Iterative & Incremental Approach

It is an iterative and incremental approach consisting of five main steps that helps to generate candidate solutions. This candidate solution can further be refined by repeating these steps and finally create an architecture design that best fits our application. At the end of the process, we can review and communicate our architecture to all interested parties.

It is just one possible approach. There are many other more formal approaches that defining, reviewing, and communicating your architecture.

Identify Architecture Goal

Identify the architecture goal that forms the architecture and design process. Flawless and defined objectives emphasize on the architecture, solve the right problems in the design and helps to determine when the current phase has completed, and ready to move to the next phase.

This step includes the following activities –

1. Identify your architecture goals at the beginning
   1. Identify the consumer of our architecture
   2. Identify the constraints

Examples of architecture activities include building a prototype to get feedback on the order-processing UI for a Web application, building a customer order-tracking application, and designing the authentication, and authorization architecture for an application in order to perform a security review.

Key Scenarios

This step puts emphasis on the design that matters the most. A scenario is an extensive and covering description of a user's interaction with the system.

Key scenarios are those that are considered the most important scenarios for the success of your application. It helps to make decisions about the architecture. The goal is to achieve a balance among the user, business, and system objectives. For example, user authentication is a key scenario because they are an intersection of a quality attribute security with important functionality how a user logs into your system.

Application Overview

Build an overview of application, which makes the architecture more touchable, connecting it to real-world constraints and decisions. It consists of the following activities –

Identify Application Type

Identify application type whether it is a mobile application, a rich client, a rich internet application, a service, a web application, or some combination of these types.

Identify Deployment Constraints
Choose an appropriate deployment topology and resolve conflicts between the application and the target infrastructure.

**Identify Important Architecture Design Styles**

Identify important architecture design styles such as client/server, layered, message-bus, domain-driven design, etc. to improve partitioning and promotes design reuse by providing solutions to frequently recurring problems. Applications will often use a combination of styles.

**Identify the Relevant Technologies**

Identify the relevant technologies by considering the type of application we are developing, our preferred options for application deployment topology and architectural styles. The choice of technologies will also be directed by organization policies, infrastructure limitations, resource skills, and so on.

**Key Issues or Key Hotspots**

While designing an application, hot spots are the zones where mistakes are most often made. Identify key issues based on quality attributes and crosscutting concerns. Potential issues include the appearance of new technologies and critical business requirements.

Quality attributes are the overall features of your architecture that affect run-time behavior, system design, and user experience. Crosscutting concerns are the features of our design that may apply across all layers, components, and tiers.

These are also the areas in which high-impact design mistakes are most often made. Examples of crosscutting concerns are authentication and authorization, communication, configuration management, exception management and validation, etc.

**Candidate Solutions**

After defining the key hotspots, build the initial baseline architecture or first high level design and then start to fill in the details to generate candidate architecture.

Candidate architecture includes the application type, the deployment architecture, the architectural style, technology choices, quality attributes, and crosscutting concerns. If the candidate architecture is an improvement, it can become the baseline from which new candidate architectures can be created and tested.

Validate the candidate solution design against the key scenarios and requirements that have already defined, before iteratively following the cycle and improving the design.

We may use architectural spikes to discover the specific areas of the design or to validate new concepts. Architectural spikes are a design prototype, which determine the feasibility of a specific design path, reduce the risk, and quickly determine the viability of different approaches. Test architectural spikes against key scenarios and hotspots.

**Architecture Review**

Architecture review is the most important task in order to reduce the cost of mistakes and to find and fix architectural problems as early as possible. It is a well-established, cost-effective way of reducing project costs and the chances of project failure.

- Review the architecture frequently at major project milestones, and in response to other significant architectural changes.
- The main objective of an architecture review is to determine the feasibility of baseline and candidate architectures, which verify the architecture correctly.
- Links the functional requirements and the quality attributes with the proposed technical solution. It also helps to identify issues and recognize areas for improvement.

Scenario-based evaluations are a dominant method for reviewing an architecture design which focuses on the scenarios that are most important from the business perspective, and which have the greatest impact on the architecture.
Following are common review methodologies –

- **Software Architecture Analysis Method (SAAM)** – It is originally designed for assessing modifiability, but later was extended for reviewing architecture with respect to quality attributes.

- **Architecture Tradeoff Analysis Method (ATAM)** – It is a polished and improved version of SAAM, which reviews architectural decisions with respect to the quality attributes requirements, and how well they satisfy particular quality goals.

- **Active Design Review (ADR)** – It is best suited for incomplete or in-progress architectures, which more focus on a set of issues or individual sections of the architecture at a time, rather than performing a general review.

- **Active Reviews of Intermediate Designs (ARID)** – It combines the ADR aspect of reviewing in-progress architecture with a focus on a set of issues, and the ATAM and SAAM approach of scenario-based review focused on quality attributes.

- **Cost Benefit Analysis Method (CBAM)** – It focuses on analyzing the costs, benefits, and schedule implications of architectural decisions.

- **Architecture Level Modifiability Analysis (ALMA)** – It estimates the modifiability of architecture for business information systems (BIS).

- **Family Architecture Assessment Method (FAAM)** – It estimates information system family architectures for interoperability and extensibility.

**Communicating the Architecture Design**

After completing the architecture design, we must communicate the design to the other stakeholders, which include development team, system administrators, operators, business owners, and other interested parties.

There are several following well-known methods for describing architecture to others: –

**4 + 1 Model**

This approach uses five views of the complete architecture. Among them, four views (the **logical view**, the **process view**, the **physical view**, and the **development view**) describe the architecture from different approaches. A fifth view shows the scenarios and use cases for the software. It allows stakeholders to see the features of the architecture that specifically interest them.

**Architecture Description Language (ADL)**

This approach is used to describe software architecture prior to the system implementation. It addresses the following concerns – behavior, protocol, and connector.

The main advantage of ADL is that we can analyze the architecture for completeness, consistency, ambiguity, and performance before formally beginning use of the design.

**Agile Modeling**

This approach follows the concept that “content is more important than representation.” It ensures that the models created are simple and easy to understand, sufficiently accurate, detailed, and consistent.

Agile model documents target specific customers and fulfill the work efforts of that customer. The simplicity of the document ensures that there is active participation of stakeholders in the modeling of the artifact.

**IEEE 1471**

IEEE 1471 is the short name for a standard formally known as ANSI/IEEE 1471-2000, “Recommended Practice for Architecture Description of Software-Intensive Systems.”
enhances the content of an architectural description, in particular, giving specific meaning to context, views, and viewpoints.

**Unified Modeling Language UML**

This approach represents three views of a system model. The *functional requirements view* functional requirements of the system from the point of view of the user, including use cases; the *static structural view* objects, attributes, relationships, and operations including class diagrams; and the *dynamic behavior view* collaborations among objects and changes to the internal state of objects, including sequence, activity, and state diagrams.