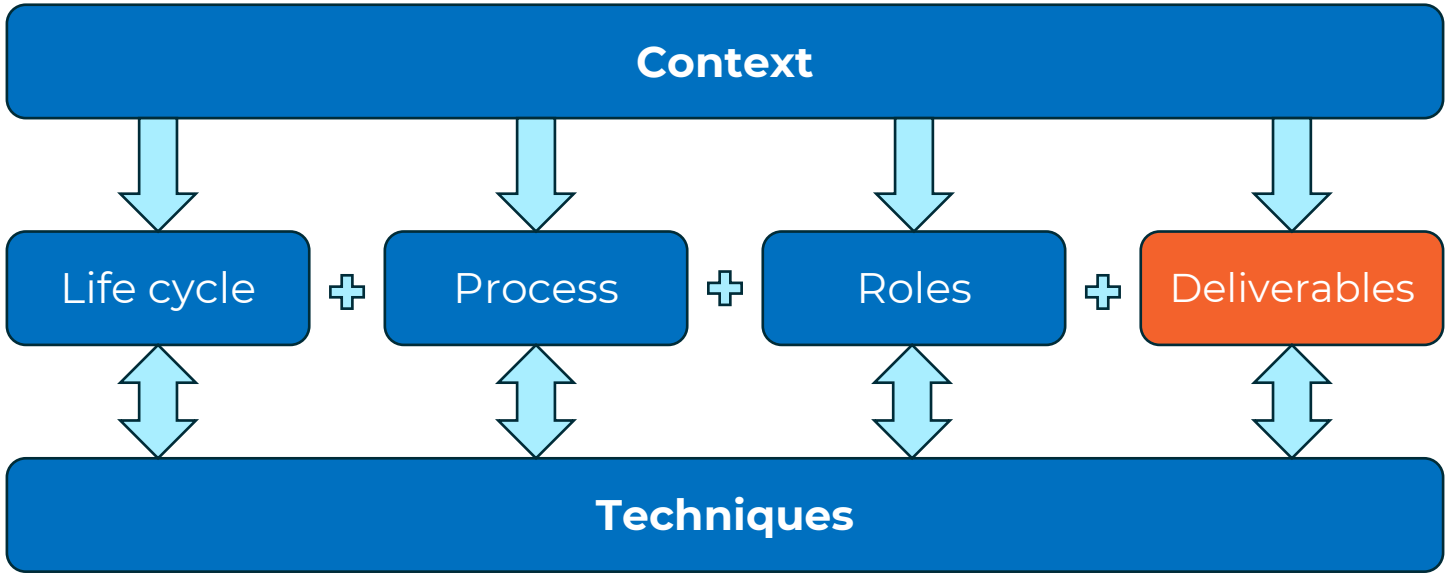

- Set of steps and actions followed to deliver outcomes within stages.
- Methods and standards support the framework within SDLC:
  - Include defined roles, deliverables, techniques, models
  - Prescriptive: specific SDLCs
  - Agnostic: various SDLCs
  - Dependent on context and lifecycle chosen

# Roles



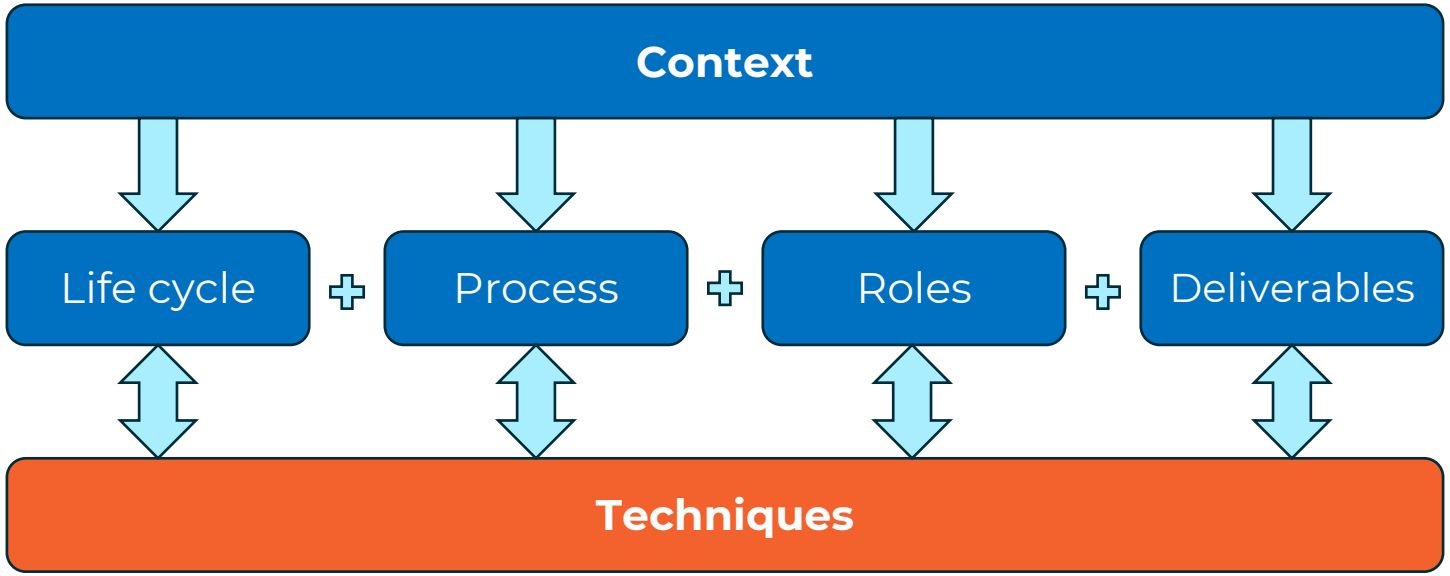

- People that carry out tasks within various SDLCs:
  - Specific to particular SDLC or generic: various titles
  - Include business, project, technical, and support roles

# Deliverables




- **Documents, models, designs, hardware and software required**:
  - Vary for different SDLCs and stages
- Detailed understanding of requirements – what, how, level of quality, when?
- Examples:
  - Requirement documents, models, system / software components
  - Test / implementation plans and scripts

# Techniques




- **Numerous**: choice dependent on team, organisation, SDLC. For example:
  - Military: Ministry of Defence Architecture Framework (MoDAF)
  - Commercial payment milestones and reviews: **Waterfall**
  - Rapid prototypes: **Agile**

- Many SDLCs: strengths / weaknesses, suit particular projects

# WHAT ARE DELIVERABLES?

Outputs / documents from each stage of the SDLC:

- Review, sign-off, and become inputs for following stage.
- Type, format, and content dependent on methodology adopted.
- Linear vs. evolutionary.
- Also known as **system documentation**.

# DELIVERABLES

**Feasibility study (initiation):**

- Project brief / charter / initiation / scope
- Terms of reference
- Business case: cost / benefit analysis
- System scope / requirements
- High level plan (Gantt chart)

**Requirements analysis**:

- Requirements report
- Threats and risks report
- Recommendations

**Design:**

- Detailed technical specifications
- Logical and physical designs

# DELIVERABLES

**Code development:**

- Developed system: software and hardware if appropriate
- Code / screens / reports
- Database
- Interfaces

**Testing:**

- Tested system
- Test plans / cases / specifications

# DELIVERABLES

**Deployment / implementation**

- Implemented system
- User documentation
- Training procedures
- Support capabilities

**Maintenance**:

- How to guides
- Completed tickets

# General considerations



**Documentation meets organisational standards:**

- Documentation content and layout
- Naming conventions
- Diagrams
- Quality control requirements

**Organisational processes and requirements:**

- Project management
- Governance

# review

POWERING POTENTIAL

# QUESTIONS AND FEEDBACK