Importance of Architecture:

* An architecture enables a system’s driving quality attributes
* A documented architecture enhances communication between stakeholders
* An architecture shows the earliest and most hardest to change decisions and are probably the fundamental building block of that system
* It defines a set of constraints on subsequent implementation
* It defines the structure of an organization
* Helps to reason about cost and schedule
* Can provide the basis for evolutionary prototyping

Inhibiting or Enabling a system’s Quality attribute:

A good architecture alone does not ensure the quality of the attributes, this has to be considered during the entirety of the software development life cycle. For example: If your system requires high performance you pay extra attention to managing the time-based behavior of elements, if modifiability is important you pay extra attention to assigning responsibilities to elements to that a change in system has minimal effect (ideally to only one element), if your system is needed to be secure then you must pay attention to communication between elements and which element has access to which information.

Reasoning about and managing change:

This refers to modifiability which can be described as the ease with which changes can be made to a system. There are three main types of changes:

1. Local change: Can be modified by making changes to a single element.
2. Non local change: Consists of multiple modifications to different elements but the architecture remains intact.
3. Architectural Change: Effects fundamentally how elements interact with each other and will probably require changes all over the system.

All of this means that the local changes are the most desirable ones so an architecture is most effective when changes required to it are local.

Predicting System Qualities:

Architecture imbues a system with quality attributes in a predictable manner. If it was not possible to make appropriate architectural decisions without the system being built, then choosing an architecture would be based on random decisions. If we know that certain architectural decisions lead to certain quality attributes then we can make those decisions and expect a suitable outcome. This helps to find out problems in your design early on during the design analysis period.

Enhancing Communication Among Stakeholders:

It represents such an abstraction of the system that if not all then most stakeholders can make sense of. It is written in a way that even people with not as much technical knowledge, with little help from the architect, can understand it enough. Each stakeholder looks at the architecture differently. For example: the user might be worried about the responsiveness and speed of the system, the manager is worried about cost and schedule and team divisions. It provides these concerns in a non technical manner so that the different stakeholders can understand it and voice any concerns early on.

Carrying Early Design Decision:

Each decision constraints the decision that follows it which is why the earliest design choices have disproportionate weight because they influence and constraint so much of what comes after them. Changing these early decisions might have enormous ramifications and can cause a ripple effect in terms of the changes that need to be made, this can also result in refactoring the architecture itself. Example: Will the system run on one processor, how much memory is required, OS for the system, communication protocol to use etc.

Defining Constraints on an implementation:

The implementation must be done as a set of prescribed elements which interact with each other in a prescribed manner. Element builders are usually only aware about the specifications of the element but not the architectural trade-offs; conversely, the architects are not supposed to know everything about algorithm design and coding but know enough to not build an architecture that is impossible to implement and must be aware of the architectural trade-offs.

Influencing the Organizational Structure:

Different portions of the system are divided and assigned to different teams. This is called the work-breakdown structure. The architecture involves the broadest decomposition of the system so it forms the basis for the work-breakdown structure which in turn dictates scheduling, budget, planning, inter-team communication, communication lines between different teams. Integration, test plans etc. Thus the software architecture not only influences the structure of the system to be built but also the structure of the project and sometimes the structure of the entire organization.

Enabling Evolutionary Prototyping:

An architecture can be analyzed and prototyped as a skeletal system. A skeletal system is one in which the entire system is not complete but some parts of its infrastructure are present. This includes initializing elements, how they interact with each other, share data, access resources, activity logs, errors etc, all of this is built much before the system’s functionality. This process can be continued, building other parts of the infrastructure little by little. This aids the development cycle because the system is executable early in its development cycle resulting in increasing its fidelity and error detection early in its life.

Improving Cost and Schedule Estimates:

An architect must be able to help the project manager in determining the cost and scheduling estimates of the project early on in the development period. The work-breakdown structure of a project is always based on architecture and these are used to determine the cost and scheduling estimates. The best way is to use both the bottom up(created by developers) and top down (created by architect and project manager) approach and reach a consensus between the two approaches.

Supplying a transferrable, reusable model:

While reuse of code is helpful, reuse of architecture provides tremendous leverage for systems with similar requirements. Not only can the code be reused but also the requirements that led to the requirements and also the infrastructure and experience used to build that architecture. A software product line or family is a set of software that are built using the same assets. Chief among these assets is the asset that was built to facilitate the needs of the entire family. Product line architects choose an architecture that supports all the family members defining what is fixed for all members and what is variable

Allowing incorporation of independently developed components:

Architecture based development focuses on building separate elements and then integrating them together. This is possible because the architecture defines the elements, possible replacements, how they communicate with the environment, how they access data and what protocols they use for communication etc. Benefits include decreased cost, decreased time to market, increased reliability, and flexibility.

Restricting the vocabulary of design alternatives:

Even though software elements can be combined in almost infinite ways, there is something to be gained by restricting ourselves in terms of elements and their interactions. By doing so complexity of the design is reduced, enhanced reuse, greater interoperability, shorter selection time etc. This guides the architect and makes the architect focus on the quality attribute of interest by restricting the greater vocabulary of design alternatives.

Providing a basis for training:

The architecture describes elements and how they interact with each other. Module views are essential for showing the structure of a project; who does what, which team does what etc. This is why It can serve as an introduction of the system to a new project member