**Schichtenarchitektur (Layered Architecture)**: Dieser Architekturstil organisiert die Software in Schichten oder Ebenen, wobei jede Schicht bestimmte Funktionen und Verantwortlichkeiten hat. Ein häufiges Beispiel ist die dreischichtige Architektur, die aus Präsentations-, Geschäftslogik- und Datenspeicherebenen besteht.

**Mikrodienste (Microservices):** Bei dieser Architektur werden Anwendungen in kleine, unabhängige Dienste aufgeteilt, die jeweils eine spezifische Aufgabe erfüllen. Diese Dienste können unabhängig voneinander entwickelt, bereitgestellt und skaliert werden.

**Monolithische Architektur:** Im Gegensatz zu Mikrodiensten handelt es sich bei monolithischen Anwendungen um eine einzige, zusammenhängende Einheit. Alle Funktionen und Komponenten sind in einem einzigen Codebase integriert.

**Ereignisgesteuerte Architektur (Event-Driven Architecture):** Diese Architektur basiert auf dem Austausch von Ereignissen oder Nachrichten zwischen verschiedenen Komponenten der Anwendung. Komponenten reagieren auf Ereignisse und können unabhängig voneinander arbeiten.

**Event Sourcing und CQRS:** Dies ist eine spezielle Art der Architektur, die auf der Idee basiert, dass alle Änderungen im System als Ereignisse gespeichert werden (Event Sourcing) und dass die Lese- und Schreibvorgänge getrennt werden (CQRS - Command Query Responsibility Segregation). Dies ermöglicht eine effiziente Verarbeitung von Änderungen und Abfragen.

**Serviceorientierte Architektur (Service-Oriented Architecture, SOA):** SOA organisiert Anwendungen in Dienste, die über Netzwerkprotokolle miteinander kommunizieren. Diese Dienste sind wiederverwendbar und bieten spezifische Funktionalitäten an.

**Architektur für lose Kopplung (Loosely Coupled Architecture)**: Dieser Architekturstil betont die Unabhängigkeit von Komponenten, sodass Änderungen in einer Komponente minimale Auswirkungen auf andere haben.

**Datenzentrierte Architektur (Data-Centric Architecture):** Hier steht die Datenverarbeitung im Mittelpunkt, und die Architektur ist darauf ausgerichtet, Daten effizient zu speichern, zu verarbeiten und abzurufen.

**Schichtenunabhängige Architektur (Tierless Architecture)**: Bei dieser Architektur werden die herkömmlichen Schichtenkonzepte vermieden, und Komponenten werden basierend auf funktionalen Einheiten organisiert.

**Big Data-Architektur**: Diese Architektur wurde speziell für die Verarbeitung und Analyse großer Datenmengen entwickelt und umfasst Technologien wie Hadoop und Spark.

**Serverless-Architektur**: In diesem Ansatz werden Anwendungen so entwickelt, dass sie auf serverlosen Plattformen wie AWS Lambda oder Azure Functions ausgeführt werden, ohne dass Entwickler sich um die Verwaltung von Servern kümmern müssen.

**Containerbasierte Architektur**: Diese Architektur verwendet Container-Technologien wie Docker, um Anwendungen und ihre Abhängigkeiten in isolierten Containern zu verpacken und bereitzustellen.

**Paketarchitektur (Component-Based Architecture)**: Hier werden Anwendungen in wiederverwendbare Pakete oder Komponenten unterteilt, die unabhängig voneinander entwickelt und integriert werden können

**Adaptive Systeme**

Diese Muster unterstützen besonders die Erweiterungs- und Anpassungsfähigkeit von Softwaresystemen.

[Dependency Injection](https://de.wikipedia.org/wiki/Dependency_Injection)

[Mikrokernel](https://de.wikipedia.org/wiki/Mikrokern)

[Reflexion](https://de.wikipedia.org/wiki/Reflexion_(Programmierung))

**Chaos zu Struktur (**[**englisch**](https://de.wikipedia.org/wiki/Englische_Sprache)**Mud-to-structure)**

Diese Muster sollen helfen, die Vielzahl der Komponenten und Objekte eines Softwaresystems zu organisieren. Die Funktionalität des Gesamtsystems wird hierbei in kooperierende Subsysteme aufgeteilt.

**[Command Query Responsibility Segregation](https://de.wikipedia.org/wiki/Command_Query_Responsibility_Segregation" \o "Command Query Responsibility Segregation)**

**[Data Context Interaction](https://de.wikipedia.org/wiki/Data_Context_Interaction" \o "Data Context Interaction)**

**[Domain-driven Design](https://de.wikipedia.org/wiki/Domain-driven_Design" \o "Domain-driven Design) und [Naked Objects](https://de.wikipedia.org/wiki/Naked_Objects" \o "Naked Objects)**

[**Pipes und Filter**](https://de.wikipedia.org/wiki/Pipes_und_Filter)

[**Schichtenarchitektur**](https://de.wikipedia.org/wiki/Schichtenarchitektur)

**Schwarzes Brett ([Blackboard](https://de.wikipedia.org/wiki/Blackboard" \o "Blackboard))**

**Interaktive Systeme**

Muster dieser Kategorie helfen, Mensch-Computer-Interaktionen zu strukturieren.

**[Model View Controller](https://de.wikipedia.org/wiki/Model_View_Controller" \o "Model View Controller) (MVC), [Model View Presenter](https://de.wikipedia.org/wiki/Model_View_Presenter" \o "Model View Presenter) und [Remote Presentation Model](https://de.wikipedia.org/wiki/Remote_Presentation_Model" \o "Remote Presentation Model)**

[**Presentation-Abstraction-Control**](https://de.wikipedia.org/wiki/Presentation-Abstraction-Control)**(PAC)**

**Verteilte Systeme (Netzwerk)**

Diese Kategorie unterstützt die Verwendung verteilter Ressourcen und Dienste in Netzwerken, wie z. B. die [Orchestrierung](https://de.wikipedia.org/wiki/Dienstekomposition#Orchestrierung). Zwei weitere Modelle ('Mikrokernel' und 'Pipes und Filter') unterstützen Verteilung zweitrangig.

**[Client-Server](https://de.wikipedia.org/wiki/Client-Server-Modell" \o "Client-Server-Modell)**

**[Peer-to-Peer](https://de.wikipedia.org/wiki/Peer-to-Peer" \o "Peer-to-Peer)**

**[Serviceorientierte Architektur](https://de.wikipedia.org/wiki/Serviceorientierte_Architektur" \o "Serviceorientierte Architektur) (SOA)**

[**Edge Computing**](https://de.wikipedia.org/wiki/Edge_Computing)