Good or Bad?

* Software architecture is not inherently good or bad
* Designed to address a set of requirements that are used to **address a problem or need**
* The **context** on where you used it will determine the quality of your architecture
* Your system will most likely use a **combination** of architectural designs

2 types of Requirements

1. Functional
2. Non-functional Requirements – not always clear

Dev team will care about

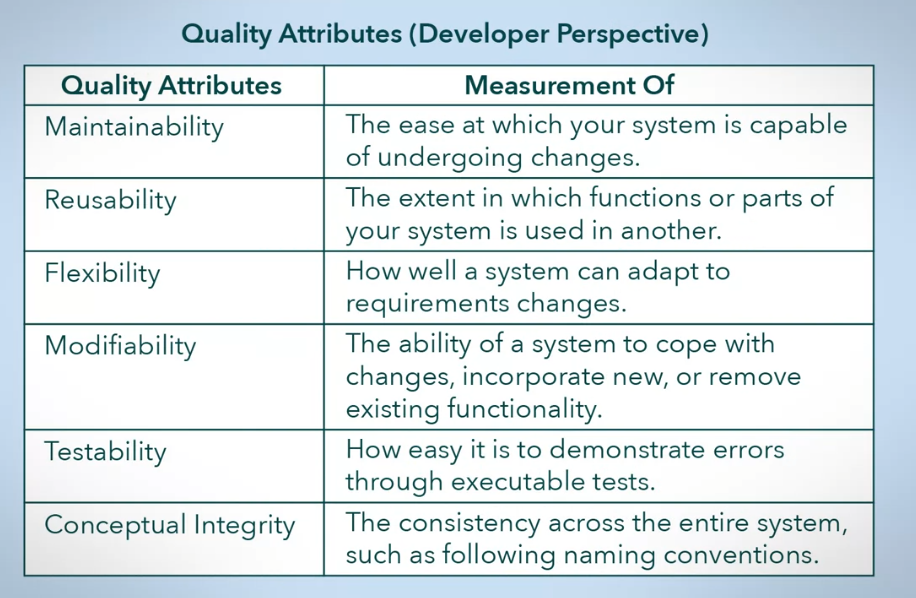
* Maintainability
* Reusability
* Testability
* Supportability

But an end user will care about

* Ease of Use
* Error Handling
* System Stability

Quality Attributes

* Measurable properties of a system used to gauge:
  + system’s design
  + run time performance
  + usability

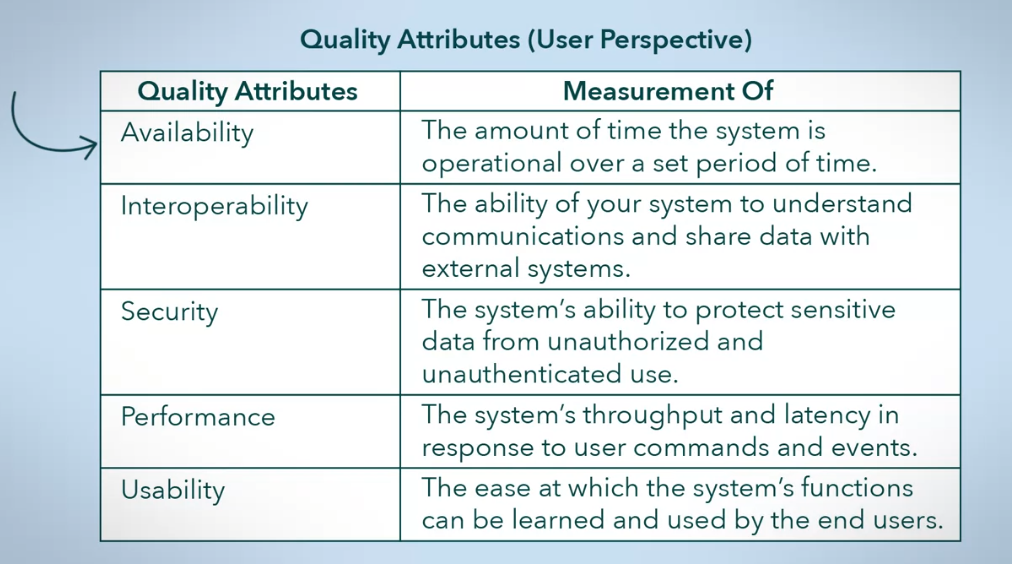


Quality Attributes (Developer’s Perspective)

1. Maintainability
   * How easy systems can undergo change
2. Reusability
   * Take functionality or parts of the system and use it in another one
3. Flexibility
   * Ability to adapt to future requirement changes in a timely and cost-efficient manner
4. Modifiability
   * Ease at which your system is able to handle changes to functions, incorporating new functionality, or remove existing ones
   * Questions to ask:
     1. What can change?
     2. Who will be making these changes?
     3. What is the possibility of changes happening?
5. Testability
   * How easy it is to demonstrate errors through executable tests
6. Conceptual Integrity
   * Consistency across the entire system (e.g., naming conventions)

How well a system performs in a user perspective?

1. Is it able to meet requirements?
2. Is it intuitive to use?
3. Does it perform its functionality in a timely manner?
4. Are the outputs of the system, correct?
5. How well can it handle errors?



Quality Attributes (User’s Perspective)

1. Availability
   * The amount of time the system is operational over a set period of time
   * How fast can the system handle the following:
     1. System errors
     2. High loads
     3. Updates
2. Interoperability
   * Ability to understand interfaces and use them to exchange information under specific conditions with external system
     1. Communication Protocols
     2. Data Formats
     3. With Whom to Exchange Information
3. Security
   * How well your system is protected from unauthorized access and use
   * Should provide **data integrity**
     1. Data integrity – controlling who can see the data versus who can also change the data
4. Performance
   * How well your system is able to response to a user command or system event
   * 2 attributes:
     1. Throughput
        + Amount of output produced over a period of time
     2. Latency
        + Time it takes to produce an output after receiving an input
   * Lower price per unit of performance
5. Usability
   * Ease at which the system’s functions can be learned and used by the end users
   * Should be:
     1. Easy and intuitive to learn
     2. Minimize user errors
     3. Provide feedback to the user
     4. Make it easy to complete tasks

How do we go about designing a high-quality system?

* + Create or choose an appropriate architectural design for your system
    1. Makes maintaining, supporting, and updating the system throughout its life cycle much easier
  + A high-quality system does not need to be ‘complex’
  + Have detailed and up-to-date documentation
  + Use a set of rules, or guidelines for the design process and how your system will be structured
  + Recognizing the importance of **quality attributes** (more on this later) and prioritizing them
  + Involving a technical lead in the design process
  + Taking a design approach from the perspective of different groups of stakeholders

Some examples of structural rules:

1. Having well defined subsystems that are assigned responsibility based on design principles
2. Having consistent implementations of functions across the entire system
3. Having a set of rules on how resources are used

Summary

* It is a matter of selecting the **appropriate** architectural solution for your problem
* You should consider
  + Involving all groups of stakeholders so all concerns are heard and addressed
  + Adopting good documentation practices
  + Setting rules for design and implementation
* A properly designed system will consider equality attributes address:
  + Developer’s perspective
    - Maintainability
    - Reusability
    - Flexibility
    - Modifiability
    - Testability
    - Conceptual Integrity
  + User’s perspective
    - Availability
    - Interoperability
    - Security
    - Performance
    - Usability