The Learning House

427 S 4th Street #300

Louisville KY 40202

What is Software Architecture?

.NET Cohort

Coding Bootcamp

SOFTWARE GUILD

Software Architecture Defined

A software architect is responsible for the acknowledgement of *functional and non-functional requirements* and making decisions about the implementation of those requirements.

A software architect is concerned with the organization of the system such that it satisfies the stakeholder's concerns.

What's a stakeholder?

A stakeholder is any individual interested or concerned about the software system. Simply put, just about anyone can be a stakeholder.

Some examples of stakeholders are end users, business analysts, executive sponsors, developers, and testers.

Different stakeholders sometimes value requirements differently. It is often the duty of the software architect to document the tradeoffs between requirements and drive consensus within the organization.

Acknowledging Requirements

Every system has a purpose… a mission. This purpose is expressed through requirements and these requirements are what drive a system's architecture.

Requirements can be functional or non-functional.

Functional vs. Nonfunctional

In the simplest sense:

A functional requirement is a behavior the system must perform in a given scenario.

A non-functional requirement is an attribute of the system requested by stakeholders.

About Functional Requirements

A functional requirement is often described in terms of input, behavior (rules) and output. At the minimum, it must document the expected behavior.

A good requirement will also document what to do in case of unexpected behaviors.

I often like to think of it as answering the questions of *what, how, and when?*

Example of a Functional Requirement

When a customer cancels their account:

1. Cancel all future invoices
2. Calculate whether there is an overpay
   - If so, queue a refund check
3. Send a confirmation email to the customer

About Non-Functional Requirements

While functional requirements are mostly about code and implementation, non-functional requirements are about architecture choices & aspects of the system that are difficult to change later.

Non-functional requirements will often drive things like language choice, frameworks, hardware setup, etc.

Some Non-Functional Requirements

- Scalability — The ability to provide good performance as the amount of work grows.
- Security — How access to system resources is granted.
- Reliability — Capability of the software to maintain performance under special circumstances.
- Portability — Can the software be used in multiple platforms or coexist with other software in the same environment?

It Depends…

Oftentimes when you ask an architect a brief question, you will get this infuriating “it depends” answer. It really does depend though.

For example:

How should your internet based application behave in the case of poor connectivity?

- Should it fail?
- Should it wait and retry?
- Should it show an error or wait message?
- Should it switch to “offline mode” and used cached data?

The answer to these questions will have significant impacts on the system architecture in terms of reliability and recoverability!

Gathering Requirements

A good requirement…

- Addresses one and only one thing
- Is clear and unambiguous
- Has a clear stakeholder
- Is not obsolete

However, no matter how much effort you put in, things will always be withheld, omitted, forgotten, or simply expressed incorrectly.

A Successful Architect Speaks the Domain

It is impossible to successfully architect a system if the architect does not fully understand the business domain.

We can’t expect business people to understand the world of tables, code, processes, and protocols, so we must make the necessary effort to learn the language of the business domain!

Architecture is About the Hard Decisions

A big mistake that newer developers make in architecture is trying to define every aspect and feature in advance.

I believe that time should be focused on only those things that are difficult to change later.