CS4135
Software Architectures

Dr. Salim Saay
Associate professor, CSIS, University of Limerick
Funded Investigator at LERO and WAVE
Salim.saay@ul.ie

2026

Lecturer

Dr. Salim Saay
Salim.saay@ul.ie

- ❖ PhD. Software Engineering, Tallinn University,
- ❖ MSc. Computer Science, University of the Western Cape
- ❖ BSc. Computer Science, Kabul University.
- ❖ Postdoc. University of Limerick.

Teaching:

- ❖ Immersive Software Engineering(ISE)
- ❖ Technological University Shannon (TUS)
- ❖ Kabul University (KU)

Team members

Mariam Ali is a PhD at UL

- Three years of professional experience in software development
- Working on the WAVE project, which aims to drive advancements in platforms and methods for low-code/no-code (LC/NC) software development.
- You can reach Mariam via email ali.mariam@ul.ie or visit her office at CSG-028A

Gabriel, PhD at UL

- Bachelor's and a Master's in Computer Science,
- PhD student at UL, working on the WAVE
- Over 7 years of experience in software development.
- His focus has around DevOps, Cybersecurity and Software Engineering.
- Reached via Teams or Email: oyeyemi.gabriel@ul.ie

Content

Part 1: Module introduction

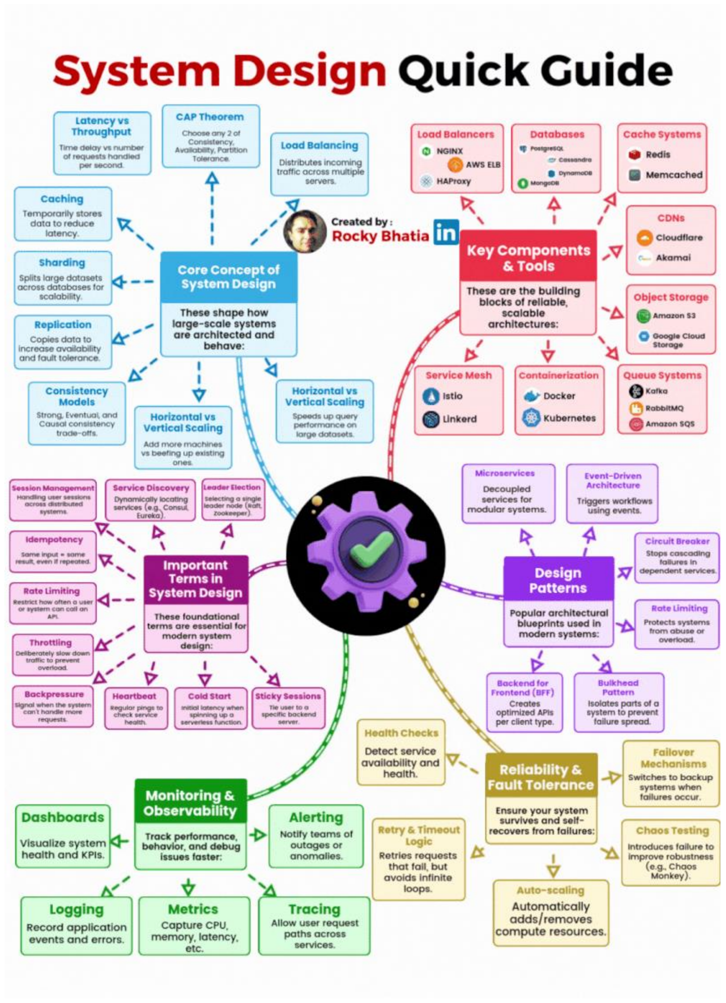
- ❖Introduction
- ❖Purpose of the Module
- ❖Learning Outcomes
- ❖Teaching Method
- ❖Assessment

Part 2: Software Architecture

- ❖Concept of SA
- ❖Software Architect
- ❖Example project

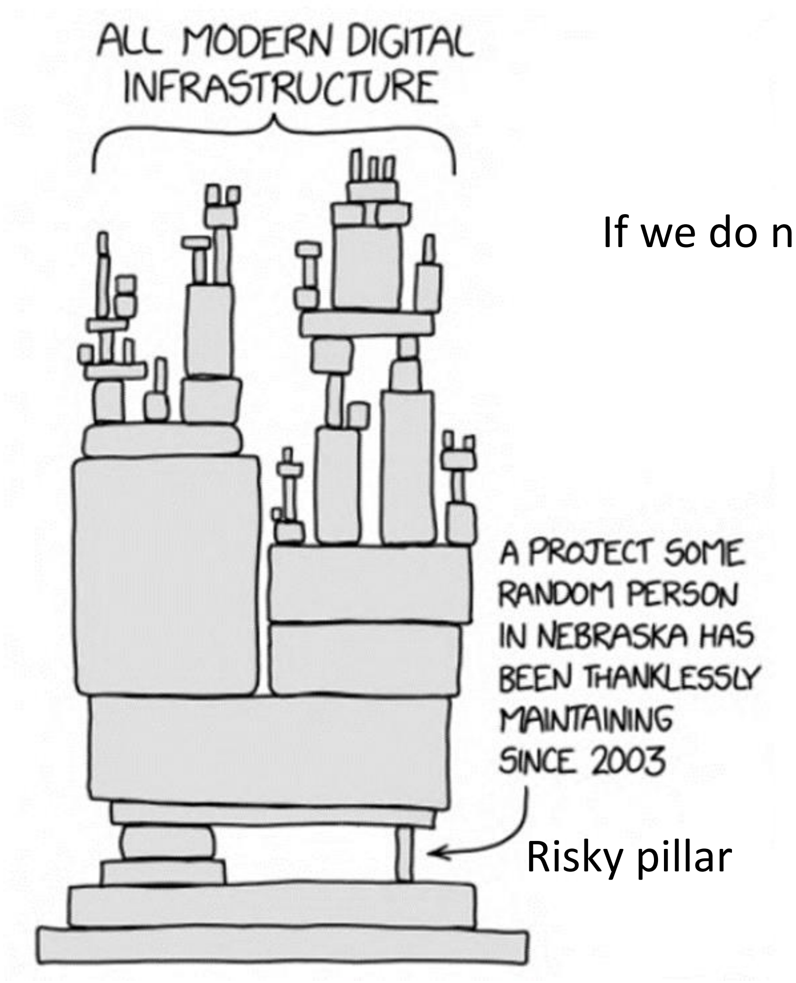
What is Software Architecture

- ❖ Software Architecture covers how a software system is constructed at the highest level.
- ❖ Development teams will decide how a software system will be broken down into **components** that work together.
- ❖ Modern digital infrastructures are interconnected with each other
- ❖ Many software development phases can be developed by Tools and GenAI. But SA is unique for each domain.



System Design Quick Guide
Created by : Rocky Bhatia
Core Concept of System Design
These shape how large-scale systems are architected and behave:
Latency vs Throughput
Time delay vs number of requests handled per second.
CAP Theorem
Choose any 2 of Consistency, Availability, Partition Tolerance.
Load Balancing
Distributes incoming traffic across multiple servers.
Caching
Temporarily stores data to reduce latency.
Sharding
Splits large datasets across databases for scalability.
Replication
Copies data to increase availability and fault tolerance.
Consistency Models
Strong, Eventual, and Causal consistency trade-offs.
Horizontal vs Vertical Scaling
Add more machines vs beefing up existing ones.
Horizontal vs Vertical Scaling
Speeds up query performance on large datasets.
Key Components & Tools
These are the building blocks of reliable, scalable architectures:
Load Balancers
NGINX
AWS ELB
HAProxy
Databases
PostgreSQL
Cassandra
DynamoDB
MongoDB
Cache Systems
Redis
Memcached
CDNs
Cloudflare
Akamai
Object Storage
Amazon S3
Google Cloud Storage
Service Mesh
Istio
Linkerd
Containerization
Docker
Kubernetes
Queue Systems
Kafka
RabbitMQ
Amazon SQS
Important Terms in System Design
These foundational terms are essential for modern system design:
Session Management
Handling user sessions across distributed systems.
Service Discovery
Dynamically locating services (e.g. Consul, Eureka).
Leader Election
Selecting a single leader node (Raft, Zookeeper).
Idempotency
Same input = same result, even if repeated.
Rate Limiting
Restrict how often a user or system can call an API.
Throttling
Deliberately slow down traffic to prevent overload.
Backpressure
Signal when the system can't handle more requests.
Heartbeat
Regular pings to check service health.
Cold Start
Initial latency when spinning up a serverless function.
Sticky Sessions
Tie user to a specific backend server.
Design Patterns
Popular architectural blueprints used in modern systems:
Microservices
Decoupled services for modular systems.
Event-Driven Architecture
Triggers workflows using events.
Circuit Breaker
Stops cascading failures in dependent services.
Rate Limiting
Protects systems from abuse or overload.
Backend for Frontend (BFF)
Creates optimized APIs per client type.
Bulkhead Pattern
Isolates parts of a system to prevent failure spread.
Reliability & Fault Tolerance
Ensure your system survives and self-recovers from failures:
Health Checks
Detect service availability and health.
Failover Mechanisms
Switches to backup systems when failures occur.
Retry & Timeout Logic
Retries requests that fail, but avoids infinite loops.
Chaos Testing
Introduces failure to improve robustness (e.g., Chaos Monkey).
Auto-scaling
Automatically adds/removes compute resources.
Monitoring & Observability
Track performance, behavior, and debug issues faster:
Dashboards
Visualize system health and KPIs.
Alerting
Notify teams of outages or anomalies.
Logging
Record application events and errors.
Metrics
Capture CPU, memory, latency, etc.
Tracing
Allow user request paths across services.

What is Software Architecture


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

Risky pillar

If we do not develop the architecture properly.

Part 1: Introduction

Purpose of the Module

Provide hands-on technical exposure to core software architecture concepts, principles, methodologies, and industry best practices.

Enable students to design and evaluate robust, scalable, and maintainable software systems.

Topics

- Definitions of software architecture; Architecture versus design; Role of a software architect; and Relationship between system qualities and software architectures.
- Software architecture process:
  - Determining, identifying and prioritising requirements;
  - Choose an architectural pattern;
  - validate using scenarios and prototyping;
- Concepts of views and viewpoints: module-structured views, component and connector views and allocation views; the role of views in specifying an architecture

Introduction: Topics

- Architectural styles/patterns:
  - Layered systems;
  - Broker architecture,
  - Event-Driven Architecture,
  - Plug-in Architecture,
  - Microservices architecture.
- Middleware architectures and technologies: distributed objects, message-oriented middleware, application servers, message brokers;
- Aspect-oriented architectures, model-driven architectures, service-oriented architectures;
  - Architectural description languages;
  - Documenting a software architecture;
  - Software architecture evaluation;
  - Architectural reuse;
- Evaluation: Architecture Trade-off analysis Method (ATAM), and Software Architecture Analysis Method (SAAM).

Learning Outcomes

- On successful completion of this module, students should be able to:
  - Identify and assess the appropriate quality attributes of a system at the architectural level.
  - Recognise software architectural styles, design patterns, and frameworks within existing software systems.
  - Debate the current trends and technologies in the field, such as model-driven, service-oriented, and aspect-oriented architectures.
  - Express a software architecture using appropriate representations.
  - Generate architectural alternatives for a problem and select among them.
  - Evaluate a software architecture;

Module Overview

Fundamental Knowledge

Hands-on Practice in Real-world projects

Assess in the final exam

Assessed through a series of five assignments

Delivery method/ time

Module Code - Title:

CS4135 - SOFTWARE ARCHITECTURES

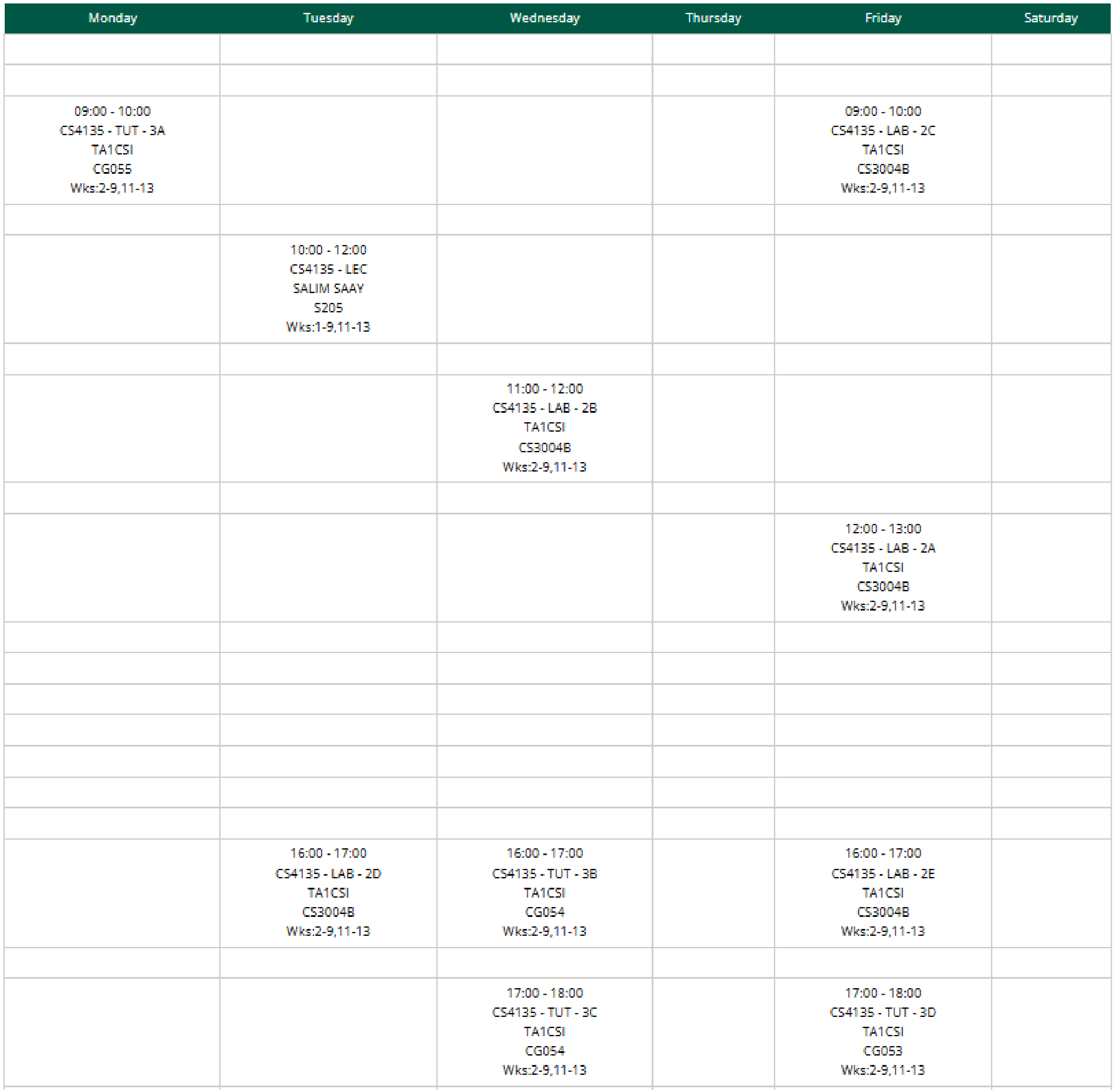
Year Last Offered:

2025/6

Hours Per Week:

Lecture	*Lab*	*Tutorial*	*Other*	*Private*	*Credits*
2	1	1	0	6	6

Timetable

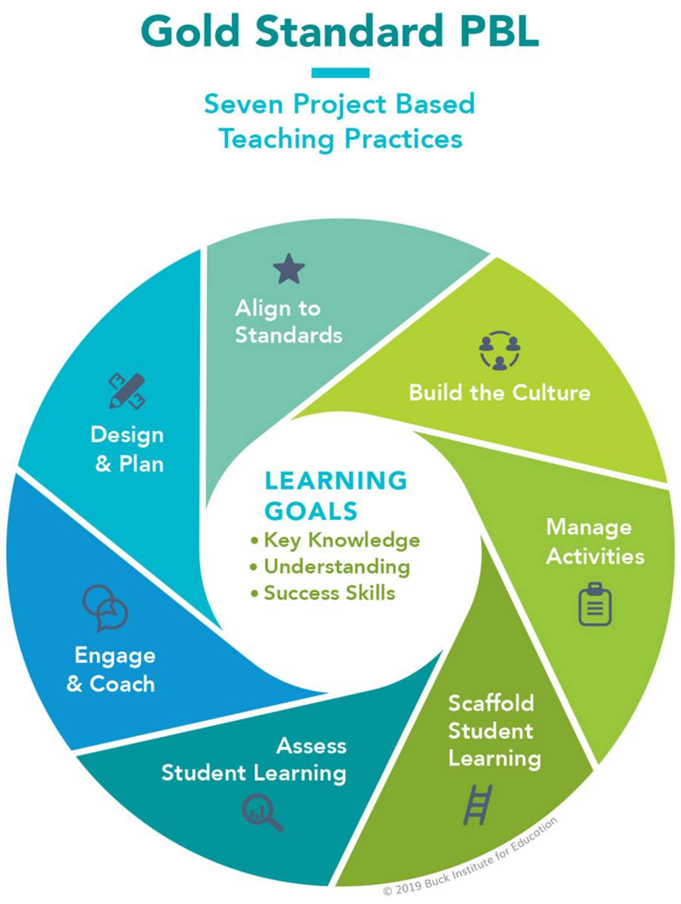


Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
09:00 - 10:00 CS4135 - TUT - 3A TA1CSI CG055 Wks:2-9,11-13				09:00 - 10:00 CS4135 - LAB - 2C TA1CSI CS3004B Wks:2-9,11-13	
	10:00 - 12:00 CS4135 - LEC SALIM SAAY S205 Wks:1-9,11-13				
		11:00 - 12:00 CS4135 - LAB - 2B TA1CSI CS3004B Wks:2-9,11-13			
				12:00 - 13:00 CS4135 - LAB - 2A TA1CSI CS3004B Wks:2-9,11-13	
	16:00 - 17:00 CS4135 - LAB - 2D TA1CSI CS3004B Wks:2-9,11-13	16:00 - 17:00 CS4135 - TUT - 3B TA1CSI CG054 Wks:2-9,11-13		16:00 - 17:00 CS4135 - LAB - 2E TA1CSI CS3004B Wks:2-9,11-13	
		17:00 - 18:00 CS4135 - TUT - 3C TA1CSI CG054 Wks:2-9,11-13		17:00 - 18:00 CS4135 - TUT - 3D TA1CSI CG053 Wks:2-9,11-13	

Teaching Method

In this module, we employ a hybrid teaching concept composed of **Lectures** and **project-based teaching**.

- ❖ Lectures cover the concept and fundamentals of software architecture.
  - ➢ You will learn different patterns and methods of architectural development. Development process, Evaluation
  - ➢ 10-minute break in between.



https://www.pblworks.org/what-is-pbl

Teaching Method

- Accompanying lab project complements the lectures.
  - Need to form teams, each consisting of 4 to 6 participants.
  - The teams are needed to pick one of the **application scenarios** that we will introduce in the form of short project profiles.
  - The scenarios are typically based on current industry or research projects we are involved in.
  - Each profile describes the context, objectives, and basic requirements and technical requirements of the project, and thus provides a framework with guardrails for the teams.
  - This ensures that the workload of students remains within the intended limits.

Teaching Method

- Students also have the option to propose and discuss their own project.
- After picking an application scenario, each team starts working on its respective project in LAB and Tutorial days.
- Your progress in this module will be assessed through a series of five lab assessments, which guide you through different phases of the project.
- The lab assignments are released incrementally and synchronised with the contents of the lectures.

Assessment

The five phases are structured as follows :

Project Timeline

Week	Lab	Deadline
Week 2-3	Project Setup	13/02/2026
Week 4-5	Requirements and UML documentation	27/02/2026
Week 6-10	Strategic architecture development	10/04/2026
Week 10-12	Evaluation and Peer review	24/04/2026
Week 13-14	Deployment and documentation	8/05/2026
Week 15	Final Exam	

Assessment

Repeats Process

- ❖ If a lower mark is achieved. Upon completion of all five assignments, your cumulative marks will be evaluated against the required overall pass mark for the module. If you do not attain the necessary pass mark, you will be required to retake the module in the upcoming semester.

- ❖ Repeat Assessment (summer): 70%

- ❖ You cannot submit an assessment that you (or someone else) have previously submitted to this or another module in UL or elsewhere, or that has similarity to such an assessment.

- ❖ For all assessments, you are required to declare your use of **GenAI** tools or other such tools that you have used in the preparation of an assessment.

Assessment

For all five assessments:

- ❖ Each assessment must be linked to Lab work.
- ❖ Group work assessments (But the contribution of each individual should be clear in the report)
- ❖ Report to be submitted to Brightspace. The deadline for submission will be on the Brightspace assignment.
- ❖ Failure to submit on time results in penalties 0%. For exceptional circumstances (e.g. illness documented with certification), alternative arrangements will be discussed on an individual basis.
- ❖ In other cases, where assessments are not submitted, an **F-**grade will be awarded, and thus the module will need to be repeated over the summer and in time for the annual repeats.

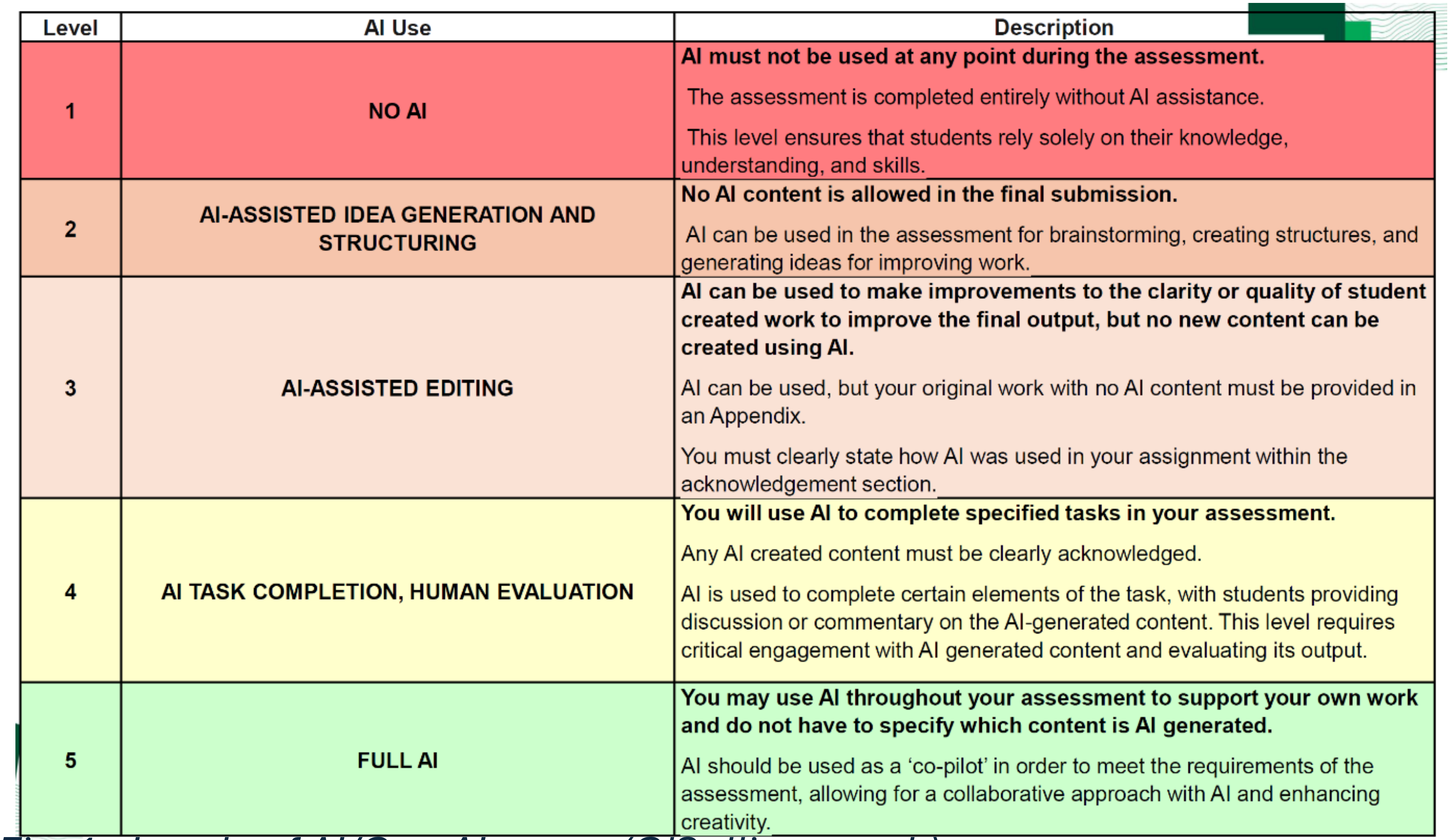
Assessment

Plagiarism

- The UL Student Charter at UL (under Integrity, page 6) states that: The University expects students not to plagiarise (i.e., present another's ideas or writings as their own), fabricate or falsify data, commission others to complete assessments or engage in academic cheating in any form whatsoever.
- Project work undertaken by the student on a taught program should be clearly defined, well-planned and where necessary, have Research Ethics approval.

- Academic Integrity and Academic Misconduct | University of Limerick

Declaration of use of GenAI



Level	AI Use	Description
1	NO AI	**AI must not be used at any point during the assessment.** The assessment is completed entirely without AI assistance. This level ensures that students rely solely on their knowledge, understanding, and skills.
2	AI-ASSISTED IDEA GENERATION AND STRUCTURING	**No AI content is allowed in the final submission.** AI can be used in the assessment for brainstorming, creating structures, and generating ideas for improving work.
3	AI-ASSISTED EDITING	**AI can be used to make improvements to the clarity or quality of student created work to improve the final output, but no new content can be created using AI.** AI can be used, but your original work with no AI content must be provided in an Appendix. You must clearly state how AI was used in your assignment within the acknowledgement section.
4	AI TASK COMPLETION, HUMAN EVALUATION	**You will use AI to complete specified tasks in your assessment.** Any AI created content must be clearly acknowledged. AI is used to complete certain elements of the task, with students providing discussion or commentary on the AI-generated content. This level requires critical engagement with AI generated content and evaluating its output.
5	FULL AI	**You may use AI throughout your assessment to support your own work and do not have to specify which content is AI generated.** AI should be used as a 'co-pilot' in order to meet the requirements of the assessment, allowing for a collaborative approach with AI and enhancing creativity.

Fig. 1: Levels of AI/Gen AI usage (O'Sullivan et al.)

Assessment and Feedback

Grading Criteria

❖For a comprehensive understanding of the grading criteria, please refer each **assessment brief.** The detailed breakdown provided in each brief will offer specific information on how your work will be evaluated.

Late Submission

❖Late submissions for student assignments will be accepted **up to one day** after the designated deadline. However, a penalty of **10% of the total marks** for the assignment will be applied to late submissions. This penalty is non-negotiable and aims to maintain fairness and consistency among all students.

❖**After the one-day grace period**, late submissions will not be accepted unless students have made prior arrangements with the lecturer and received confirmation via email. It is essential to communicate with the lecturer well in advance to discuss any unforeseen circumstances that may prevent timely submission.

Grading

Each assessment has a percentage grade:

1. Project setup - 5% of the total grades
2. Requirements - 5 % of the total grades
3. Strategic architecture development - 30% of the total grades
4. Evaluation and Peer review- 10% of the total grades
5. Technical Documentation -10% of the total grades
6. Final Exam 40% of the total grade.

Mark	Grade	QCA
80%	A1	4.00
72%	A2	3.60
64%	B1	3.20
60%	B2	3.00
56%	B3	2.80
52%	C1	2.60
48%	C2	2.40
40%	C3	2.00
35%	D1	1.60
30%	D2	1.20
<30%	F	0.00

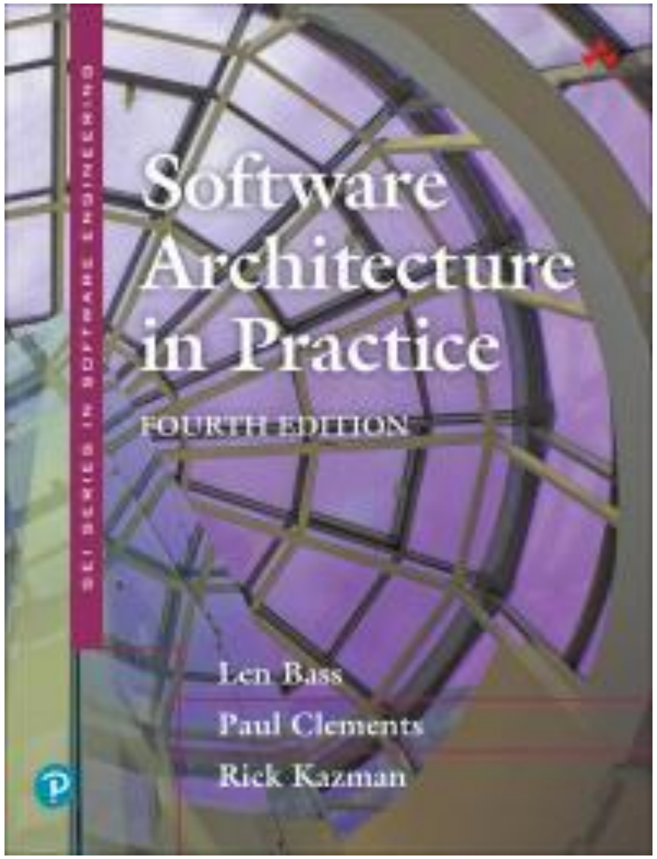
References for the assessment

1. O'Sullivan, A. Risquez, M.C. Kennedy and F. McGrath, "Academic Integrity and Assessment" Presentation from Centre for Transformative Learning and Academic Integrity Unit, University of Limerick.

Required tools and Materials

- Slides
- Books and up-to-date research papers
- UML https://online.visual-paradigm.com/
- ArchiMate Untitled | Visual Paradigm Online (visual-paradigm.com)
- BPMN (Business Process Model and Notation) https://www.bpmn.org/
- Install Eclipse, Java EE or Visual Studio
- Spring Boot
- React.js

Part 2: Software Architecture and Software Architect


SEI SERIES IN SOFTWARE ENGINEERING
Software Architecture in Practice
FOURTH EDITION
Len Bass
Paul Clements
Rick Kazman

The concept of Software Architecture

- Software architecture is an **essential activity** for the development of software systems
  - Enabling reasoning about system properties
  - **Early** in the development lifecycle.
- The issue is on how to organize a system to, simultaneously:
  - provide the required **functional** services,
  - guarantee the required **quality of service**.

The concept of Software Architecture

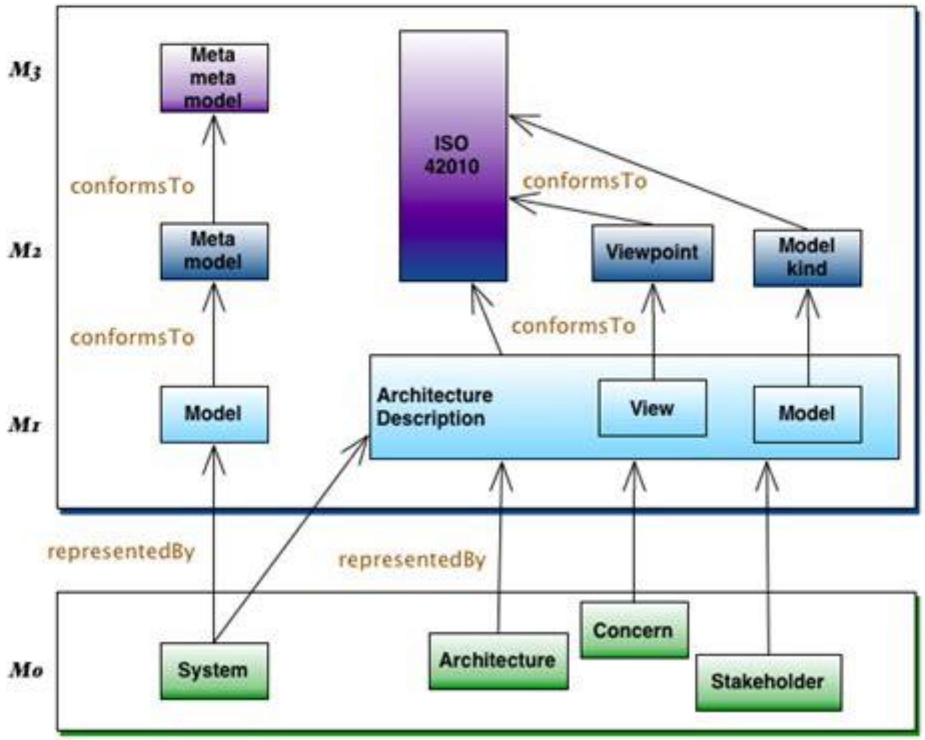
- Software architecture is **concerned** with the n**on-functional** aspects of a system, such as
  - Scalability, reliability, maintainability, and security,
- The **functional** requirements of the system.
  - The structure of the system,
  - The technologies to be used,
  - The design patterns to be employed, and
  - The trade-offs between various competing factors.

The concept of Software Architecture

- Since 2011, we have had an official definition of software architecture [ISO/IEC Standard 42010] "
  - **The fundamental properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution".**
- To understand the concept and definition of software architecture, need to know:
  - **structure**: what is the form of the interrelated elements
  - **behaviour**: what is the functionality that the interrelated elements provide
  - **execution**: how the functionality provided by the interrelated elements is carried out

The concept of Software Architecture

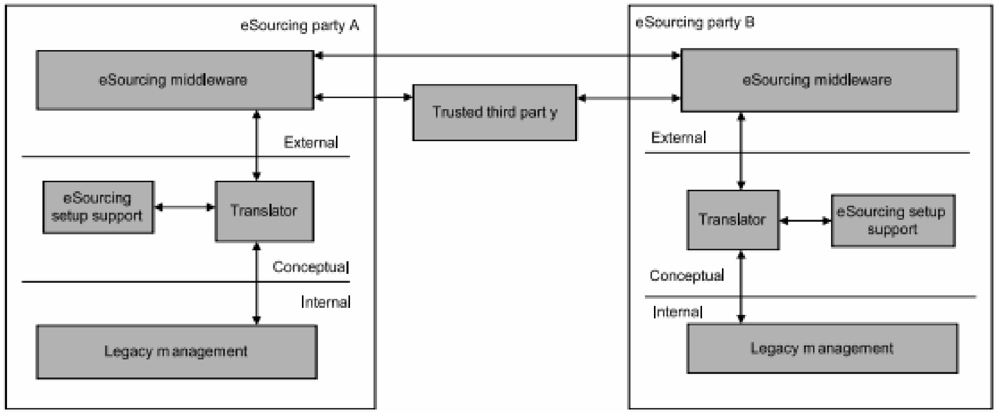
- Three matters in an architecture (ISO/IEC Standard 42010 definition)
  - **Elements**
  - **Relationships** between elements
  - **Principles** in the design and evolution of these interrelated elements



Conceptual model of an architecture description proposed by the ISO/IEC/IEEE 42010:2011 standard.

Software Architecture and software architect

- **Software architecture** is a set of architectural elements of the system
- **The relationship between the elements** is how the elements work together
- **Architecture** is used to understand and reason about the behavior, structure, quality attributes and geographic deployment
- **Functionality** of the system to meet the user requirements, while the cost the time, also need to be considered.
- **Quality attributes**, such as performance, security, …. Are important in the software architecture design.
- **Elegance** is the architecture easy to understand and modify
- Ex. A System architecture

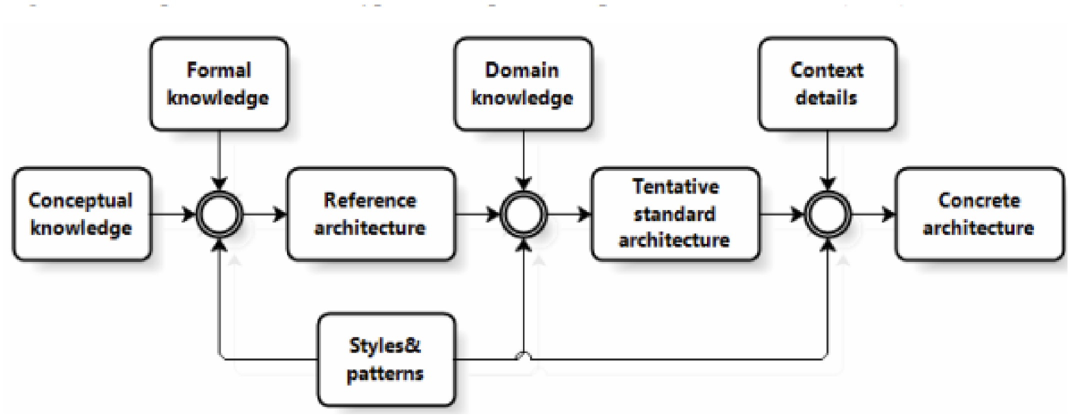


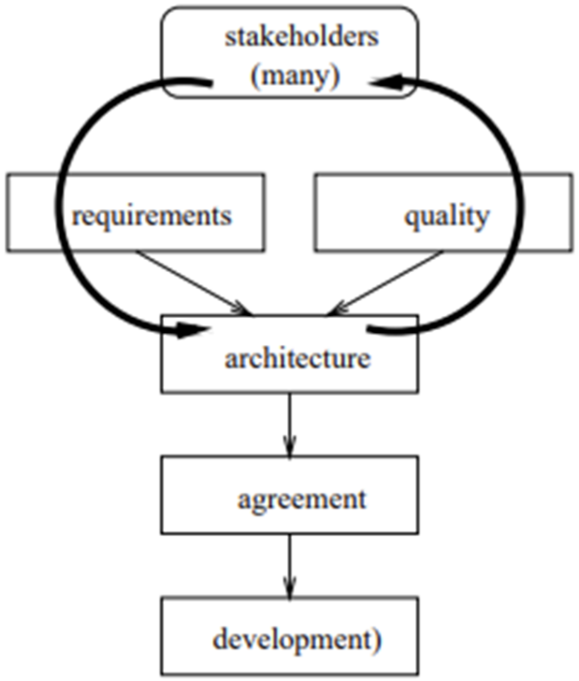
Software Architecture and software architect

- ❖ When you work as an **architect**, you need to consider different components of the system and need to know how they work together.
  - ➢ Software architect needs to focus on the **big picture and** the whole system.
  - ➢ **While software developers** need to understand deeply **one or two components**,
  - ➢ Architects need to understand **many components** but at a high level
  - ➢ The responsibility of an architect is beyond the design and architect but is also responsible for the **performance** of the system.
  - ➢ Architect is an **evaluator** and needs to provide continuous feedback.
  - ➢ Architects need to **continuously learn new things**
  - ➢ Architects need to **review the plan** of software development and guide developers.

Software Architecture and software architect

Developing software architecture requires broader knowledge

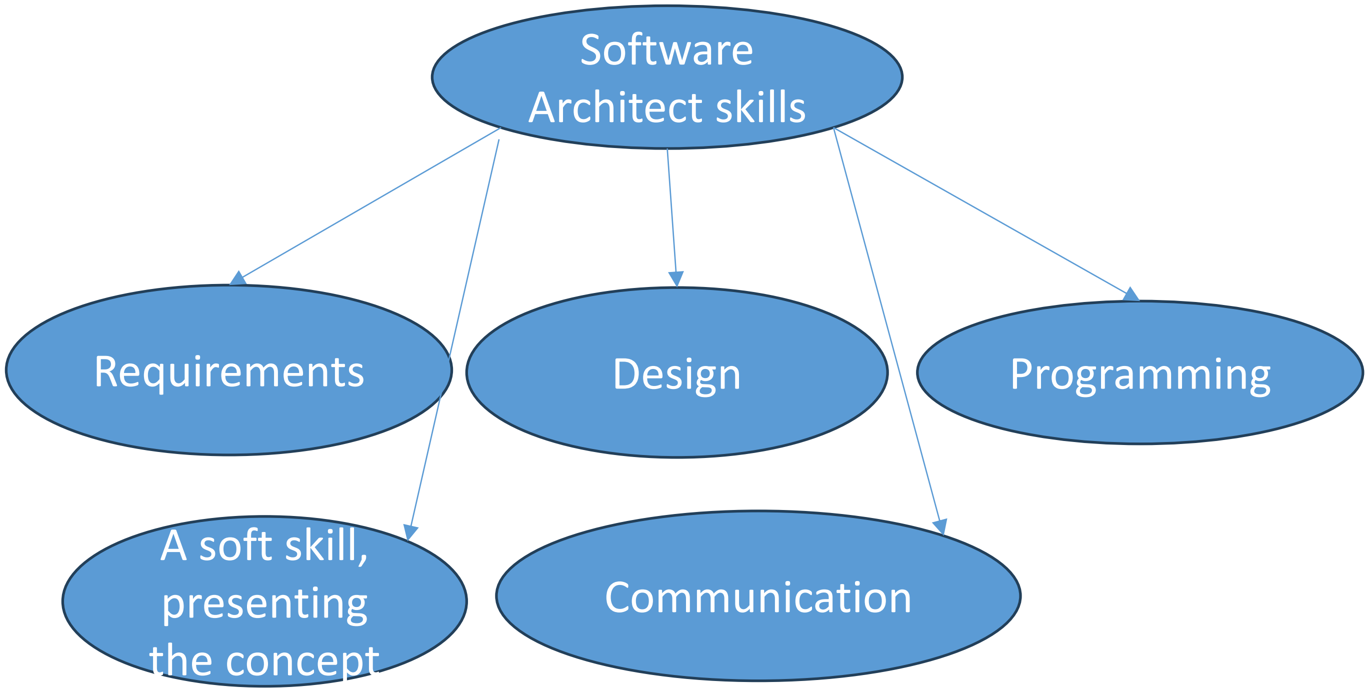




Software Architecture and software architect

Software architecture is not just putting together existing packages as if they were building blocks.

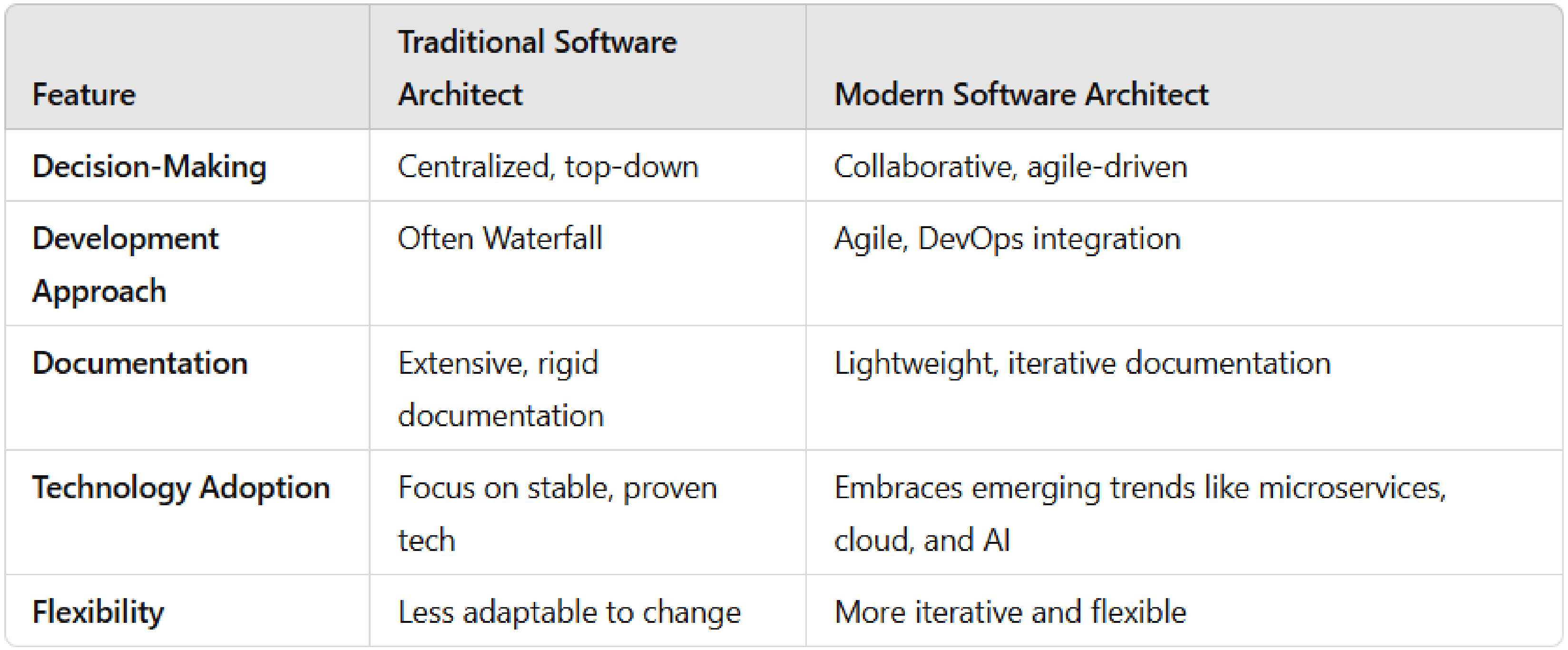
In programming, you learn simplicity, maintainability and testability


Software Architect skills
Requirements
Design
Programming
A soft skill, presenting the concept
Communication

Software Architecture and software architect

Traditional architects

Modern architects



Feature	Traditional Software Architect	Modern Software Architect
Decision-Making	Centralized, top-down	Collaborative, agile-driven
Development Approach	Often Waterfall	Agile, DevOps integration
Documentation	Extensive, rigid documentation	Lightweight, iterative documentation
Technology Adoption	Focus on stable, proven tech	Embraces emerging trends like microservices, cloud, and AI
Flexibility	Less adaptable to change	More iterative and flexible

Conclusions