Design pattern vs Architecture pattern

Design patterns and architecture patterns are both important concepts in software development, but they serve different purposes and operate at different levels of abstraction.

Design Patterns:  
Design patterns are reusable solutions to common design problems that occur within the confines of a single class or a small group of classes. They focus on specific design issues and provide guidelines for implementing them effectively. Design patterns are concerned with the structure, behavior, and interaction of individual objects and classes within a system. Examples of design patterns include the Singleton pattern, Observer pattern, and Factory pattern. Design patterns help to improve code organization, reusability, and maintainability at a lower level of granularity.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Architecture Patterns:  
Architecture patterns, on the other hand, deal with larger-scale design decisions and address the overall structure, organization, and behavior of an entire software system. They define the high-level components, their interactions, and the overall flow of control in the system. Architecture patterns are concerned with the system's architecture, including its layers, modules, components, and their relationships. Examples of architecture patterns include the Layered architecture, Microservices architecture, and Model-View-Controller (MVC) architecture. Architecture patterns guide the overall design and structure of the system, focusing on aspects such as scalability, performance, security, and maintainability.

In summary, design patterns provide reusable solutions to specific design problems within smaller units of code, while architecture patterns define the overall structure and behavior of a software system. Both design patterns and architecture patterns contribute to creating well-designed, maintainable, and scalable software systems, but they operate at different levels of abstraction and address different aspects of the design process.