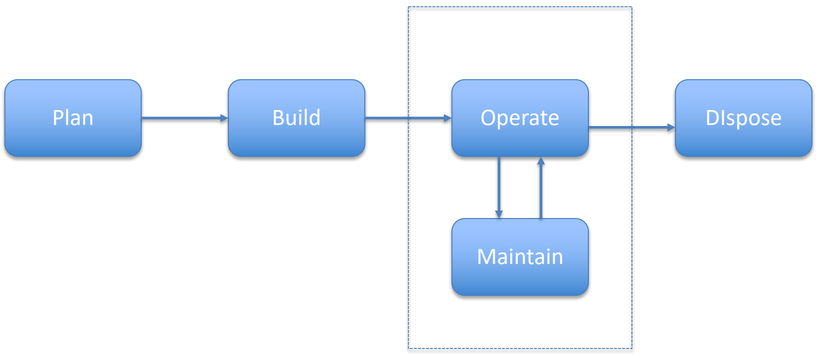
IF3141 SISTEM INFORMASI
System Development LifeCycle

Semester II 2024/2025

1

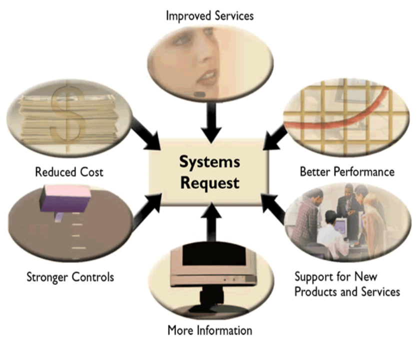
System LifeCycle



2

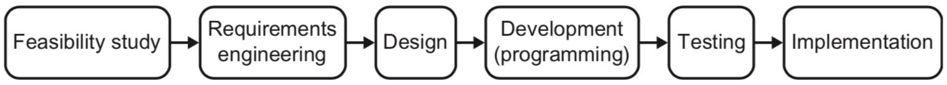
Reasons for Systems Projects

- Main Reasons for Systems Projects
  - Systems request
  - Improved service
  - Support for new products and services
  - Better performance
  - More information



3

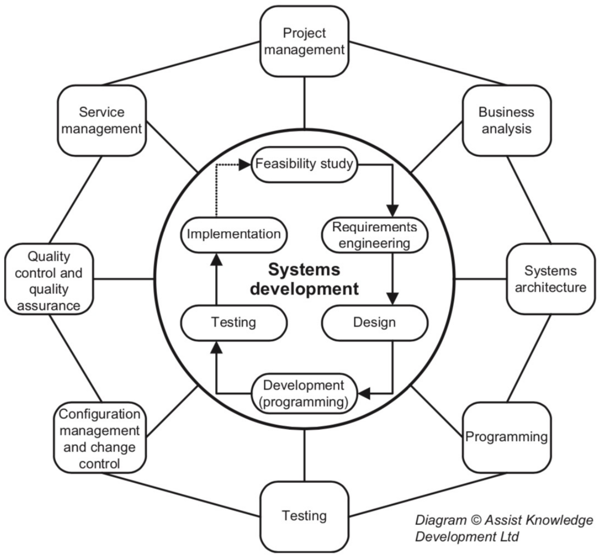
System Development LifeCycle (SDLC)



A structured methodology used to develop an information system, covering all its components:
- Infrastructure (Hardware & Network)
- Applications & Software
- Business Processes & Workflows
- Data & Information Management
- Organizational Structure & Roles
- Brainware (Users, IT Staff, and Training)**

4

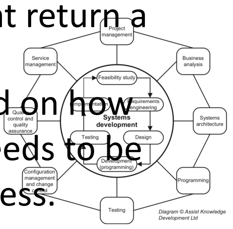
System Development LifeCycle (SDLC)



5

Feasibility Study

- Involves investigation and research in order to evaluate the project's potential for success to support decision-making.
  - (Resources and costs) vs Value that will be attained
  - Return on investment (ROI): ratio between net income and investment
  - Only those projects and systems that return a reasonable ROI will be supported.
- The time spent in this stage will depend on how big and complex is the problem that needs to be solved within the organisation or business.

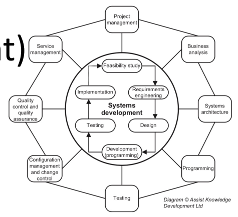


6

Feasibility Study

Includes

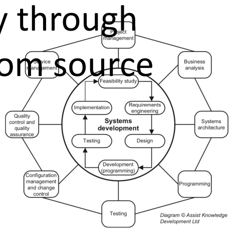
- Technical Feasibility (Can we built it?)
- Economic Feasibility
  - Total cost of ownership (TCO)
  - Tangible benefits
  - Intangible benefits
- Operational Feasibility (Will users accept it?)
- Regulatory Feasibility
- Schedule Feasibility (Time constraint)



7

Requirements Engineering

- To secure understanding of what the business needs the proposed system to do.
  - Eliciting, Analysing, Documenting and Validating user requirements
  - The requirements will be stored and managed throughout the system development lifecycle
  - The requirements become the input to system design and can be traced all the way through the system development lifecycle from source to implementation.

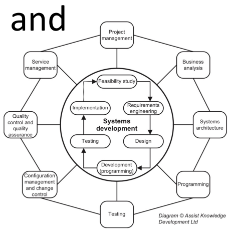


8

Design

- Possible solutions that meet the requirements are considered and weighed against each other.
- The chosen design is then elaborated in sufficient detail to allow the developers to implement it.

- Activities
  - Develops System Architecture & UX/UI Design
  - Defines Applications & Data Models, and
  - Designs Processes.

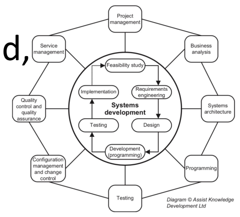


9

Development

- Technical components (both hardware and software) are created, procured or configured.
- The developers follow the design to ensure that the system does what is needed for the business or customer.

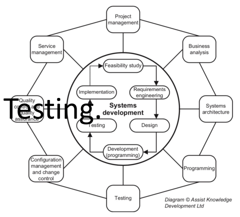
- Activities
  - Implements Applications & Coding,
  - Sets up Infrastructure (Servers, Cloud, Network), and
  - Configures Data Storage & Flow.



10

Testing

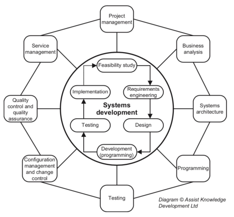
- The components produced during development are tested to make sure they are working properly and that the system does what it is supposed to do.
- Different levels of testing
  - unit testing,
  - component testing,
  - system testing and
  - acceptance testing
- Activities
  - Validates Applications & Software,
  - Ensures Processes work correctly, and
  - Conducts Security Evaluations & Software Testing.



11

Implementation

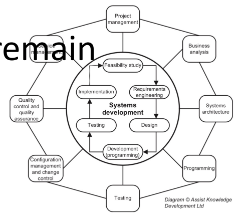
- Before a system is ready to use it must be commissioned into the 'live' or 'operational' environment.
- Moving from a reference or test environment(to protect other systems from faults within the new system) to the live environment
- Activities
  - Sets up Infrastructure,
  - Conducts User Training & Change Management, and
  - Implements Data Migration.



12

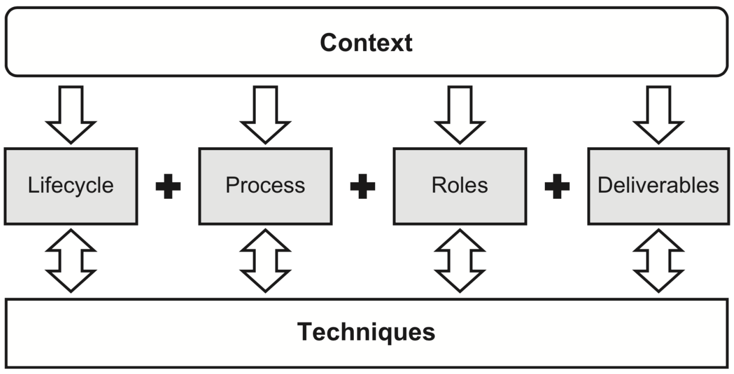
Maintenance

- After a system operates, monitoring its performances is conducted continually.
- Small improvement often needs to solve operational problems.
- Activities
  - Manages System Upgrades & Performance Monitoring,
  - Updates Applications, and
  - Ensures Infrastructure & Processes remain efficient.

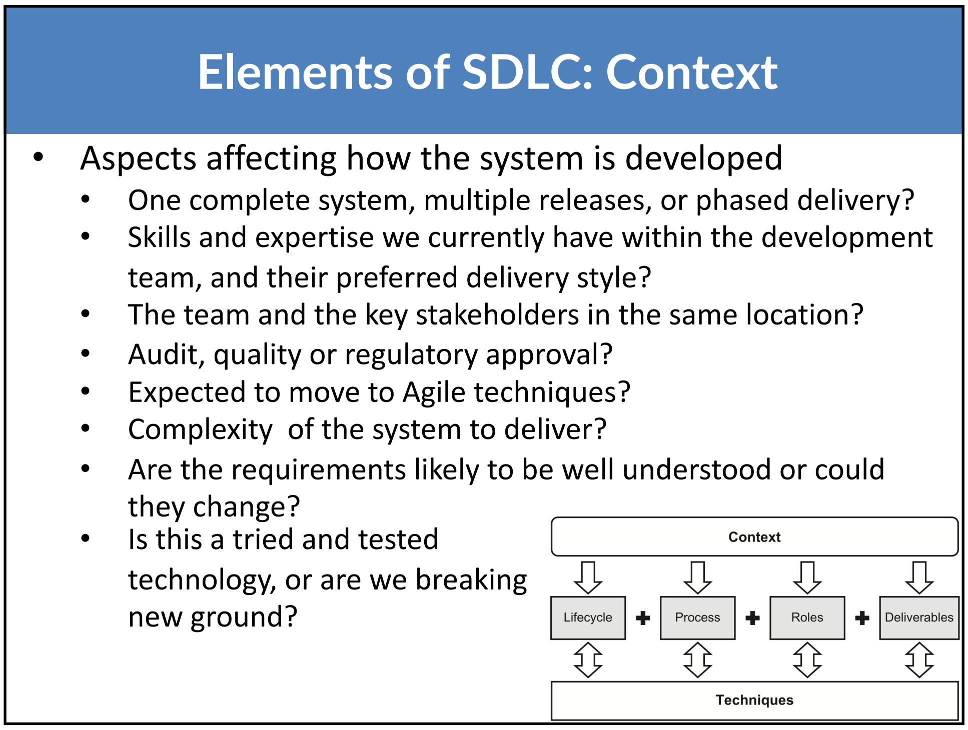


13

Elements of SDLC

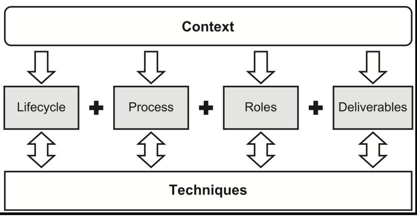


14

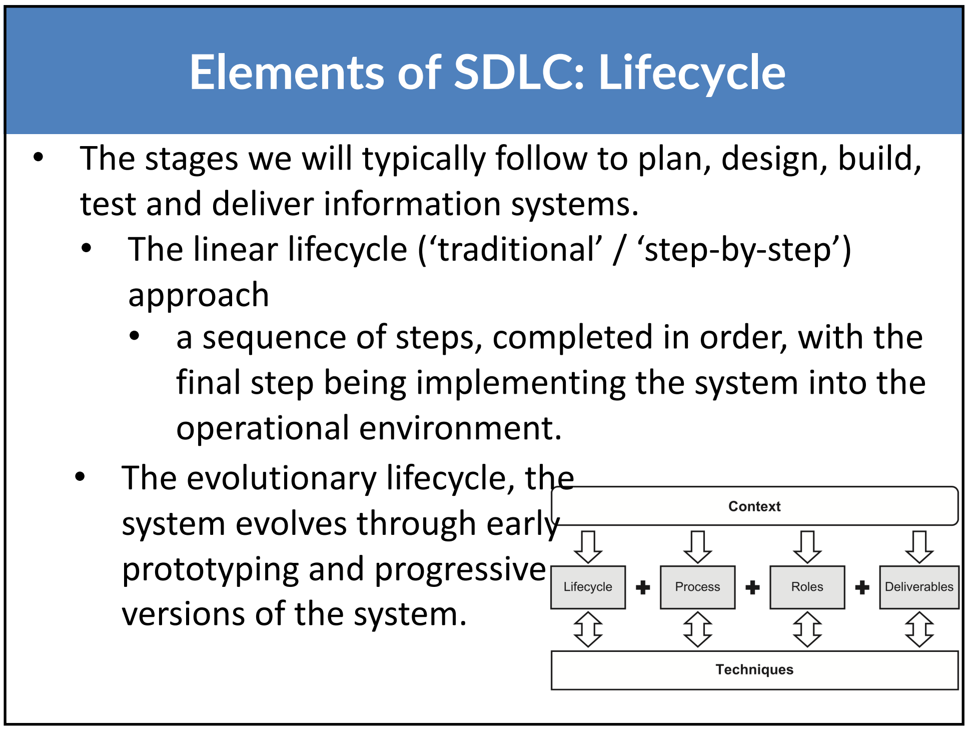


Elements of SDLC: Context

- Aspects affecting how the system is developed
  - One complete system, multiple releases, or phased delivery?
  - Skills and expertise we currently have within the development team, and their preferred delivery style?
  - The team and the key stakeholders in the same location?
  - Audit, quality or regulatory approval?
  - Expected to move to Agile techniques?
  - Complexity of the system to deliver?
  - Are the requirements likely to be well understood or could they change?
  - Is this a tried and tested technology, or are we breaking new ground?

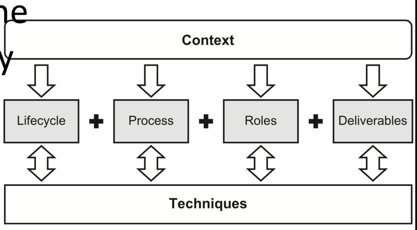


15



Elements of SDLC: Lifecycle

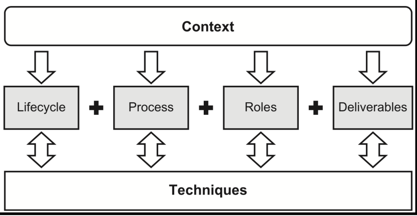
- The stages we will typically follow to plan, design, build, test and deliver information systems.
  - The linear lifecycle (‘traditional’ / ‘step-by-step’) approach
    - a sequence of steps, completed in order, with the final step being implementing the system into the operational environment.
  - The evolutionary lifecycle, the system evolves through early prototyping and progressive versions of the system.



16

Elements of SDLC: Process

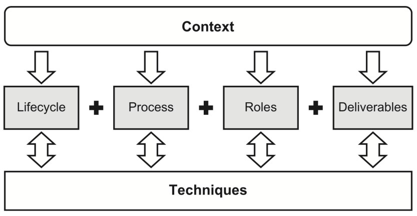
- A set of actions or steps that, if followed, will deliver a particular outcome.
- System development processes to use should be in line with the 'context' and 'lifecycle' that has been chosen or is adopted within the organisation.



17

Elements of SDLC: Roles

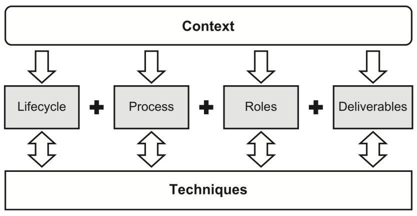
- Business roles;
  - sponsor or senior responsible owner
  - business analyst
  - domain expert
  - end users
- Project roles
  - project manager
  - team leader
  - work package manager
- Technical roles
  - technical architect
  - solution developer
  - solution tester
- Implementation and support roles
  - release manager
  - database administrator
  - system administrator



18

Elements of SDLC: Deliverables

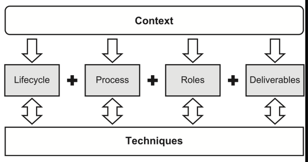
- The deliverables are **a way of detailing what is understood about the system** in terms of **what** it needs to do, **how** it needs to do it, **how** well it does it and **how** it gets delivered
- Includes
  - requirement documents
  - models such as:
    - class or entity relationship models
    - use case models
    - process models
    - state transition diagrams
    - sequence diagrams
    - component diagrams
  - test plans
  - test scripts
  - implementation plans
  - system components and working software.



19

Elements of SDLC: Techniques

- They vary depending on team and organisation preferences, life cycle choice, and system development approach selected
- Such as
  - MoDAF (Ministry of Defence Architecture Framework – UK) or DoDAF (Department of Defense Architecture Framework – USA)
  - Teams used to rapid prototyping or Test Driven Development may struggle with a linear approach and work better in an Agile model.
  - Commercial projects with payment milestones that require formal sign-off and structured review dates may benefit from the rigour provided by a linear 'Waterfall' lifecycle.
  - Detailed UML designs may not be appropriate where team members and project sponsors do not understand them.
  - Another factor to consider when selecting the techniques is whether there are any formal standards or procedures that need to be adopted within the organisation.



20

Types of SDLC

- Linear
  - A sequence of tasks that are completed in order, only moving to the next step once the previous step is complete.
  - Basic types
    - Waterfall
    - 'V' model
    - Incremental
- Evolutionary approach
  - The solution evolves through **progressive versions**, each more complete than the last, and often uses a prototyping approach to development.
  - Basic types:
    - Iterative
    - Spiral

21

Types of SDLC: Linear

+ Breaks down the problem into distinct clear stages
+ Everything is agreed in advance of being used
+ Helps provide structure to complex systems
+ Suits a detailed design decomposition and specification approach
+ It makes easier to control cost and scope creep
+ Simple and intuitive for smaller problems

- Depends greatly on each stage being done properly
- For complex problems**, long timescales** required
- Doesn't cope well with **changing requirements**;
- Customer or **business value** is not available until the end
- If the project is stopped early there is little of business value to show for the cost.

22

Types of SDLC: Evolutionary

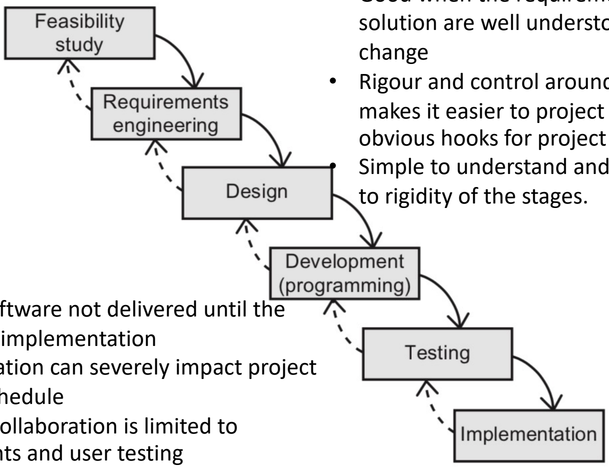
+ Early delivery of value to the customer
+ Copes well with complex requirements – fast changing, uncertain or complicated
+ Encourages collaboration with the users throughout, so customer buy-in is higher
+ Allows 'just enough' to be done knowing that it can be refined later on.

- Can be hard to project manage due to multiple iterations; especially hard with multiple iteration teams and complex products
- Without careful management, the evolving requirements can result in scope creep
- Overall costs can be higher due to the additional inte-gration and test across multiple teams and iterations;
- Easy to over-promise on early functionality.

23

Types of SDLC: Linear

- **Waterfall**



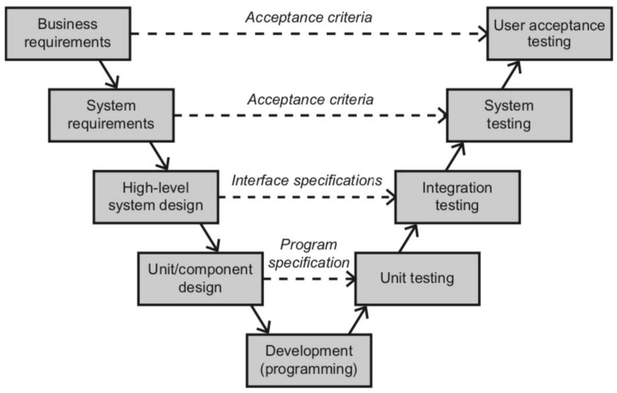
- Good when the requirements and likely solution are well understood and unlikely to change
- Rigour and control around each stage makes it easier to project manage, with obvious hooks for project milestones
- Simple to understand and easy to use due to rigidity of the stages.

- Working software not delivered until the final stage, implementation
- Poor estimation can severely impact project cost and schedule
- Customer collaboration is limited to requirements and user testing
- On long projects this can lead to a lack of customer buy-in, particularly where stakeholders change.

24

Types of SDLC: Linear

- **‘V’ Modell**



25

Types of SDLC: Linear

- **‘V’ Modell**

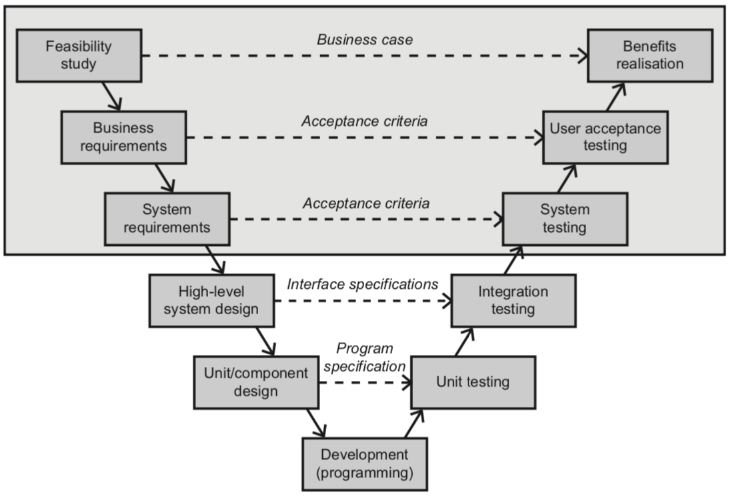
+ Additional focus on testing leads to a high quality product
+ Ideal for systems with complex or high quality attributes (such as safety critical, challenging to integrate or very large)
+ Easy to plug in highly complex supply chains and distributed teams due to clear interfaces and boundaries
+ Rigour and control around each stage makes it easier to project manage
+ Easy to scale.

- Heavy testing and integration cost that can be unnecessarily expensive for simpler systems
- Cost and impact of change increases significantly as further progress down and up the ‘V’ is made
- Early stages take longer to complete as there are test artefacts to complete
- Working system not delivered until the final stage, which could be a long time after the customer was last involved;
- Development teams can often omit to put enough effort into planning the right-hand side as they progress down the left; This significantly increases risk, particularly in the integration of complex systems.

26

Types of SDLC: Linear

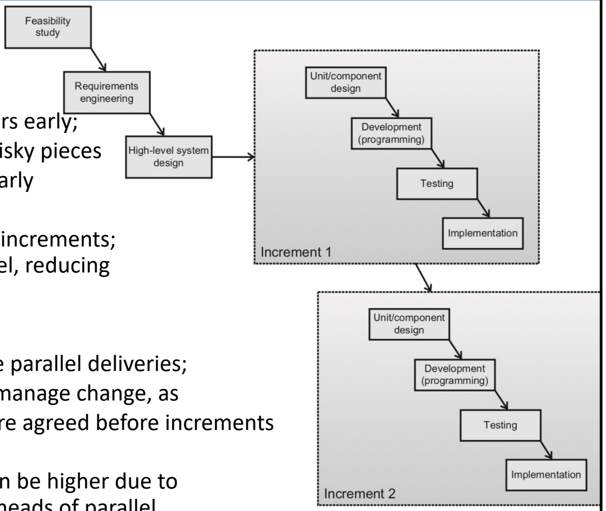
- **Extended 'V' Modell**



27

Types of SDLC: Linear

- **Incremental**



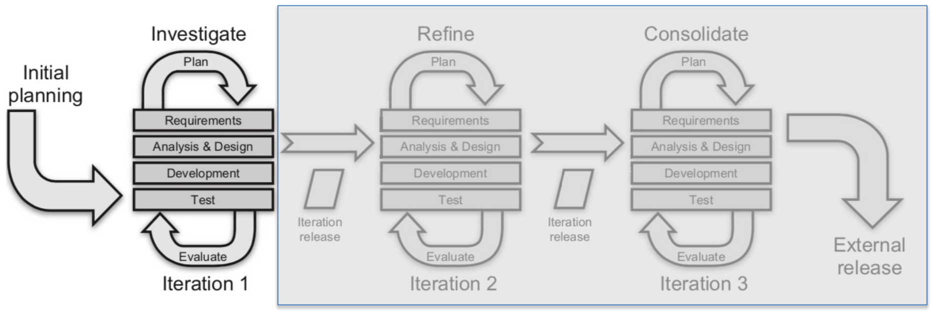
+ Delivers working software to users early;
+ Easier to manage risks because risky pieces can be identified and handled in early increments;
+ Easier to test and debug smaller increments;
+ Increments can be built in parallel, reducing overall time to market.

- Hard to manage parallel deliveries;
- still difficult to manage change, as requirements are agreed before increments decided;
- Overall costs can be higher due to additional overheads of parallel implementation and additional 'regression testing'.

28

Types of SDLC: Evolutionary

- **Iterative**

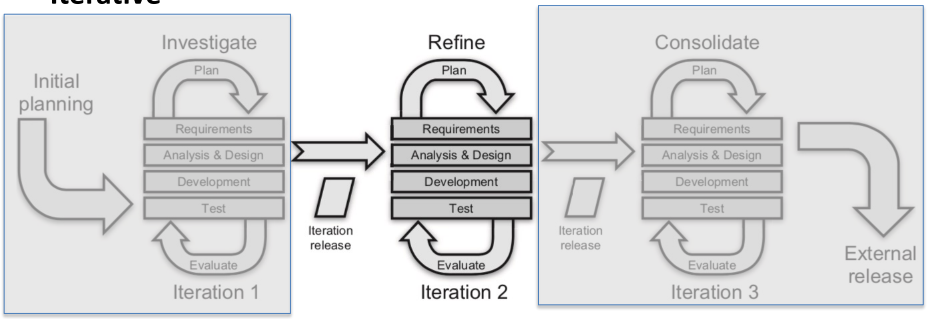


- Investigate
  - to investigate things such as risks around the technology or risks around user functionality to understand more about what is required to deliver the external iteration or first release.

29

Types of SDLC: Evolutionary

- **Iterative**

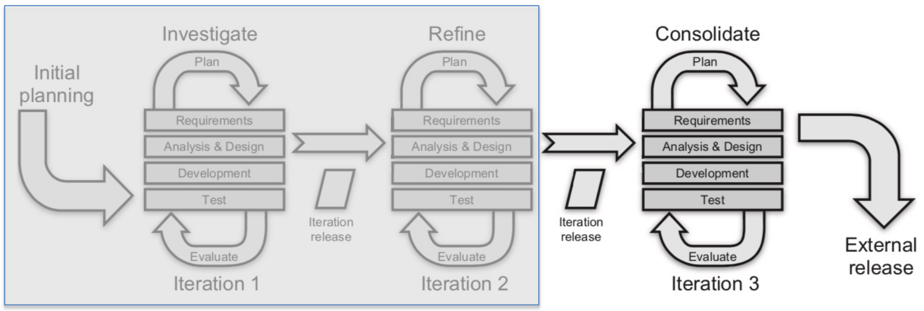


- Refine
  - The requirements are fully detailed and designed and the system is developed to meet the business goal. There are sometimes multiple refinement iterations.

30

Types of SDLC: Evolutionary

- **Iterative**

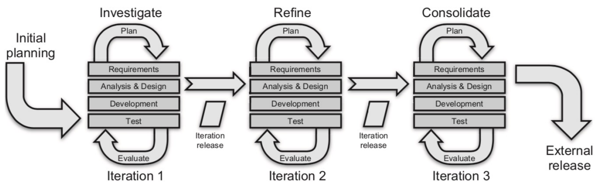


- Consolidate
  - To ensure that the code and design developed is stabilised and fully integrated with any other components or iterations so that the overall business goal is achieved in the working system, the external release.

31

Types of SDLC: Evolutionary

- **Iterative**



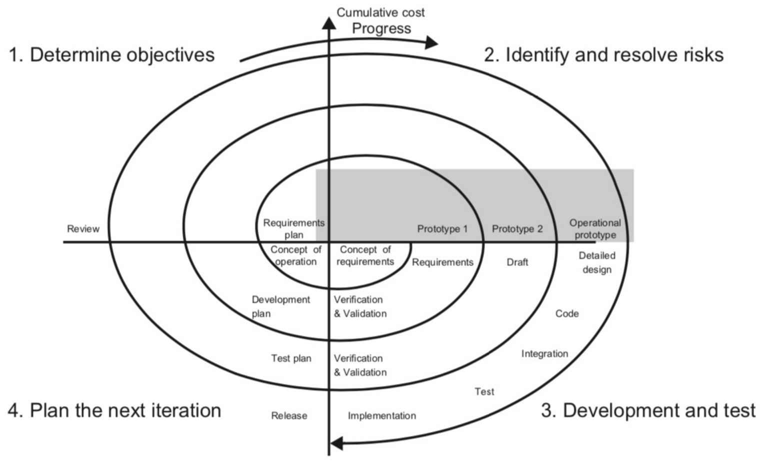
+ Requirements evolve through each iteration;
+ Encourages collaboration with the users throughout so customer buy-in is higher;
+ Change is easier to manage as requirements are not locked down early;
+ Cost can be controlled.

- Can be hard to project manage due to multiple iterations and multiple iteration teams;
- Without careful management, the evolving requirements can result in scope creep;
- If poorly managed, can still appear like a single, final release;
- Overall costs can be higher due to the additional integration and test across multiple teams and iterations.

32

Types of SDLC: Evolutionary

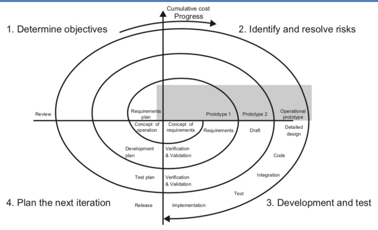
- **Spiral**



33

Types of SDLC: Evolutionary

- **Spiral**



+ Risks are addressed early so risk avoidance is enhanced
+ Requirements evolve, additional functionality can be added at a later stage
+ Collaboration with users throughout is key, so customer buy-in is good
+ A working system is produced early in the development lifecycle.

- There is a high chance of scope creep due to evolving nature of requirements
- Overall costs can be higher due to the many implementations of working software into the operational environment
- The focus is on a working system, documentation of the system can become de-prioritised, making the system harder to maintain post-project delivery;
- It can seem like a lot of iterations are required, so can be a poor choice for small teams or simple problems.

34

Comparing SDLC Model

- Choosing the right SDLC model depends on project needs.

SDLC Model	Use Case	Pros	Cons
Waterfall	Well-defined requirements	Structured, easy to manage	Rigid, late user feedback
Agile	Fast-changing needs	Flexible, quick delivery	Hard to control scope
Spiral	Risk-heavy projects	Risk reduction, iterative	High cost, complex
V-Model	High-reliability systems	Quality-driven, structured	Expensive, time-consuming
Incremental	Early partial releases	Quick benefits, risk handling	More testing & integration

35

Real-World SDLC Applications

SDLC models are used in:

ERP Implementation:
Using an Incremental approach for modules (HR, Finance, Supply Chain).

Banking Systems:
Waterfall for core banking but Agile for customer-facing applications.

Government e-Services:
Using the V-Model to ensure security and compliance.

Startup AI Applications:
Agile or Iterative models for rapid testing and deployment.

36

IS Component in SDLC

- Planning
  - Defines required Infrastructure,
  - Determines Organizational Structure, and
  - Aligns Business Strategy & IT goals.
- Analysis
  - Identifies Business Processes,
  - Gathers Data requirements, and
  - Conducts Business Process Modeling & Requirement Gathering.
- Design
  - Develops System Architecture & UX/UI Design
  - Defines Applications & Data Models, and
  - Designs Processes.
- Development
  - Implements Applications & Coding,
  - Sets up Infrastructure (Servers, Cloud, Network), and
  - Configures Data Storage & Flow.

37

IS Component in SDLC

- Testing
  - Validates Applications & Software,
  - Ensures Processes work correctly, and
  - Conducts Security Evaluations & Software Testing.
- Deployment
  - Sets up Infrastructure,
  - Conducts User Training & Change Management, and
  - Implements Data Migration.
- Maintenance

  Manages System Upgrades & Performance Monitoring,
  Updates Applications, and
  Ensures Infrastructure & Processes remain efficient.

38