# Querschnittliche Konzepte

## Definition

Technische bzw. querschnittliche Konzepte in Softwarearchitekturen haben – wie der Name andeuten soll – häufig die Eigenschaft, über mehrere Teile oder gar über die gesamte Architektur hinweg zu wirken.

Quelle Basiswissen für Softwarearchitekten

Im Buch Effektive Softwarearchitektur wird stattdessen der Begriff Übergreifende Konzepte verwendet.

## Beispiele übergreifender Konzepte

Quelle Effektive Softwarearchitekturen8. Auflage, Kapitel 7

|  |  |
| --- | --- |
| Persistenz | Datenspeicherung, insbesondere von Objekt- und Klassenstrukturen auf  Tabellen, Entkopplung von Fachdomäne und (Datenbank-)Infrastruktur |
| Geschäftsregeln | Behandlung domänenspezifischer Zusammenhänge und Regeln als eigenständige  („externalisierte“) Einheiten |
| Integration | Einbindung bestehender Systeme in einen neuen oder veränderten Kontext |
| Verteilung | Verteilung von Systembestandteilen auf getrennte Ablaufumgebungen |
| Kommunikation | Übertragung von Daten zwischen Systemkomponenten, innerhalb und außerhalb von Prozess- oder Adressräumen |
| Grafische  Oberflächen | Steuerung von Navigation und Zustandswechseln in grafischen  Benutzeroberflächen, Verarbeitung von Benutzereingaben und Ereignissen |
| Ablaufsteuerung,  Geschäftsprozessmanagement | Steuerung systemübergreifender Abläufe, Koordination verschiedener Softwaresysteme, Workflowmanagement, Steuerung von Geschäftsprozessen |
| Sicherheit | Methoden zur Gewährleistung von Datenschutz und -sicherheit,  Verhinderung von Datenmissbrauch |
| Logging, Protokollierung,  Tracing | Sammeln von Informationen über den Programmzustand während der Laufzeit |
| Ausnahme- und  Fehlerbehandlung | Abweichungen von „erwarteten“ Situationen, Fehlverhalten oder Defekte  von Systemteilen, unerwartetes Verhalten von Benutzern oder anderen  Systemen |
| Batchverarbeitung | (Stapelverarbeitung): Datensatzweise Verarbeitung großer Datenbestände  ohne Interaktion mit Benutzern |
| Transaktionsbehandlung | Transaktionen sind nicht teilbare Arbeitsabläufe, die in sich komplett abgeschlossen sind. Transaktionen müssen immer komplett und vollständig abgearbeitet oder aber „zurückgerollt“ werden. Solche Transaktionen müssen die ACID-Forderungen erfüllen: Atomicity, Consistency, Isolation und Durability. |
| Konfigurierbarkeit | Anpassung des Systems an besondere Umwelt- oder Umfeldbedingungen  durch Änderungen von Installations-, Start- oder Laufzeitparametern |
| Zustands-, Session- oder  Sitzungsbehandlung | Verwaltung von Client- und Server-Zuständen, Zuordnung von Clients zu spezifischen Sessions („Sitzungen“), insbesondere im verteilten Client/Serverund Webumfeld wichtig |
| Plausibilisierung und Validierung von Eingabe- oder  Eingangsdaten | Wo und wann sollen Datenprüfungen stattfinden, um einen angemessenen  Kompromiss zwischen Wartbarkeit („zentralisierte Prüfung“) und Laufzeit  („Prüfung bei Dateneingang oder -eingabe“) zu finden? |

Tabelle Architekturaspekte nach arc42

Diese Aufzählung ist sehr komplex und hoch. Nach dem Buch Basiswissen Softwarearchitekur wird nur Persistenz, Fehlerbehandlung, Logging und Sicherheit erwähnt.

Die Aufgabe eines Softwarearchitekten ist dafür zu sorgen, dass ähnliche Probleme auf dieselbe Art und Weise gelöst werden.

Offen sind noch folgende Punkte

Querschnittskonzepte entscheiden und entwerfen, beispielsweise Persistenz, Kommunikation, GUI, Fehlerbehandlung, Nebenläufigkeit

mögliche wechselseitige Abhängigkeiten dieser Entscheidungen erkennen und beurteilen.