Lernzettel

Architekturklassifikation nach Architekturtyp und Anwendungszweck

Universität: Technische Universität Berlin
Kurs/Modul: Softwaretechnik und Programmierparadigmen

Erstellungsdatum: September 19, 2025

Lernzettel: Architekturklassifikation nach Architekturtyp und Anwendungszweck

(1) Grundkonzepte. Eine Architekturklassifikation ordnet Softwaresysteme nach ihrer Struktur (Architekturtyp) und ihrem Einsatzzweck (Anwendungszweck). Der Architekturtyp beschreibt die Bauweise der Komponenten und deren Beziehungen. Der Anwendungszweck gibt an, welche Qualitäten im Vordergrund stehen (z. B. Skalierbarkeit, Verfügbarkeit, Wartbarkeit).

(2) Architekturtypen (Strukturelle Klassifikation). - Schichtenarchitektur (Layered).
Definition: Mehrere Schichten mit klaren Schnittstellen (z. B. Präsentation, Logik, Persistenz). Vorteile: Hohe Trennung der Belange, einfache Wartbarkeit. Nachteile: Potenzieller Overhead, starre Schichtgrenzen.

- Client-Server.
Definition: Client-Anwendungen fordern Dienste von einem Server an. Vorteile: Zentralisierte Dienste, einfache Skalierung einzelner Funktionsbereiche. Nachteile: Netzwerklatenz, Abhängigkeiten vom Server.

- Pipe-and-Filter.
Definition: Verarbeitungsketten aus Filterkomponenten, die Datenströme transformieren. Vorteile: Leichte Wiederverwendbarkeit, klare Datenflüsse. Nachteile: Koordination der Filter, potenzielle Latenz bei langen Pipelines.

- Komponentenbasierte Architektur.
Definition: Systeme als Zusammenfügung modularer, wiederverwendbarer Komponenten. Vorteile: Wiederverwendbarkeit, klare Schnittstellen. Nachteile: Kompositionserwägungen, Dependencies.

- Mikroservices.
Definition: Eine Anwendung aus vielen kleinen, unabhängig deploybaren Diensten. Vorteile: Skalierbarkeit, Fehlertoleranz, unabhängige Entwicklung. Nachteile: Komplexität der Koordination, Netzwerkverkehr.

- Mikrokernel (Plug-in-Architektur).
Definition: Kernsystem mit austauschbaren Plug-ins. Vorteile: Erweiterbarkeit, geringe Modifikationskosten am Kern. Nachteile: Abhängigkeiten der Plug-ins, Kompatibilitätsmanagement.

- Ereignisgesteuerte Architektur (EDA).
Definition: Systeme reagieren auf Ereignisse über asynchrone Messaging-Kanäle. Vorteile: Hohe Skalierbarkeit, Entkopplung. Nachteile: Komplexität beim Debugging, eventual consistency.

- Model-View-Controller (MVC) / Architekturmuster.
Definition: Trennung von Modell, Sicht und Steuerelementen zur Strukturierung von UI-Anwendungen. Vorteile: Klarheit der Verantwortlichkeiten, einfache Anpassung der UI. Nachteile: Overhead durch Abstraktionen, nicht immer passende Trennung in Nicht-UI Bereichen.

(3) Architekturtypen nach Anwendungszweck (Anwendungszweckorientierte Klassifikation). - Hohe Skalierbarkeit und Verfügbarkeit.
Beispiele: Mikroservices, skalierbare Client-Server-Architekturen. Vorteile: Lastspitzen besser abfangen, Ausfallteile isoliert. Herausforderungen: Verteilte Konsistenz, Deployment-Komplexität.

- Echtzeit-/latenzempfindliche Systeme.
Beispiele: Eventgesteuerte Architekturen mit Low-Latency-Messaging. Vorteile: Schnelle Reaktionszeiten, deterministisches Verhalten. Herausforderungen: Scheduling, QoS, deterministische

Latenzen.

- Datenzentrierte bzw. data-intensive Systeme.
Beispiele: Pipes-and-Filters, Dienste mit datenorientierten Pipelines. Vorteile: Datenflussklarheit, leichte Transformation von Daten. Herausforderungen: Datenpersistenz, Konsistenz-Modelle.

- Integrations- und Enterprise-Szenarien.
Beispiele: Serviceorientierte Architekturen (SOA), ESB, API-Gateways. Vorteile: Systemübergreifende Interoperabilität. Herausforderungen: Governance, Sicherheit, Versioning.

- Erweiterbarkeit und Plug-in-Fähigkeit.
Beispiele: Mikrokernel, modulare Frameworks. Vorteile: Schnellere Anpassung an neue Anforderungen. Herausforderungen: Kompatibilität, Ökosystem-Management.

(4) Bewertungskriterien (Qualitäten aus Architekturperspektive). Kopplung und Kohäsion der Bausteine.

Skalierbarkeit und Verfügbarkeit.

Wartbarkeit und Erweiterbarkeit.

Performance und Ressourcenverbrauch.

Verständlichkeit der Architektur und Dokumentation.

Flexibilität bei Änderungsanforderungen und Portabilität.

(5) Beispiele und typische Anwendungsfälle. - Schichtenarchitektur. Webanwendung mit Präsentation, Geschäftslogik und Persistenzschicht.

- Client-Server. Desktop- oder Webanwendung mit zentralem Serverdienst.

- Pipe-and-Filter. Kompilier-Pipeline oder Datenverarbeitungs-Workflow.

- Mikroservices. Cloud-basierte E-Commerce-Plattform mit vielen Diensten.

- Mikrokernel. Erweiterbare IDE oder Editor mit Plug-ins.

- EDA. IoT- oder Event-Processing-System mit asynchronen Nachrichten.

- MVC. Web-Frontend mit klarer UI-Logiktrennung.

(6) Hinweise für die Praxis. Architekturentscheidungen wirken sich direkt auf Wartbarkeit, Skalierbarkeit und Sicherheit aus. Bei der Wahl eines Architekturtyps sollten Anforderungen, Risiken und vorhandene Ressourcen beachtet werden. Die Kombination mehrerer Typen (Hybridarchitektur) ist häufig sinnvoll, um Stärken verschiedener Ansätze zu nutzen.

(7) Weiterführende Begriffe. Kopplung, Kohäsion, Modularisierung, Schnittstellen, Schnittstellen-Design, Deployment-Strategien, Governance.