**Software architecture refers to the high level structures of a software system, the discipline of creating such structures, and the documentation of these structures. These structures are needed to reason about the software system. Each structure comprises software elements, relations among them, and properties of both elements and relations. The architecture of a software system is a metaphor, analogous to the architecture of a building.**

**As a result, software architecture has become increasingly important in the last two decades in software engineering community. At the heart of every well-engineered software system is always its software architecture. Software architecture deals with the high-level building blocks that represent an underlying software system. Software architectures that have been proved to be useful for families of systems are often codified into architectural styles (patterns). Also, a good architecture of a software system can lead to better quality attributes including reusability, extensibility, scalability, maintainability, and ect.**

**A software design pattern is a general reusable solution to a commonly occurring problem within a given context in software design. It is not a finished design that can be transformed directly into source or machine code. It is a description or template for how to solve a problem that can be used in many different situations. Design patterns are formalized best practices that the programmer can use to solve common problems when designing an application or system.**

**As a result, good designers do not solve every problem from first principles. They reuse design patterns to help them solve architectural design problems. Design patterns are preferred because they are a common design vocabularies, thus allowing engineers to abstract a problem and talk about that abstraction in isolation from its implementation. Design patterns also capture design expertise and allow that expertise to be communicated, thus promoting design reuse and avoiding mistakes. What’s more, design patterns help to improve documentation (less is needed) and understandability (patterns are described well once) for software architecture design.**

**For this workflow-centric scientific social network project, we firstly analyzed the architecture of the legacy system by using ATAM (Architecture Tradeoff Analysis Method) systematically. After that, we carefully abstracted and analyzed the architecture and architectural approaches that the legacy system has adopted. After we had a clear and intensive understanding of the architecture of the whole project, we tried to improve the existing architectural design of the legacy system by applying some useful design patterns. For each design pattern, we carefully analyzed its definition, benefits, constraints, the place where we adopted a specific design pattern, and also detailed information about how to implement a specific design pattern using class diagram and real code.**