Chapter 4 - Architectural Characteristics

Introduction

- Architects may collaborate on **defining the domain** or **business requirements**, but one key responsibility entails defining, discovering, and otherwise analyzing all the things the software must do that isn'e directly related to the domain functionality: **Architectural Characteristics**.

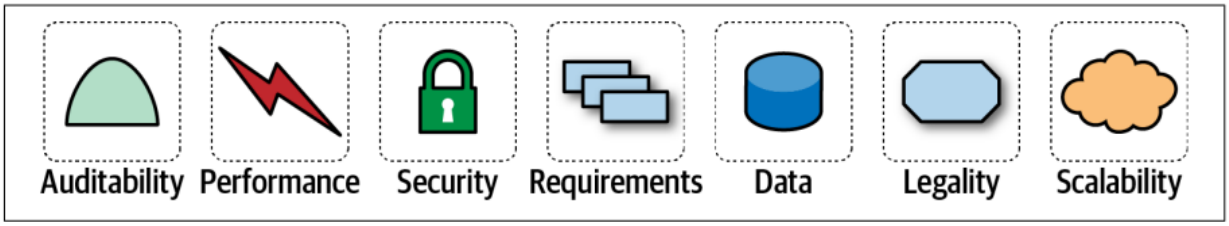

Figure 4-1. A software solution consists of both domain requirements and architectural characteristics

Features of an Architecture Characteristic

An architecture characteristic meets three criteria:

1. Specifies a **nondomain design consideration**
2. Influences some **structural aspect of the design**.
3. Is critical or important to **application success**

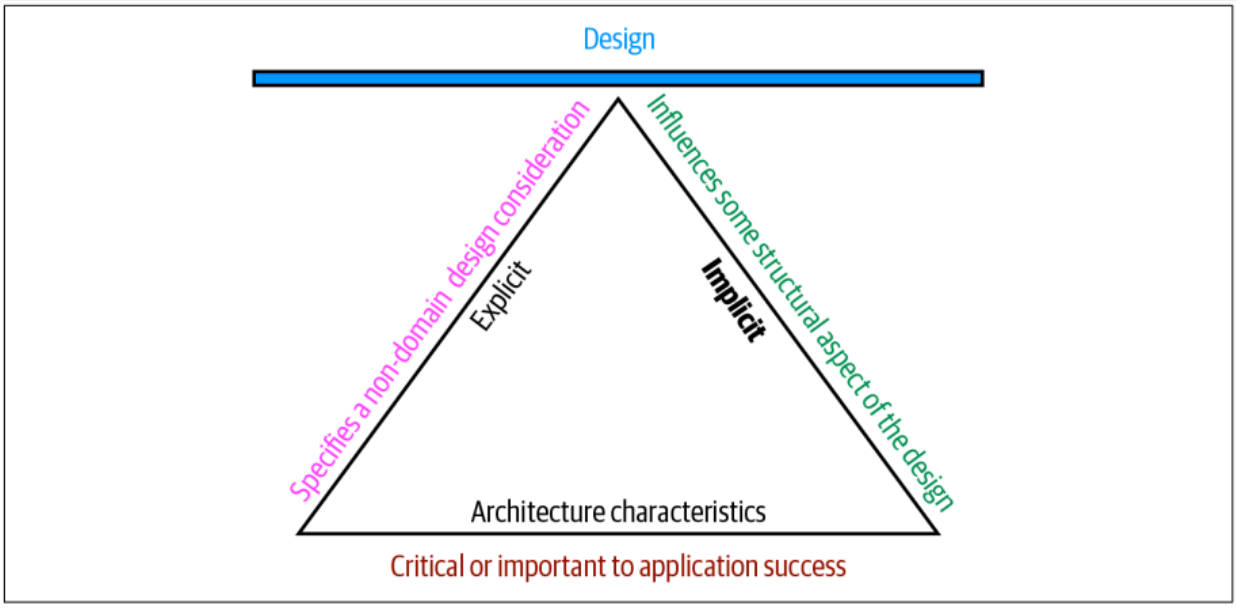

Figure 4-2. The differentiating features of architecture characteristics

1. Specifies a Nondomain Design Consideration

When designing an application:

1. **The requirements** specify what the application should do.
2. **Architecture Characteristics** specify operational and design criteria for success, concerning **how to implement the requirements** and **why certain choices were made**.
3. No requirements document states "prevent technical debt" or "have high performance" but it is a common design consideration for architects and developers.

2. Influences Some Structural Aspect Of the Design

- The **primary reason** architects try to describe architecture characteristics on projects concerns **design considerations**: *Does this architecture characteristic require special structural consideration to success?*

3. Critical or Important to Application Success

- Applications **could** support a huge number of architecture characteristics, but **shouldn't**.
- This is because support for each architecture characteristic **adds complexity to the design**.
- Thus the art of the architect is in choosing **the fewest** architecture characteristics **rather than the most possible**.

> **Note**
> Each of the definition elements supports the others, which in turn support the overall design of the system.

Architectural Characteristics (Partially) Listed

- No true universal standard exists for the definitions and listing of architectural characteristics, but we will try to list them here.
- One reason for this is that the software ecosystem changed so fast.
- Despite the volume and scale, architects commonly separate architecture characteristics into 3 broad categories:
    1. **Operational**
    2. **Structural**
    3. **Cross-Cutting**

Operational Architecture Characteristics

Table 4-1. Common operational architecture characteristics

Term	Definition
Availability	How long the system will need to be available (if 24/7, steps need to be in place to allow the system to be up and running quickly in case of any failure).
Continuity	Disaster recovery capability.
Performance	Includes stress testing, peak analysis, analysis of the frequency of functions used, capacity required, and response times. Performance acceptance sometimes requires an exercise of its own, taking months to complete.
Recoverability	Business continuity requirements (e.g., in case of a disaster, how quickly is the system required to be on-line again?). This will affect the backup strategy and requirements for duplicated hardware.
Reliability/ safety	Assess if the system needs to be fail-safe, or if it is mission critical in a way that affects lives. If it fails, will it cost the company large sums of money?
Robustness	Ability to handle error and boundary conditions while running if the internet connection goes down or if there's a power outage or hardware failure.
Scalability	Ability for the system to perform and operate as the number of users or requests increases.

> **Note**
>
> Operational architecture characteristics heavily overlap with operations and DevOps concerns.

Structural Architecture Characteristics

Table 4-2. Structural architecture characteristics

Term	Definition
Configurability	Ability for the end users to easily change aspects of the software's configuration (through usable interfaces).
Extensibility	How important it is to plug new pieces of functionality in.
Installability	Ease of system installation on all necessary platforms.
Leverageability/ reuse	Ability to leverage common components across multiple products.
Localization	Support for multiple languages on entry/query screens in data fields; on reports, multibyte character requirements and units of measure or currencies.
Maintainability	How easy it is to apply changes and enhance the system?
Portability	Does the system need to run on more than one platform? (For example, does the frontend need to run against Oracle as well as SAP DB?
Supportability	What level of technical support is needed by the application? What level of logging and other facilities are required to debug errors in the system?
Upgradeability	Ability to easily/quickly upgrade from a previous version of this application/solution to a newer version on servers and clients.

Cross-Cutting Architecture Characteristics

Table 4-3. Cross-cutting architecture characteristics

Term	Definition
Accessibility	Access to all your users, including those with disabilities like colorblindness or hearing loss.
Archivability	Will the data need to be archived or deleted after a period of time? (For example, customer accounts are to be deleted after three months or marked as obsolete and archived to a secondary database for future access.)
Authentication	Security requirements to ensure users are who they say they are.
Authorization	Security requirements to ensure users can access only certain functions within the application (by use case, subsystem, webpage, business rule, field level, etc.).
Legal	What legislative constraints is the system operating in (data protection, Sarbanes Oxley, GDPR, etc.)? What reservation rights does the company require? Any regulations regarding the way the application is to be built or deployed?
Privacy	Ability to hide transactions from internal company employees (encrypted transactions so even DBAs and network architects cannot see them).
Security	Does the data need to be encrypted in the database? Encrypted for network communication between internal systems? What type of authentication needs to be in place for remote user access?
Supportability	What level of technical support is needed by the application? What level of logging and other facilities are required to debug errors in the system?
Usability/ achievability	Level of training required for users to achieve their goals with the application/solution. Usability requirements need to be treated as seriously as any other architectural issue.

> **Important**
>
> Any list of architecture characteristics will necessarily be an incomplete list; any software **may invent important characteristics** based on unique factors.

ISO's Definition of Architecture Characteristics

- **Performance Efficiency**: Measure of the performance relative to the amount of resources used under known conditions.
  - *Time Behavior*: Measure of response, Processing time, Throughput rates
  - *Resource Utilizations*: Amounts and types of resources used
  - *Capacity*: Degree to which the maximum established limits are exceeded.
- **Compatibility**:
  - *Coexistence*: Can perform its required functions efficiently while sharing a common environment and resources with other products
  - *Interoperability*: Degree to which two or more systems can exchange and utilize information
- **Usability**: Users can use the system effectively
  - *Appropriate Recognizability*: Users can recognize whether the software is appropriate for their needs

    - *Learnability*: How easily users can learn how to use the software
    - *User Error Protection*: Protection against users making errors
    - *Accessibility*: Make the software available to people with the widest range of characteristics and capabilities
- **Reliability**: Degree to which a system functions under specified conditions for a specified period of time.
    - *Maturity*: Does the software meet the reliability needs under normal operation
    - *Availability*: Software is operational and accessible
    - *Fault Tolerance*: Does the software operate as intended despite hardware or software faults
    - *Recoverability*: Can the soft- ware recover from failure by recovering any affected data and reestablish the desired state of the system.
- **Security**: Degree the software protects information and data so that people or other products or systems have the degree of data access appropriate to their types and levels of authorization
    - *Confidentiality*: Data is accessible only to those authorized to have access
    - *Integrity*: The software prevents unauthorized access to or modification of software or data
    - *Nonrepudiation*: Can actions or events be proven to have taken place
    - *Accountability*: Can user actions of a user be traced
    - *Authenticity*: Proving the identity of a user
- **Maintainability**: Represents the degree of effectiveness and efficiency to which developers can modify the software to improve it
    - *Modularity*: Degree to which the software is composed of discrete components
    - *Reusability*: Degree to which developers can use an asset in more than one system or in building other assets
    - *Analyzability*: How easily developers can gather concrete metrics about the software
    - *Modifiability*: Degree to which developers can modify the software without introducing defects or degrading existing product quality
    - *Testability*: How easily developers and others can test the software
- **Portability**: Degree to which developers can transfer a system, product, or component from one hardware, software, or other operational or usage environment to another
    - *Adaptability*: Can developers effectively and efficiently adapt the software for different or evolving hardware, software, or other operational or usage environments

    - *Installability*: Can the software be installed and/or uninstalled in a specified environment
    - *Replaceability*: How easily developers can replace the functionality with other software
- **Functional Suitability**: The degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions.
  - *Functional Completeness*: Degree to which the set of functions covers all the specified tasks and user objectives.
  - *Functional Correctness*: Degree to which a product or system provides the correct results with the needed degree of precision.
  - *Functional Appropriateness*: Degree to which the functions facilitate the accomplishment of specified tasks and objectives.

> **Note**
> These functional requirements are not architecture characteristics but rather the motivational requirements to build the software. This illustrates how thinking about the relationship between architecture characteristics and the problem domain has evolved.

Trade-Offs and Least Worst Architecture

Applications can only support a few of the architecture characteristics.

1. First, each of the supported characteristics requires design effort and perhaps structural support.
2. Second the bigger problem lies with the fact that each architecture characteristic often has an impact on others.
   - For example, if an architect wants to improve security, it will almost certainly negatively impact performance.

> **Important**
> - Never shoot for the best architecture, but rather shoot for the **least worst architecture**.

- Too many architecture characteristics leads to generic solutions that are trying to solve every business problem, and those architectures rarely work because the design become unwieldy.