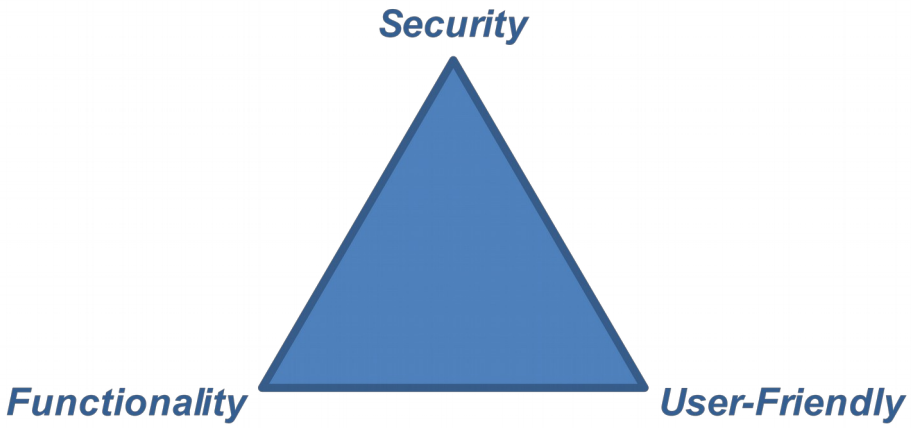
Secure Software Development



CISSP®

Diarmuid Ó Briain
CEng, FIEI, FIET, CISSP

diarmuid@obriain.com

Software Development Controls

CISSP®

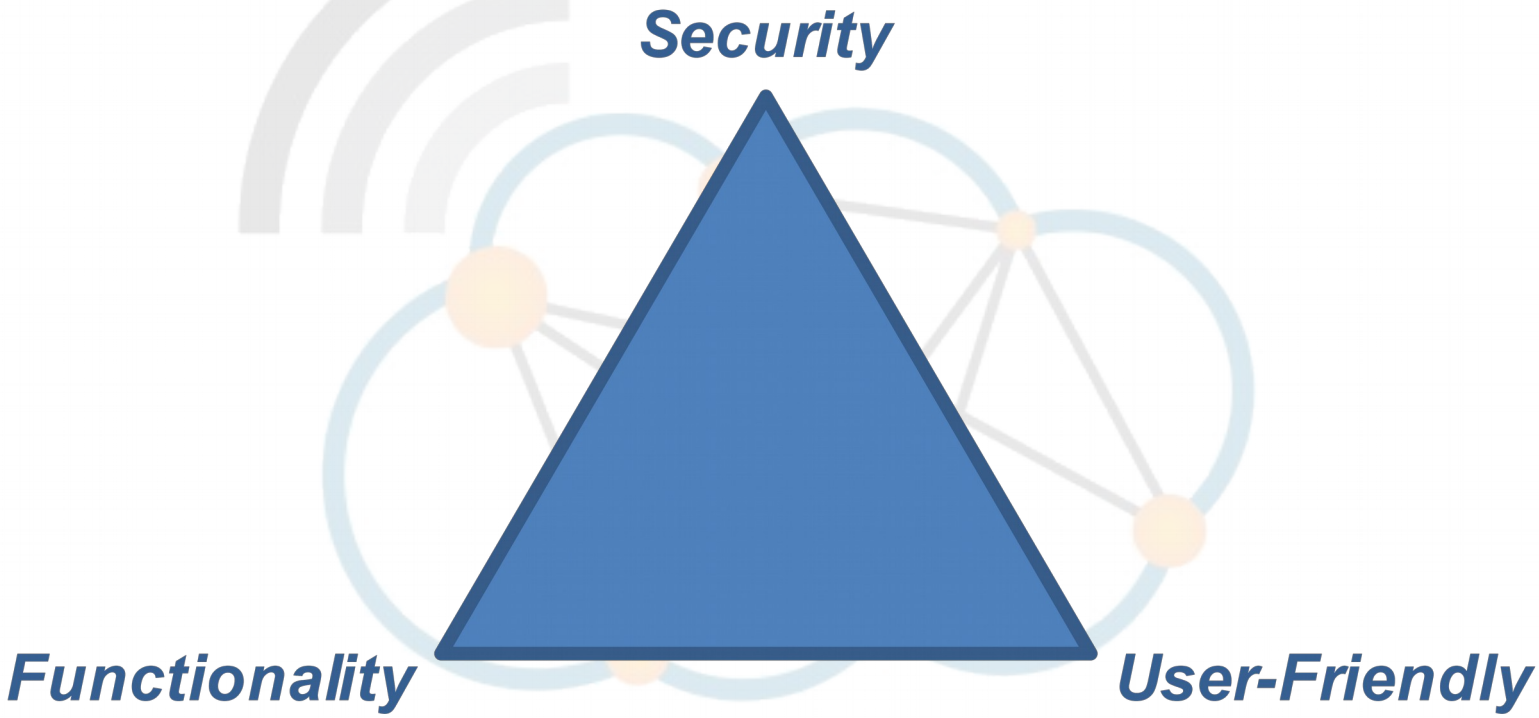
Diarmuid Ó Briain
CEng, FIEI, FIET, CISSP

diarmuid@obriain.com

Software Assurance (SwA)

- SwA "*the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle and that the software functions in the intended manner*".
- Avoiding System Failure
  - **Fail-secure -** highest level of security assumed until the problem is diagnosed and repaired.
  - **Fail-open -** allows users to bypass failed security controls in the event of a failure.

Programming Languages



Programming Languages

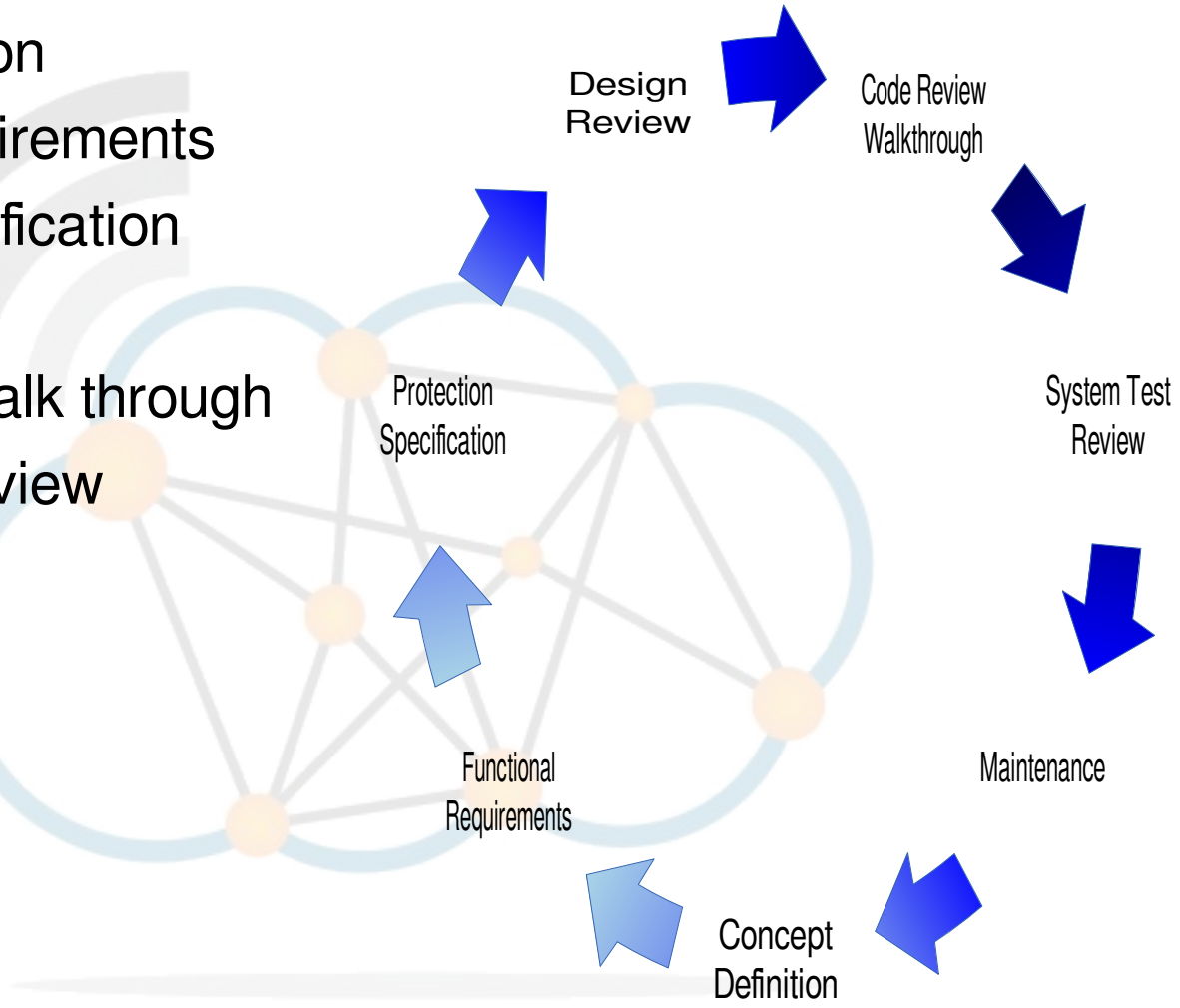
- **First Generation Languages (1GL)**
  - Machine-level programming language.
- **Second Generation Languages (2GL)**
  - Assembly languages.
- **Third Generation Languages (3GL)**
  - First introduced in the late 1950s, For example: Fortran, ALGOL and COBOL.
  - Most 'modern' languages (BASIC, C, C++, C#, Pascal, and Java) are 3GL.
- **Fourth Generation Languages (4GL)**
  - High-level computer language (such as SQL) that allows non-programmer users to write (usually short) programs to query databases and to generate custom reports.
- **Fifth Generation Languages (5GL)**
  - Solve problems using constraints given to the program, rather than using an algorithm written by a programmer.
  - Most constraint-based and logic programming languages and some declarative languages are fifth-generation languages.
  - Used mainly in artificial intelligence research, like Prolog.

Object-oriented Programming (OOP)

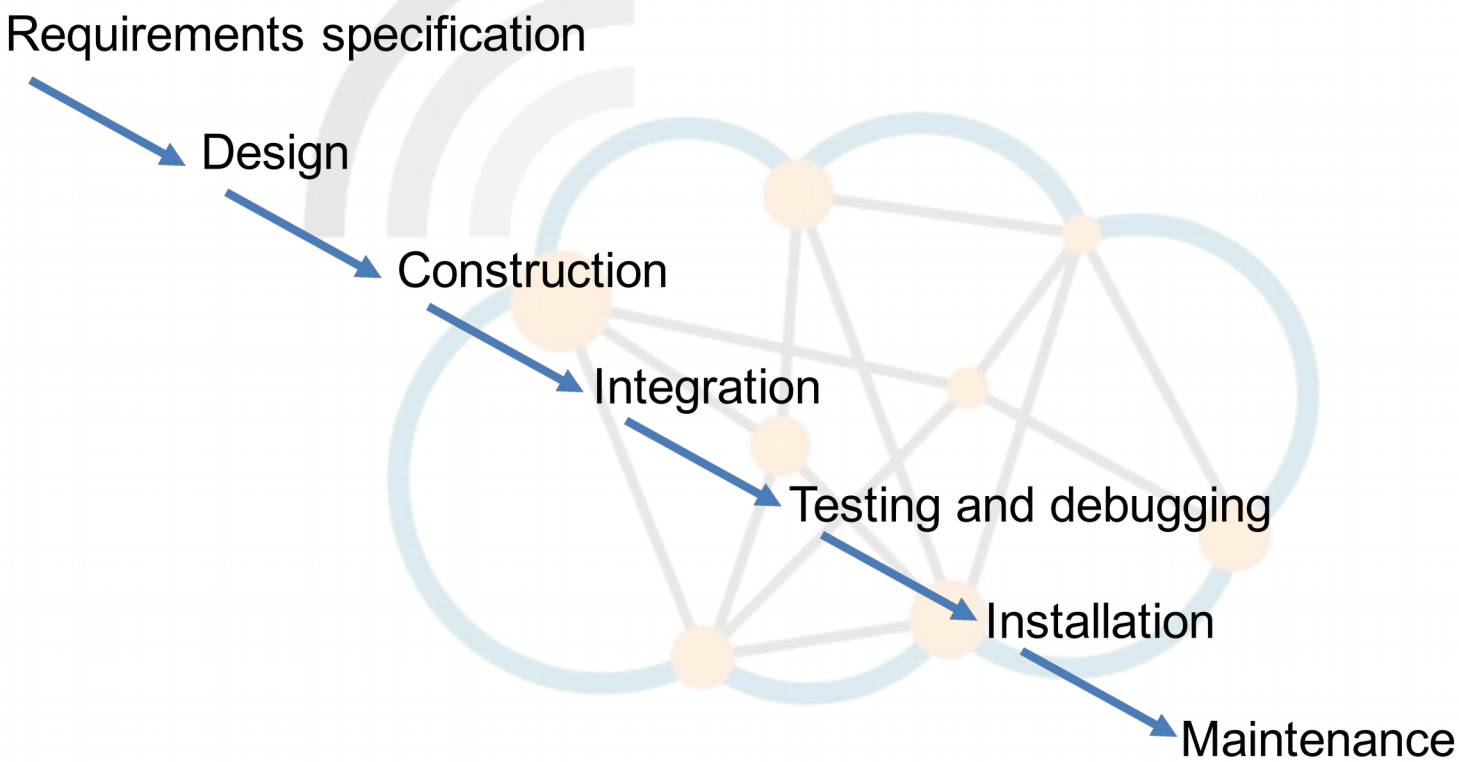
- Example: C++, Java
- OOP is a programming paradigm that uses "*objects*", data structures consisting of data fields and methods.
- OOP may thus be viewed as a collection of interacting objects, as opposed to the conventional model, in which a program is seen as a list of tasks (subroutines) to perform.
- In OOP, each object is capable of receiving messages, processing data, and sending messages to other objects and can be viewed as an independent 'machine' with a distinct role or responsibility.
  - **Class**
  - **Object**
  - **Instance**
  - **Method**
  - **Message passing**
  - **Inheritance**
  - **Abstraction**
  - **Encapsulation**
  - **Polymorphism**
  - **Decoupling**

Systems Development Life Cycle (SDLC)

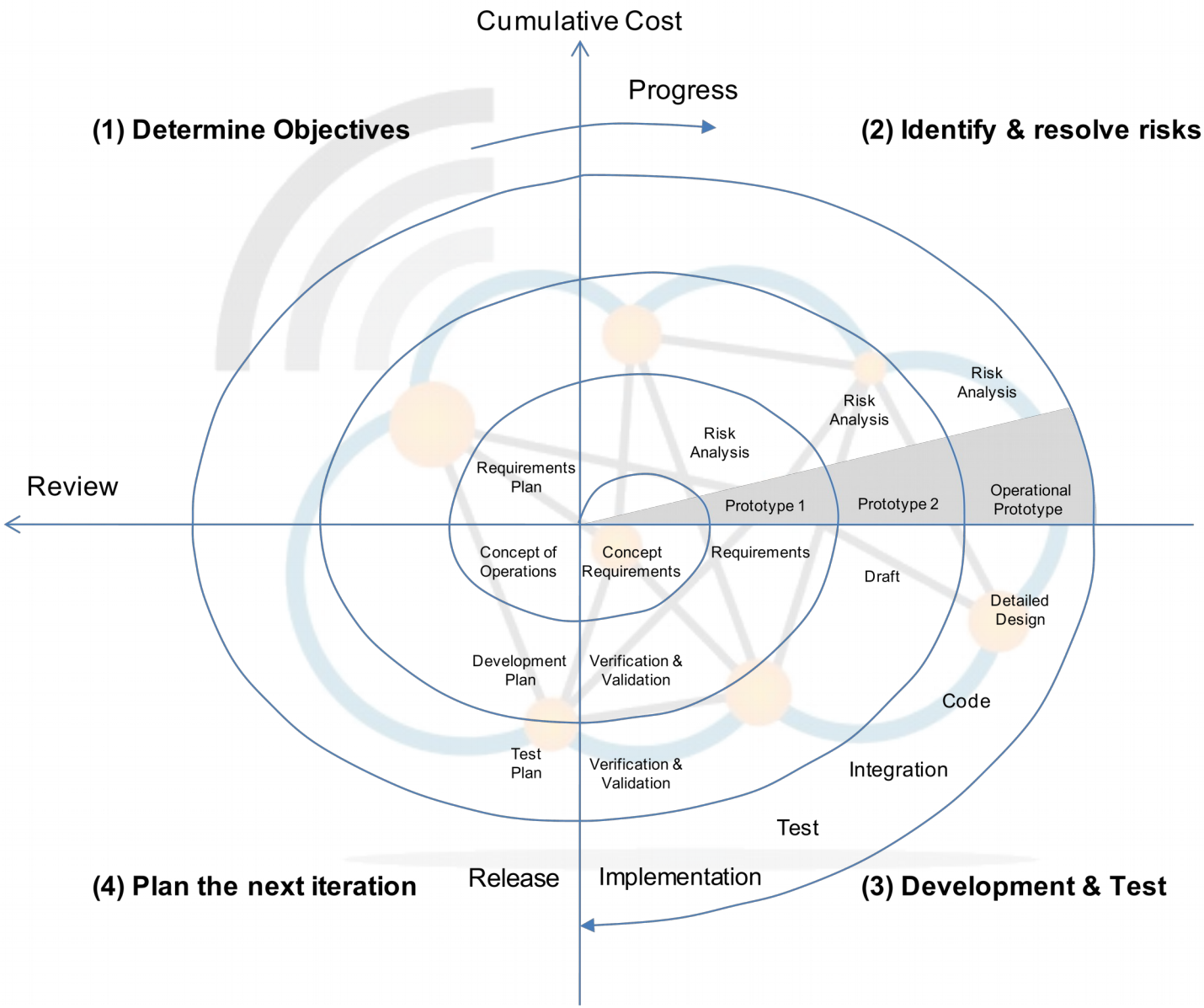
- Concept Definition
- Functional Requirements
- Protection Specification
- Design Review
- Code Review Walk through
- System Test Review
- Maintenance



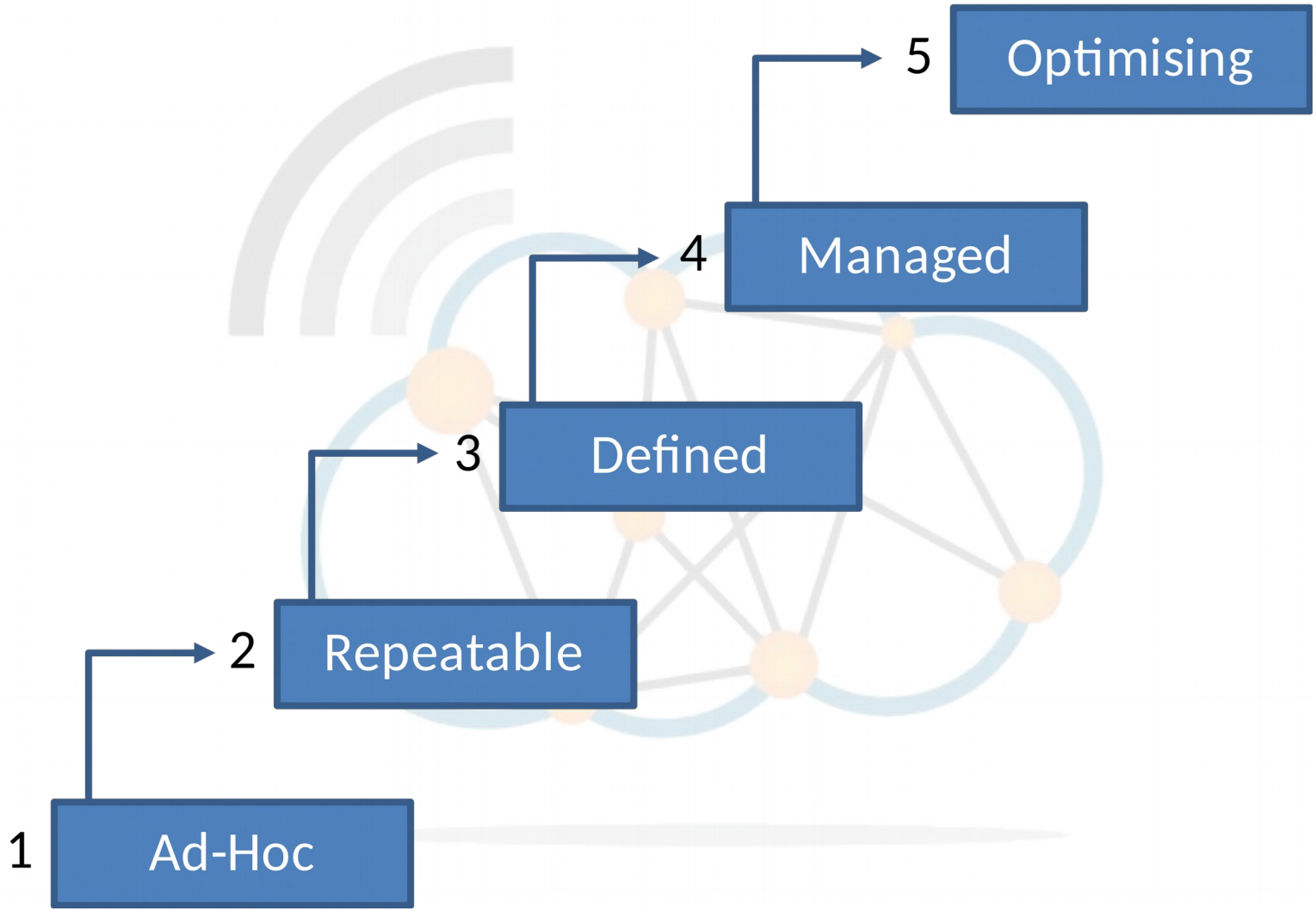
Waterfall model



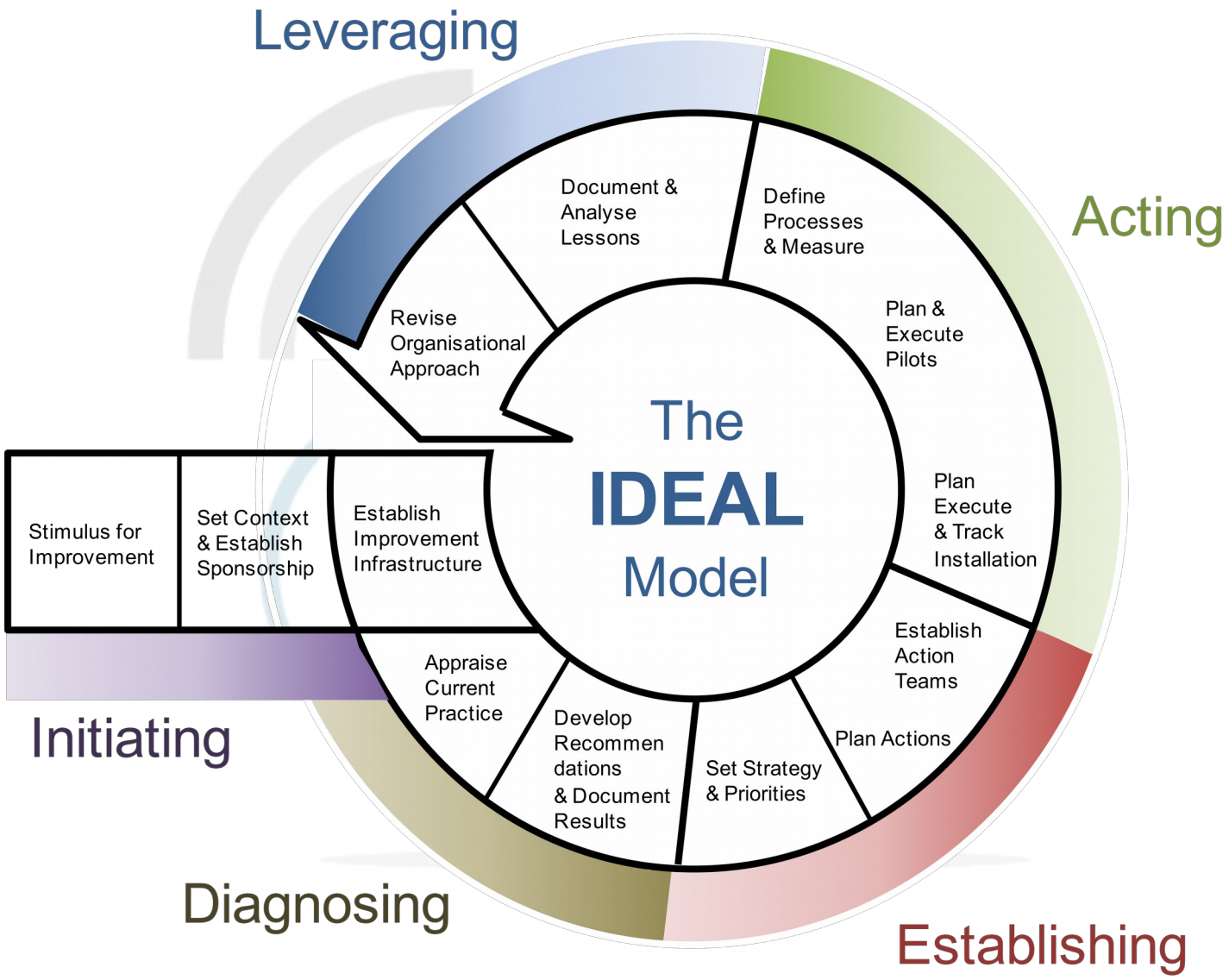
Spiral model



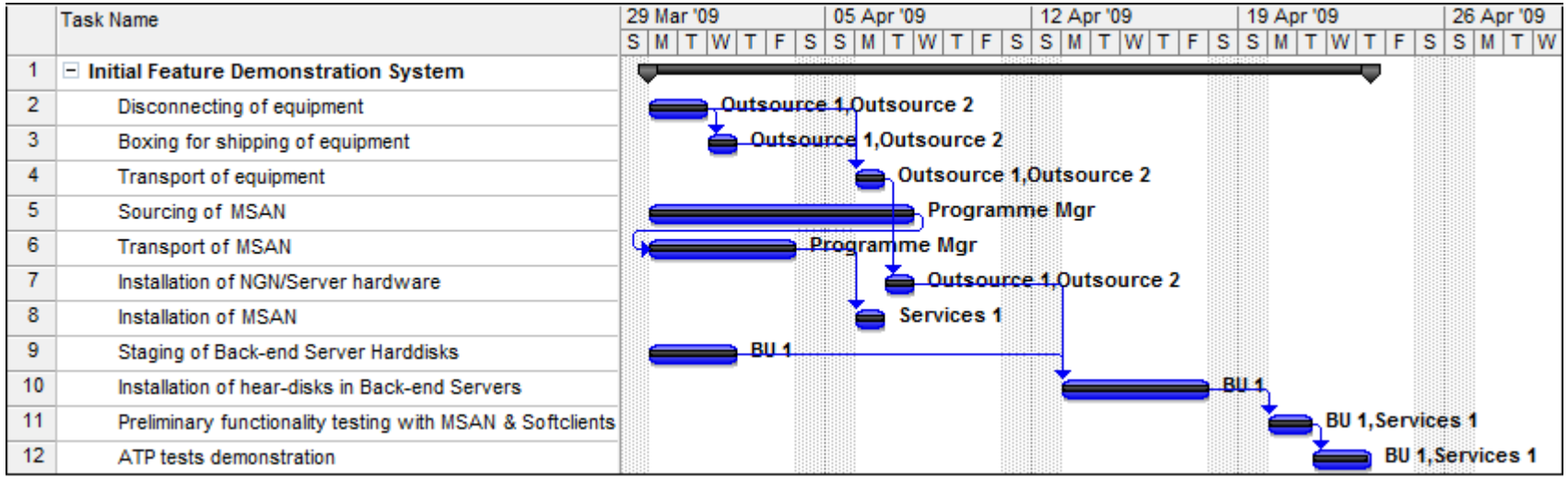
Software Capability Maturity Model



IDEAL Model


Leveraging
Acting
Initiating
Diagnosing
Establishing
The IDEAL Model
Stimulus for Improvement
Set Context & Establish Sponsorship
Establish Improvement Infrastructure
Appraise Current Practice
Develop Recommendations & Document Results
Set Strategy & Priorities
Plan Actions
Establish Action Teams
Plan Execute & Track Installation
Plan & Execute Pilots
Define Processes & Measure
Document & Analyse Lessons
Revise Organisational Approach

Gantt Chart



- Specialist bar chart that illustrates a project schedule.
- Illustrate the start/finish dates of the elements of a project.
- Elements comprise the Work Breakdown Structure (WBS).
- Dependency relationships between activities.
- Current schedule status.
- TODAY line.

Program Evaluation and Review Technique (PERT)

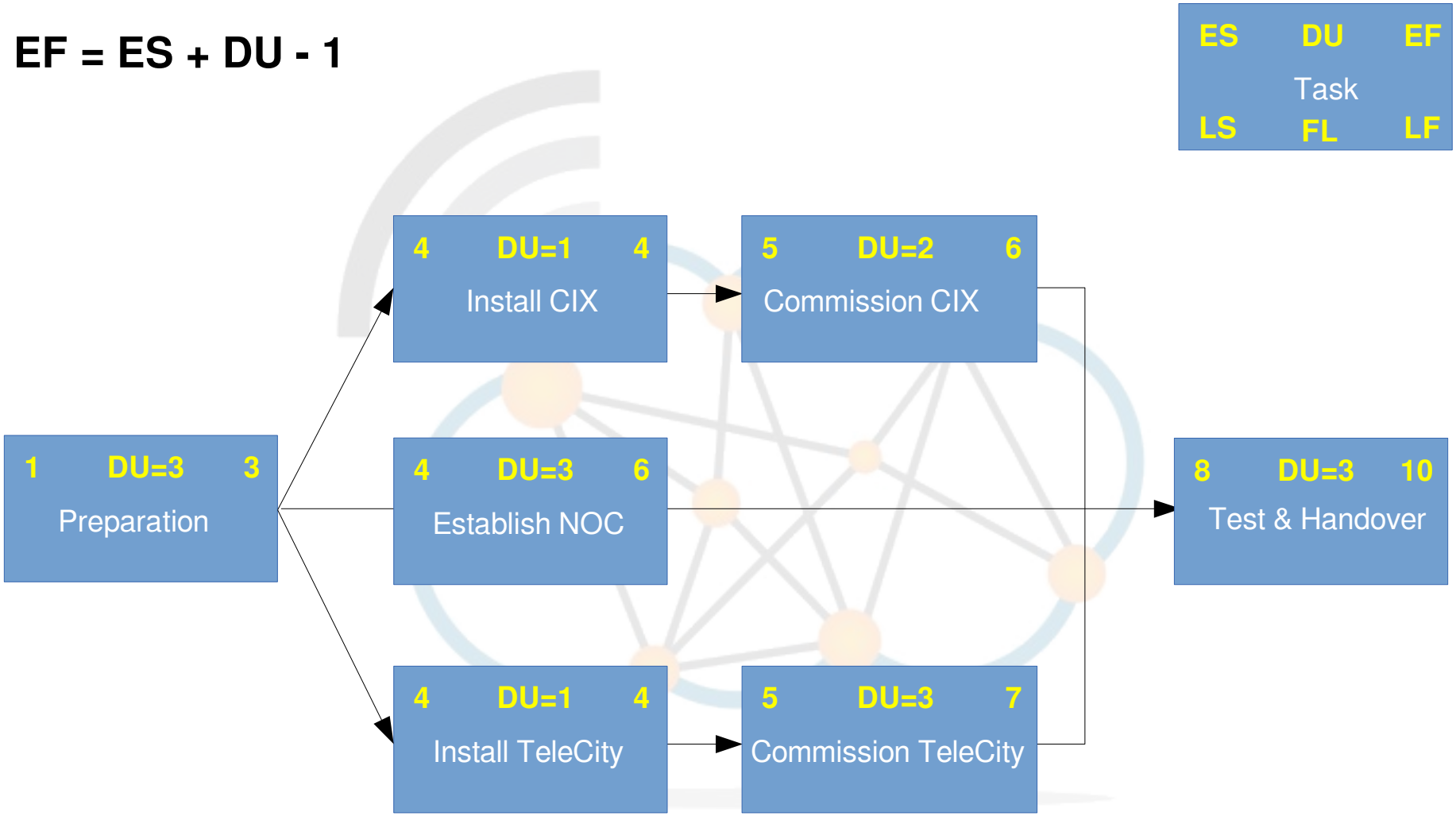
- A method to analyse the involved tasks in completing a given project, especially the time needed to complete each task, and identifying the minimum time needed to complete the total project.
- Developed primarily to simplify the planning and scheduling of large and complex projects.

Forward pass definitions

- **Early Start Date (ES)**
  - Earliest possible point in time an activity can start.
- **Duration (DU)**
  - Number of work periods required for activity.
- **Early Finish Date (EF)**
  - Earliest possible time the activity can finish.
- **Forward Pass**
  - Starting at the beginning (left) of the network develop early start and early finish dates for each task, progressing to end (right-most box) of the network.

Program Evaluation and Review Technique (PERT) Tool – Forward pass

EF = ES + DU - 1



Forward pass calculation

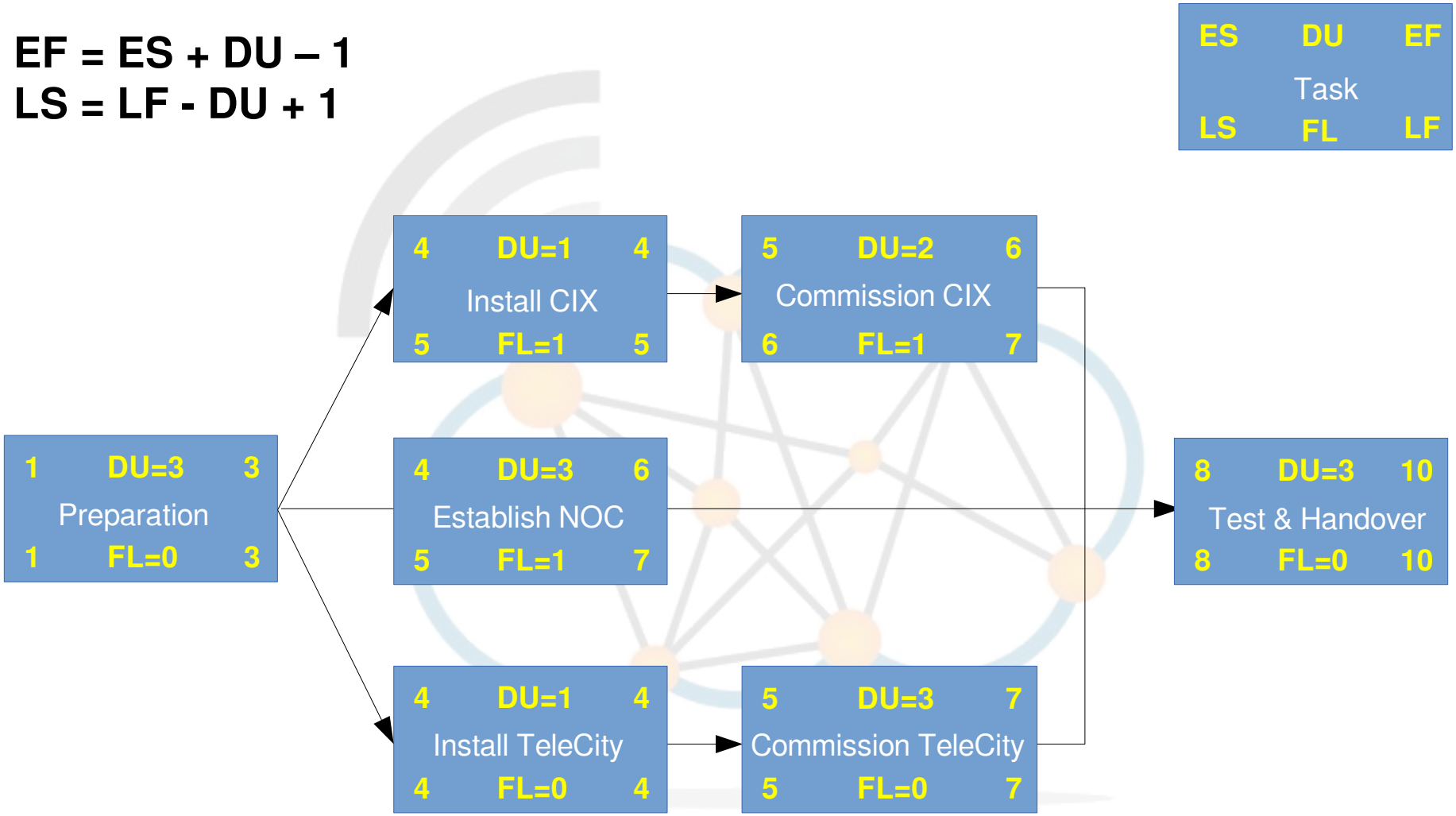
Forward pass calculation			
Name	**Duration**	**ES**	**EF**
Preparation	3	1	3
Install CIX	1	4	4
Establish NOC	3	4	6
Install TeleCity	1	4	4
Commission CIX	2	5	6
Commission TeleCity	3	5	7
Test & Handover	3	8	10

Backward pass definitions

- **Late Start Date (LS)**
  - Latest point in time that an activity may begin.
  - If the activity is on the critical path, the project end date will be affected.
- **Float or Slack (FL)**
  - Latest point in time a task may be delayed from its earliest start date.
- **Late Finish (LF)**
  - Latest point in time a task may be completed.
  - Activity is on the critical path, the project end date will be affected.
- **Backward Pass**
  - Calculate late start and late finish dates by starting at project completion, using finish times and working backwards.

PERT Tool - Backward pass calculation

EF = ES + DU – 1
LS = LF - DU + 1

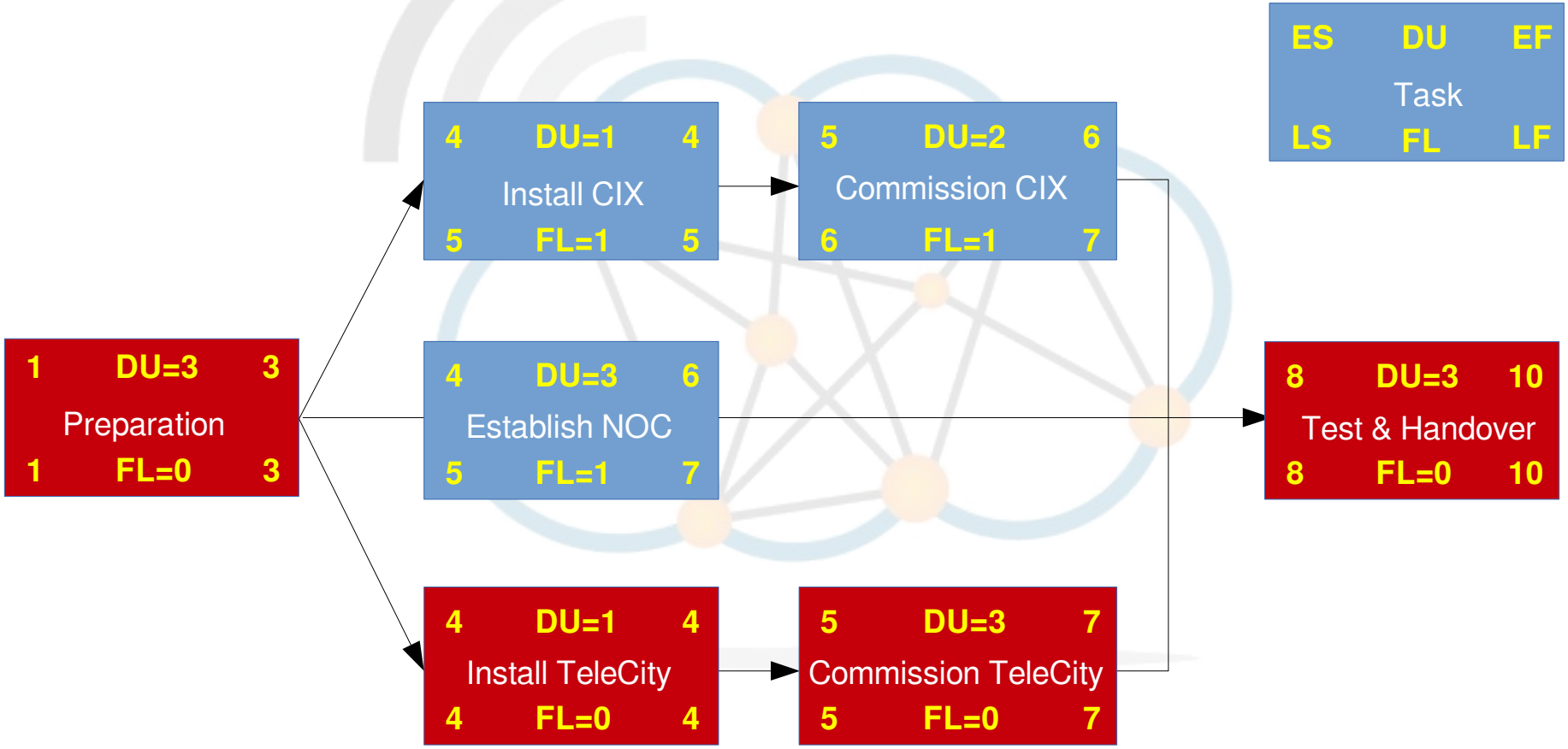


Forward and backward pass calculation

Forward and backward pass calculation						
Name	**Duration**	**ES**	**EF**	**LS**	**LF**	**FL**
Preparation	3	1	3	1	3	0
Install CIX	1	4	4	5	5	1
Establish NOC	3	4	6	5	7	1
Install TeleCity	1	4	4	4	4	0
Commission CIX	2	5	6	6	7	1
Commission TeleCity	3	5	7	5	7	0
Test & Handover	3	8	10	8	10	0

Critical Path

- The Critical Path is the longest possible continuous pathway taken from the initial event to the terminal event.



Change Control

- **Request** Control
  - Process framework whereby users can request modifications, managers can conduct cost/benefit analysis and developers can prioritise tasks.
- **Change** Control
  - Developer's process to recreate a situation reported by a user in order to make appropriate changes to remedy the situation.
- **Release** Control
  - Once changes are made a release control process must be adhered to.
  - Does the change warrant going through system test group etc. ?

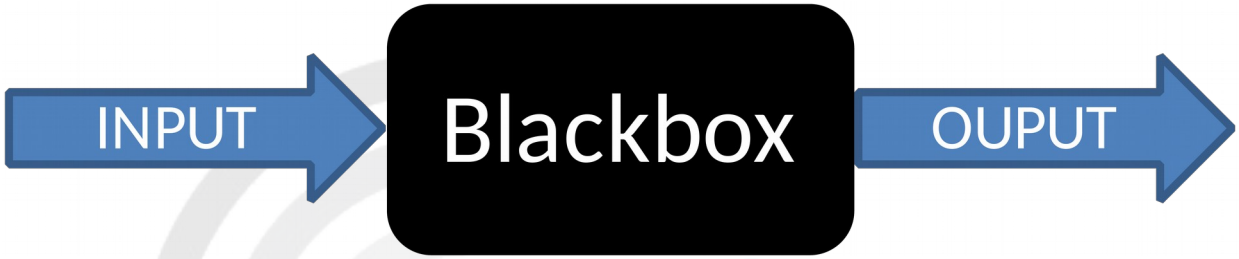
Configuration Management

- **Configuration Identification**
  - Documentation of the configuration on systems.
- **Configuration Control**
  - If changes to the configuration become necessary they must be made in accordance to a change control process.
- **Configuration Status Accounting**
  - Formalised procedures to track of all authorised changes that take place.
- **Configuration Audit**
  - Periodic audits to ensure the production system is consistent with the accounting records and that unauthorised changes have not occurred.

Whitebox testing

- Uses an internal perspective of the system to design test cases based on internal structure.
- Applicable at the unit, integration and system levels of the software testing process, it is typically applied to the unit.
- Can uncover an overwhelming number of test cases, it might not detect unimplemented parts of the specification or missing requirements.
- Typical white box test design techniques include:
  - Control flow testing
  - Data flow testing
  - Branch Testing.

Blackbox testing



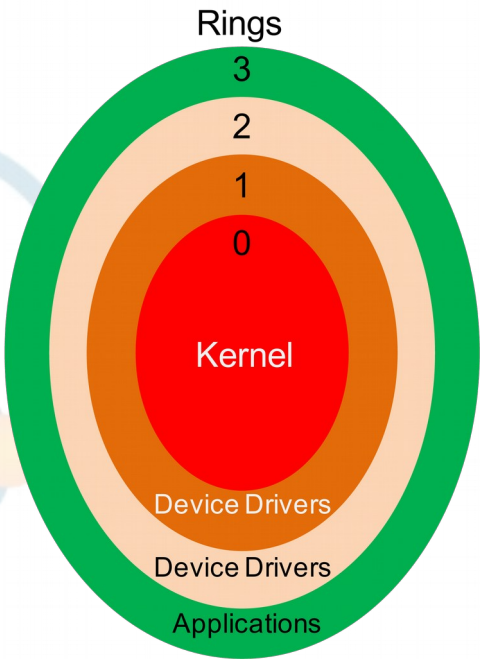
- Takes an external perspective of the test object to derive test cases.
- Can be functional or non-functional, though usually functional.
- The test designer selects valid and invalid inputs and determines the correct output.
- There is no knowledge of the test object's internal structure.
- This method of test design is applicable to all levels of software testing: unit, integration, functional testing, system and acceptance.

Greybox testing

- Grey box testing involves having access to internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black box level.
- Manipulating input data and formatting output do not qualify as grey box, because the input and output are clearly outside of the "black box" that we are calling the system under test.
- This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test.

Security Control Architecture

- **Process Isolation**
  - A set of different hardware and software technologies designed to protect each OS process from other processes. It does so by preventing process A from writing into process B.
- **Protection Rings**
  - Protection rings, are a mechanism to protect data and functionality from faults (fault tolerance) and malicious behaviour (computer security).
  - Rings are arranged in a hierarchy from most privileged (i.e. zero) to least privileged (highest ring number).
  - On most OSs, Ring 0 is the level with the most privileges and interacts most directly with the physical hardware such as the CPU and memory.



Service Level Agreement (SLA)

- An SLA is a negotiated agreement between two parties where one is the customer and the other is the service provider. This can be a legally binding formal or informal "contract".
- The SLA records a common understanding about services, priorities, responsibilities, guarantees, and warranties.
- It is important to note that the "agreement" relates to the services the customer receives, and not how the service provider delivers that service.

Open Web Application Security Project (OWASP)

Open Web Application Security Project (OWASP)

- Open community dedicated to enabling organistions to have trusted applications.
- OWASP helps organisations:
  - Conceive
  - Develop
  - Acquire
  - Operate
  - Maintain

  } applications that can be trusted.
- All of the OWASP tools, documents, forums, and chapters are free and open to anyone interested in improving application security.

OWASP – Top 10 Web vulnerabilities

- A1: Injection
- A2: Broken Authentication and Session Management
- A3: Cross-Site Scripting (XSS)
- A4: Broken Access Control
- A5: Security Misconfiguration
- A6: Sensitive Data Exposure
- A7: Insufficient Attack Protection
- A8: Cross-Site Request Forgery (CSRF)
- A9: Using Components with Known Vulnerabilities
- A10: Under-protected APIs.

OWASP – Top 10 Mobile vulnerabilities

- M1: Improper Platform Usage
- M2: Insecure Data Storage
- M3: Insecure Communication
- M4: Insecure Authentication
- M5: Insufficient Cryptography
- M6: Insecure Authorisation
- M7: Poor Code Quality
- M8: Code Tampering
- M9: Reverse Engineering
- M10: Extraneous Functionality..

OWASP – Top 10 Pro Active Controls

- C1: Verify for Security Early and Often
- C2: Parameterise Queries
- C3: Encode Data
- C4: Validate All Inputs
- C5: Implement Identity and Authentication Controls
- C6: Implement Appropriate Access Controls
- C7: Protect Data
- C8: Implement Logging and Intrusion Detection
- C9: Leverage Security Frameworks and Libraries
- C10: Error and Exception Handling.

Open Web Application Security Project (OWASP)

	A1-Injection	A2-Broken Authentication and Session Management	A3-Cross-Site Scripting (XSS)	A4-Insecure Direct Object References	A5-Security Misconfiguration	A6-Sensitive Data Exposure	A7-Missing Function Level Access Control	A8-Cross-Site Request Forgery (CSRF)	A9-Using Components with Known Vulnerabilities	A10-Unvalidated Redirects and Forwards
C1: Verify for Security Early and Often	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C2: Parameterize Queries	✓									
C3: Encode Data	✓		✓							
C4: Validate All Inputs	✓		✓							✓
C5: Implement Authentication Controls		✓								
C6: Implement Appropriate Access Controls				✓			✓			
C7: Protect Data						✓				
C8: Implement Logging and IDs	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C9: Leverage Security Frameworks	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C10: Error and Exception Handling	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Thank you