Lernzettel

Architektur von Anwendungssystemen

Universität: Technische Universität Berlin
Kurs/Modul: Architektur von Anwendungssystemen
Erstellungsdatum: September 19, 2025

(1) Einführung. Die Architektur von Anwendungssystemen beschreibt die Struktur eines Systems aus Bausteinen (Komponenten, Dienste) und deren Schnittstellen. Bei verteilten Anwendungssystemen erstreckt sich diese Struktur über mehrere Rechner und Netze hinweg. Ziel ist es, Systeme so zu gestalten, dass sie robust, skalierbar, sicher und wartbar bleiben. Architekturentscheidungen betreffen dabei sowohl die Anwendungslogik als auch die zugrundeliegende Technologiebene (Middleware, Web-Technologien) und deren Interaktionen.

(2) Architekturstile verteilter Anwendungssysteme. Verteilte Architekturen nutzen unterschiedliche Stile, um Anforderungen aus Bereichen wie Skalierbarkeit, Verfügbarkeit und Flexibilität abzubilden. Wichtige Stile sind:

- Schichtenarchitektur (Layered): klare Aufteilung in Ebenen wie Präsentation, Logik und Datenzugriff. Vorteil: gute Trennung, Nachteil: potenzieller Overhead durch Vermittlerlagen.
- Client-Server: Clients fordern Dienste von zentralen Servern an. Vorteil: Zentrale Ressourcen- und Sicherheitskontrolle; Nachteil: Engpässe am Server.
- 3-Tier-Architektur: Präsentationsschicht, Anwendungsschicht und Datenhaltung liegen auf separaten Rechnern. Vorteil: bessere Skalierbarkeit und Wartbarkeit als Monolith; Nachteil: komplexeres Deployment.
- Serviceorientierte Architektur (SOA): Funktionen werden als lose gekoppelte Services bereitgestellt, die über definierte Schnittstellen kommunizieren. Vorteil: Wiederverwendbarkeit von Diensten; Nachteil: Governance-Kosten.
- Microservices: kleine, unabhängige Services, die iterativ entwickelt, bereitgestellt und skaliert werden. Vorteil: hohe Flexibilität; Nachteil: verlagerte Komplexität bei Schnittstellen und Orchestrierung.
- Event-getriebene Architektur (EDA): Kommunikation über Ereignisse (Asynchronität). Vorteil: lose Kopplung, Skalierbarkeit; Nachteil: komplexere Konsistenzmodelle.

(3) Middleware- und Web-Technologien. Zur Umsetzung verteilter Architekturen kommen unterschiedliche Technologien zum Einsatz. Wichtige Gruppen sind:

- Web-APIs und Kommunikation: REST, GraphQL, gRPC; REST ist oft verbreitet für lose Kopplung, GraphQL für Abfragevielfalt, gRPC für effiziente binäre Kommunikation.
- Messaging und Middleware: AMQP (z. B. RabbitMQ), Kafka (Event-Streaming), MQTT für IoT-Kommunikation; Einsatz bei asynchroner Kommunikation und Events.
- Pub/Sub-Modelle und API-Gateways: Publish/Subscribe-Muster, API-Gateways zur Schnittstellenkapselung, Service Discovery zur Erkennung von Diensten.
- Echtzeit-Kommunikation: WebSockets, Server-Sent Events für langanhaltende Verbindungen.
- Identität und Sicherheit: OAuth 2.0, OpenID Connect, JWT; Zugriffskontrolle und Authentifizierung in verteilten Systemen.
- Transaktionsmodelle: Sagas zur Koordination verteiltranzaktionaler Abläufe ohne verteilte ACID-Transaktionen; eventual consistency wird oft angestrebt.

(4) Entwurf und Bewertung von Anwendungsarchitekturen. Beim Entwurf werden Anforderungen aus Technik, Fachbereich und Organisation in eine passende Architektur überführt. Zentrale Aspekte sind:

- Lose Kopplung und hohe Kohäsion der Komponenten.
- Klare Schnittstellen (APIs) und stabile Verträge zwischen Diensten.
- Wahl zwischen synchroner vs. asynchroner Kommunikation je nach Latableit, Konsistenzbedarf und Fehlertoleranz.
- Berücksichtigung von Skalierbarkeit, Verfügbarkeit, Wartbarkeit und Sicherheit.
- Einsatz geeigneter Middleware- und Web-Technologien entsprechend dem Anwendungsfall.

(5) Nicht-funktionale Eigenschaften. Wichtige Qualitätskriterien verteilter Architekturen:

- Verfügbarkeit: Bereitschaftsdauer des Systems trotz Fehlern.
- Skalierbarkeit: horizontale/vertikale Ausbaumöglichkeit bei steigender Last.
- Konsistenz: konsistentes Verhalten trotz Verteilung; verschiedene Modelle (strong, eventual, causal) je nach Anforderung.
- Latenz und Durchsatz: Reaktionszeit und bearbeitete Anfragen pro Zeiteinheit.
- Sicherheit: Vertraulichkeit, Integrität, Authentisierung und Autorisierung.
- Wartbarkeit und Erweiterbarkeit: einfache Anpassung von Funktionen und Diensten.

(6) Musterarchitekturen. Typische Muster, die in Kursen zu verteilten Anwendungssystemen diskutiert werden, umfassen:

- 3-Tier-Architektur als Grundmuster: Präsentation, Logik, Datenhaltung.
- Microservice-Architektur: viele kleine, unabhängige Dienste mit eigener Datenhaltung.
- Event-Driven Architecture (EDA): Produzenten und Konsumenten verbinden über Events, asynchrone Verarbeitung.
- Layered Architecture mit API-Schicht: klare Trennung der Schnittstellen.

(7) Übungen und Praxisbeispiele. Zum Abschluss dieses Abschnitts stehen praxisnahe Aufgaben zum Verständnis:

- Übungsbeispiel 1: Entwerfe für einen Online-Shop eine passende Architektur. Begründe die Wahl zwischen einem monolithischen, einer Microservice- oder einer Event-getriebenen Lösung. Skizziere die Hauptdienste, deren Schnittstellen und eine grobe Kommunikationsstrategie (Synchrone vs. asynchrone Wege).
- Übungsbeispiel 2: Ein IoT-System mit vielen Sensoren soll Daten an eine zentrale Auswertungsplattform senden. Welche Architekturstile und Technologien kommen sinnvoll zum Einsatz? Welche Nicht-funktionalitäten müssen beachtet werden?