# Software Architecture: A Travelogue

## 1. **INTRODUCTION**.

A critical issue in the design and construction of any complex software system is its architecture: that is, its organization as a collection of interacting elements – modules, components, services, etc. A good architecture can help ensure that a system will satisfy its key functional and quality requirements, including performance, reliability, portability, scalability, and interoperability. A bad architecture can be disastrous.Over the past two and a half decades software architecture has received increasing attention as an important subfield of software engineering.

## 2. WHAT IS SOFTWARE ARCHITECTURE?

While there are numerous definitions of software architecture, at the core of most of them is the notion that the architecture of a system describes its gross structure. This structure illuminates the top-level design decisions, including things such as how the system is composed of interacting parts, what are the principal pathways of interaction, and what are the key properties of the parts and the system as a whole. Additionally, an architectural description includes sufficient information to allow high-level analysis and critical appraisal.

Software architecture typically plays a key role as a bridge between requirements and implementation

To elaborate, software architecture can play an important role in at least six aspects of software development:

1. **Understanding**: Software architecture simplifies our ability to comprehend large systems by presenting them at a level of abstraction at which a system's high-level design can be easily understood . Moreover, at its best, architectural description exposes the high-level constraints on system design, as well as the rationale for making specific architectural choices.
2. **Reuse**: Architectural design supports reuse of both components and also frameworks into which components can be integrated. Domain-specific software architectures, frameworks, platforms and architectural patterns are various enablers for reuse, together with libraries of plugins, add-ins and apps
3. **Construction**: An architectural description provides a partial blueprint for development by indicating the major components and dependencies between them. For example, a layered view of an architecture typically documents abstraction boundaries between parts of a system's implementation, identifies the internal system interfaces, and constrains what parts of a system may rely on services provided by other parts
4. **Evolution**: Architectural design can expose the dimensions along which a system is expected to evolve. By making explicit a system’s "load-bearing walls," maintainers can better understand the ramifications of changes, and thereby more accurately estimate costs of modifications [32]. In many cases such evolution and variability constraints are manifested in product lines, frameworks and platforms, which dictate how the system can be instantiated or adapted through the addition of application-specific features and components
5. **Analysis**: Architectural descriptions provide opportunities for analysis, including system consistency checking [3][7], conformance to constraints imposed by an architectural style [1], satisfaction of quality attributes [16], and domain-specific analyses for architectures built in specific styles
6. **Management**: For many companies the design of a viable software architecture is a key milestone in an industrial software development process. Critical evaluation of an architecture typically leads to a much clearer understanding of requirements, implementation strategies, and

potential risks, reducing the amount of rework required to address problems later in a system's lifecycle

## 3. THE PAST

In the early decades of software engineering, architecture was largely an ad hoc affair. Within industry, two trends highlighted the importance of architecture. The first was the recognition of a shared repertoire of methods, techniques, patterns and idioms for structuring complex software systems. For example, the box-and-linediagrams and explanatory prose that typically accompany a highlevel system description often refer to such organizations as a "pipeline," a "blackboard-oriented design," or a "client-server system."

The second trend was the concern with exploiting commonalities in specific domains to provide reusable frameworks for product families. Such exploitation was based on the idea that common aspects of a collection of related systems can be extracted so that each new system can be built at relatively low cost by "instantiating" the shared design and reusing shared artifacts.

## 4. SOFTWARE ARCHITECTURE TODAY

Although there is considerable variation in the state of the practice, today software architecture is widely visible as an important and explicit design activity in software development. In addition, the technological and methodological basis for architectural design has improved dramatically. Four important advances have been (1) the codification and dissemination of architectural design expertise; (2) the emergence of platforms and product lines, and their associated ecosystems; (3) the development of principles, languages and tools for architecture description; and (4) the integration of architectural design into the broader processes of software development, and, in particular the relationship between architecture and agility

## 5. CHALLENGES AHEAD

What about the future? Although software architecture is on a much more solid footing than two decades ago, it is not yet fully established as a discipline that is taught and practiced across the software industry. One reason for this is simply that it takes time for new approaches and perceptions to propagate. Another reason is that the technological basis for architecture design (as outlined earlier) is still maturing. In both of these areas we can expect that a natural evolution of the field will lead to steady advances.

However, the world of software development, and the context in which software is being used, is changing in significant ways. These changes promise to have a major impact on how architecture is practiced.So for that some of the important trends and their implications for the field of software architecture.(1) Network-Centric Computing (2) Pervasive Computing and Cyber-physical Systems (3) Fluid Architectures (4) Socio-technical Ecosystems

## 6. CONCLUSION

The field of software architecture is one that has experienced considerable growth over the past two and a half decades, and it promises to continue that growth for the foreseeable future. Although architectural design has matured into an engineering discipline that is broadly recognized and practiced, there are a number of significant challenges that will need to be addressed. Many of the solutions to these challenges are likely to arise as a natural consequence of dissemination and maturation of the architectural practices and technologies that we know about today. Other challenges arise because of the shifting landscape of computing and the roles that software systems play in our world: these will require significant new innovations.