SE 3112

SOFTWARE ARCHITECTURE

Prepared by: Dams Gabriel Lazarus
ICT University, Fall 2022

CHAPTER 3:

Architecture Viewpoints and Quality Attributes

Topic Overview

- What is Architectural View point?
- 4+1 View Model
- Relationships between view points
- Software Quality Attributes (SQAs)
  - Non-functional Requirements
- Categories of SQAs
- Ranking of SQAs

What is Architectural Viewpoint?

- It is a set of representations of an architecture that covers stakeholder issues.
- It also represents functional and non-functional requirements of software application.
- It means different things to different stakeholders. For example
  - A network engineer would be interested in the hardware and network configuration of the system
  - A project manager – in the key component to be developed and their timelines
  - A developer – in classes that make up a component
  - A tester – in testable scenarios etc..

What is Architectural Viewpoint?

- Architectural models of a software system can be used to focus discussion about the software requirements

- Architectures may be documented from several views

- A view describe the structure of a system and Architectural views may be used to document a design so that it can be used as a basis for more detailed design and implementation

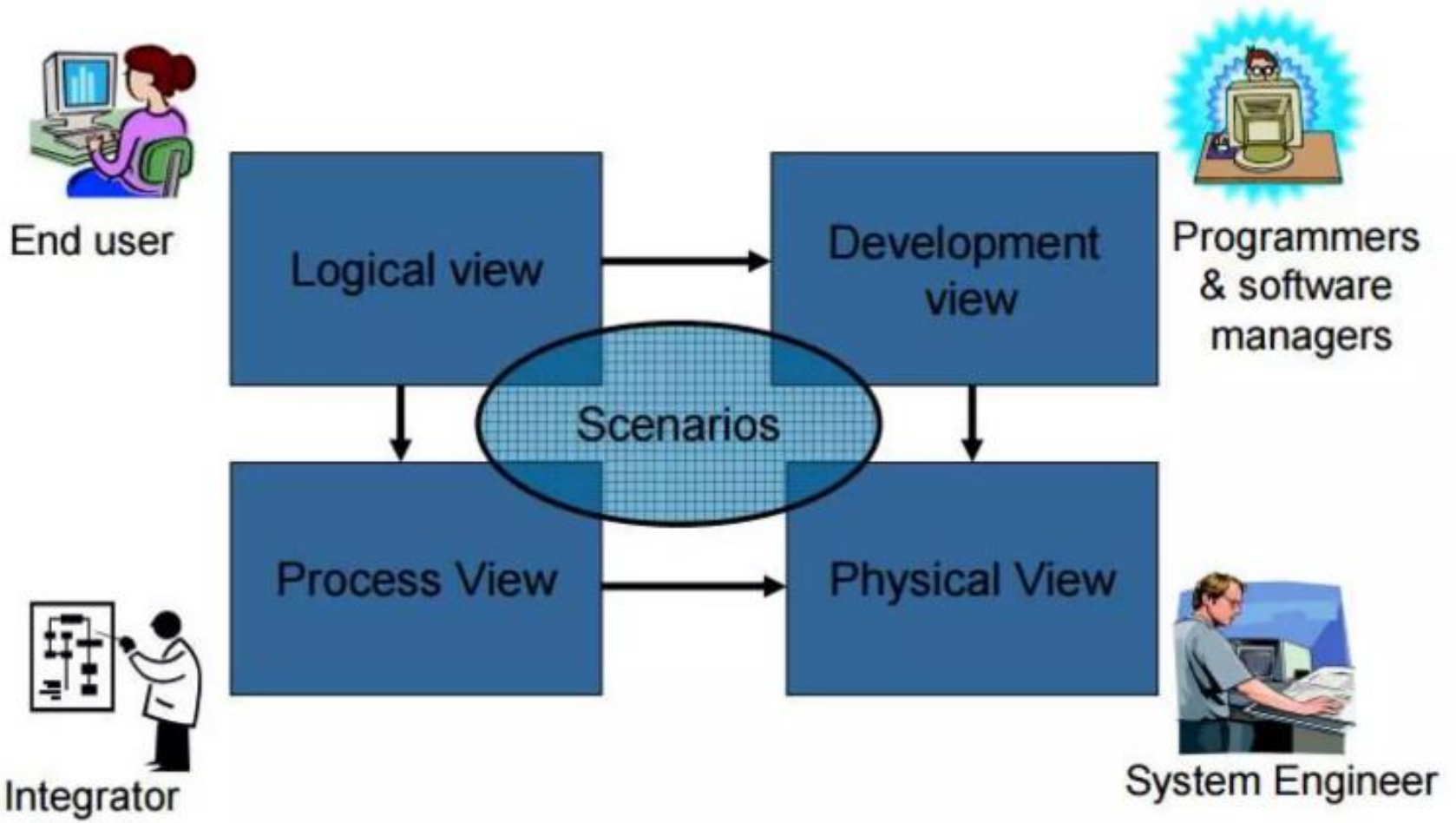
Concepts used in Architectural Views

- Two concepts are used in Architectural views
  - What views or perspectives are useful when designing and documenting a systems architecture? (VIEWS)
  - What notations should be used for describing architectural model? (Structure)

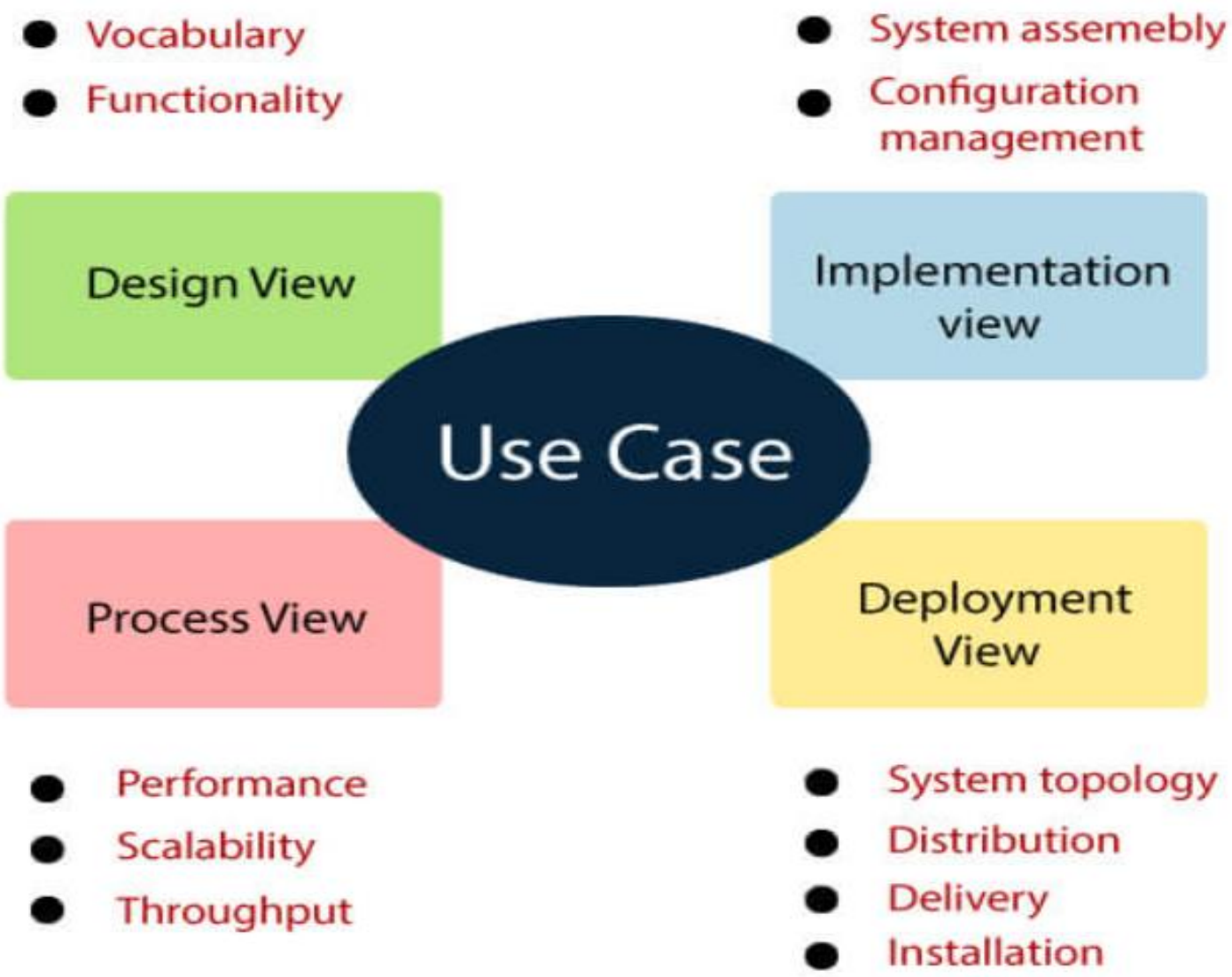
"4+1" View Model

- The 4+1 model views are used to describe the system from the viewpoint of different stakeholders, such as end-users, developers, project manager etc..
- It describe different aspects of the system.
- It provides a better organization with better separation of concern.

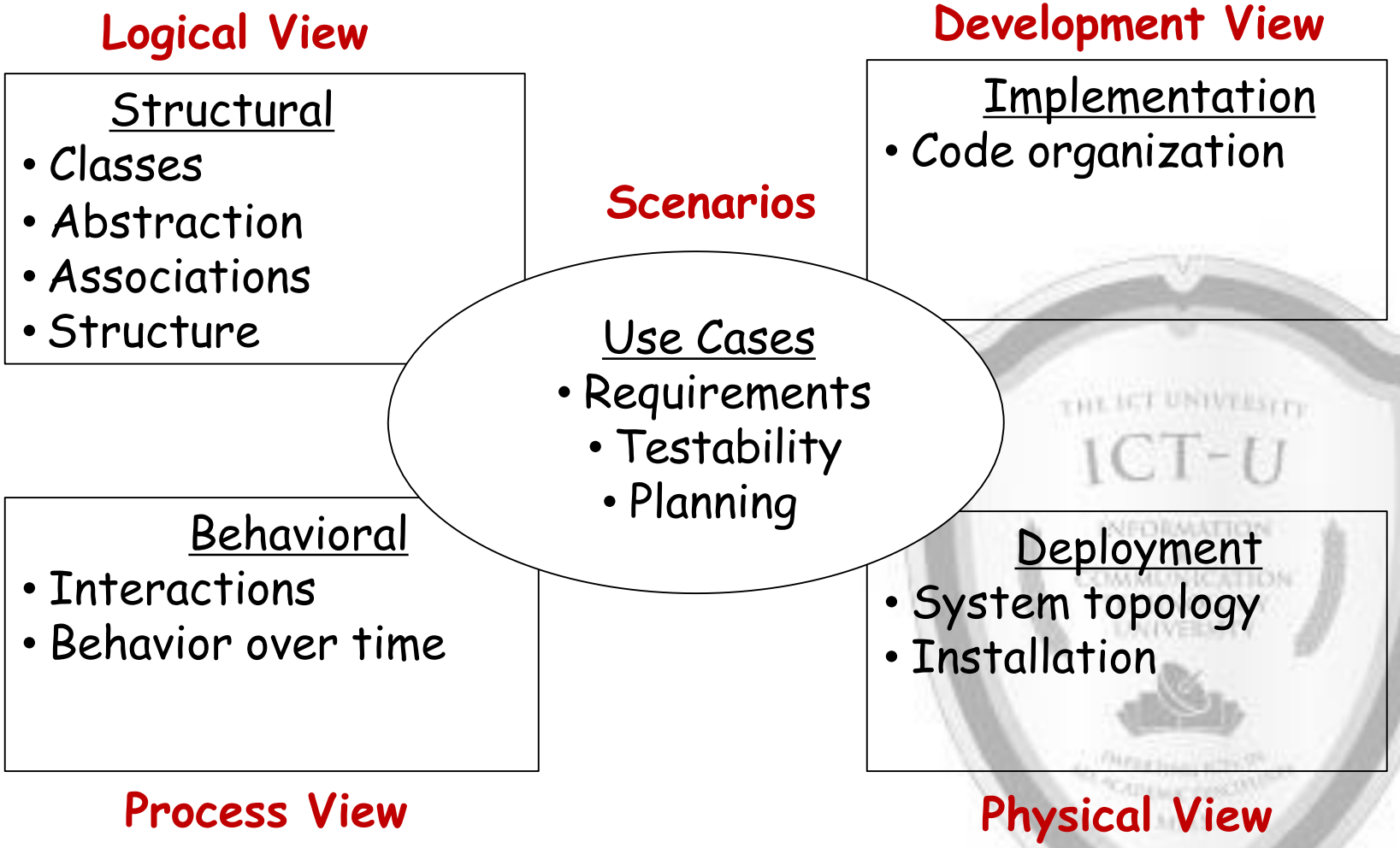
"4+1" View Model



"4+1" View Model

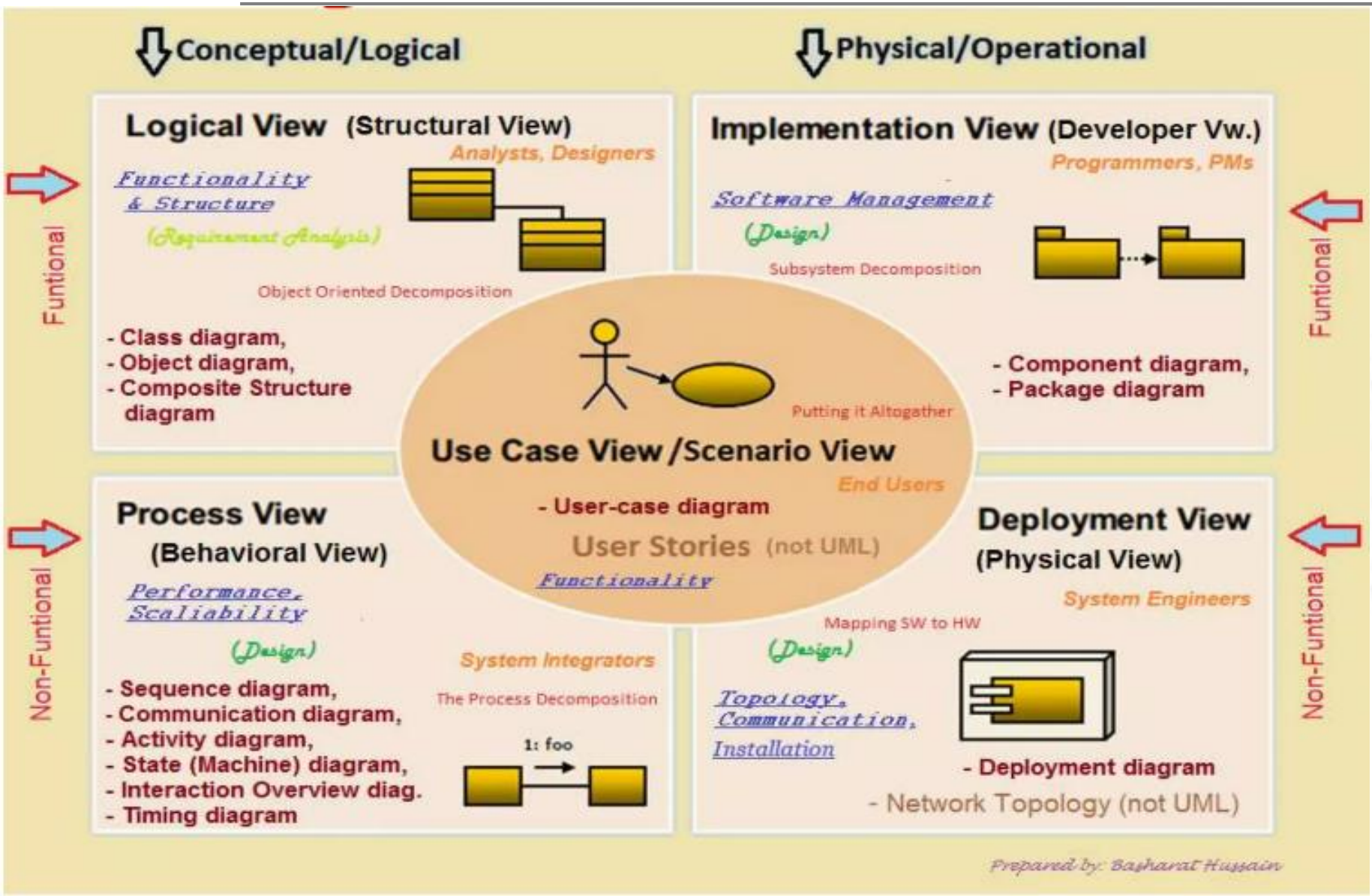


"4+1" View Model



Multiple View (4 + 1) Model

"4+1" View Model



1. Logical View

- **Audience**: End-User, Analyst, Designers
- **Considers**: Functional Requirements
- **What this view shows?**
  - Shows the components (objects) of the system as well as their interaction/relationship
- **UML diagrams**: Class, Object, State Machine, Composite Structure diagrams

2. Process View

- **Audience**: System Integrators(s)
- **Considers**: Non-functional requirements
- **What this view shows?**
  - Shows processes of the system and how those processes communicate with each other
- **UML diagrams**: Sequence, Communication, Activity, Timing, Interaction diagrams

3. Development View

- **Audience**: Programmers and Software Managers
- **Considers**: Software module organization
- **What this view shows?**
  - Shows the building blocks of the system
- **UML diagrams**: Component, Package diagrams

4. Physical View

- **Audience**: System Engineers
- **Considers**: Non-functional requirements for hardware (Topology, Communication)
- **What this view shows?**
  - Shows the system execution environment.
- **UML diagrams**: Deployment diagrams
- **Non-UML diagrams**: Network Topology (not in UML)

5. Use-case View

- **Audience**: All users of other views and evaluators
- **Considers**: System consistency, validity
- **What this view shows?**
  - Shows the validation and illustration of the system completeness. This view is redundant with other views
- **UML diagrams**: Use-case diagram, User stories.

Relationships between Views

- The Logical view and Process views are at a conceptual level and are used from analysis to design
- The Development and Deployment views are at the physical level and represent the actual application components built and deployed
- The Logical and Development views realizes the non-functional aspects using behavioral and physical modeling

Relationships between Views

- Use Case view leads to structural elements being analyzed in the Logical view and implemented in the Development view.

- The scenarios in the Use-Case view are realized in the Process View and deployed in the Physical View.

Why is it called "4+1" instead of Just 5?

- The Use-Case view: The use case view has a special significance.
- Views are effectively redundant (i.e. views are interconnected)
- However, all other views would not be possible without use case view.
- It details the high levels requirements of the system
- Other views detail how those requirements are realized.

Software Quality Attributes (SQA)

- When looking at any requirements other than the core functionalities of system, always look out for quality attributes a.k.a non functional requirements (NFRs).
- SQAs are features that facilitate the measurement of performance of a software product by Software Testing professionals.
- It helps in evaluating the software overall performance.

Software Quality Attributes (SQA)

- Overall factors that affects run-time behavior, system design and user experience.
- Software quality is the degree to which a software possesses a desired combination of attributes (IEEE 1061)
- Architecture cannot achieve quality by itself.
- Architecture should include factors of interest for each attribute.

Common Quality Attributes Category

- Quality attributes (NFRs) are categorized into various specific areas:
  - Design qualities
  - Runtime qualities
  - System qualities
  - User qualities
  - Non-runtime qualities
  - Architecture qualities
  - Business qualities

Design Quality Attributes

- Conceptual Integrity:
  - Defines the consistency and coherence of the overall design
- Maintainability:
  - Ability of the system to undergo changes with a degree of ease
- Reusability:
  - Defines the capability for components and subsystems to be suitable for use in other applications.

Runtime Quality Attributes

- Interoperability:
  - Ability of a system(s) to operate successfully by communicating and exchanging information with other external systems written and run by external parties
- Manageability:
  - Defines how easy it is for system administrators to manage the application
- Reliability:
  - Ability of a system to remain operational over time.

Runtime Quality Attributes

- Availability:
  - Proportion of time that the system is functional and working whenever needed.

System Quality Attributes

- Supportability:
  - Ability of a system to provide information helpful for identifying and resolving issues when it fails to work correctly

- Testability:
  - Measure of how easy it is to create test criteria for the system and its components

User Quality Attributes

- Usability:
  - Defines how well the application meets the requirement of the user.
  - Measures how easy user interfaces are to use.

Non-runtime Quality Attributes

- Portability:
  - Ability of a system to run under different computing environments
- Reusability:
  - Degree to which existing applications can be reused in new applications
- Integrability:
  - Ability to make the separately developed components of the system work correctly together.

Non-runtime Quality Attributes

- Modifiability:
  - Ease with which a software system can accommodate changes to its software

Architecture Quality Attributes

- Correctness:
  - Accountability for satisfying all requirements of the system

Business Quality Attributes

- Cost and Schedule:
  - Cost of the system with respect to time to market, expected project lifetime and utilization of legacy systems

- Marketability:
  - Use of the system with respect to market competition

  - Check for more:
    - https://en.wikipedia.org/wiki/List_of_system_quality attributes

Ranking Quality Attributes (NFRs)

- Ranking of the quality attributes depends on the stake holder you are communicating with.

- Customer Perspective
  - Usability > Security

- Security Perspective
  - Security > Usability

- As an architect, you must always find a balance between stakeholders

Q & A