**Abstract:**

As software systems have evolved, so has software architecture, with practices growing to meet each era's new challenges. The next phase of evolution--intelligent connected systems--promises to be an exciting time for software architects.

**Summary**

It is nearly 25 years since the seminal paper by Perry and Alexander Wolf marked the recognition of Software architecture as a recognizable discipline, and it has changed considerably over that time, from the basic problems of designing static software structures, to today’s global always-on, Internet connected systems with their ever evolving architectures.

**The Five Ages of Software System**

When we look back over recent software history, we can see five identifiable evolutions of software systems, each one roughly aligning with a decade.   
Through the 1980s, software systems were largely monolithic and tended to run on single computers and software was developed as “programs” and architecture was largely a vendor concern, inherited from the platform that the software developers were using.As we moved into the 1990s, distributed systems became mainstream, a lot of batch processing moved to online processing, and three-tier client server follow for enterprise system with more architectural decisions to be made than before. In the late 1990s Internet became a mainstream technology, organizations needed Internet-connected systems, which were “always on” rather than just “online”. These were initially websites but over time started to incorporate public user-interfaces for B2C and B2B processing too. The architecture of these systems now had to support unpredictable and challenging non-functional qualities (particularly performance, scalability and security)/In 2010s, the he Internet has become a basic service and always being used from anywhere for anything” as they are Internet-native systems. Following current trends, it seems that the next phase of evolution will be to Intelligent Connected systems, as artificial intelligence (machine learning in particular) becomes mainstream and fast, reliable networks become ever more ubiquitous, allowing us to connect our “things” into systems as well as traditional compute- rs. These systems will move beyond providing “access anywhere” to providing “intelligent assistance” to their users. This is going to need a new architectural focus on data and algorithms as well as a move to “emergent” architecture which is only formed at runtime.

**Five Ages of Software Architecture**

Software architecture has developed as a very pragmatic discipline and as the software industry has evolved through its different stages, the techniques and concerns of software architects have evolved in parallel. During the monolithic systems development, software architecture had not been recognized as a discipline, but pioneers were already thinking about large scale software design, leading to structural design techniques like the use of modules for structuring and information hiding for encapsulating. As technology moved to distributed systems, the increasing complexity of and their environments led to more complex design decision making. This led to software architecture being recognized as a discipline and the introduction of techniques to deal with more complex environment, such as viewpoints and views for architectural description, explicit identification and management of stakeholders, the definition of proven and reusable architectural styles and the introduction of architectural assessment

Techniques to allow structured comparison of architectural options. Internet connected systems introduced further new challenges that resulted in a new architectural focus on their challenging non-functional qualities and how to achieve them he importance of well-made, clearly-communicated architectural decisions, and how architectural design could support the agility. In our current era, Internet-native systems have resulted in further changes to the role of software architecture, as systems are now often more malleable and dynamic, being composed from fine-grained network services. Systems are often built on PaaS platforms and so can composed from a mix platform services, network connected services from other suppliers and the system’s own unique services.

**The Future of Software Architecture**

While structural design isn’t going to go away, the need to integrate dynamic architecture and intelligence into systems is going to push us to reconsider how important data and algorithms are Another trend that seems very likely is a move from structure being defined “up front” through an architectural design process, to the definition of architectural structure formed at runtime by combining a large number of network services to form a system. It’s worth noting that this is quite different to the sort of “emergent architecture” which some agile practitioners talk about - they’re talking about conventional “static” architecture, which is developed incrementally as a project progresses. A fairly recent evolution in software practice has seen the architect acting less as an “up-front” designer of structures, and becoming responsible for a stream of informed, significant decisions, made just in time for the project. The combination of these factors means that software architects will need to deal much more with probability rather than certainty (or an assumption of certainty). Factors such as composing systems from 3rd party services, the use of machine learning and analytics in system design, integrating very large numbers of small (IoT) devices and the use of dynamic runtime environments such as PaaS platforms all imply that architecture will involve dealing with statistical characteristics and trends more than statically defined structures and definite quantities. The operational aspects of our systems are going to evolve too, which will also affect architecture practice. Whereas today architects may well be involved in defining and advising on operational processes, large dynamic systems will require policy-driven automation rather than today’s carefully designed step-by-step processes.

**Conclusion**

As software systems have evolved, so has software architecture, with each era of architecture practice evolving to meet the changing challenges of software practice at that time. This has successfully led software architects from basic concepts like information hiding, through the foundation of the discipline to techniques like viewpoints and architectural assessment which are mainstream today. More recently architecture practice has evolved further to the point where architecture has become an implicit part of mainstream practice and is often seen as an activity rather than a distinct role, focusing more on decisions and principles than the previous focus on defining structure.