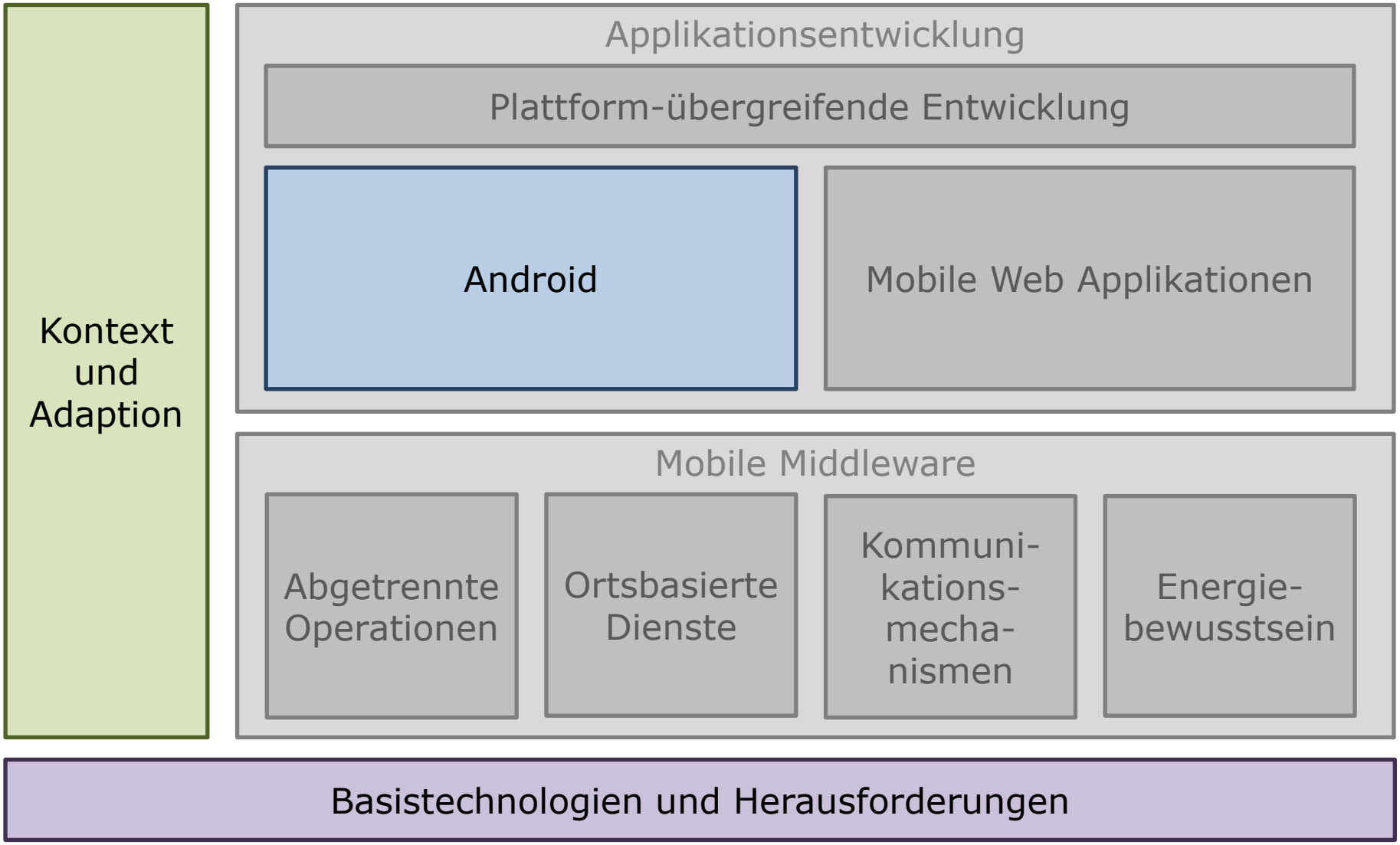
Web- und App-Programmierung

Kontext und Adaption

mit Skriptmaterial von Dr.-Ing. T. Springer

Prof. Dr.-Ing. Tenshi Hara
fragen@lern.es

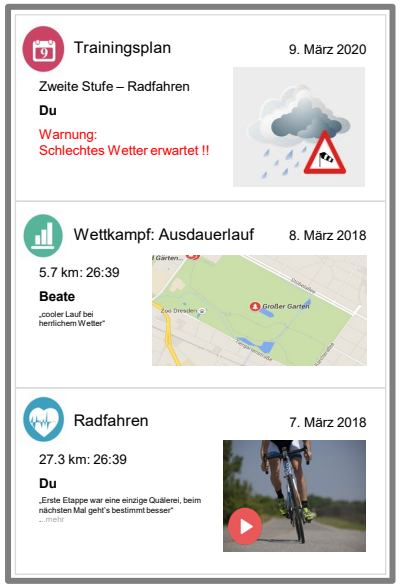
Aufbau der Lehrveranstaltung

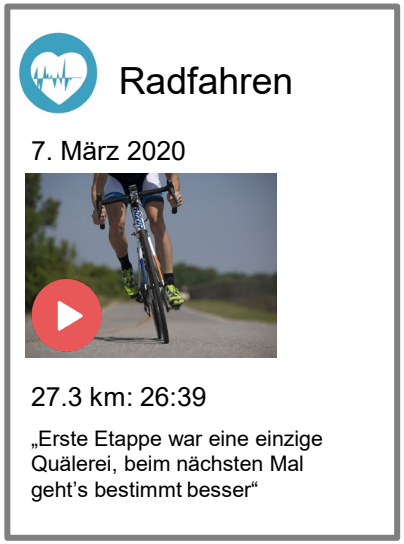


Beispiel 1

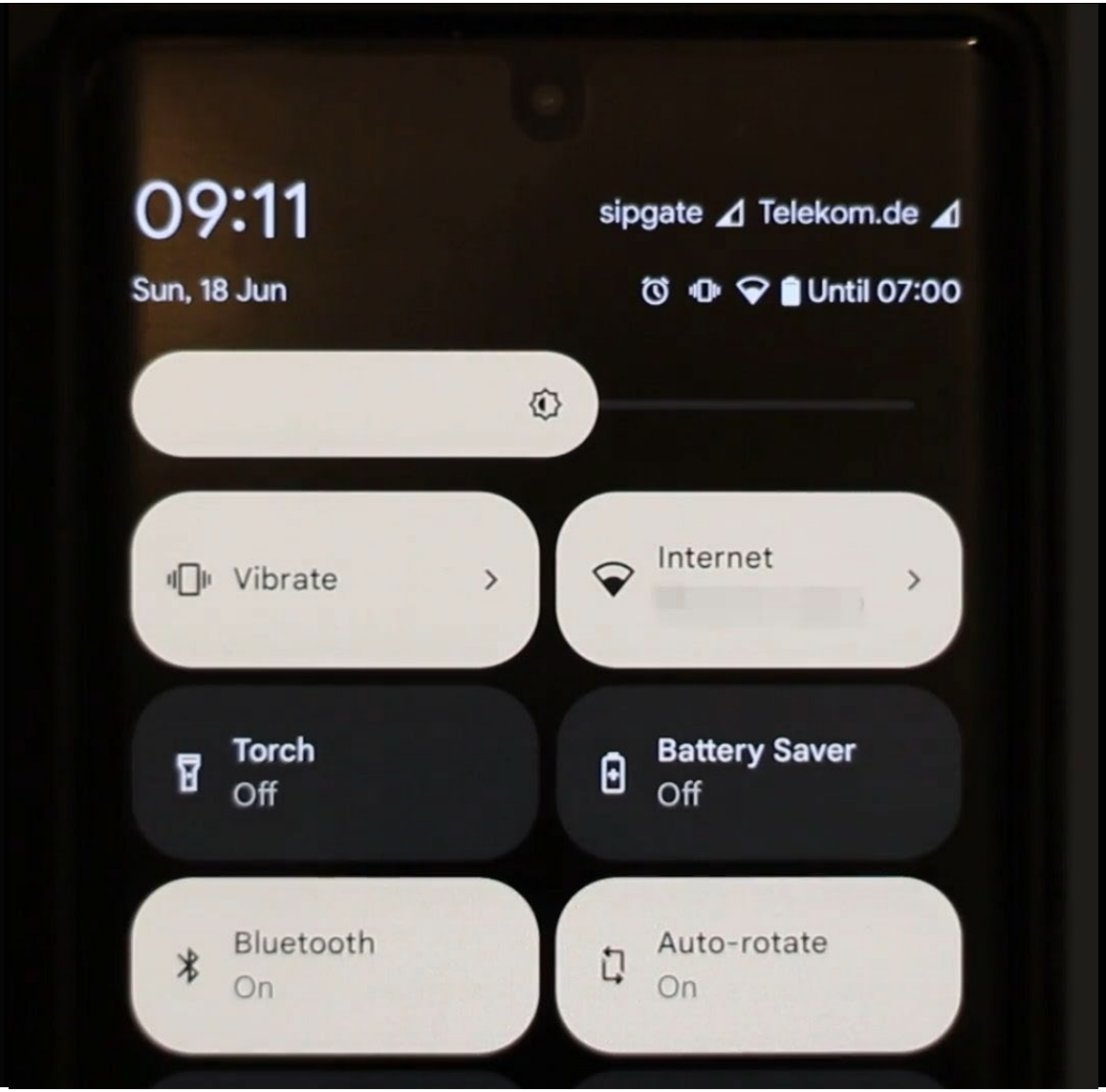
Social-Fitness-App

- Armband/Smartwatch mit Smartphone/Tablet-App
- Server-Komponente für Datenhaltung und Nutzerverwaltung
- Funktionalität
  - automatische Aktivitätserkennung, -aufzeichnung und -(um)planung
  - Aufzeichnung der Aktivitätszustände und korrelierter Medien- und Sensordaten (Bilder, Videos, Pfade, Pulskurven, …)
  - Zeitlinie der Aktivitäten
    → Posten der eigenen Aktivitäten
    → Sehen der Aktivitäten anderer
    → integrierte Zeitlinie mit Medien
  - Verwaltung des Trainingsplans
  - Verwaltung von Wettkämpfen und Platzierungen





Beispiel 2


09:11
sipgate Telekom.de
Sun, 18 Jun
Until 07:00
Vibrate
Internet
Torch
Off
Battery Saver
Off
Bluetooth
On
Auto-rotate
On

Problemanalyse

- Konnektivität
  - geringe Datenrate und hohe Verzögerung im Mobilfunk
  - intermittierende, teilweise schwache, wechselnde Verbindung
- Offline-Phasen
  - begrenzte Abdeckung, Unerreichbarkeit von Servern
  - Energiespareinstellungen, Datenkontingente, Kostenbremsen
- Formfaktor
  - begrenze Ressourcen (CPU, Speicher, Bildschirmgröße)
  - Diversität der Geräteklassen (Smartwatch, Smartphones, Tablets, …) und Geräteplattformen (iOS, Android, Windows Mobile, …)
- Energie
  - begrenzte Batteriekapazität
  - verbraucht von Bildschirm, Berechnungen, Kommunikation, Sensoren

Adaptionsbeispiele

- Automatische Erkennung und Erfassung von Sportaktivitäten
  - basierend auf Beschleunigungssensordaten
  - GPS nur während des Laufens und Radfahrens aktiv → spart Energie
  - höhere Abtastrate während des Laufens oder Wanderns
- Teilen der Aktivitäten mit dem Social-Network (Daten-Up- & -Download)
  - Up- und Download-Warteschlange
  - Filterregeln (nur Radfahraktivitäten), verlustbehaftete Konversion und *Lazy Evaluation* für Daten-Download
  - *Prefetching* und *Caching* von Daten
- Terminieren von Trainingsaktivitäten
  - Beobachtung von Wetterinformationen → Umplanen falls es regnet
- Verlust der Netzwerkverbindung
  - Nutzung der gecacheten Daten für die Zeitlinie
  - lokales (Zwischen)Speichern der Aktivitätsdaten

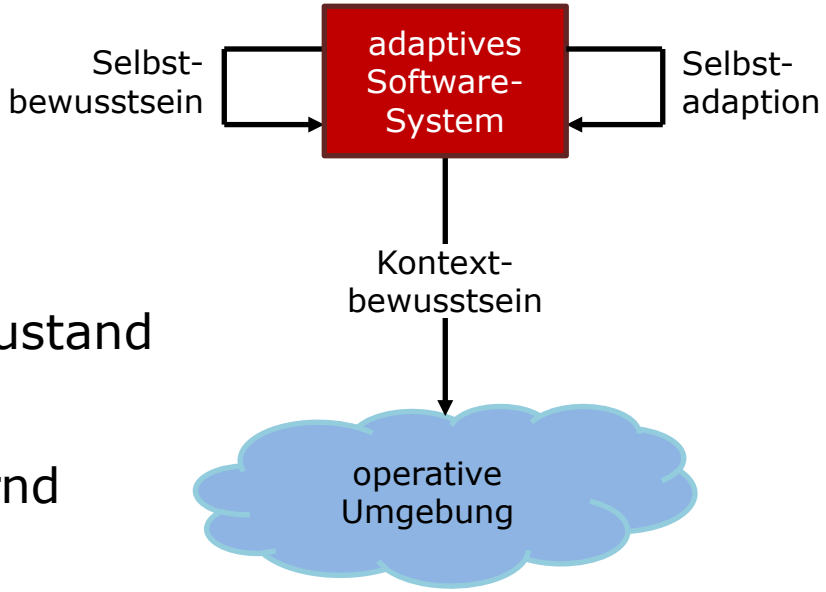
SELBSTADAPTIVE SYSTEME

Selbstadaptive Software modifiziert ihr eigenes Verhalten als Reaktion auf Veränderungen ihrer Laufzeitumgebung. Dabei ist die Laufzeitumgebung alles, was von der Software erfasst werden kann, bspw. Nutzereingaben, Sensorinformationen oder Programmausstattung.

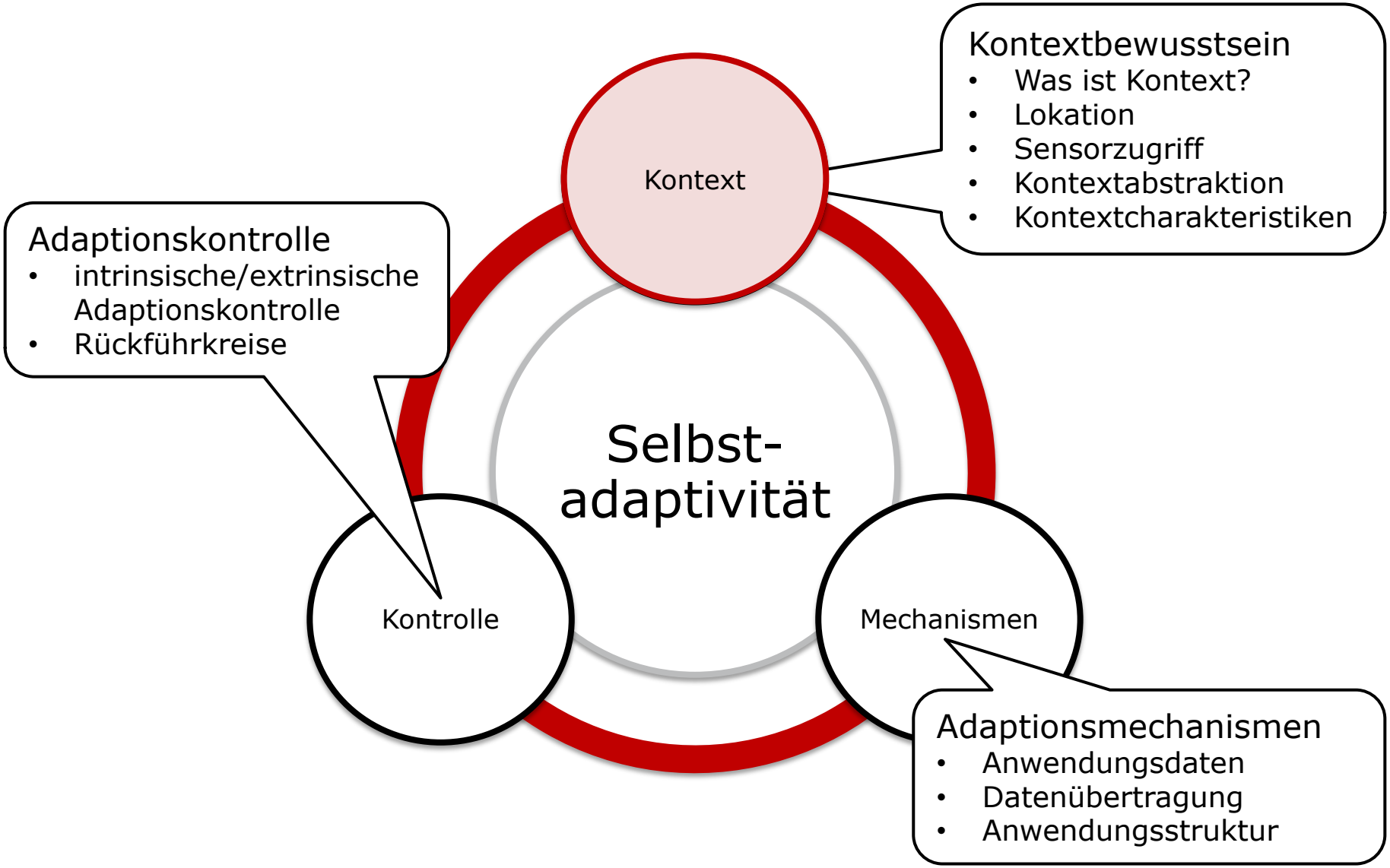
nach Oreizy et.al

Drei Notwendigkeiten

- Kontextbewusstsein:
  Wissen über operative Umgebung
- Selbstbewusstsein:
  Wissen über aktuellen Konfigurationszustand
- Selbstadaption:
  sich selbst während der Laufzeit ändernd

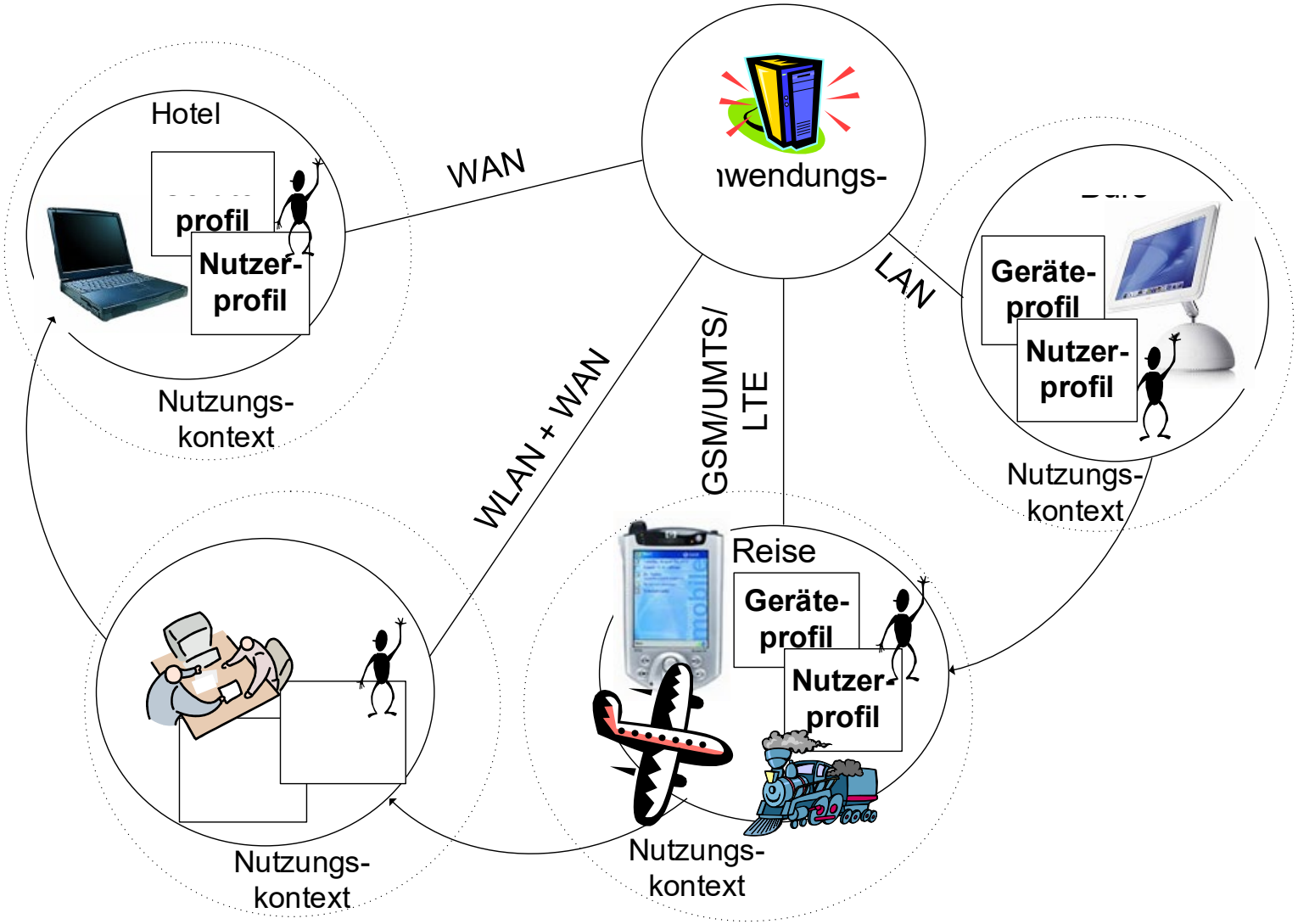


Kapitelübersicht



Selbstadaptivität: Kontext

Kontextbewusstsein macht den Unterschied



Motivation

soziale Umgebung und Aktivität
Berechnungsumgebung, Gerät und Netzwerkressourcen
Mobilität und Lokation
Interaktionsmöglichkeiten und Nutzeraufmerksamkeit
Kontextbewusste Anwendungen
Verständnis der aktuellen Nutzersituation
Kontrolle des Adaptionsprozesses
situations-/orts-abhängige Information
proaktive, autonome Systeme
personalisierte, vereinfachte Nutzerinteraktionen
adaptives Systemverhalten

Definition: Kontext

Alles außer explizite Ein- und Ausgaben einer Anwendung. [LiS00]

Das, was etwas anderes umgibt und ihm Bedeutung gibt. [o.V.00]

Kontext ist ein subjektives Konzept das durch seine es wahrnehmende Entität definiert wird. […] Kontextzustände […] sind inhärent mit spezifischen Objekten assoziiert. [Pascoe]

Kontext ist ein operativer Term: Etwas ist Kontext auf Grund der Art, wie es während einer Interpretation verwendet wird, nicht wegen seiner inhärenten Eigenschaften. […] Eigenschaften der Welt werden zu Kontext durch die Art ihrer Verwendung. [Winograd]

Kontext ist jede Art Information die verwendet werden kann, um die Situation einer Entität zu charakterisieren. Eine Entität ist eine Person, ein Ort oder ein Objekt, die/der/das als relevant für die Interaktion zwischen Mensch und Anwendung betrachtet wird, einschließlich des Nutzers und der Anwendung selbst. [Anind K. Dey]

Typen von Kontextinformationen

relevant für die Beispielanwendung

Dimension	zugehörige Kontextinformation
physisch	Ort, Zeit, Temperatur, Helligkeit und Lautstärke
technisch	Netzwerk (Bandbreite, Latenz, Fehlerrate), Gerät (Ein- und Ausgabe-Fähigkeiten, Speicher, Software-Unterstützung), Backend (verfügbare Dienste, Dienstvoreinstellungen)
persönlich	Adresse, Telefonnr., Zahlungsinformationen, Zeitplan
sozial	Personen in der Nähe, Gruppen/Teams des Nutzers, Familie und Verwandte, Freund und Bekannte
operativ	Rollen, Aktivitäten, Merklisten, Posteingang

Kontext in der Social-Fitness-App

?

Wie feststellen, dass der Nutzer losläuft?

Notify

Situation von Interesse:

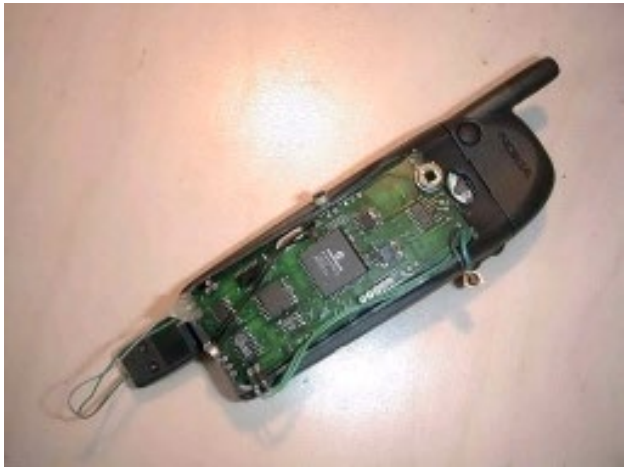
Nutzer beginnt Aktivität, insbesondere Laufen

Aktion

Wechsel in Aufzeichnungs-modus:

- aktiviere GPS und setze Abtastrate auf 1Hz
- beginne Aufzeichnung der GPS-Koor-dinaten und Aktivitätsdaten
- Kündige der Gruppe den Beginn der Aktivität an

Technology for Enabled Awareness (TEA)

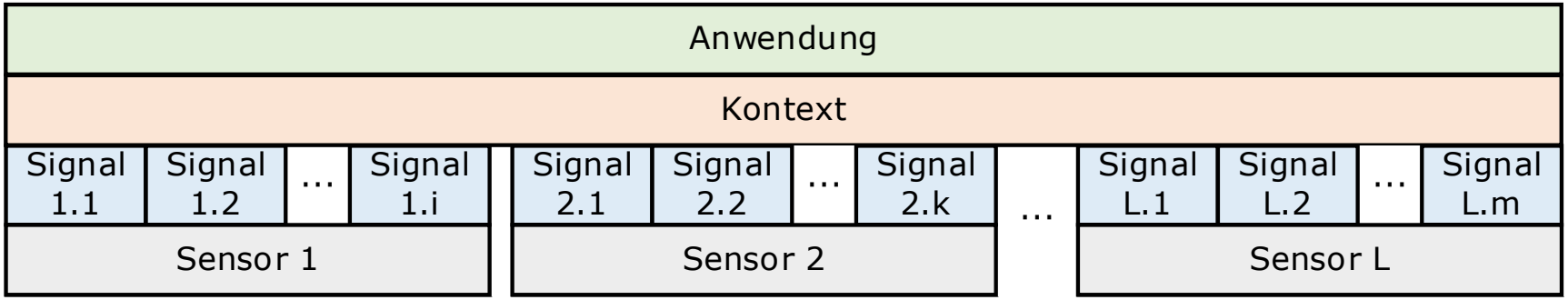


- Sensoren im Mobilgerät
- Ziel: Erkennen der Situation des Geräts und seines Nutzers
  - Ort und Nutzung des Mobilfunks
  - automatische Auswahl des Telefon-Profils
- dafür verwendete Sensoren
  - zwei Fotodioden
  - zwei Mikrofone
  - Zwei-Achsen-Beschleunigungssensor
  - Temperatursensor
  - Berührungssensor
  - durch Erweiterungsschächte um weitere Sensoren erweiterbar

TEA – Auswertungsebenen

4-schichtiges Model

1. Sensor (physisch, logisch)
2. Signal (Verarbeitung einer Sensorausgabe)
   - beinhaltet Wertefeld und Operationen auf diesen Werten (bspw. Mittelwert, Standardabweichung, Quantil)
3. Kontext (beschreibt disjunkte Situationen mit ihrer Wahrscheinlichkeit)
   - abstrahierte Beschreibung der aktuellen Situation
   - basierend auf Logikregeln angewendet auf Signalen
4. Anwendung (situations- und anwendungsspezifisch)
   - Definition von Aktionen basierend auf Skripten (ausgelöst wenn bestimmte Situation im Wahrscheinlichkeitsmodell eintreten/enden)



TEA – Auswertungsebenen

Signale

- keine Direktwerte von den Sensoren, nur berechnete Werte
- Berechnungsmethode: statistische Funktionen
  - Mittelwert
    - für Werte eines einzelnen Sensors über ca. 60s
    - Licht-, Beschleunigungs-, Temperatur- und Drucksensoren
  - Standardabweichung
    - für Werte über mehr als 60s
    - Licht- und Beschleunigungssensoren, passive IR-Sensoren
  - Basisfrequenz
    - Lichttypen (bspw. Flackern), Beschleunigungsmuster in Bewegung
  - erste Ableitung
    - passive IR-Sensoren, Beschleunigungssensoren

TEA – Beispiel

Exklusive Kontexte

- Regelmenge zur Erkennung von Situation in denen das Gerät in der Hand, auf dem Tisch oder im Koffer ist (basierend auf Licht- und Beschleunigungsdaten in der XY-Ebene)
- Konstanten `Dx`, `Dy`, `L`, `Xnormal`, `Ynormal`, `D`, und `Q` vorab auf Basis von Erfahrungswerten/Beobachtungen definiert
- Einschränkung
  - des Orts des Mobilgeräts: Hand, Tisch oder Koffer
  - der Gerätenutzung: innen/außen, Pkw/Bus/Zug, stationär/mobil

```
Hand(t):-        standard_deviation(accelX,t) > Dx,
                 standard_deviation(accelY,t) > Dy,
                 % Gerät bewegt sich leicht in X- und Y-Richtung
                 average(light,t) > L. % es ist nicht dunkel
Table(t):-       abs(average(accelX,t) - Xnormal) < D,
                 abs(average(accelY,t) - Ynormal) < D,
                 % Gerät ist in XY-Ebene tariert
                 quartile(accelX,t) < Q, quartile(accelY,t) < Q
                 % Gerät bewegt sich nicht
                 average(light,t) > L. % es ist nicht dunkel
Suitcase(t):-    average(light,t) < L. % es ist dunkel
```

Verwaltung von Kontextinformationen [Korpipää et al.]

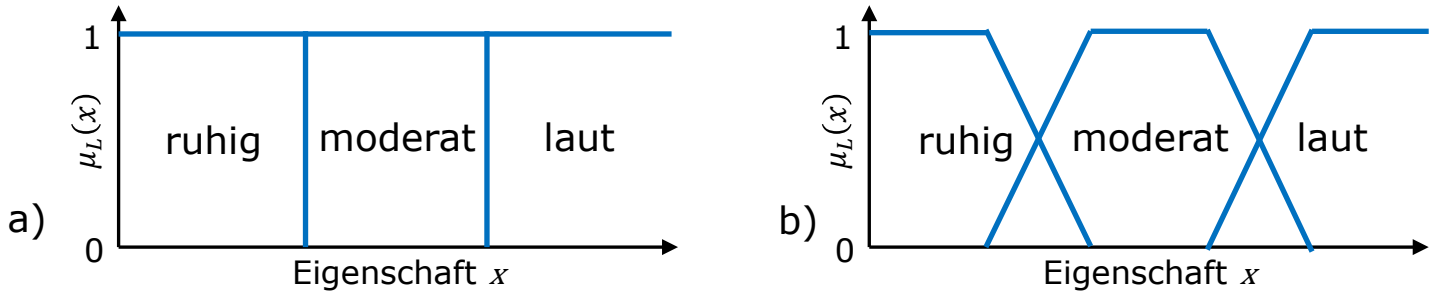
- Anwendungsszenario
  - Umgebungslautstärke bestimmt Lautstärke der Systemgeräusche
  - Schriftgröße, Bildschirmhelligkeit und Dienstinhalt werden an Nutzeraktivität und Umgebungshelligkeit angepasst
- verwendete Sensoren
  - Mikrofon, Beschleunigungssensor, Zwei-Kanal-Lichtsensor, Thermometer, Hygrometer/Psychrometer, Berührungssensor
- berechneter Kontext höherer Ebene
  - `Location {indoor, outdoor}`
  - `SoundType {Car, Elevator, RockMusic, ClassicalMusic, TapWater, Speech, OtherSound}`

Verwaltung von Kontextinformationen [Korpipää et al.]

1. Phase: Abstraktion der Sensorrohdaten (ähnlich Signalen)

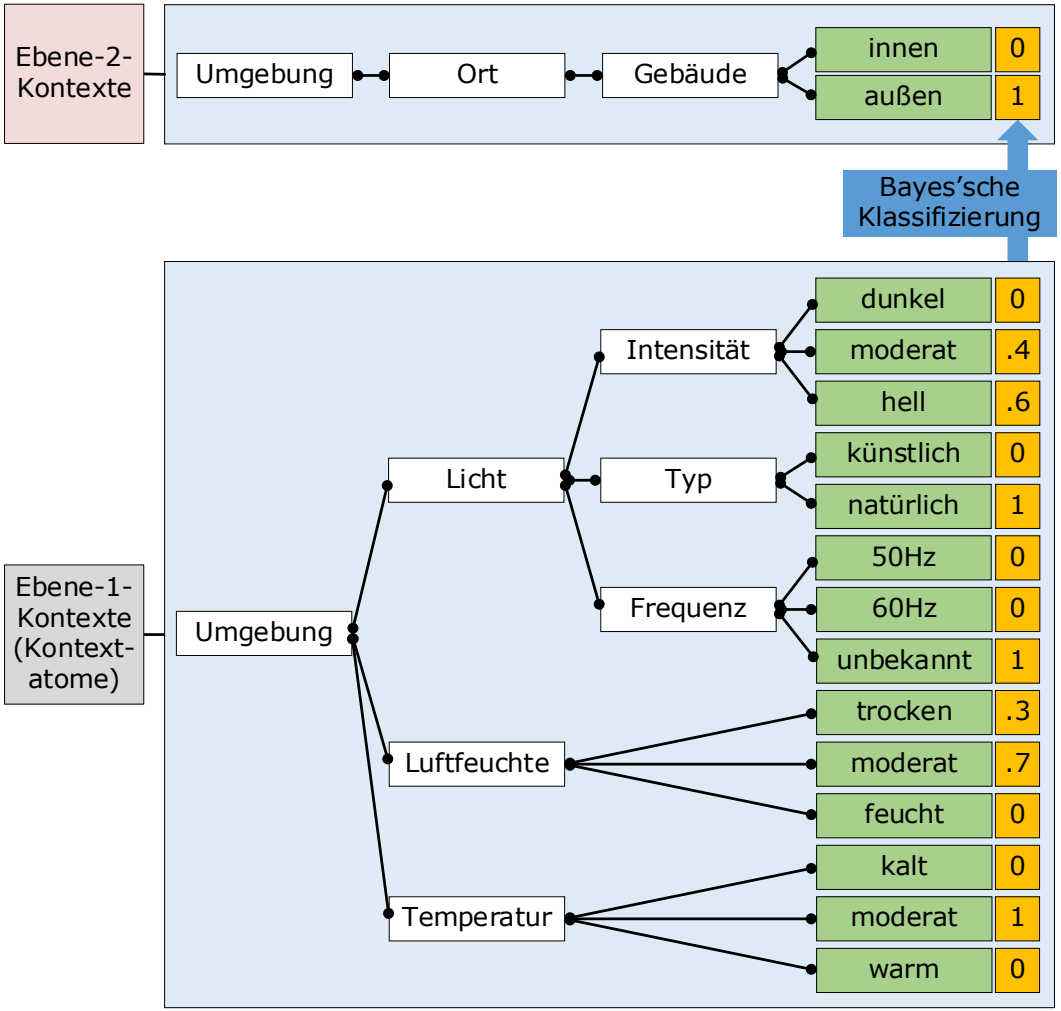
   a) scharfe Grenzen: wahr/falsch-Markierung der Sensordaten (bspw. Umgebungsgeräusche: ruhig, moderat, laut)

   b) schwammige Grenzen: sich überlagernde Bereiche (bspw. [0,10] ruhig, [7,23] moderat, [17,30] laut)



Verwaltung von Kontextinformationen [Korpipää et al.]

2. Phase: Berechnen des höherebenen Kontextes durch ein naives Bayes'sches Netz



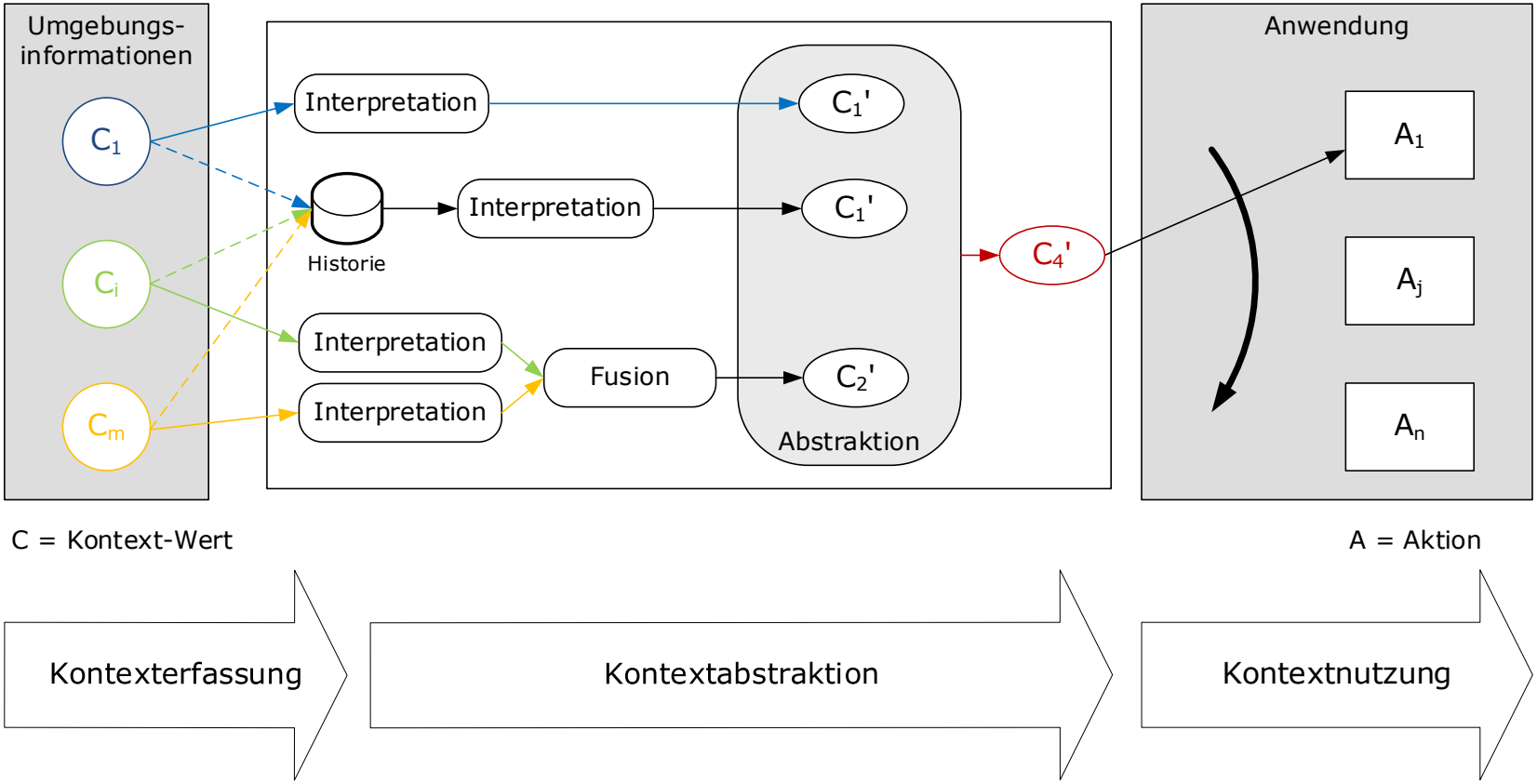
Kontextabstraktion – Regelbasierter Ansatz

Situation	Argumentationsregel
schlafend	(?u locatedIn Bedroom) ∧ (Bedroom lightLevel LOW) ∧ (Bedroom drapeStatus CLOSED) ⇒ (?u situation SLEEPING)
duschend	(?u locatedIn Bathroom) ∧ (WaterHeater locatedIn Bathroom) ∧ (Bathroom doorStatus CLOSED) ∧ (WaterHeater status ON) ⇒ (?u situation SHOWERING)
kochend	(?u locatedIn Kitchen) ∧ (ElectricOven locatedIn Kitchen) ∧ (ElectricOven status ON) ⇒ (?u situation COOKING)
fernsehend	(?u locatedIn Livingroom) ∧ (TVSet locatedIn Bathroom) ∧ (TVSet status ON) ⇒ (?u situation WATCHINGTV)
essend	(?u locatedIn Diningroom) ∧ (?v locatedIn Diningroom) ∧ (?u owl:differentFrom ?v) ∧ (WaterHeater status ON) ⇒ (?u situation HAVINGDINNER)

- Anwendungsfall: Smart Home
- Schlussfolgerungen über Situation des Nutzers zu Hause
- Modellierung:
  - CONON (CONtext ONtology)
  - OWL, RDF Triples
- Argumentation
  - regelbasiert (einfache Mustererkennungsalgorithmen)
  - basierend auf OWL light und RDF-Tripel

Kontextabstraktionsprozess

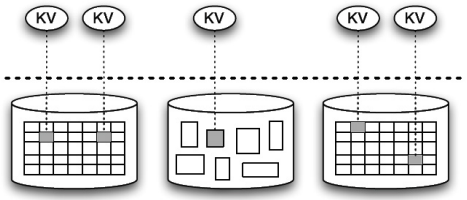
Temperatur	(12°C)	kalt	Wolkenbildung,	empfehle dem Nutzer
Luftfeuchte	(90%)	hoch	Regen und Sturm	die Radfahraktivität
Luftdruck	(1005hPa)	hoch	sehr wahrscheinlich	zu verschieben

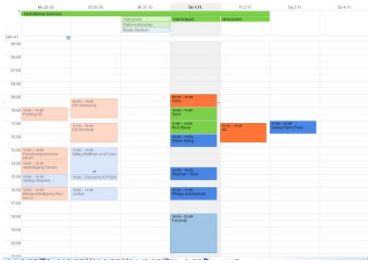


Mögliche Quellen der Kontexterfassung

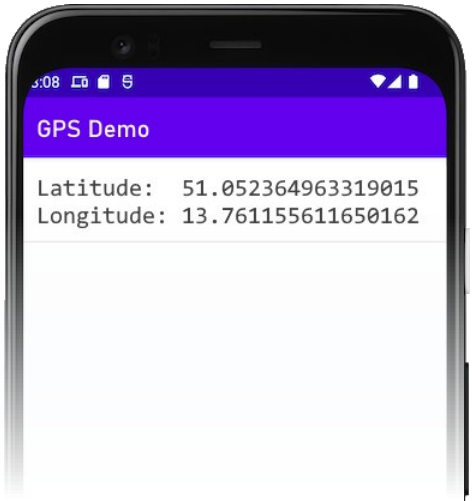
Kontextquellen

- Sensorgeräte
  (GPS, Thermometer, Fotodiode, Mikrofon, …)
- Datenbanken
  (bspw. Kontextextraktion aus strukturierten Daten in der Datenbank)
- Anwendungsdaten (bspw. Terminplaner)
- Nutzerbeobachtung und -eingaben
  (bspw. Ort und Art der aktuellen Nutzeraktivität)





Ermittlung der GPS-Koordinaten in Android




```
public class GpsActivity extends Activity {
    private FusedLocationProviderClient fusedLocationClient;
    // …

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // …
        fusedLocationClient = LocationServices
            .getFusedLocationProviderClient(this);
        fusedLocationClient.getLastLocation()
            .addOnSuccessListener(this, new OnSuccessListener<Location>() {
                @Override
                public void onSuccess(Location location) {
                    // Lokationsbezug kann NULL zurückgeben…
                    if (location != null) {
                        // Applikationslogik für Verarbeitung der Position
                        TextView tv = new TextView(GpsActivity.this);
                        tv.setText("Latidude:  " + location.getLatitude() + "\n"
                                + "Longitude: " + location.getLongitude());
                        setContentView(tv);
                    }
                }
            });
    }
}
```

Sensorzugriff in Android

- Zugriff über das Android SDK
  - `android.hardware.SensorManager`
  - `sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);`
- Prüfen, ob Sensor unterstützt wird
  - `sensorManager.getSensorList(SensorManager.SENSOR_ALL)`

Observer Pattern

- Abonnieren von Updates wenn die Activity im Vordergrund läuft

```
protected void onResume( ) {
    super.onResume( );
    sensorManager.registerListener(accListener,
        sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
        sensorManager.SENSOR_DELAY_NORMAL);
}
```

- EventListener beobachtet Änderungen der Seonsordaten

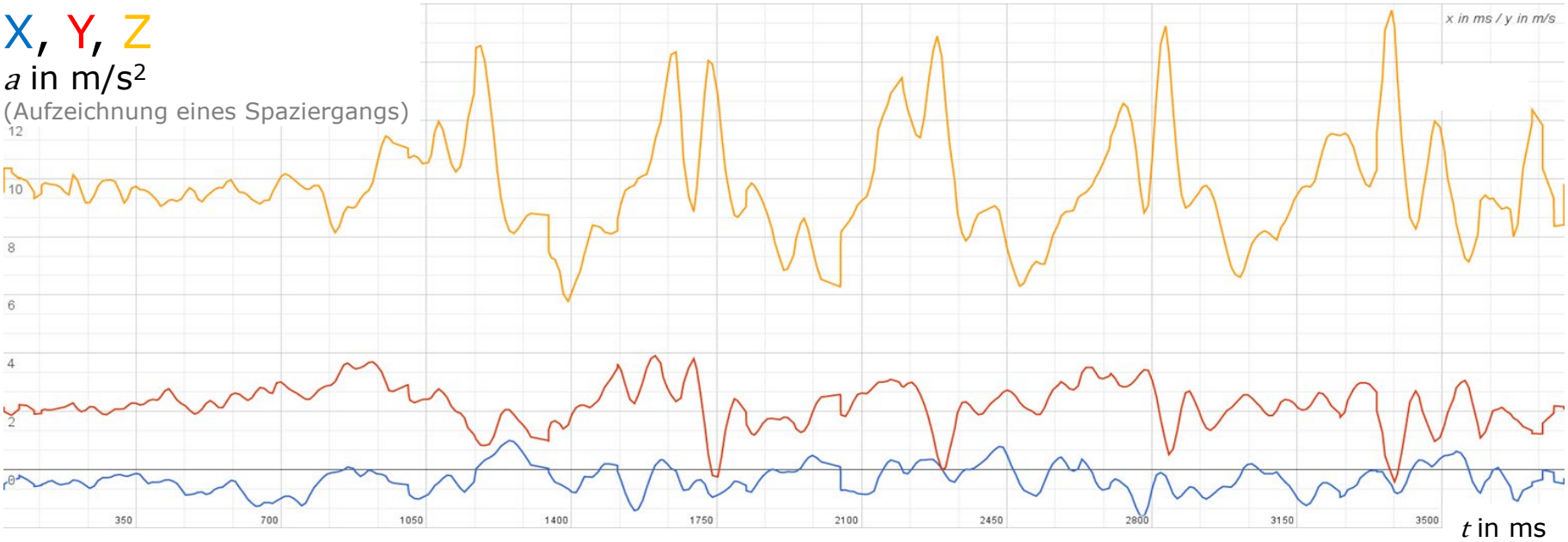
```
AccListener implements SensorEventListener {
    public void onAccuracyChanged(Sensor sensor, int accuracy) {…}
    public void onSensorChanged
class (SensorEvent event) {…}
```

Abstraktion der low-level-Kontextinformation

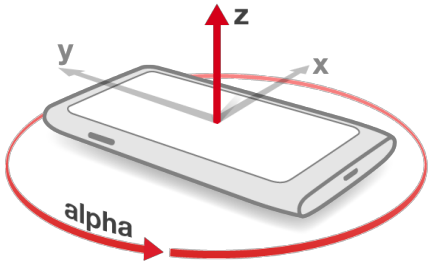
X, Y, Z

a in m/s²

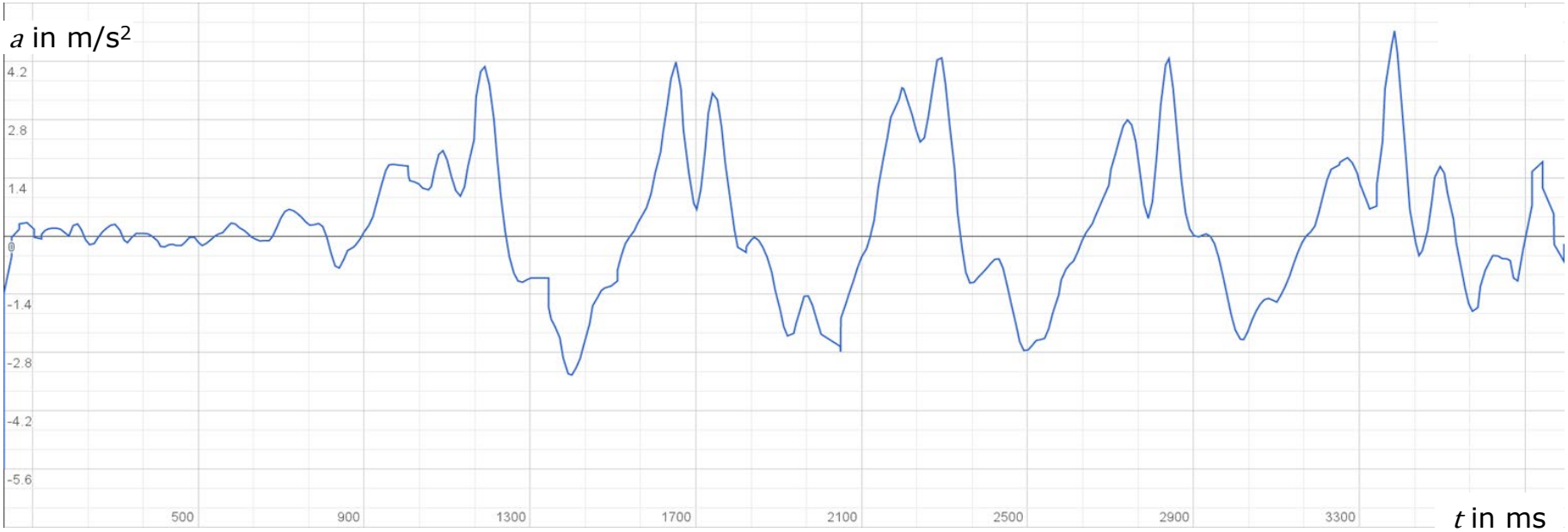
(Aufzeichnung eines Spaziergangs)



- Daten jeder Achse als eigene Funktion
- signifikante Muster für unterschiedliche Bewegungstypen (bspw. Schrittmuster in der Z-Achse)
- zu beachtende Herausforderungen
  - Ergebnisse sind achsenabhängig
  - Gravitation beeinflusst Beschleunigungssensor

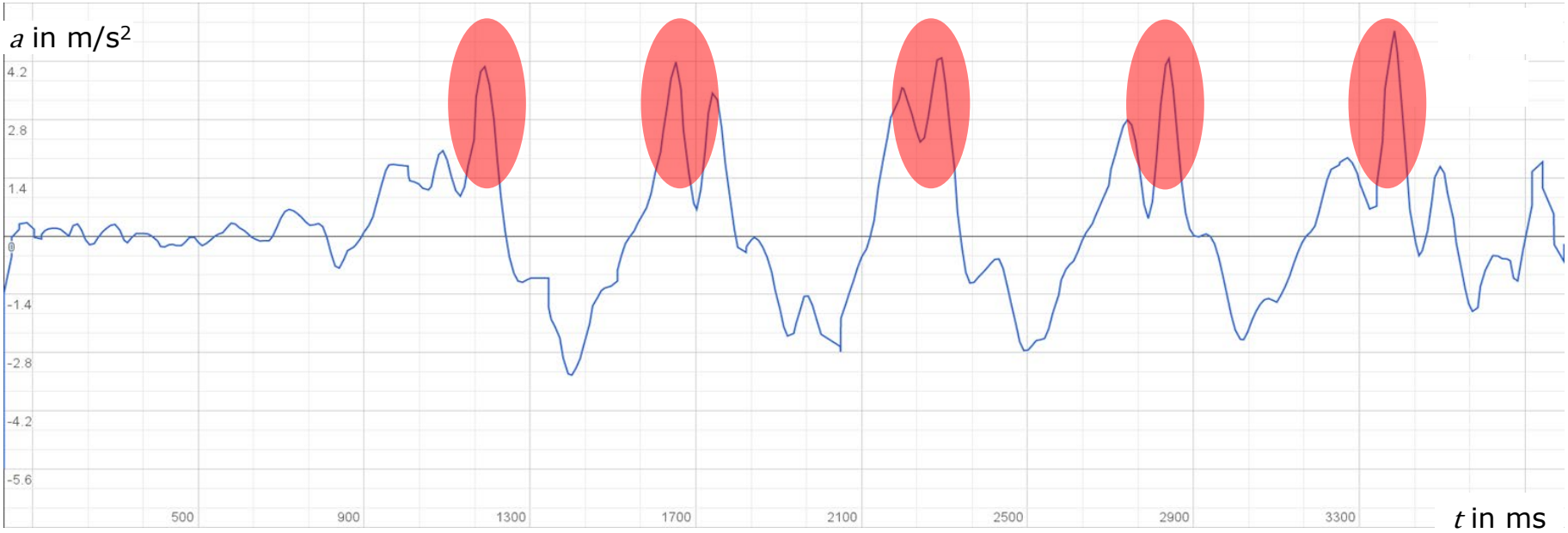


Schritterkennung mit dem Beschleunigungssensor



- um etwas erkennen zu können, ist Datenverarbeitung entscheidend
  - Achsennormalisierung (Inertialsystem)
  - Herausrechnen der Gravitation
  - Filtern der Rohdaten mit Tiefpass
- liefert eine signifikante, ausrichtungsunabhängige Kurve

Schritterkennung mit dem Beschleunigungssensor

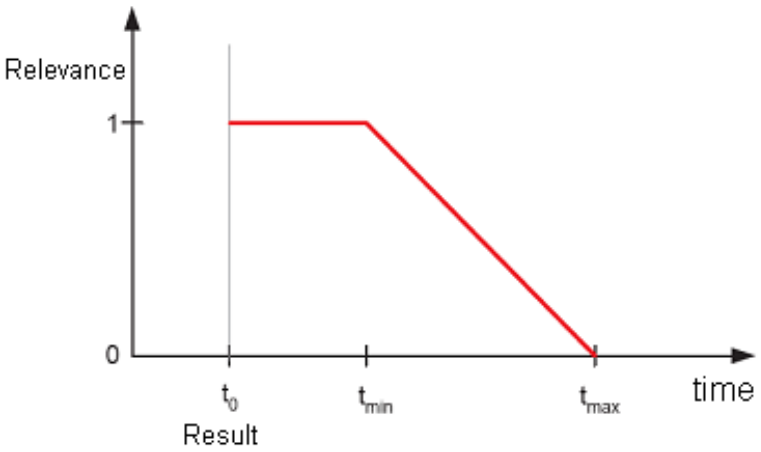


- Schritterkennung durch lokale Maxima der Datenfunktion
- daraus lässt sich ableiten
  - Schrittanzahl und -frequenz
  - zurückgelegte Distanz (falls Schrittlänge bekannt; sonst Schätzung)
  - mit Distanz: Geschwindigkeit

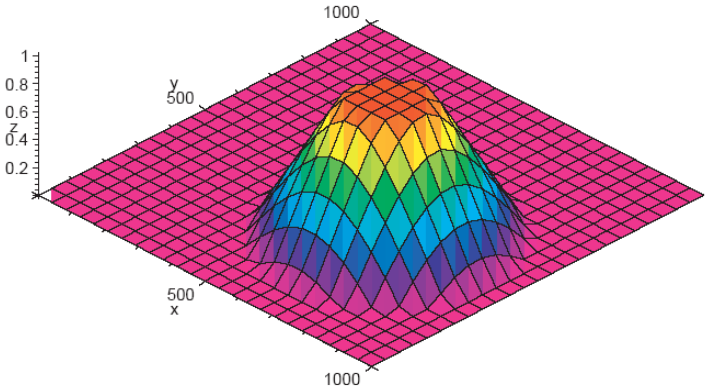
⇒ Aktivität = Joggen, falls Geschwindigkeit und Schrittfrequenz innerhalb definierter Randbedingungen

Eigenschaften von Kontextattributen

- Zeitabhängigkeit
  - höchste Relevanz im Moment der Erfassung
    → nimmt konstant ab
  - Historie: repräsentiert Werte zu verschiedenen Zeitpunkten

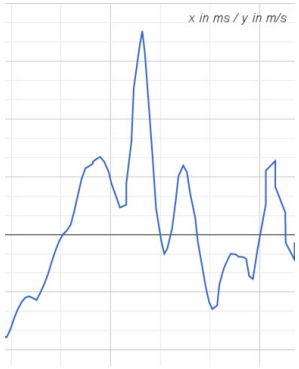


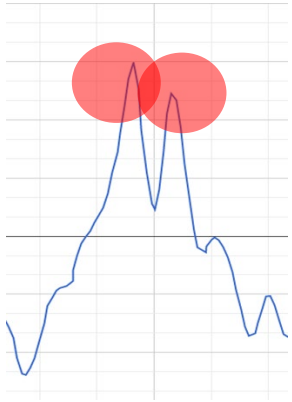
- Ortsabhängigkeit
  - höchste Relevanz am Erfassungsort
    → nimmt mit zunehmender Entfernung ab



Eigenschaften von Kontextattributen

- Imperfektion
  - unbekannte Art – Sensorfehler oder mangelnde Verfügbarkeit (bspw. funktioniert GPS nicht im Innenraum)
  - fehlerhafte Daten– auf Grund von Messfehlern oder wegen falscher Annahmen
    - bspw. wenn kein klares Muster erkannt werden kann
    - bspw. führt falsche Schrittlängenannahme zu falscher Distanz
  - inakkurate Daten – auf Grund von systematischen Fehlern oder zu hoher Toleranz in der Anwendung oder Heuristik
    - bspw. zu viele erkannte Schritte wegen fehlerhafter Extrempunktbestimmung
  - mehrdeutige Daten – auf Grund sich widersprechender Werte
    - bspw. Divergenz zwischen Kompass und aus GPS-Daten ermittelter Bewegungsrichtung



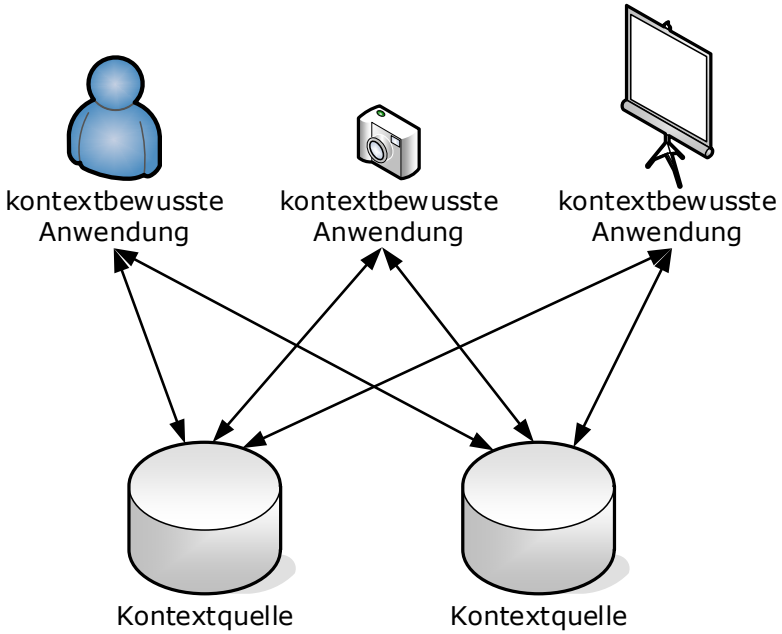


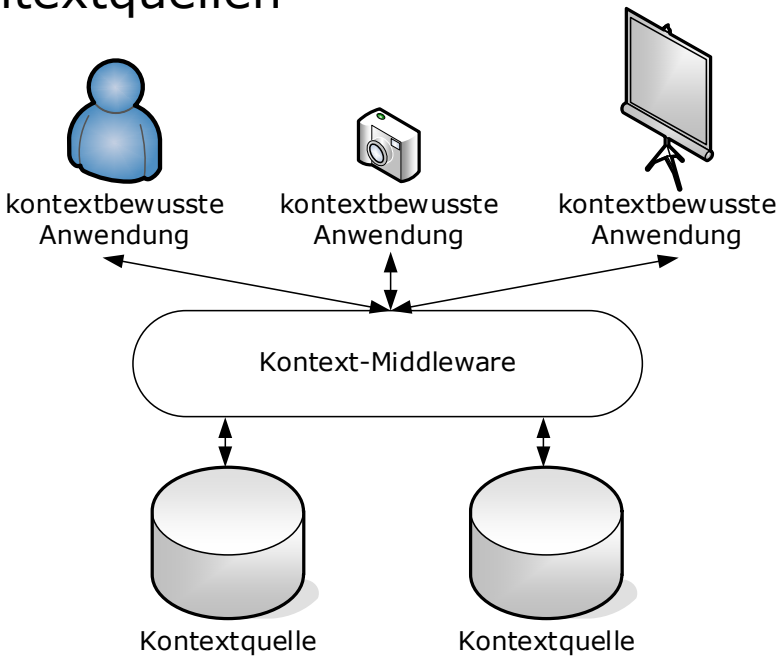
Kontext-Middleware

- Ziele
  - Verfügbarkeit des abstrakten Kontextes für Anwendungen
  - Austausch des Kontextes zwischen Anwendungen
- Aspekte
  - Architekturen
  - Kontextmodellierung

Warum Kontextvermittlung?

- Wiederverwendbarkeit generischer Funktionalität
  - Sammeln, Abstrahieren, Speichern, Transport und Anwendungszugriff
  - vermittelt Kontextinformationen zwischen Kontextquellen und Kontextnutzern
  - ermöglicht Kontextaustausch zwischen Anwendungen
- Interessenstrennung (Separation of Concerns)
  - entkoppelt Anwendungen und Kontextquellen

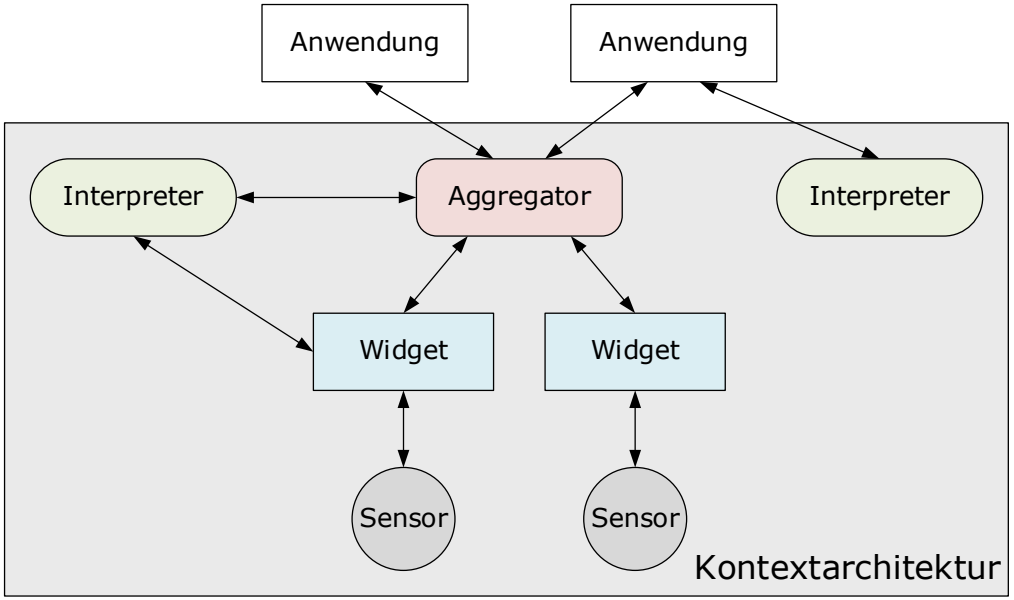




Kontext-Toolkit

Widget

- Sensorabstraktion
- repräsentiert einzelnen Kontextwert
- kapselt Details von Sensoren und anderen Quellen
- aktueller Wert, Historie und Abonnements



Interpreter

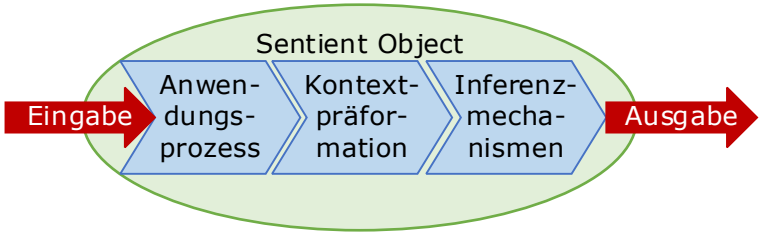
- Verarbeitung von Sensordaten
- Generierung des höherebenen Kontextes
- ein oder mehr Eingabewerte
- kann von Widgets, Aggregatoren, anderen Interpretern und Anwendungen verwendet werden

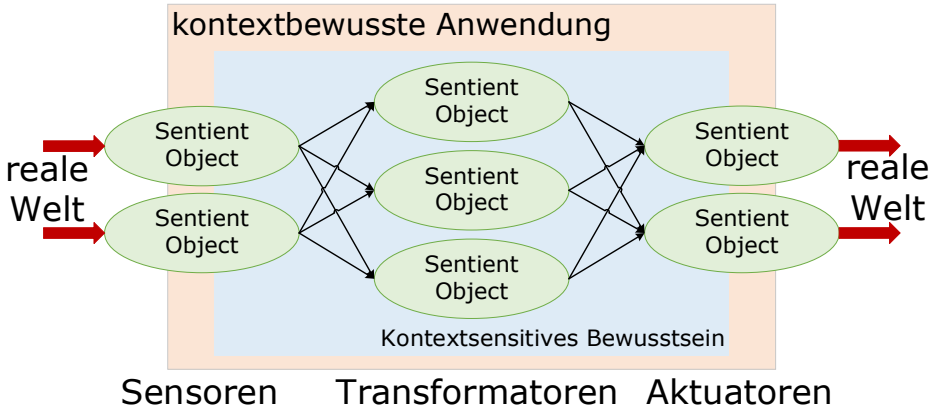
Aggregator

- weist diverse Kontextdaten (Widgets, Interpreter) Entitäten zu
- vereinfacht die Verwaltung

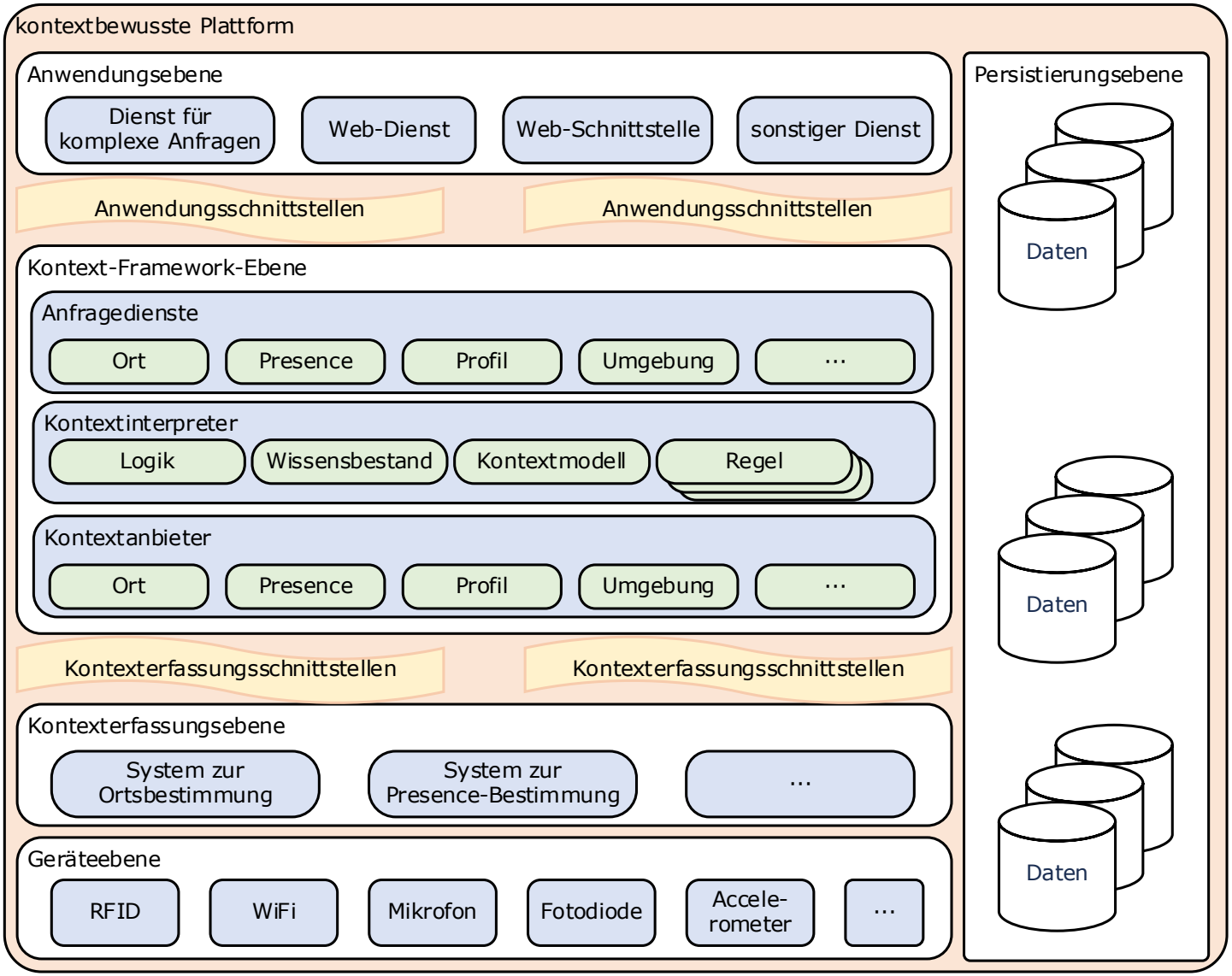
Bewusste Verarbeitung (Sentient Computing)

- Systembausteine im bewussten Zusammenhang (Sentient Objects)
- jedes Sentient Object repräsentiert kontextbewusste Funktionalität für Erfassung, Verarbeitung oder Anwendung von Kontextinformationen
- unterschiedliche Objekttypen
  - Sensoren
  - Transformer
  - Aktuatoren
- Sentient Objects enthalten interne Repräsentation des Kontextes und den zugehörigen Transformationskode (bspw. Inferenzmechanismen)
- Kombination von Sentient Objects bildet kontextbewusste Anwendung





Kontext-Middleware



Kontext-Middleware

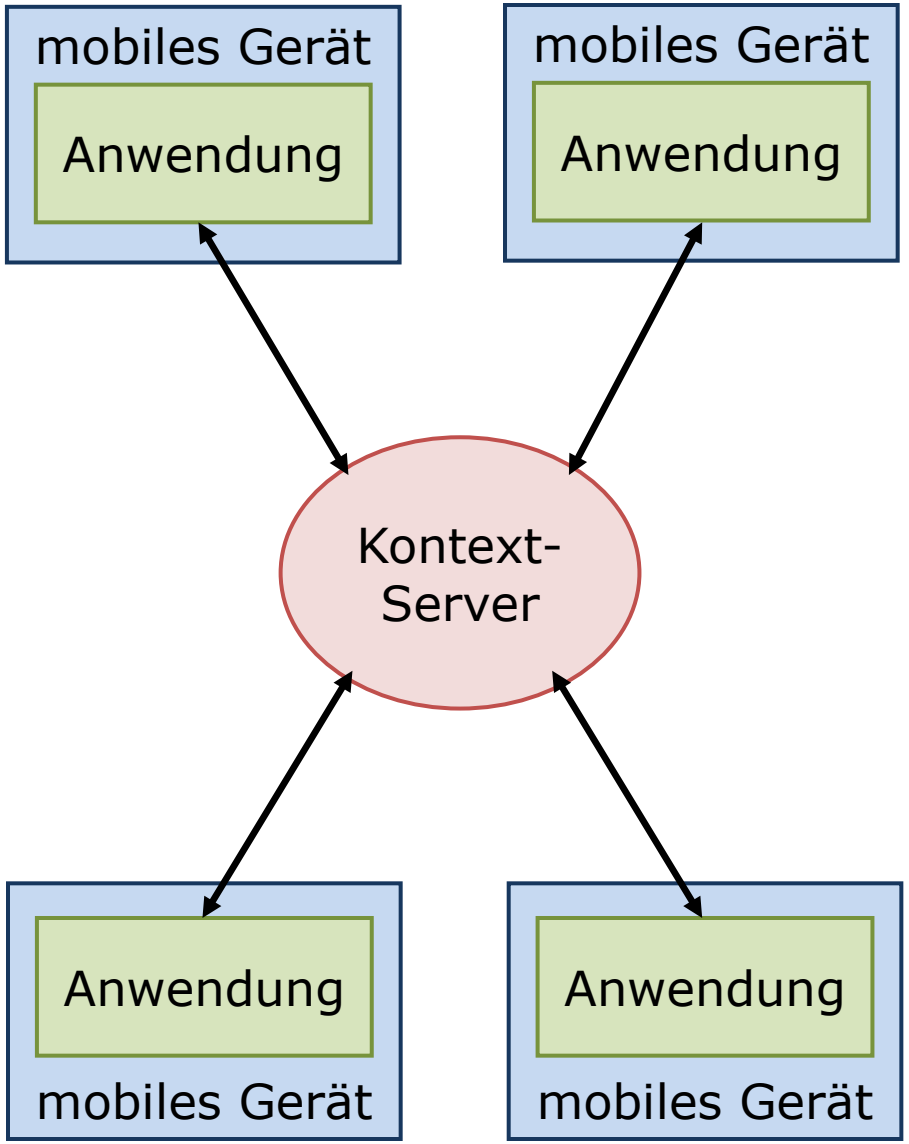
5 Ebenen mit modularem Design (verteiltes Bereitstellung möglich)

1. Persistierungsebene
   - persistente Speicherung von Kontextinformationen (aktuell, historisch)
2. Geräteebene
   - alle Geräte die Kontextinformationen liefern können
3. Kontexterfassungsebene
   - ermittelt spezielle Kontextinformationen
     - Ort, Profilinformationen, etc.
   - implementiert Schnittstellen zur Kontext-Framework-Ebene
4. Kontext-Framework-Ebene
   - erhält Kontextinformationen aus Kontexterfassung; übersetzt nach OWL
   - regel-basierte Logik zur Ableitung höherebenen Wissens
   - implementiert Schnittstellen zur Anwendungsebene
5. Anwendungsebene
   - Schnittstellen und Web-Dienste zur Abfrage der Wissensdatenbank

Zentraler Kontext-Server

+ Server bietet hohe Speicher- und Berechnungskapazitäten
+ einfach zu finden und nutzen (Discover & Manage)
+ zentrale Zugriffskontrolle

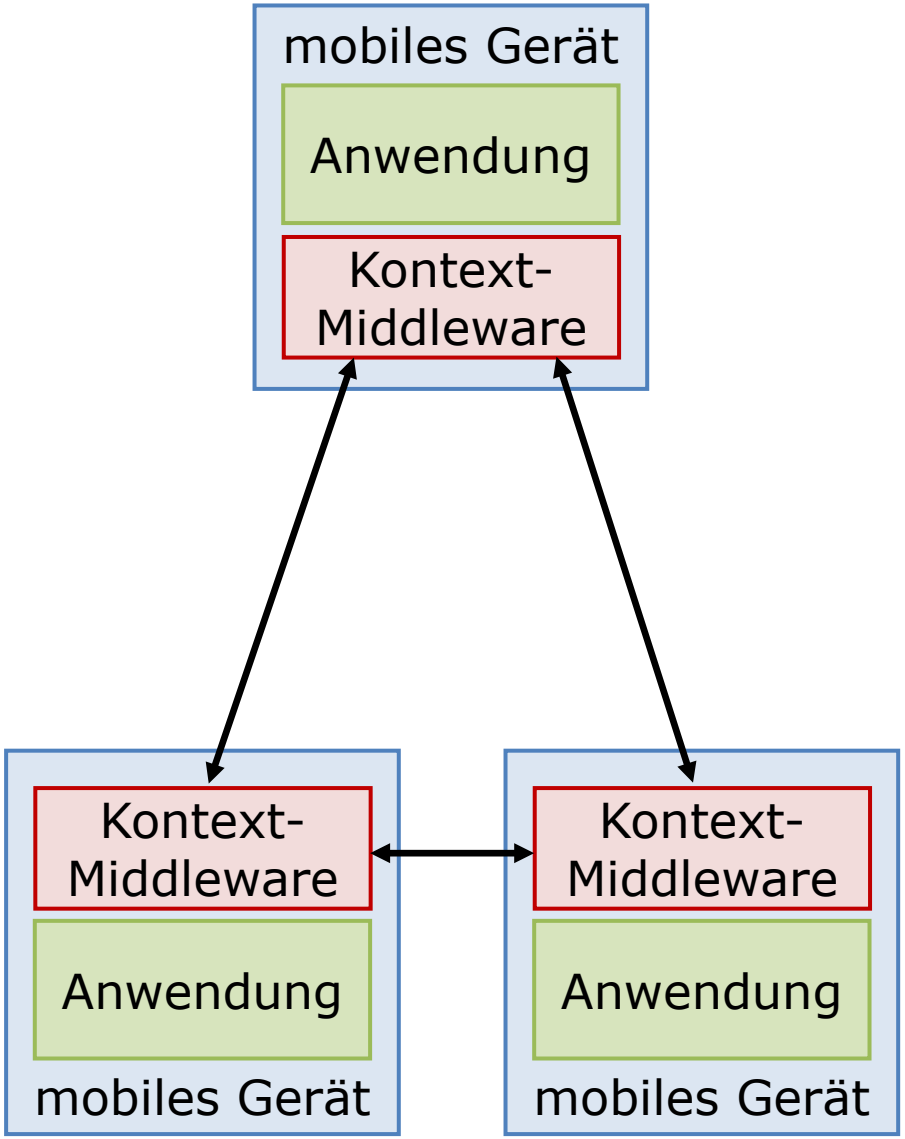
– alleiniger Ausfallpunkt (Single Point of Failure)
– begrenzte Skalierbarkeit der zentralen Ressourcen über sehr viele Clients
– Netzwerkverbindung zwischen mobilen Geräten und Server zwingend für Kontextzugriff



DEZENTRALISIERTE ARCHITEKTUR

+ unabhängig von Netzwerk-verbindung zu bestimmtem Server
+ unterstützt ad-hoc-Szenarien
+ hohe Skalierbarkeit auf Grund Verteilung des Kontextzugriffs

– Informationsverfügbarkeit korreliert stark mit Geräteverfügbarkeit
– begrenzte Nutzerprofilverfügbarkeit
– begrenzte Speicher- und Berechnungskapazitäten auf mobilen Geräten
– keine zentrale Kontrolle über Kontextzugriff

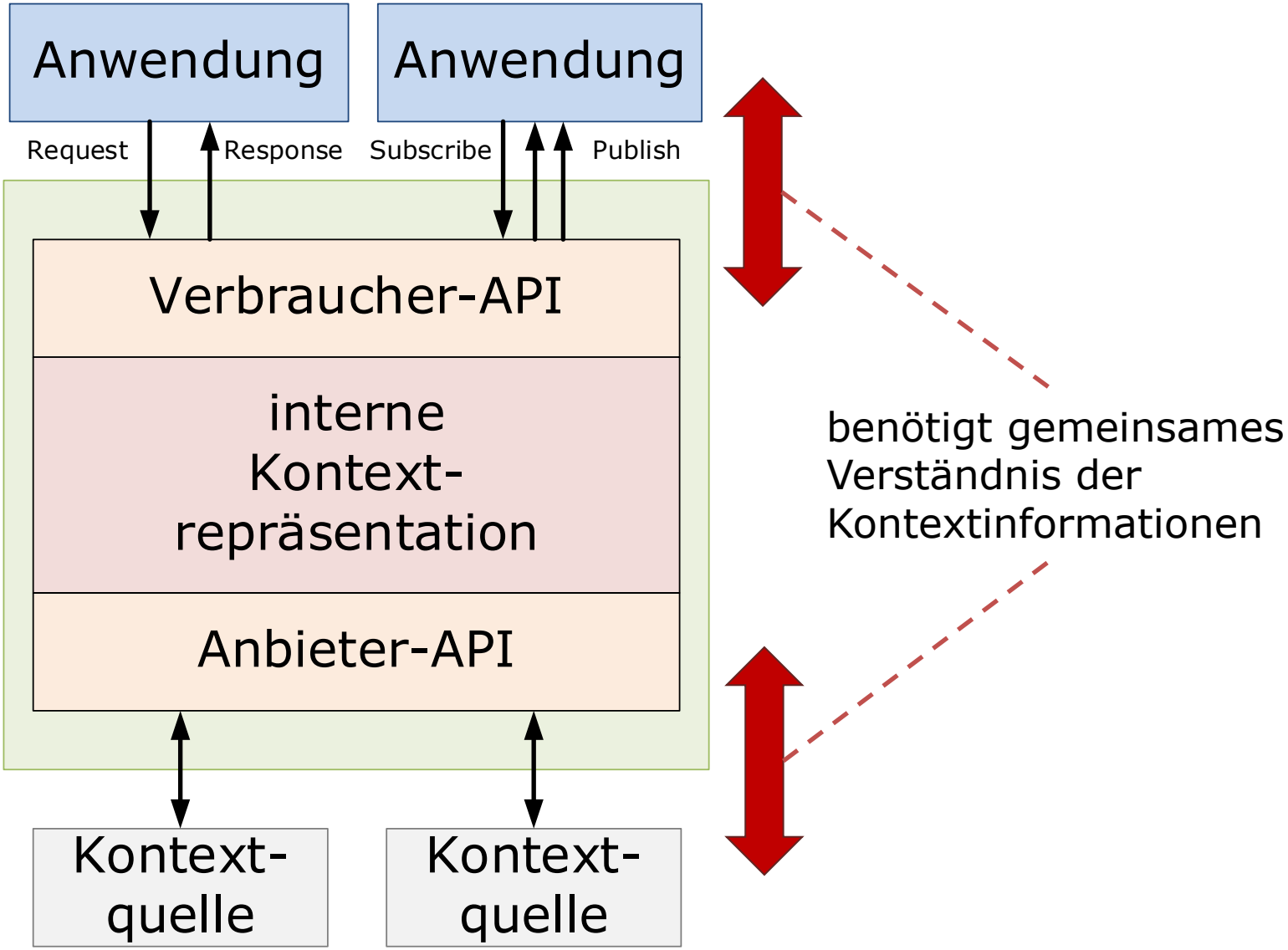


Informationskategorien

- Zustandsinformationen
  - Anwendung fordert explizit benötigten Kontext an (Pull; Request/Response-Modell)
  - Zugriff auf aktuelle und historische Daten
  - Beispiele: aktuelle Position, Geräte-Identifikation, Verbindungstyp
- Änderungsereignisse
  - Anwendung abonniert bestimmte Ereignisse und wartet passiv
  - Kontextdienst benachrichtigt Abonnenten über Zustandsänderungen, die das Abonnement betreffen im Moment der Änderung (Push; Publish/Subscribe-Modell)
  - Beispiele: Ortsänderung, Netzwerkunterbrechung, Batterieschwelle

	Zustand	**Ereignis**
Information	Aktuelle Position des Nutzers, Liste der Drucker in der Nähe	Person X betritt den Raum
Aktion	Drucke ein Dokument auf dem nächstgelegenen Drucker	Beginne Kommunikation mit Person X nachdem diese den Raum betreten hat

Kontext-Middleware-API

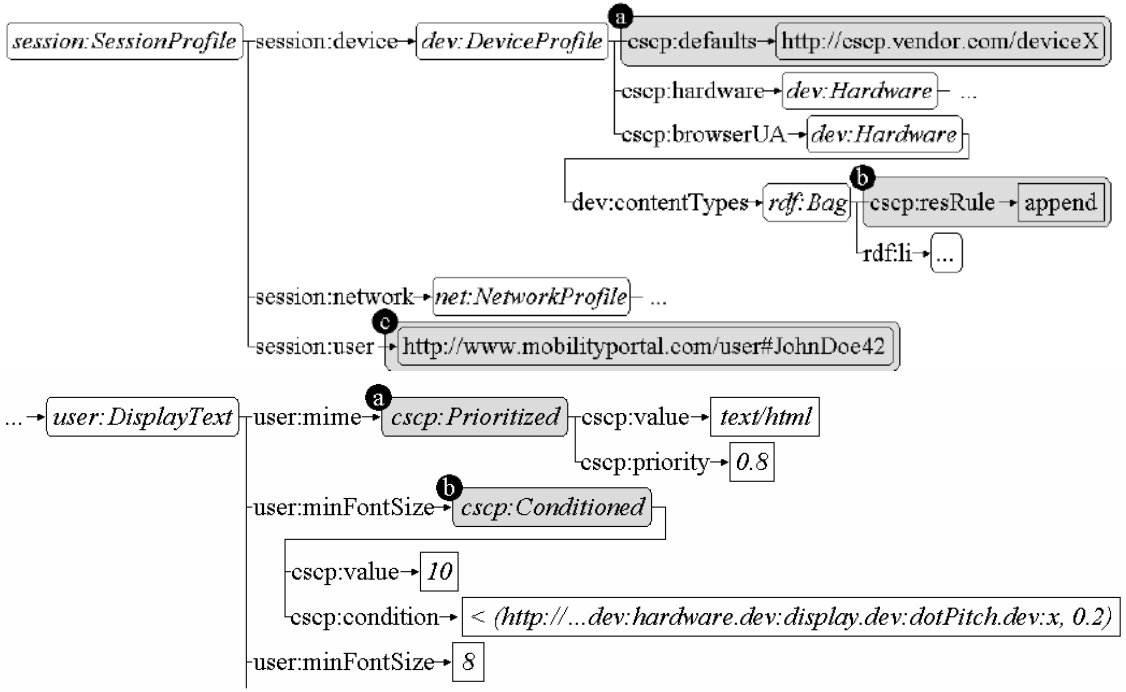


Kontextmodellierung

- gemeinsames Verständnis der Kontextinformationen zwischen Quellen und Middleware
- Teilen der Kontextinformationen zwischen Middleware und Anwendung
- Modell für Anwendungsentwicklung
  → Basis für die Entscheidungsfindung
- Ansätze
  - Kontextprofile
  - Modelle mit Objektrollen (Object Role Models)
  - ontologiebasierte Modelle

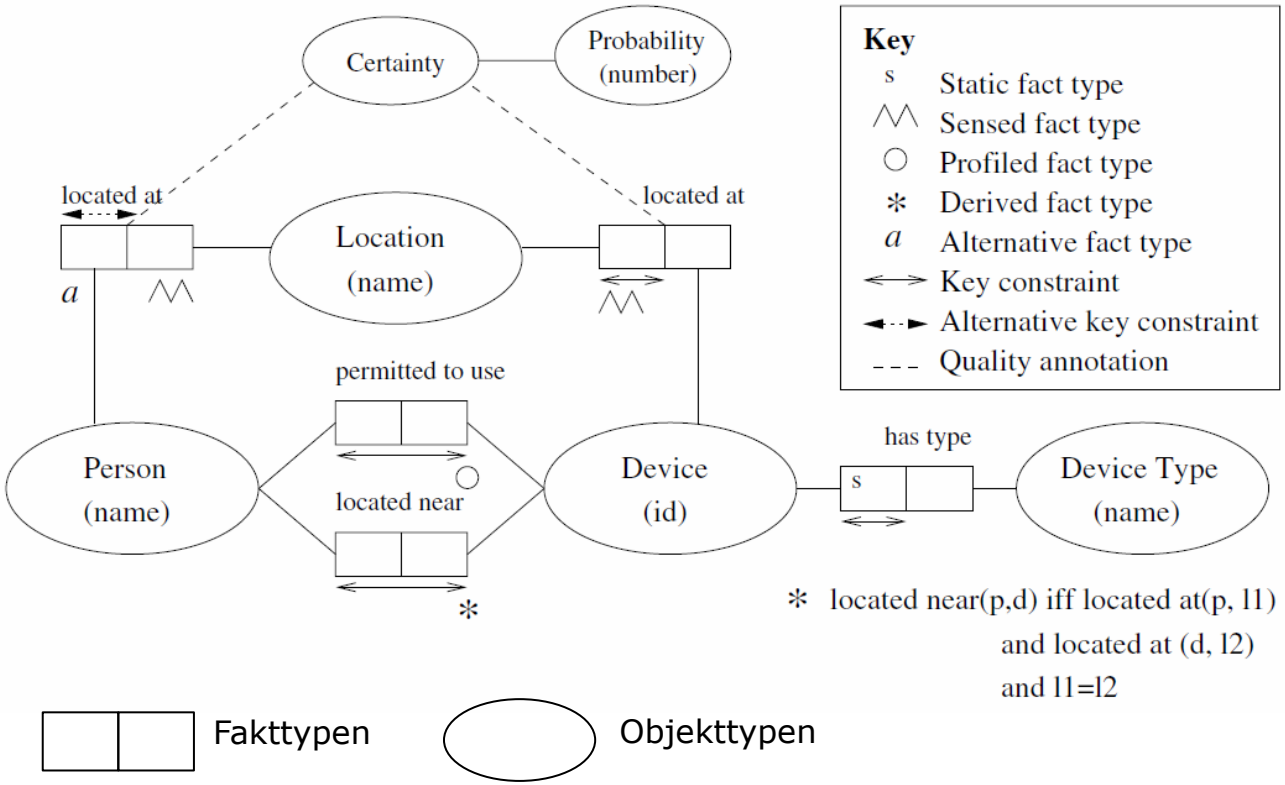
KONTEXTPROFILE

- Schlüssel/Wert-Paare (Key/Value Pairs):
  - `username="John Doe", temperature=21`
- Profile
  - Tripel – Subjekt, Prädikat, Objekt
    - `<https://www.example.com> - author - "John Doe"`
  - RDF (Resource Description Framework)
  - CC/PP (Composite Capabilities/Preference Profiles)
  - CSCP (Comprehensive Structured Context Profiles)



Object Role Model [Henricksen, Indulska]

- Kontext wird auf Basis von Fakten, Objekten und Rollen modelliert
- Qualitätsinformationen können hinzugefügt werden
- Modell kann auf relationale Datenbank abgebildet werden



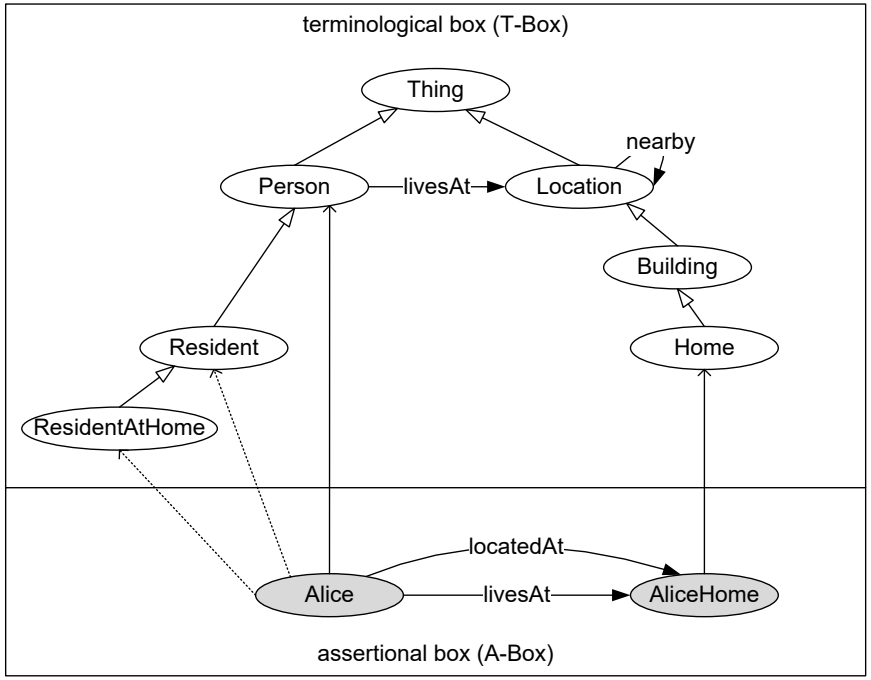
ONTOLOGIEBASIERTE KONTEXTMODELLE

Eine Ontologie ist eine explizite Spezifikation einer Konzeptualisierung.

[Gru93]

- beschreibt reale Welt als formalisierte Konzepte
- beinhaltet Konzepte und Rollen (T-Box)
- Individuen als Instanzen eines Konzepts (A-Box)

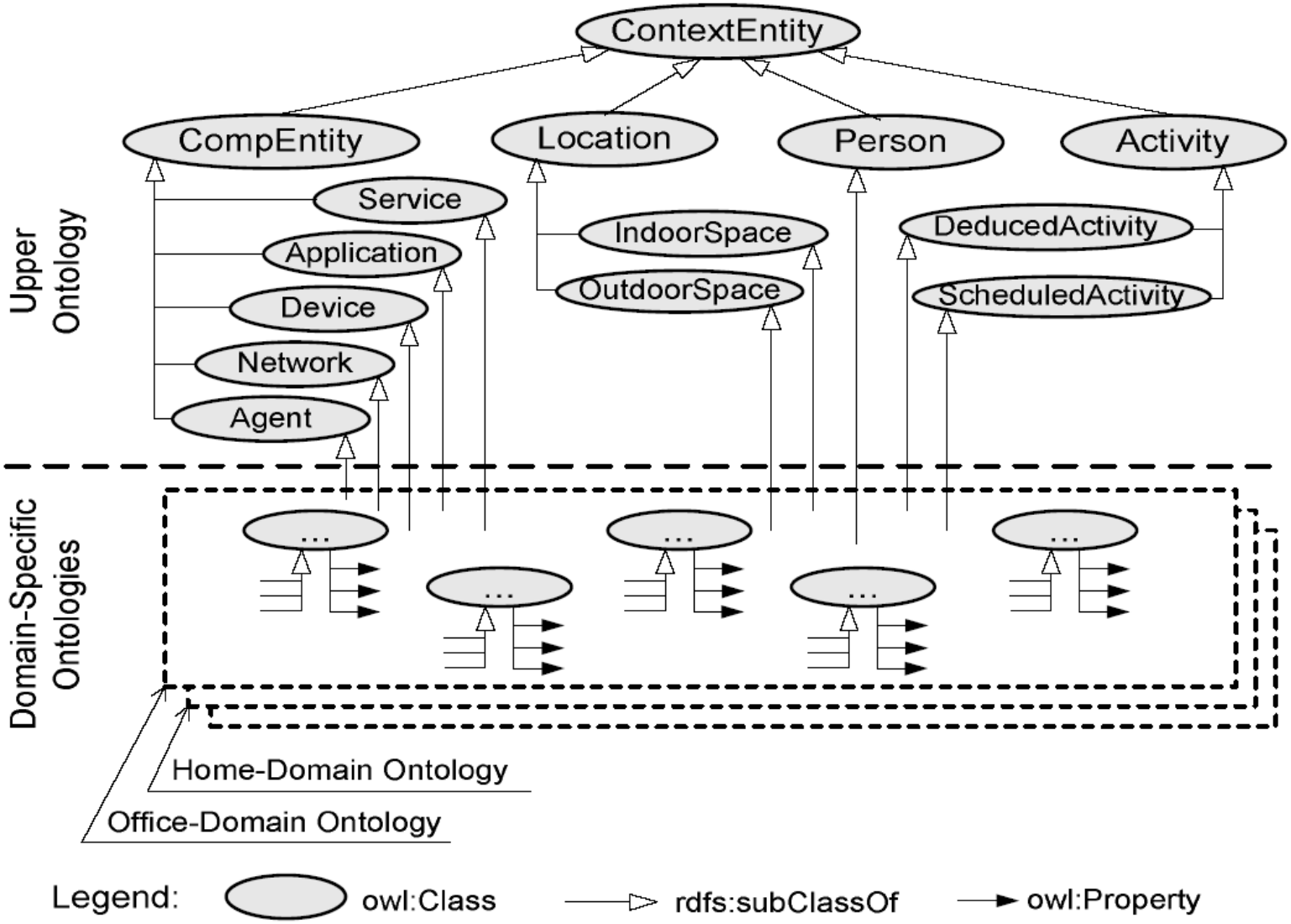
- Web Ontology Language (OWL)
  - definiert durch das W3C
  - basiert auf RDF
  - 3 Varianten
    - OWL lite
    - OWL DL
    - OWL Full



Resident = Person and (livesAt some Home)

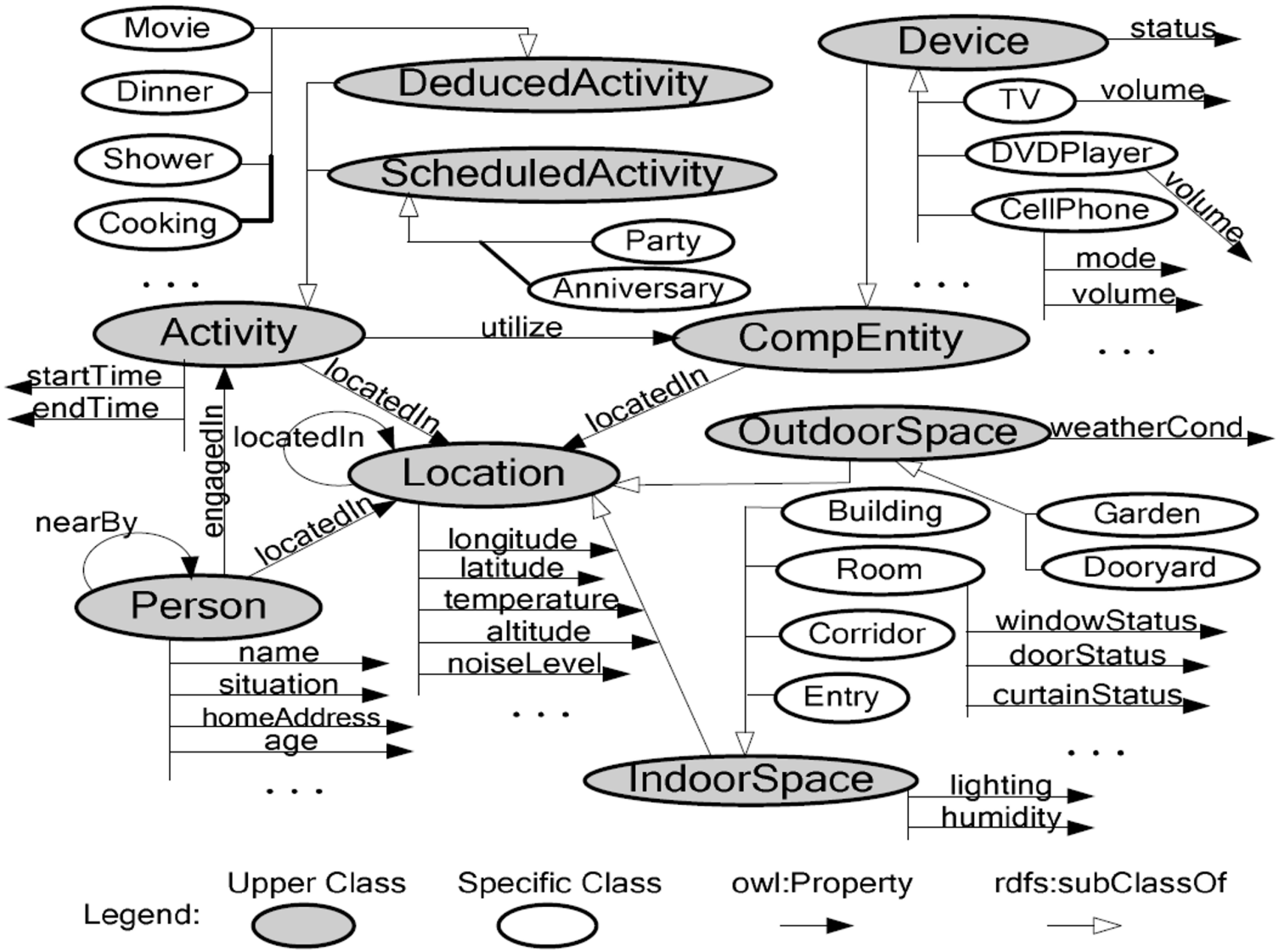
ResidentAtHome = Resident and (locatedAt some (Home or (isPartOf some Home)))

CONON – CONTEXT ONTOLOGY



Infocom Research, Singapore (Zhang)

CONON – Domänenspezifische Ontologie



Infocom Research, Singapore (nachZhang)

Vergleich der Kontextmodelle

- Schlüssel/Wert-Paare
  - \+ simpelster Ansatz: einfach zu verwalten
  - – fehlende Struktur und Semantik
- Kontextprofile
  - \+ hierarchisch strukturiert; einfache Semantik
  - – hauptsächlich für statische, selten geänderte Kontexte → keine Updates
  - – grobe Granularität für Kontextaustausch → ganze Profile oder Profilsegmente
- Object Role Model
  - \+ umfangreiche Struktur basierend auf Fakten, Objekten und Rollen
  - \+ kann auf relationale Datenbanken abgebildet werden
  - – keine weite Verbreitung
- Ontologien
  - \+ kann komplexes Wissen darstellen
  - \+ klare Struktur und Wiederverwendung basierend auf höheren Ontologien
  - \+ Inferenzen basieren auf Entscheidern der Beschreibungslogik
  - – hohe Komplexität; Performanz kann schlecht sein

Zusammenfassung

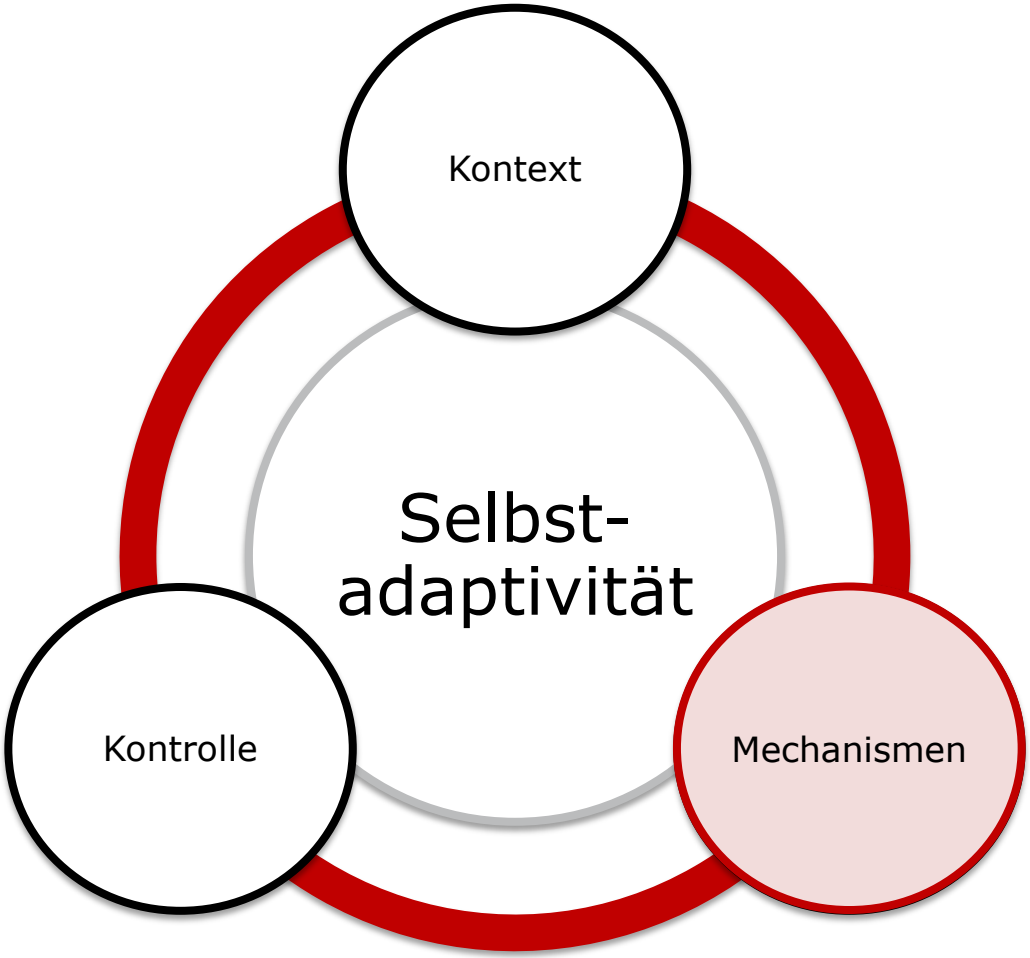
- Kontextinformationen ermöglichen Bewusstsein über die Ausführungsumgebung
- Voraussetzung zur Kontrolle des Adaptionsprozesses
  - bedarf in der Regel Abbildung von Kontextinformationen auf Kontrollparameter und -informationen
- Hauptphasen der Kontextverarbeitung
  - Erfassen
  - Abstraktion
  - Entscheidung
- Kontextmodelle erlauben Teilen der Kontextinformationen
- Kontext-Middleware stellt allgemeine Funktionalität für Erfassung, Verarbeitung, Speicherung, Verteilung und Zugriff auf die Kontextinformationen bereit

Aufgaben

- Überlegen Sie gemeinsam mit Ihren Kommilitonen konkrete Beispiele für Kontext aus Ihrem täglichen Umgang mit Smartphones.
- Wie adaptieren die von Ihnen zusammengetragenen Beispiele?
- Diskutieren Sie mit Ihren Kommilitonen verschiedene Sensoren und wie diese bei der Kontexterfassung eingesetzt werden können.
  → Welche Probleme können unterschiedliche Sensoren haben?
  → Können Sie sich Aktuatoren vorstellen, die bei der Adaption helfen?
- Diskutieren Sie mit Ihren Kommilitonen sinnvolle Szenarien für Kontext-Middlewares. Welche Komponenten sind allgemein wichtig? Welche sind eher optional?

Selbstadaptivität: Mechanismen

Kapitelübersicht



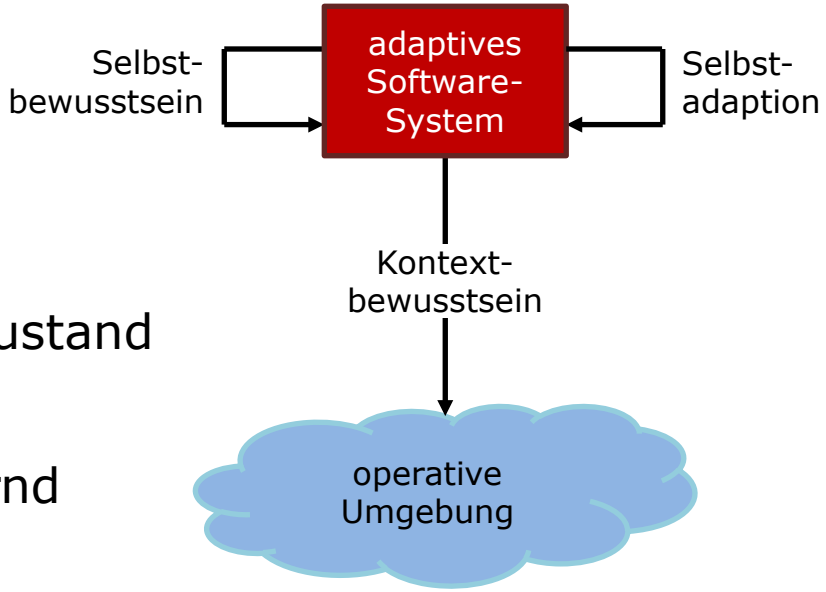
Wiederholung: Selbstadaptive Systeme

Selbstadaptive Software modifiziert ihr eigenes Verhalten als Reaktion auf Veränderungen ihrer Laufzeitumgebung. Dabei ist die Laufzeitumgebung alles, was von der Software erfasst werden kann, bspw. Nutzereingaben, Sensorinformationen oder Programmausstattung.

nach Oreizy et.al

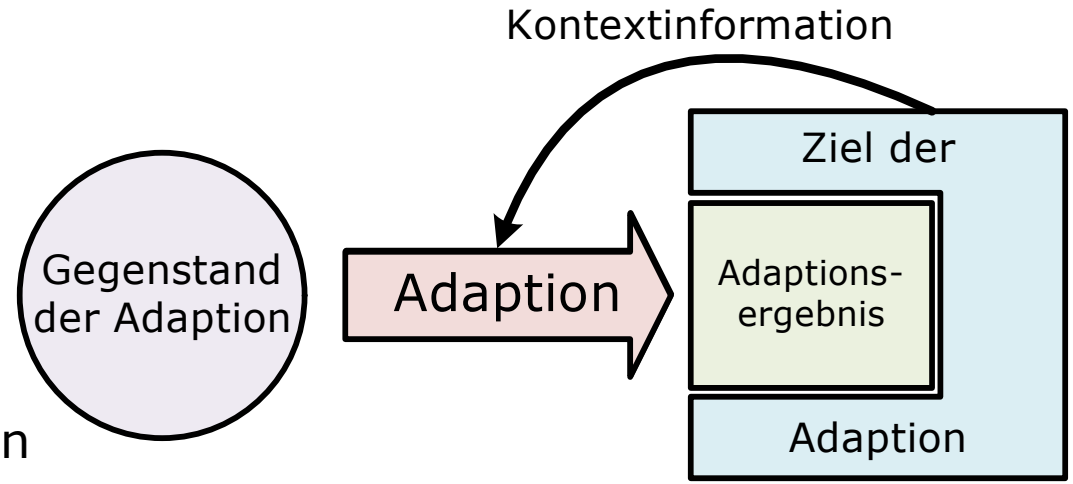
Drei Notwendigkeiten

- Kontextbewusstsein:
  Wissen über operative Umgebung
- Selbstbewusstsein:
  Wissen über aktuellen Konfigurationszustand
- Selbstadaption:
  sich selbst während der Laufzeit ändernd



Adaption

Was wird zu was adaptiert?

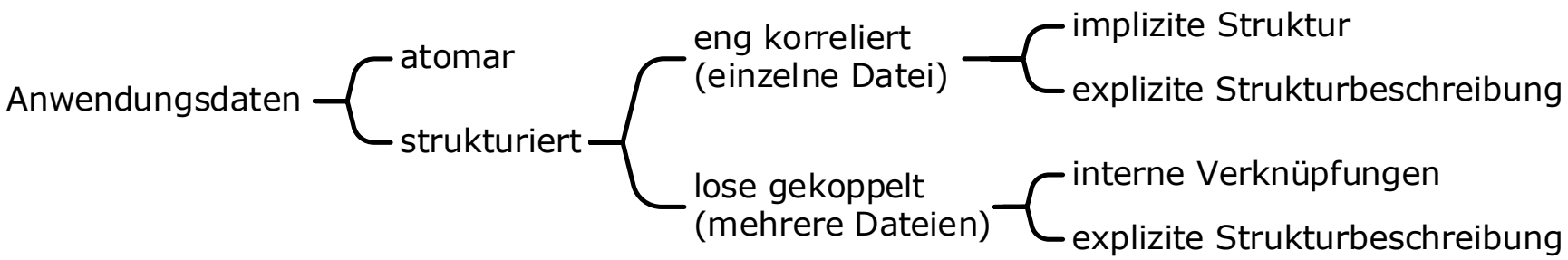


- Gegenstand der Adaption
  - Anwendungsdaten
  - Datenübertragung
  - Verarbeitungskomponenten (Struktur, Platzierung)
- Ziel der Adaption
  - verfügbare Ressourcen der Ausführungsumgebung (Gerät selbst, Peripherie, Netzwerk, …)
  - Nutzerpräferenzen, aktuelle Situation
- Kontextinformation repräsentiert Informationen über das Adaptionsziel
  → notwendige Vorbedingung für (erfolgreiche) Adaption

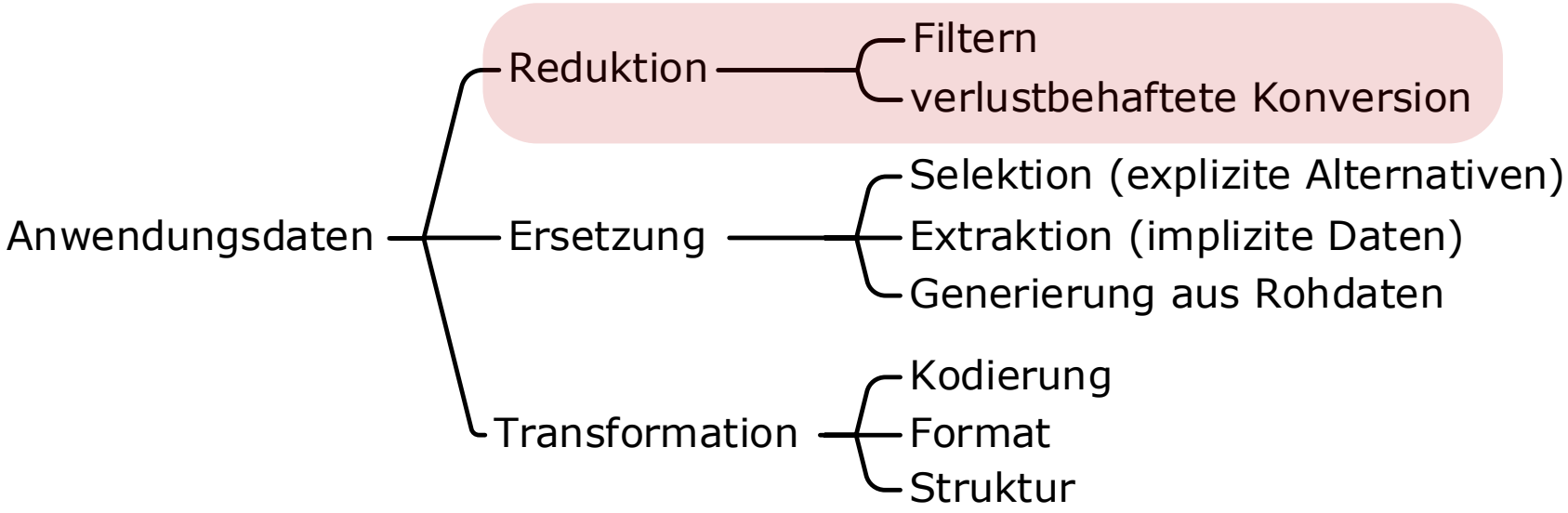
Adaption von Anwendungsdaten

Vorbetrachtung: Datencharakteristik

- Typ (Text, Video, Audio, …)
- Format (PDF, MPEG, WAV, MP3, …)
- Kodierung (ASCII, ANSI, UTF-8, …)
- Struktur
  - atomar (z.B. Bild: logisch atomar, Eingabefeld und Label)
  - eng korreliert (eine Datei)
    - implizite Struktur (Textdokument)
    - explizite Struktur (E-Mail mit MIME-Erweiterung)
  - lose gekoppelt (mehrere Dateien)
    - interne Verknüpfungen (Web-Inhalte)
    - explizite Strukturbeschreibung (I-, P- und B-Frames eines Videos)



Adaption der Anwendungsdaten



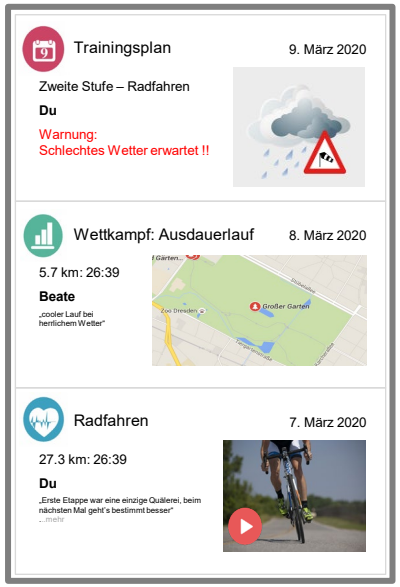
- Wichtig: Adaption von Multimediainhalten
- Balance zwischen Leistung, Datenvolumen und Informationsverlust
  - Reduktion – schneller aber grobgranularer
  - Ersetzung und Transformation – langsamer aber feingranularer

Reduktion – Filtern

- Ziel: Reduktion des Datenvolumens (mit Inhaltsverlust)
  - grobgranularer Ansatz (keep or drop)
- kurze Verarbeitungszeit und geringer Ressourcenverbrauch
  - keine Datenverarbeitung, dafür Selektieren und Entfernen
- Filterregeln definiert auf Datenselektion
- Filtern auf Basis unterschiedlicher Kriterien
  - externe Kriterien (Pakete im Router basierend auf Warteschlangenlänge verwerfen)
  - Datenstruktur (Entfernen aller E-Mail-Anhänge oder aller P- und B-Frames eines MPEG-Datenstroms)
  - Datentyp (Entfernen von Videos aus einer Webseite)
  - Datenformat (Entfernen von WebP-Bildern aus einer Webseite)
  - Semantik (Entfernen von E-Mails anhand eines Schlüsselwortes)

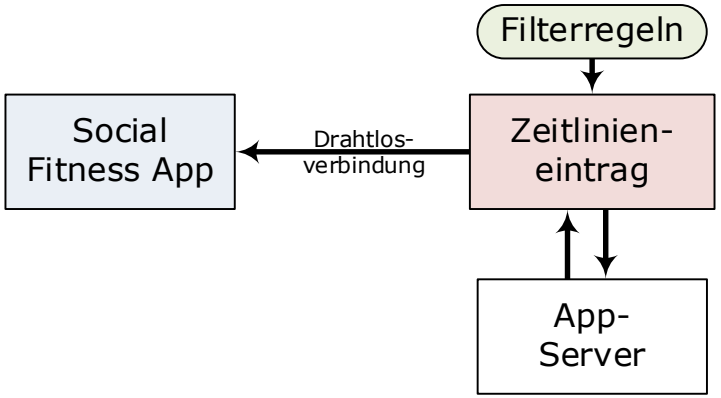
⇒ i.d.R. zur Laufzeit angewendet

Reduktion – Filterbeispiel



Reduktion der Aktivitätszeitlinie

- Kopf
  - Aktivitäten bestimmter Typen (Radfahren, Outdoor)
  - Aktivitäten eines bestimmten Wettkampfs (gewählte Gruppe UND Sportart UND Ort)
- Inhalt
  - Filtern anhand bestimmter Schlüsselwörter in Beschreibung (Team Challenge, neuer Rekord)
- Anhänge
  - verwerfe Anhänge ab einer bestimmten Größe (große Bilder)
  - verwerfe alle Videoanhänge



Reduktion – verlustbehaftete Konversion

- Ziel: Reduktion des Datenvolumens (ohne Inhaltsverlust)
- arbeitet auf Quelldaten/Rohdaten
- erzeugt verwandte Datenobjekte ähnlichen Typs
  - bewahrt Inhalt (begrenzt)
- Operationen hängen vom Daten-/Medientyp ab
  → i.d.R. werden spezifische Eigenschaften der Medienobjekte manipuliert
  - Bilder: Auflösung, Farbtiefe, Kontrast, …
  - Video: Bildrate, Größe, Farbtiefe, …
  - Ton: Abtastrate, Wertebereich, …
  - spezifische Typanpassungen: Word→PDF, Video→GIF
- feingranulare Reduktion von Mediendaten in Abhängigkeit von Gerätefähigkeiten und Verbindung (bspw. Video auf Auflösung des Smartphones verkleinern, Tonspur auf Mono reduzieren)

Reduktion – verlustbehaftete Konversion

Präadaption von Daten

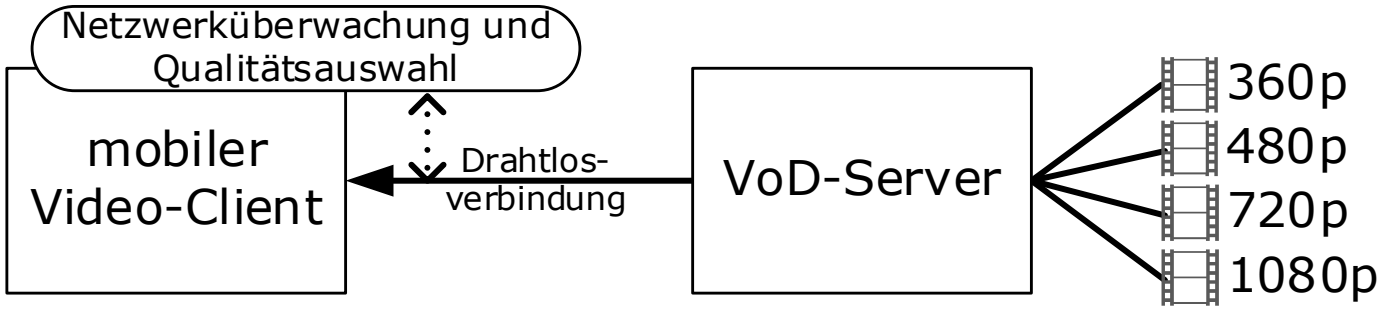
- Erzeugung diverser Qualitätsstufen der Anwendungsdaten
- Client wählt Qualitätsstufe während der Laufzeit anhand Netzwerkkapazität und Geräteressourcen
- Unterstützung dynamischer Wechsel zwischen Qualitätsstufen

+ keine Konversionsberechnungen während der Laufzeit

– hohe Anforderungen an Speicherkapazität begrenzen Anzahl unterschiedlicher Versionen

– begrenzte Flexibilität für Adaption

Beispiel: Video-on-Demand (Youtube)



Reduktion – verlustbehaftete Konversion

On-the-Fly-Adaption zur Laufzeit

→ nutze aktuellen Kontext (Bandbreite, Displaygröße, …) zur Anpassung der Datenobjektparameter

+ erhöhte Flexibilität der Adaption

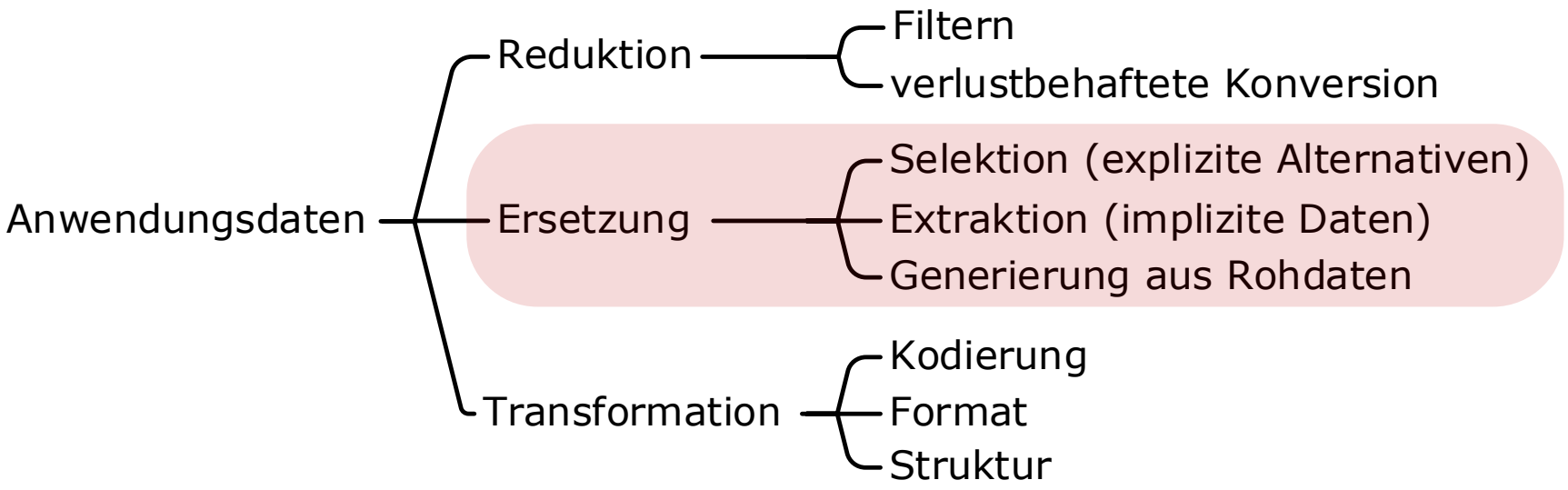
+ keine weiteren Speicheranforderungen

– Bedarf an Berechnungsressourcen zur Laufzeit

– kann Antwortzeit erhöhen

Beispiel: verlustbehaftete Bildkonversion zur Reduktion der Transfergröße

Adaption der Anwendungsdaten



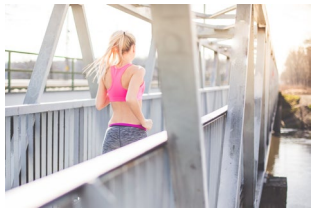
- Wichtig: Adaption von Multimediainhalten
- Balance zwischen Leistung, Datenvolumen und Informationsverlust
  - Reduktion – schneller aber grobgranularer
  - Ersetzung und Transformation – langsamer aber feingranularer

Ersetzung – Selektion

- Ziel: Reduktion des Datenvolumens durch alternative Darstellung
- Ersetzung der Quelldaten durch alternative Daten anderen Typs (synonyme Ersetzung)
- i.d.R. höhere Datenvolumeneinsparung im Vergleich zur verlustbehafteten Konversion

- Beispiel: alternative Daten sind explizit verfügbar
  - alternativer Text für Bild auf Webseite (alt-Attribut für HTML-img-Tag)
  - Ersetzung des Bildes durch alternativen Beschreibungstext

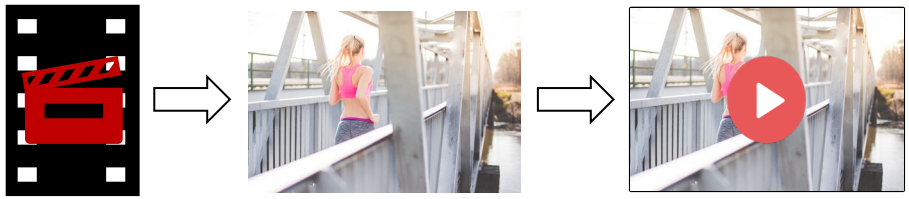
```
<img src="/img/jogginggirl.jpg"
     alt="Eine junge Frau joggt über
          eine Brücke." />
```



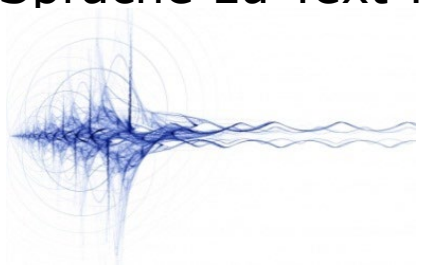
Eine junge Frau joggt über eine Brücke.

Ersetzung – Extraktion und Generierung

- Extraktion
  - alternative Daten aus Quelldaten abgeleitet
    - repräsentatives Bild als Platzhalter für ein Video
    - Rahmengröße auf Basis der Bildauflösung



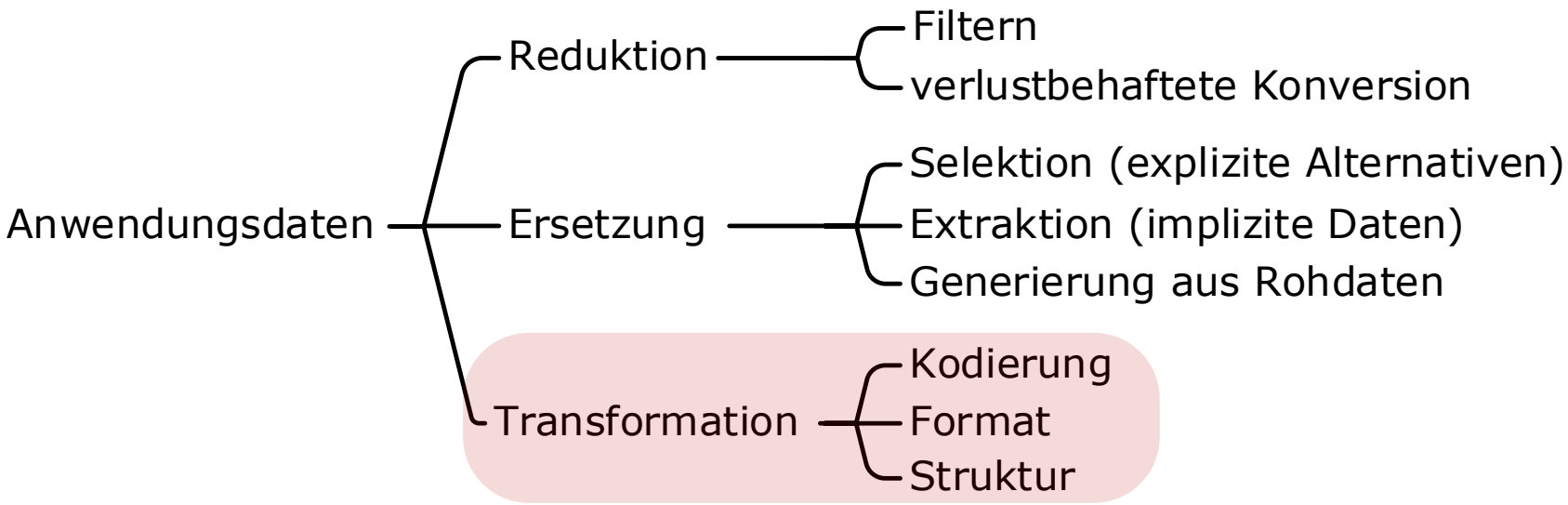
- Generierung
  - alternative Daten aus Quelldaten erzeugt
    - Sprache-zu-Text-Konversion für Tondaten



„Blah blubb“

    - Vektorgrafik aus Kantenerkennung in Bitmap

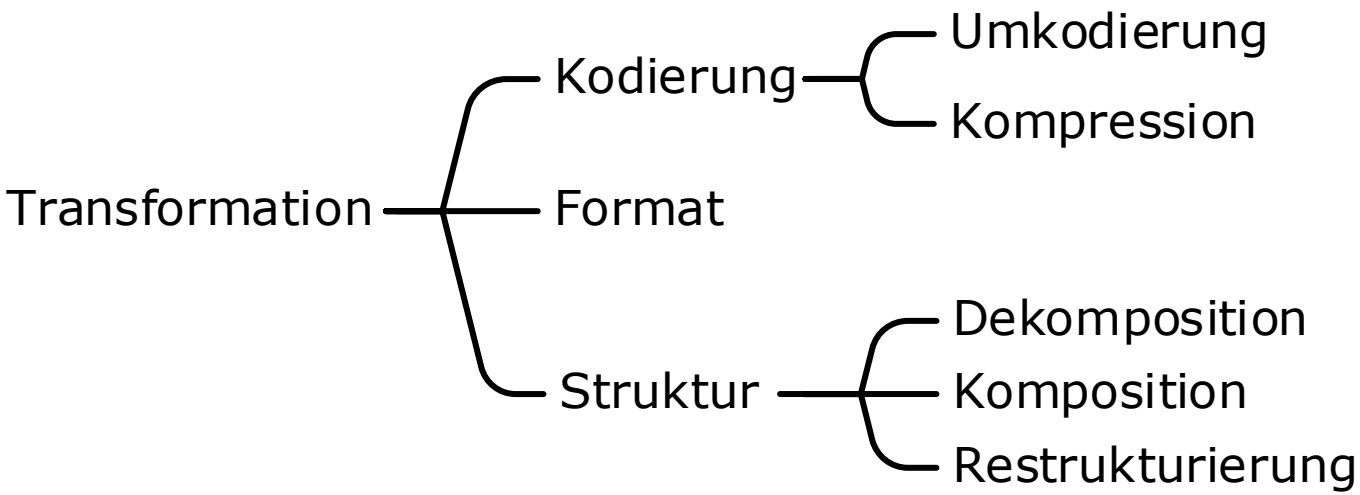
Adaption der Anwendungsdaten



- Wichtig: Adaption von Multimediainhalten
- Balance zwischen Leistung, Datenvolumen und Informationsverlust
  - Reduktion – schneller aber grobgranularer
  - Ersetzung und Transformation – langsamer aber feingranularer

Transformation

Ziel: Ändern der Datenrepräsentation ohne Informationsverlust



Transformation – Kodierung

- Ziel: Ändern der internen Datenrepräsentation ohne Informationsverlust
- keine Änderung des Datentyps, Datenformats oder der Datenstruktur

- Rekodierung: Vermeidung von Inkompatibilitäten und Effizienzerhöhung
  - Wechsel zwischen Farbräumen
  - ASCII-Text mit HTML-Entitäten zu UTF-8

- (verlustfreie) Kompression: Datenmengenreduktion (Entropiekodierung)
  - generische Mechanismen: Huffman-Kodierung, Lauflängenkodierung
  - Kompressionsraten ≤ 1:8

Transformation – Format

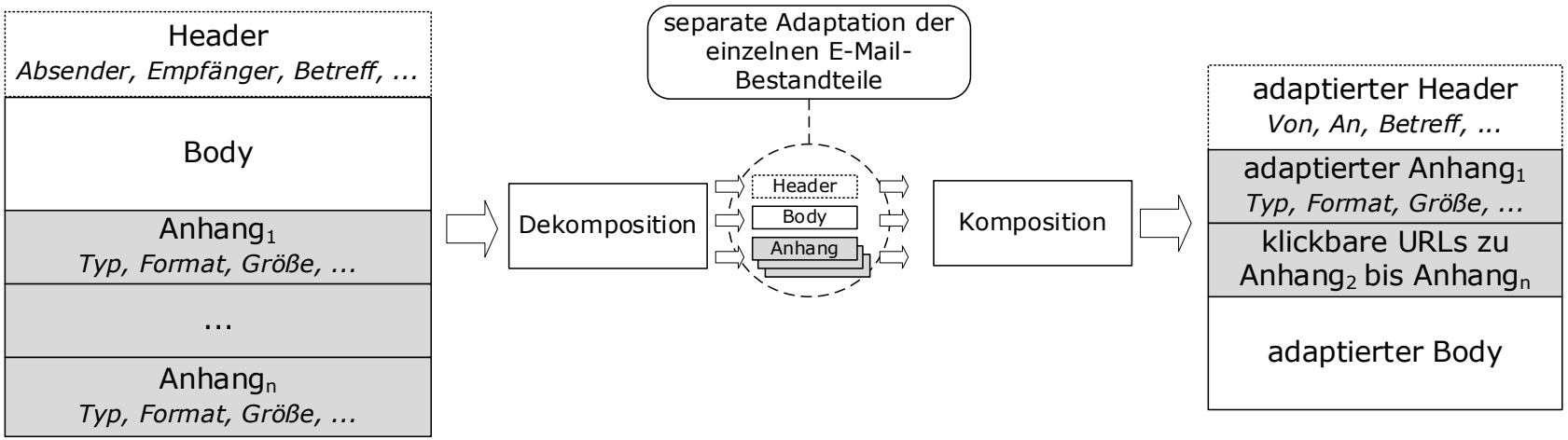
- Ziel: Ändern der Datenrepräsentation zur Beseitigung von Inkompatibilitäten (ohne Informationsverlust)
- Eingangsdefinition zu erhaltender Informationen notwendig
  → Informationen, die nicht von der Definition abgedeckt sind, dürfen verworfen werden
- Ändern des Formats der Quelldaten
- keine Änderung des Datentyps

- Beispiele
  - PDF zu Purtext (konserviert „Text"-Information)
  - TIFF zu PNG (konserviert „Bild"-Information)

- Informationsverlust sobald Zielformat weniger Expressiv (PDF zu Purtext: Schriftsatz geht verloren)

Transformation – Struktur

- Dekomposition
  - Ziel: separate Adaption von Fragmenten strukturierter Daten
  - Zerlegung der Datenobjekte in atomare Datenobjekte (Fragmente)
- Komposition
  - Ziel: Erzeugung kompakten Repräsentationsform aus (atomaren) Datenobjekten
  - Rekonstruktion der ursprünglichen Datenstruktur: inverse Operation zur Dekomposition

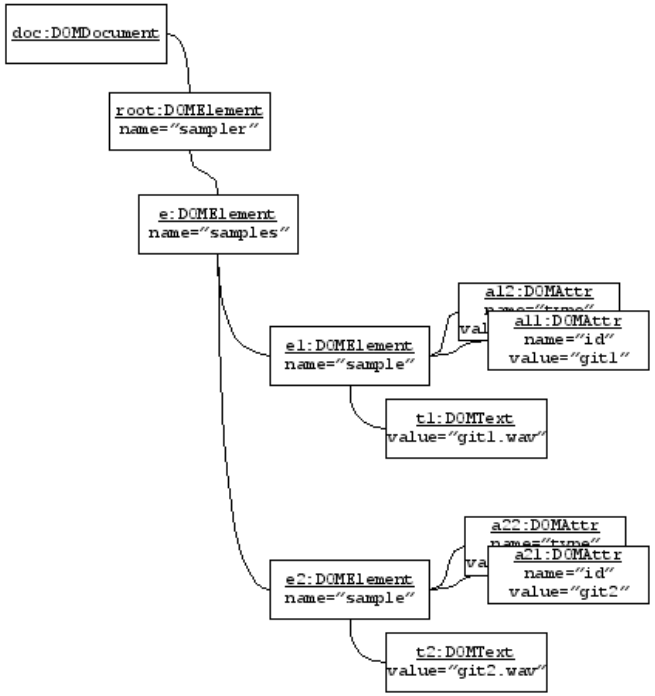


Transformation – Struktur

- Restrukturierung
  - Ziel: einfachere Verarbeitung und Interoperabilität
  - Operationen auf strukturierten Daten (ohne (De)Komposition)
  - Überführe XML-Dokument in DOM-Repräsentation
  - Überführe implizite Textdokumentstruktur in explizite XML-Struktur

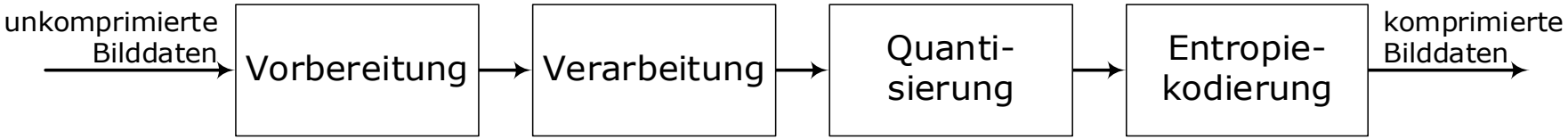
```
<sampler>
    <samples>
        <sample id="git1" type="wave">git1.wav</sample>
        <sample id="git2" type="wave">git2.wav</sample>
    </samples>
</sampler>
```

⇔

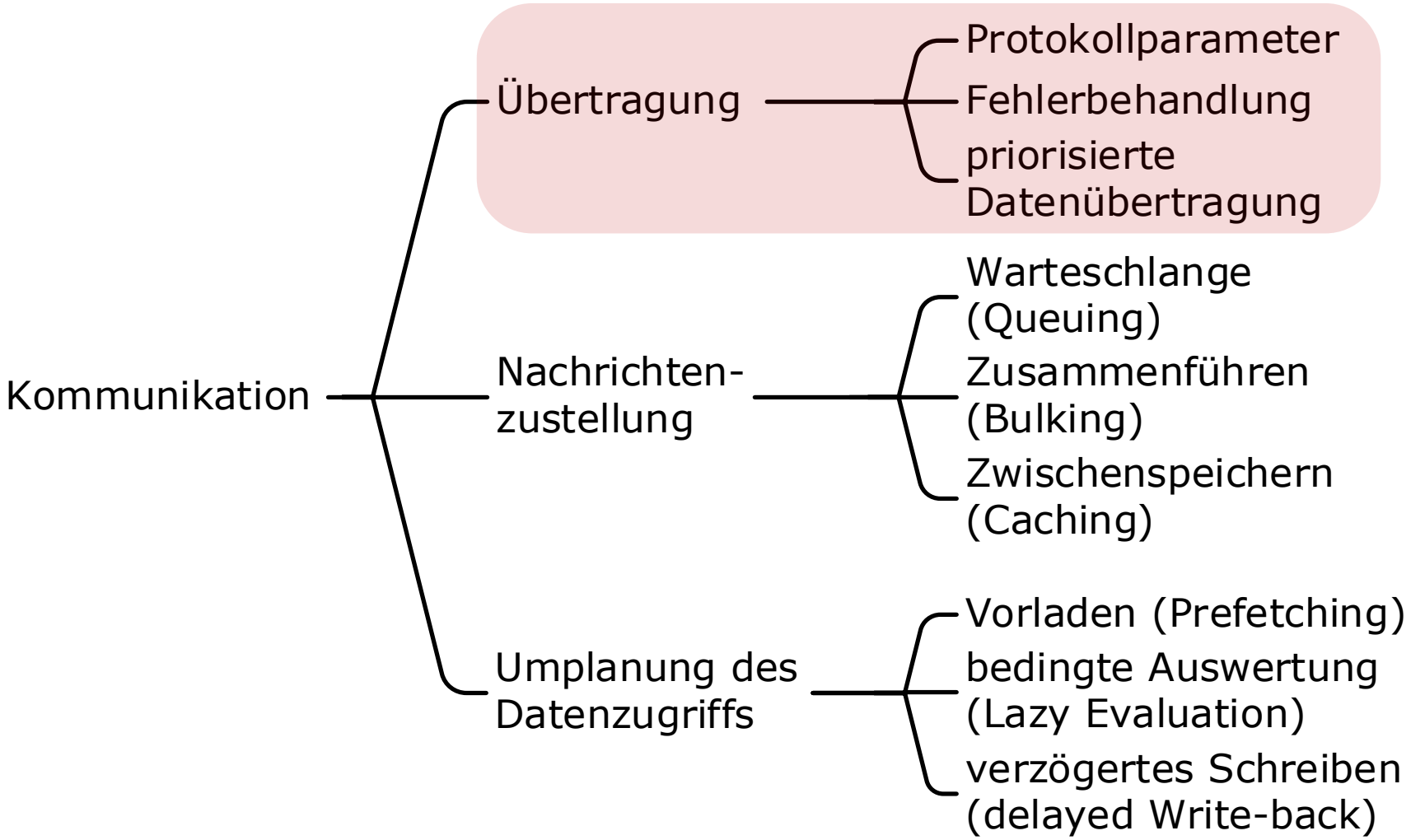


Adaption von Mediendaten

- Notwendigkeit, Multimediadaten unter geringem Informationsverlust zu konvertieren
- Kombination unterschiedlicher Mechanismen
- Beispiel: JPEG
  - Vorbereitung: Dekomposition in 8x8Px-Blöcke
  - Verarbeitung: Rekodierung vom Wertebereich in den Frequenzbereich (Discrete Cosinus Transformation)
  - Quantisierung: verlustbehaftete Konversion (reduziert Genauigkeit der Informationen)
  - Entropiekodierung: verlustfreie Kompression

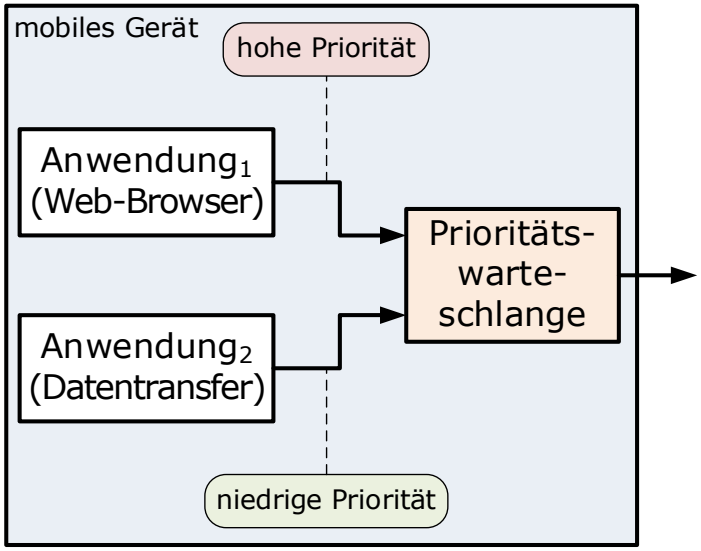


Kommunikationsadaption



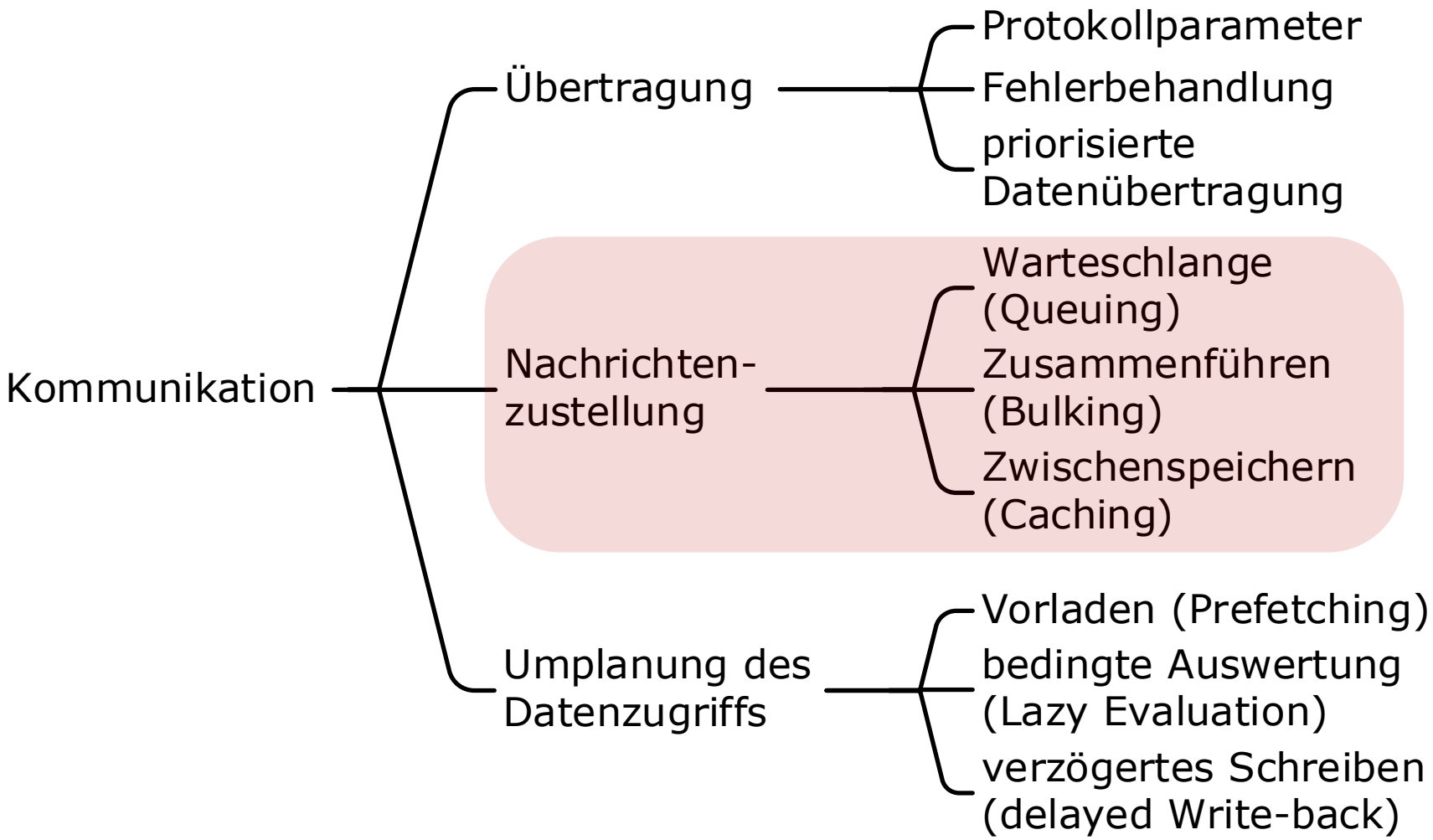
Übertragung

- Protokollparameter
  - Rahmengröße, Paketgröße (Maximum Transfer Unit)
  - Fenstergröße
  - Time-out
  - Header-Reduktion
- Fehlerbehandlung
  - Fehlererkennung
  - Forward Error Correction
  - optimierte Wiederholungsmechanismen

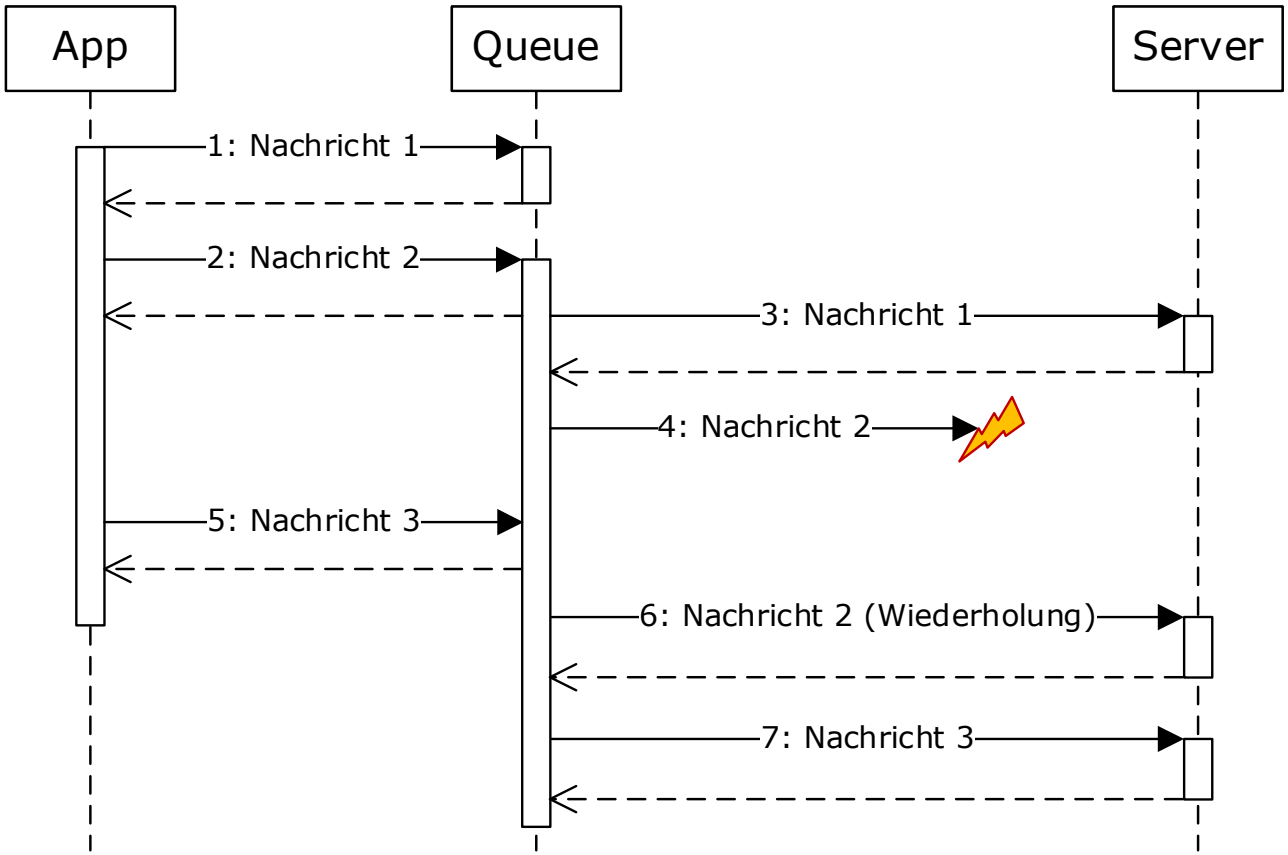


- priorisierte Datenübertragung
  - Klassifikation der Datenströme (interaktiv, Hintergrund, …)
  - Prioritätszuweisung für Klassen oder individuelle Datenströme

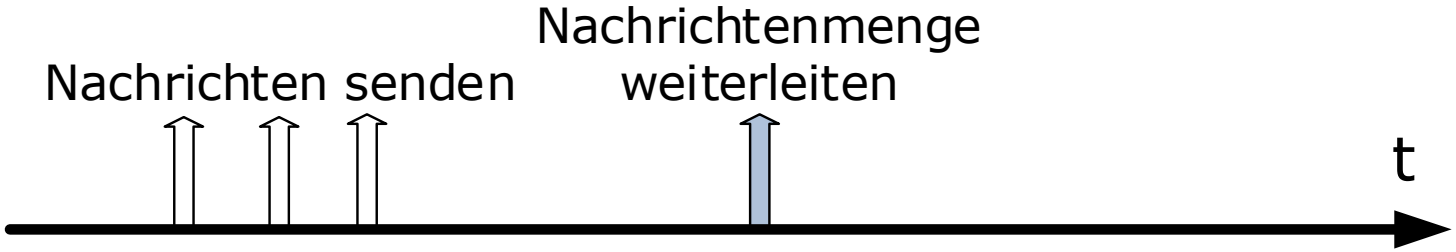
Kommunikationsadaption

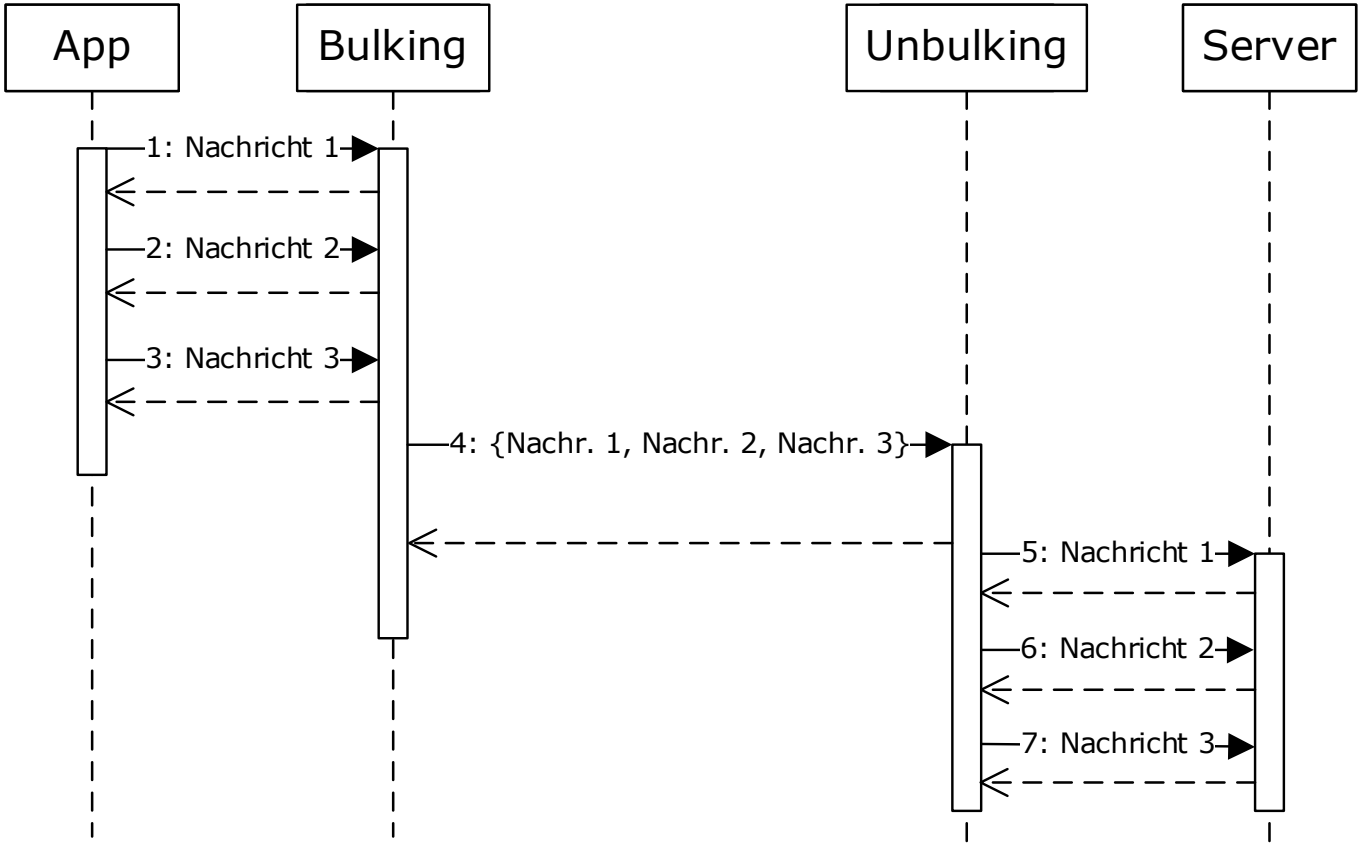


Nachrichtenzustellung – Queuing

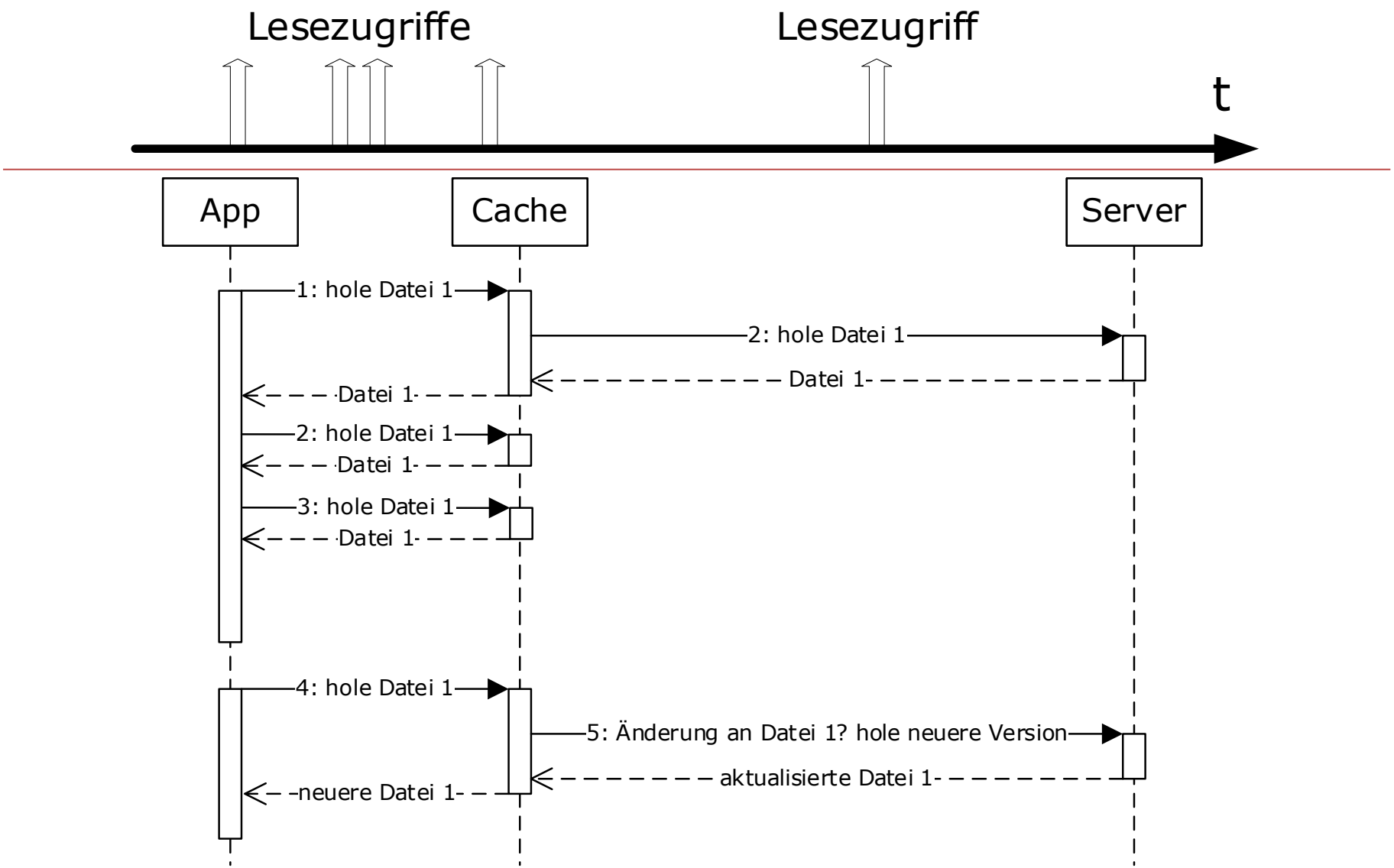


Nachrichtenzustellung – Bulking

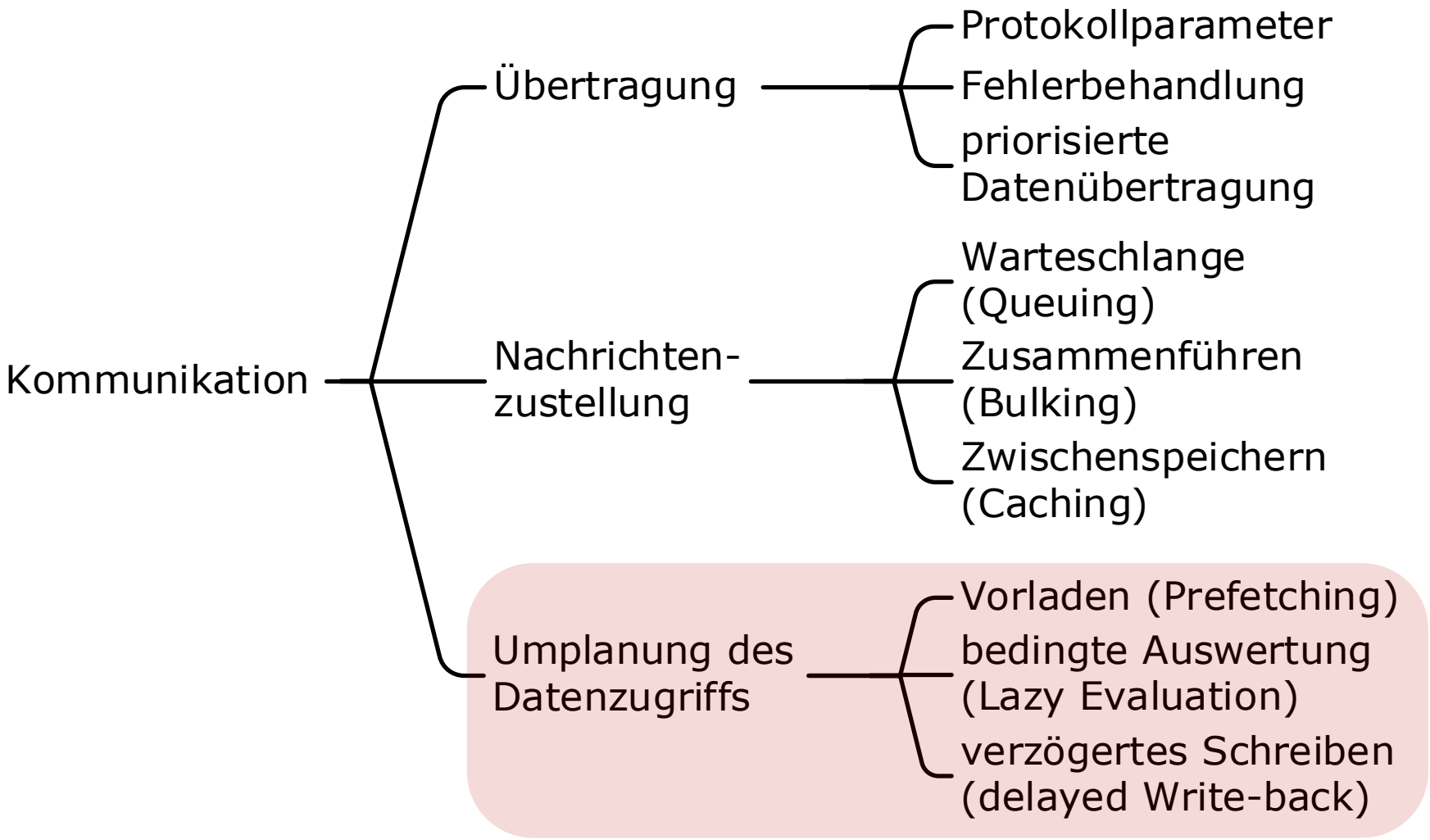




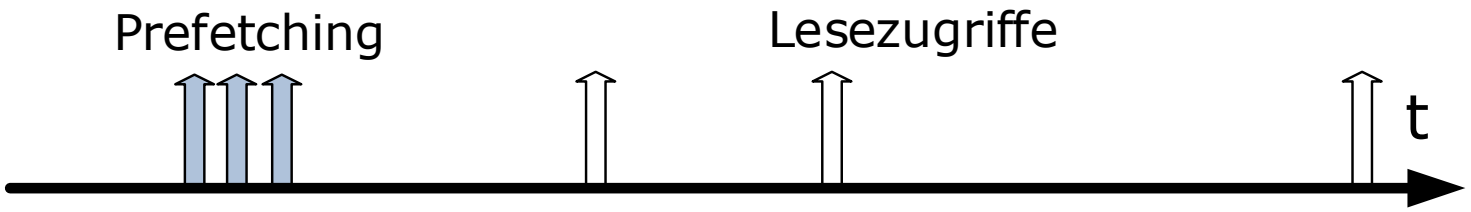
Nachrichtenzustellung – Caching

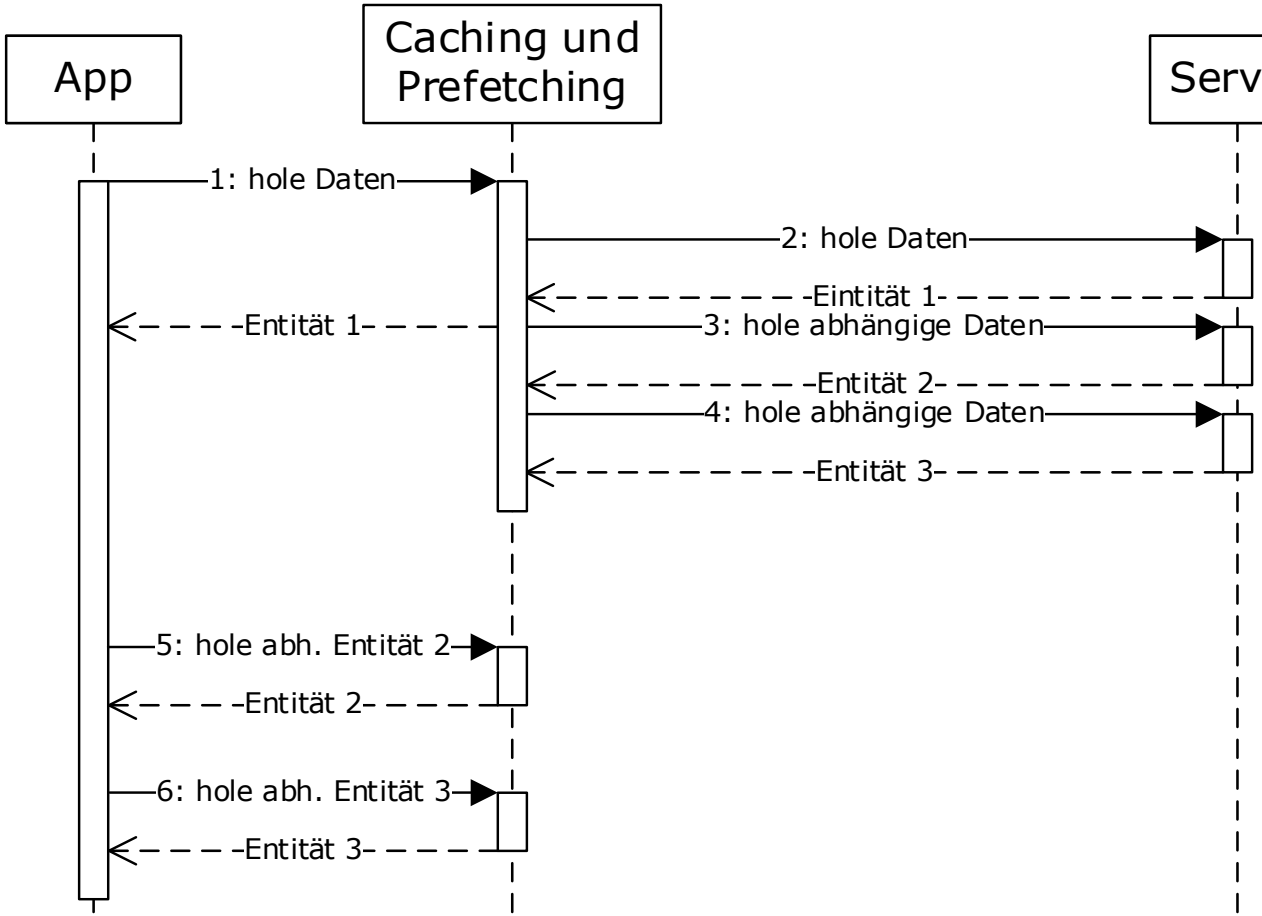


Kommunikationsadaption

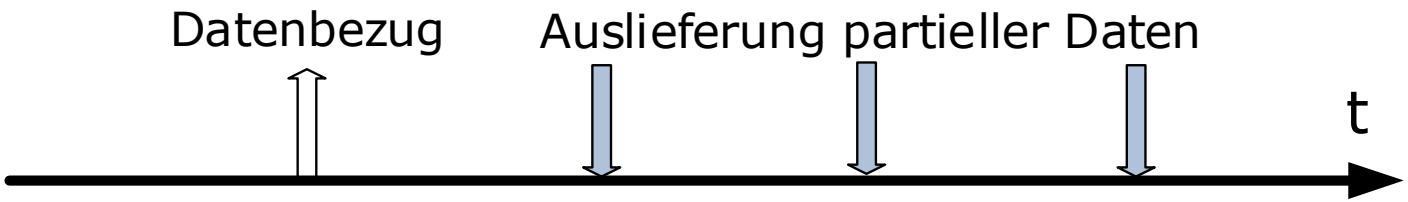


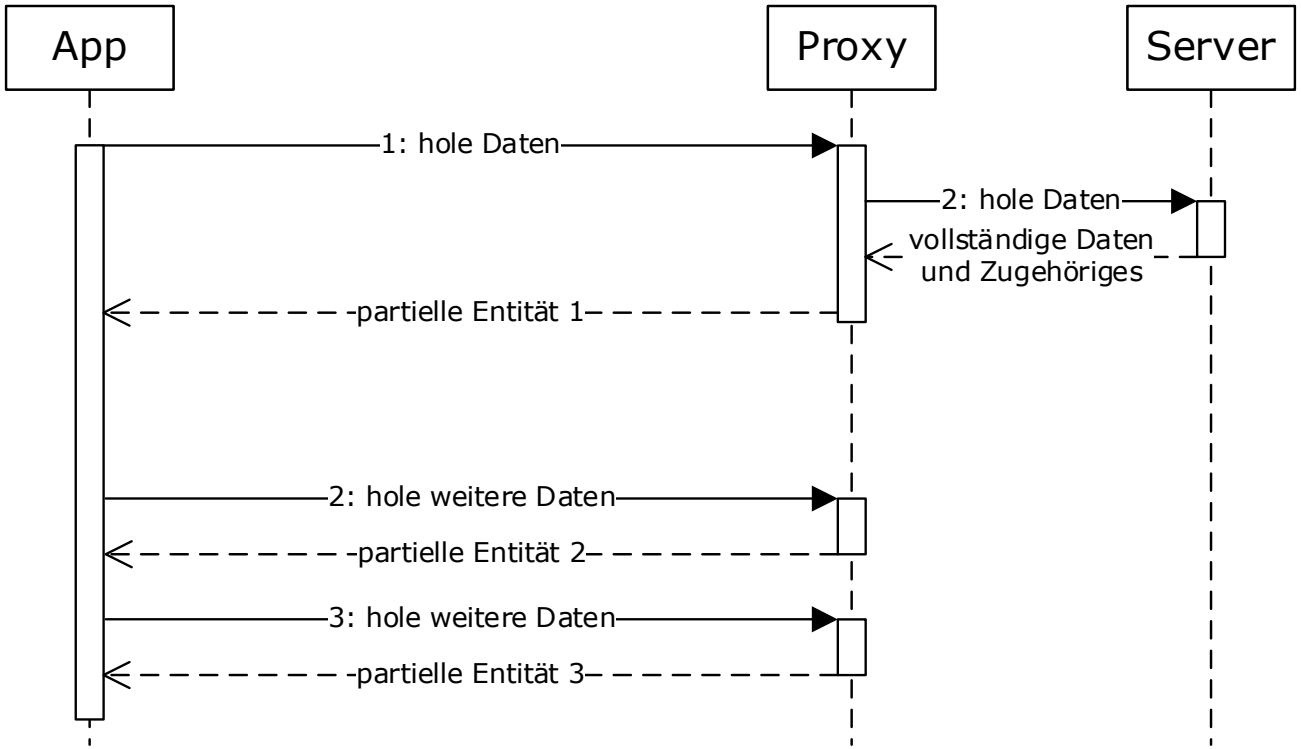
Umplanen des Datenzugriffs – Prefetching





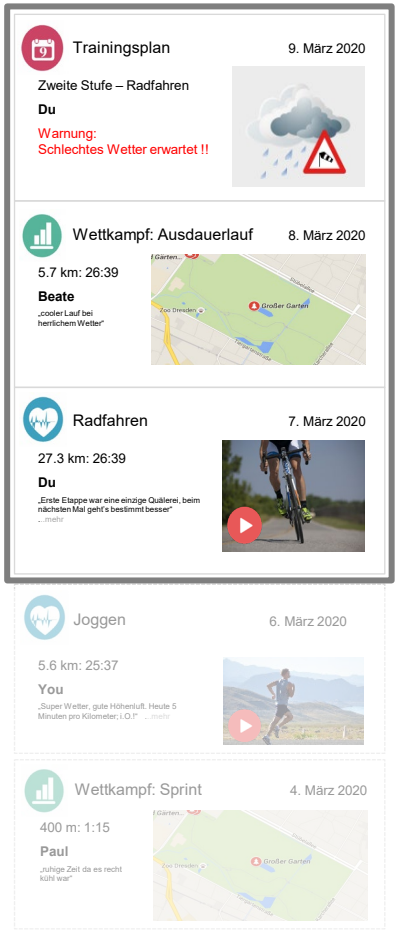
Umplanen des Datenzugriffs – Lazy Evaluation





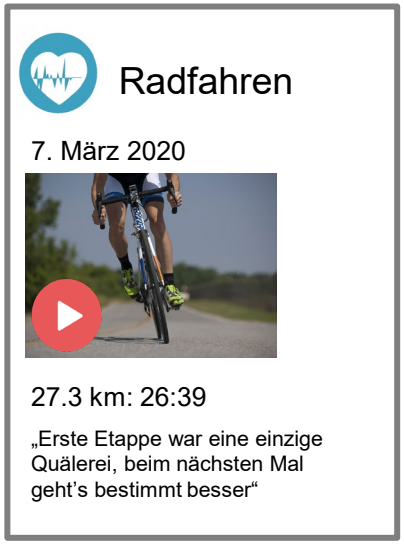
Umplanen des Datenzugriffs – Lazy Evaluation – Beispiel

- nur die ersten drei Elemente der Zeitlinie werden direkt geladen
- sobald der Nutzer scrollt werden die nächsten Elemente der Zeitlinie nachgeladen
- Details zu einzelnen Elementen werden erst geladen, wenn der Nutzer ein Element öffnet

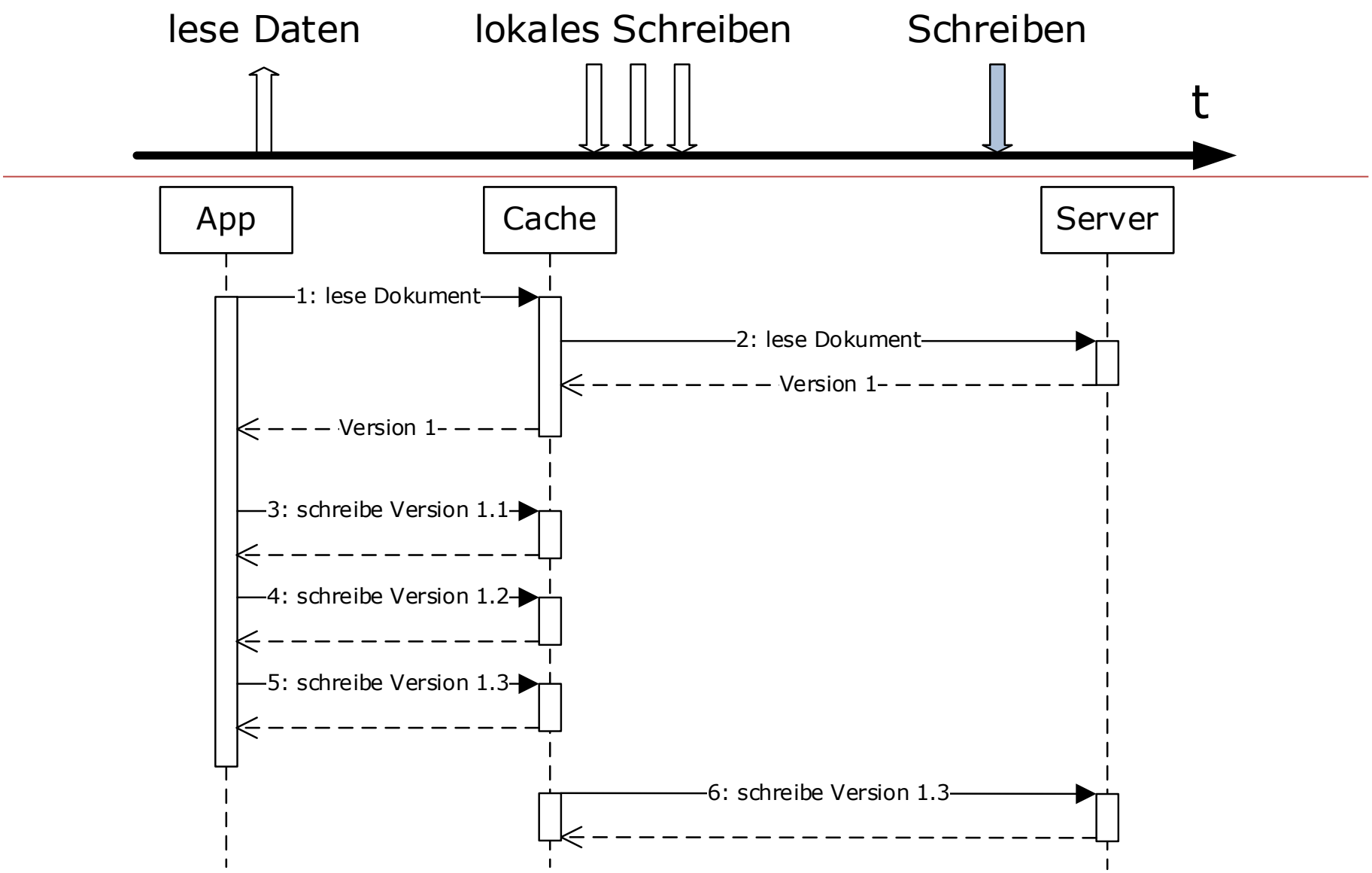


Detailansicht

Lade erst beim Scrollen



Umplanen des Datenzugriffs – Delayed Write-back



Umplanen des Datenzugriffs

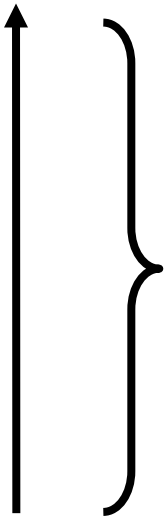
- Vorladen / Prefetching
  - Übertragungsoperation(en) vor dem eigentlichen Zugriff
  - hohe Verfügbarkeit, kurze Antwortzeiten
  - benötigt effizienten Vorhersagemechanismus
    → nutzt Netzwerkkapazitäten besser aus
    → verschwendet Netzwerkkapazitäten falls Vorhersage inkorrekt
- bedingte Auswertung / Lazy Evaluation
  - Übertragungsoperation(en) nach Datenzugriff
  - balanciert Netzwerkzugriff
  - vermeidet Lastspitzen
- verzögertes Schreiben / Delayed Write-back
  - lokale Änderungen an Ressourcen bleiben lokal
  - Rückschreiben erst wenn Ressource geschlossen wird
  - spart Bandbreite
  - vermeidet unnötige Aktualisierungen

Adaption der Anwendungsstruktur

- Filtern
- Konversion und Kompression
- Caching
- Prefetching

Adaption durch Parametrisierung

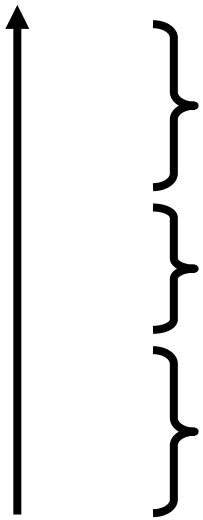
Wertebereich (z.B. Bandbreite)



Mechanismus arbeitet über gesamten Bereich

Adaption durch Strukturänderungen

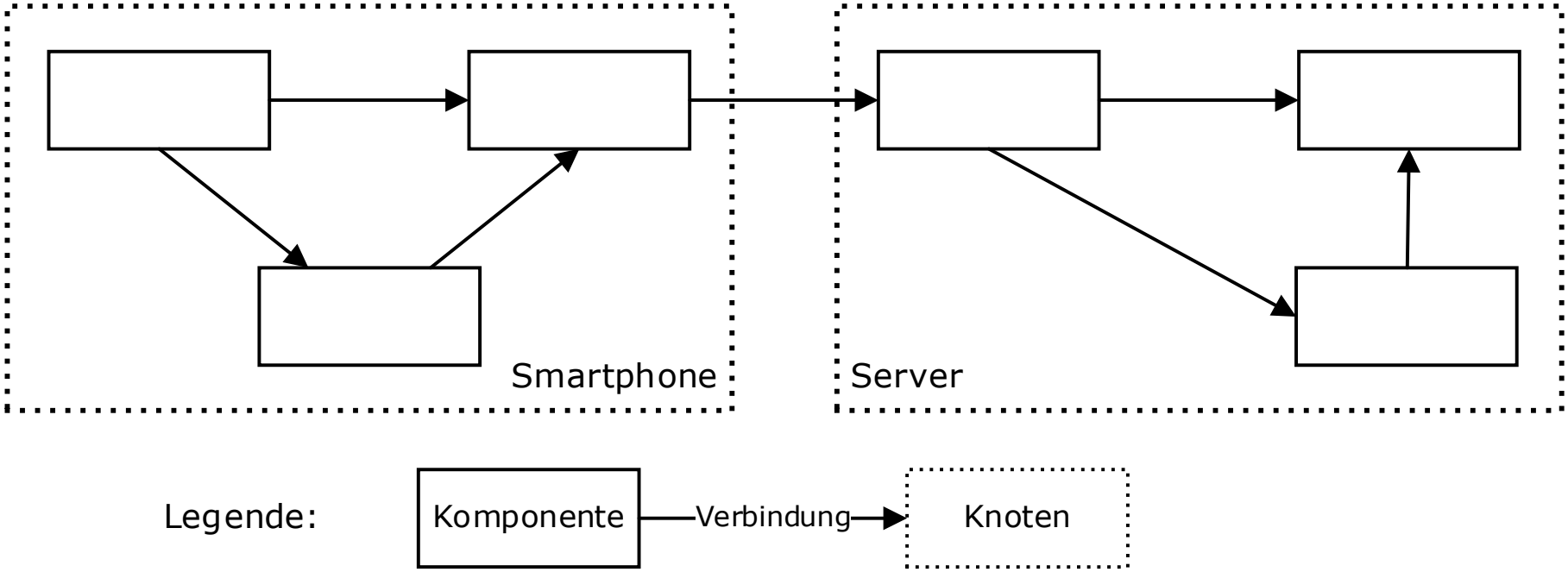
Wertebereich (z.B. Fehlerrate)



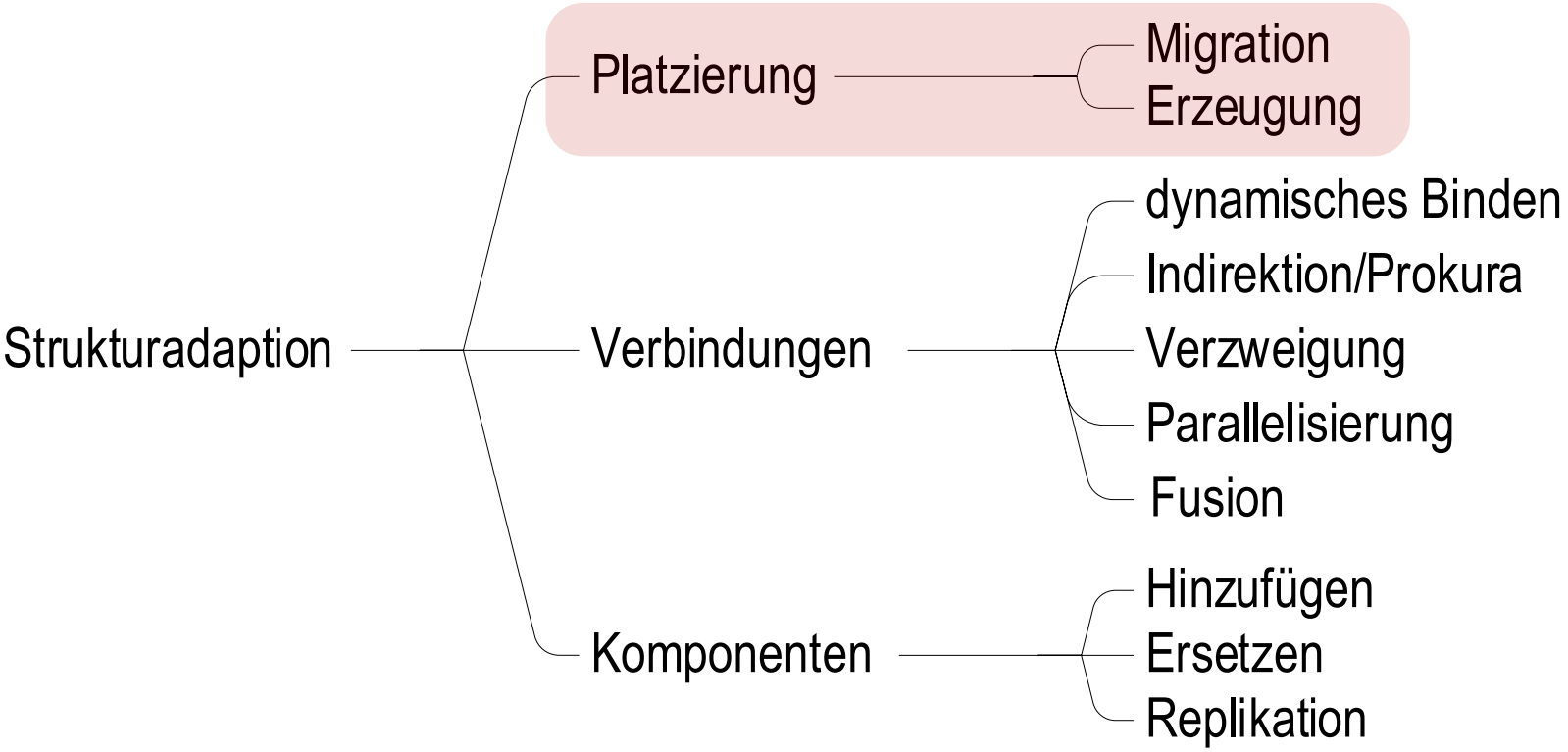
mehrere Mechanismen arbeiten in Teilbereichen

Änderungen der Ressourcenverfügbarkeit erfordern Anwendungsstrukturänderungen

Adaption der Anwendungsstruktur

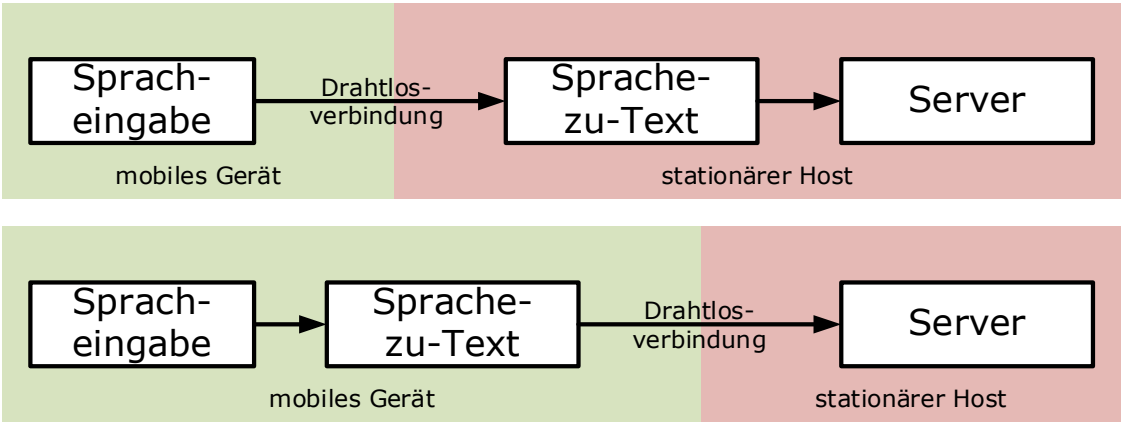


Adaption der Anwendungsstruktur

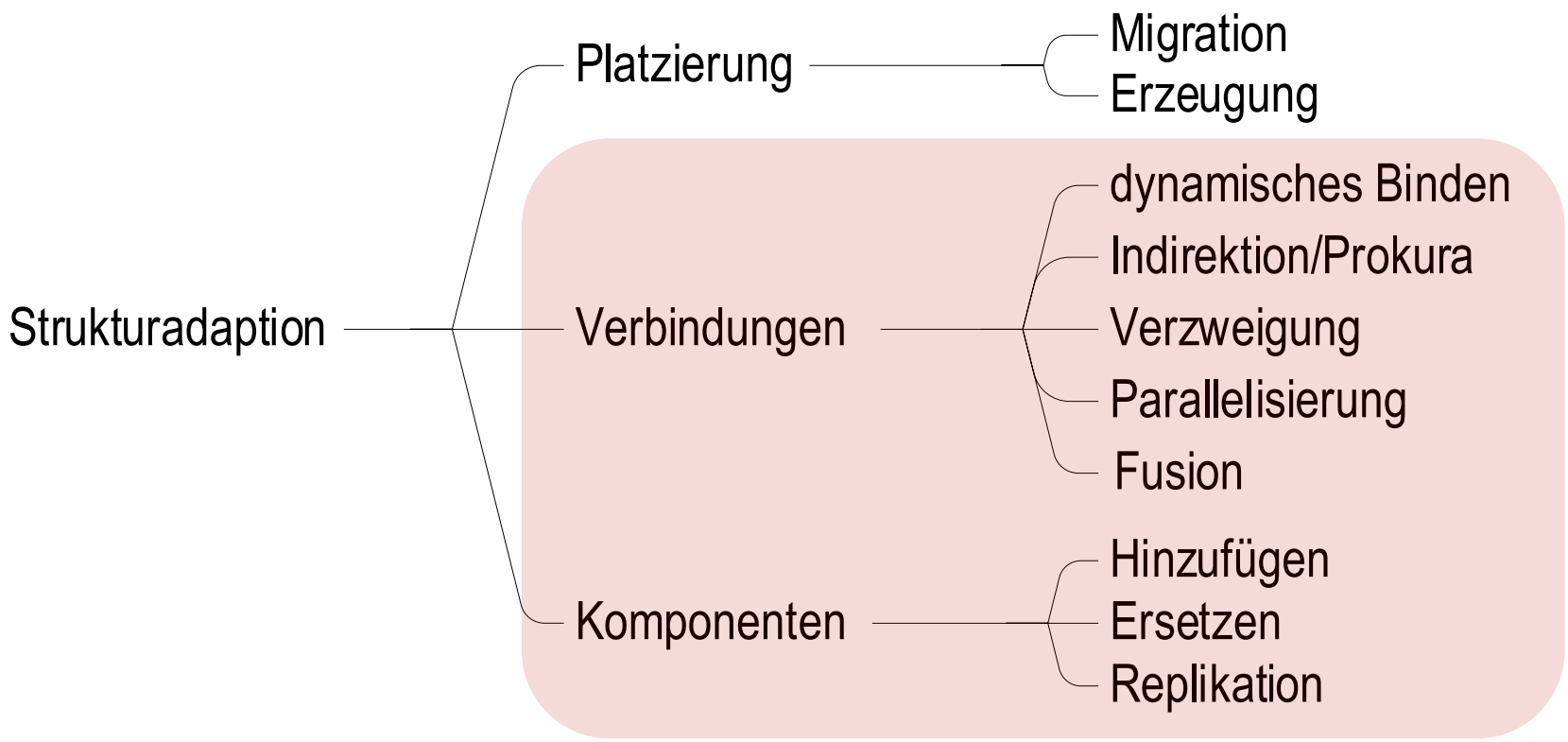


Adaption der Platzierung

- Migration
  - Ändern der Komponentenplatzierung während der Laufzeit
  - Komponenten „folgen" dem Nutzer in der Infrastruktur
  - Übergabe von Anwendungskomponenten zwischen Knoten
- Erzeugung
  - Bereitstellen von Komponenten auf geeignetem Knoten
  - Bereitstellung insbesondere auf Basis der verfügbaren Ressourcen



Adaption der Anwendungsstruktur



Adaption der Komponenten

Hinzufügen/Entfernen

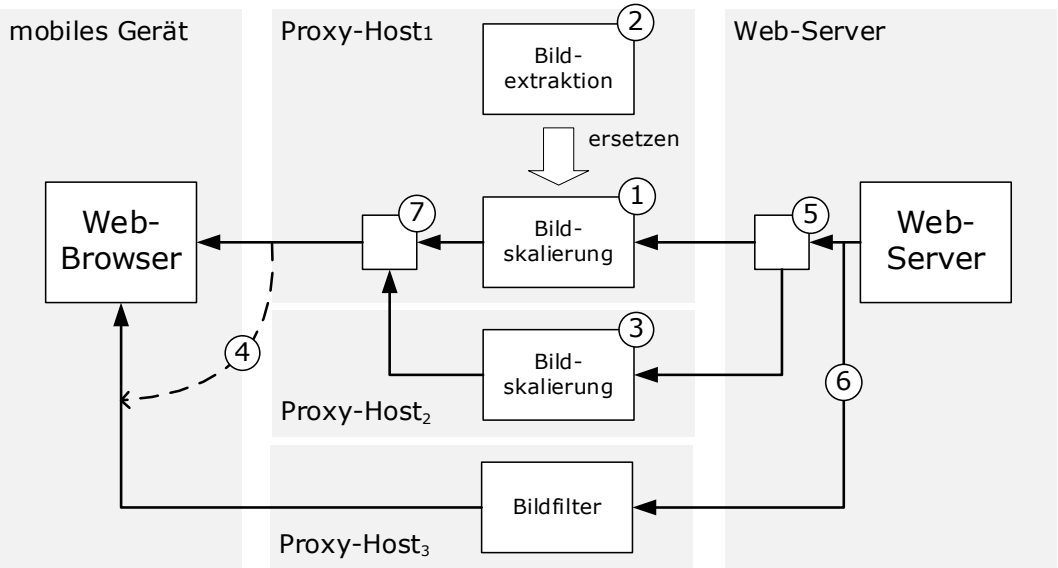
1. Bildskalierungskomponente hinzufügen für Datenvolumenverringerung

Ersetzen

2. ersetze Bildskalierungs- durch Bildextraktionskomponente

Replikation

3. repliziere Bildskalierungskomponente für Lastverteilung



Adaption der Verbindung

dynamisches Binden

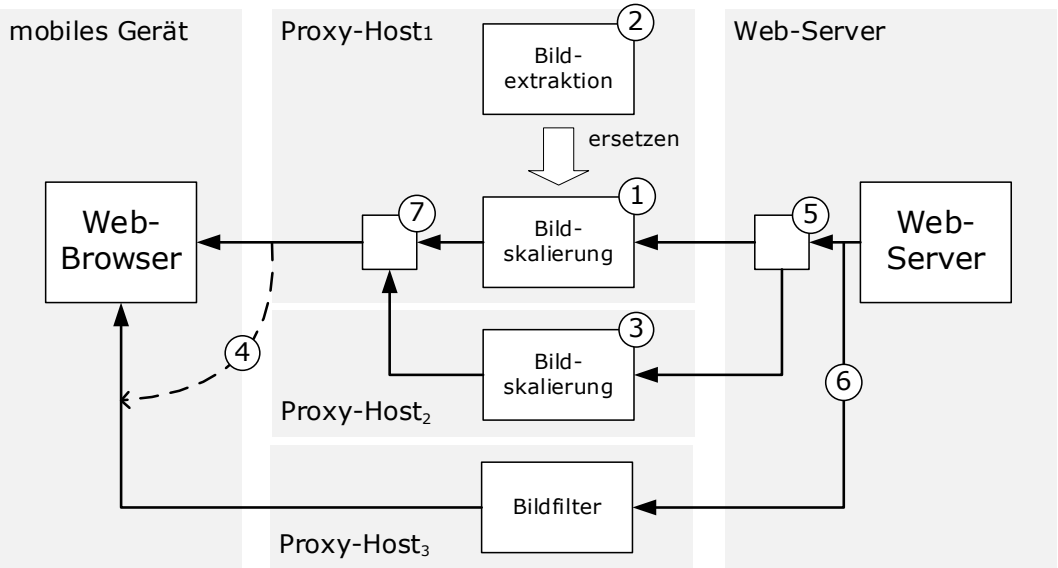
4. Neubinden an alternativen Proxy

Verzweigen

5. nutze alternative Proxys um Last zu verteilen

Parallelisierung

6. nutze unterschiedliche Komponenten zur gleichzeitigen Verarbeitung von nicht-abhängigen Daten



Adaption der Verbindung

Fusion

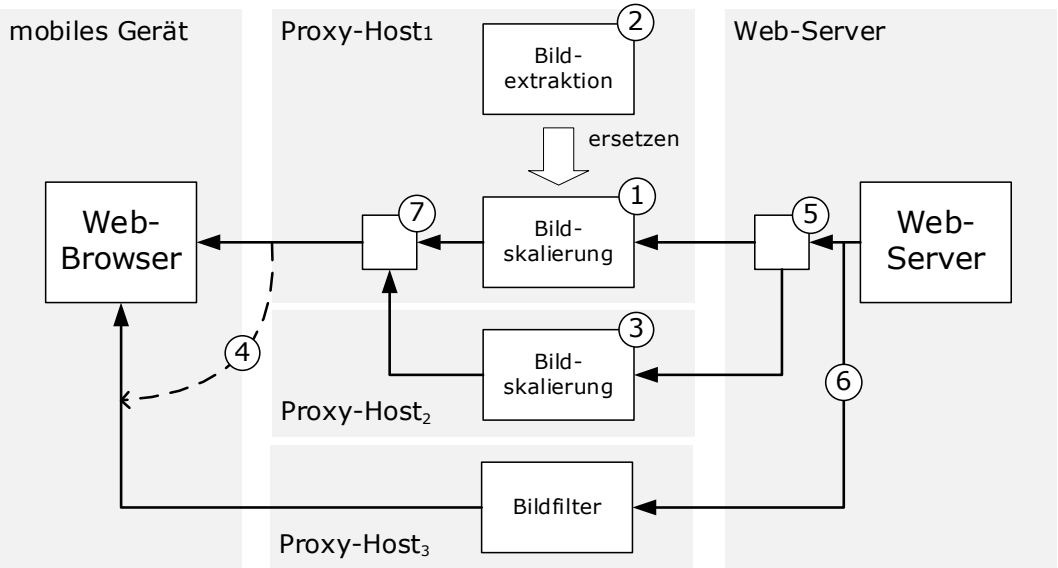
7. kombiniere Daten mehrerer Komponenten zu einem Datenstrom

Indirektion/Prokura

- nutze das Proxy-Pattern um Komponenten zu entkoppeln

Kombination

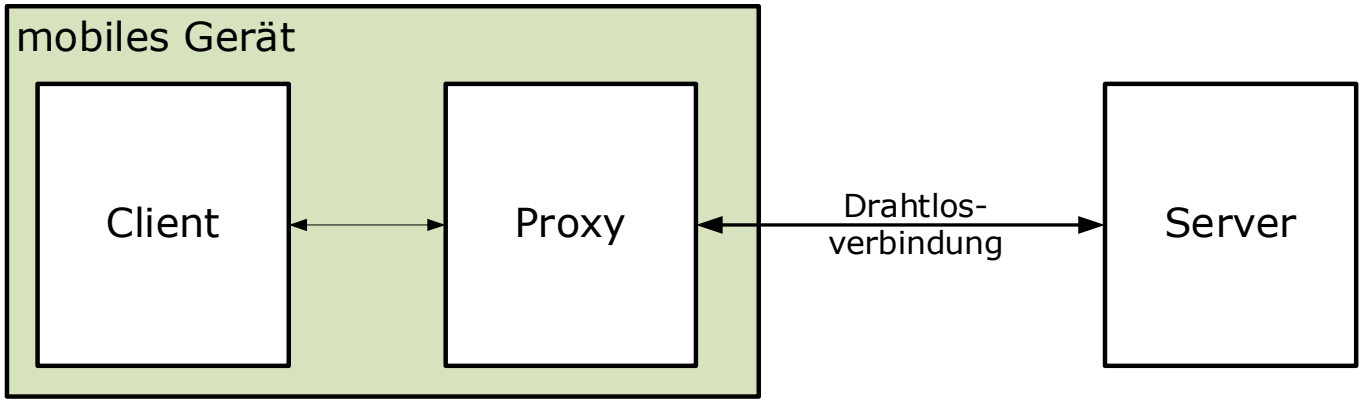
- gleichzeitige Nutzung mehrerer Strukturadaptionsmechanismen



Proxy Pattern

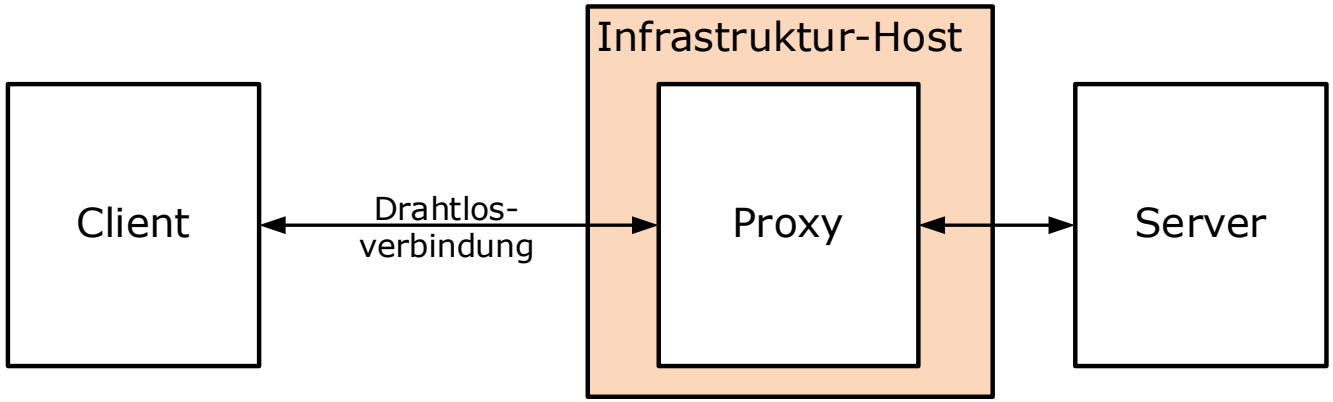
- grundlegendes Muster der Softwarearchitektur (Pattern)
- vermittelnde Abstraktionsebene zwischen Kommunikationspartnern
- indirekte Kommunikation durch den Mediator (Proxy)
- Ziel:
  - Einfügen zusätzlicher Funktionalität
  - Repräsentativkommunikation (Erreichbarkeit, Prokura)
- kann zur Erweiterung existierender Anwendungen und Systeme verwendet werden

Client-Proxy



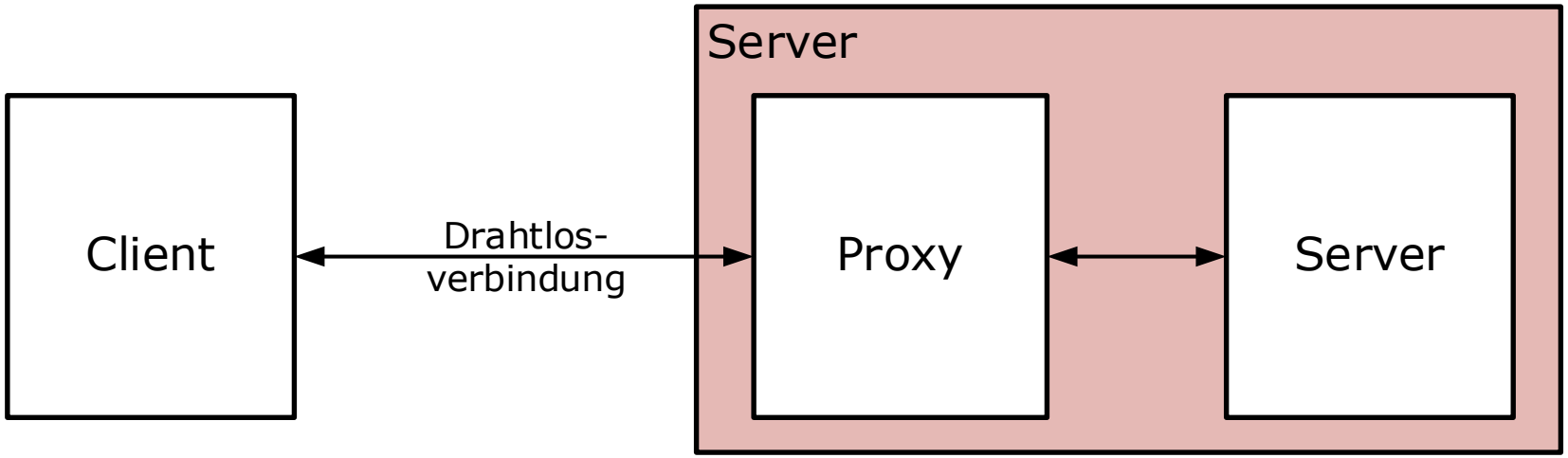
- ermöglicht Erweiterung der Funktionalität des Clients
- Unterstützung von Offline-Operationen (Disconnected Operations)
  - Server-Emulation
  - Caching, Prefetching
- Adaption der Kommunikation vom Client zum Server
- keine Datenreduktion für die Kommunikation vom Server zum Client
- Client bleibt unerreichbar während Offline-Phasen

Infrastruktur-Proxy



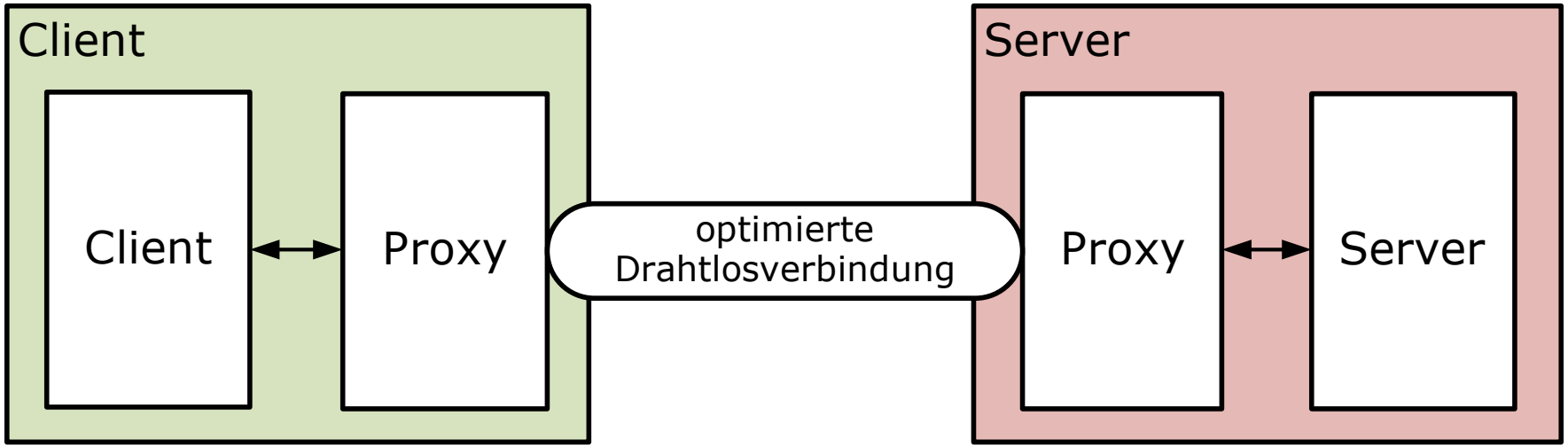
- repräsentiert den Client innerhalb der Infrastruktur
  - Client bleibt während Offline-Phasen (virtuell) erreichbar
- unabhängig von bestimmtem Server
- Datenreduktion für Kommunikation vom Server zum Client
  - i.d.R. als generischer Proxy – erlaubt nur generische Adaption

Server-Proxy



- ermöglicht Erweiterung der Funktionalität des Servers
- Datenreduktion für Kommunikation vom Server zum Client
- anwendungsspezifische Adaptionsfunktionalität
  - Datenfilterung
  - Adaption von Multimediainhalten

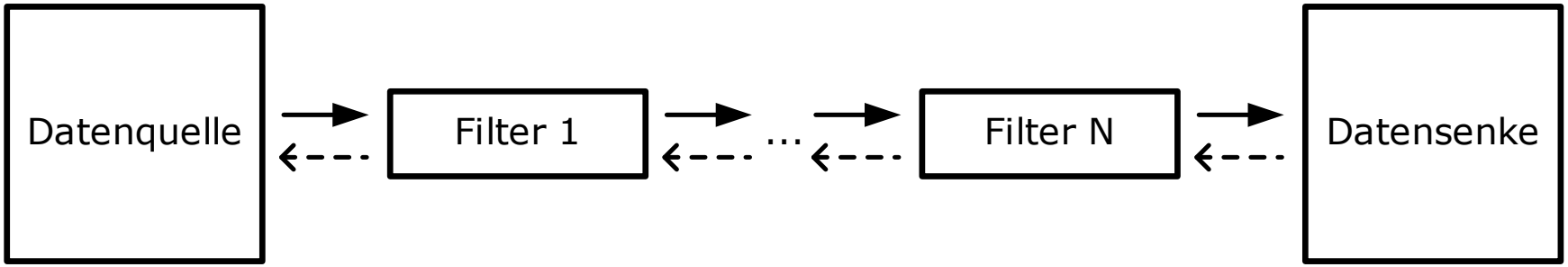
Dumbbell-Proxy



- ermöglicht Erweiterungen auf Client und Server für symmetrische Mechanismen
  - verbesserte/angepasste Kommunikationsprotokolle
  - Fehlerkorrektur
  - Verschlüsselung/Entschlüsselung
  - Enkodierung/Dekodierung

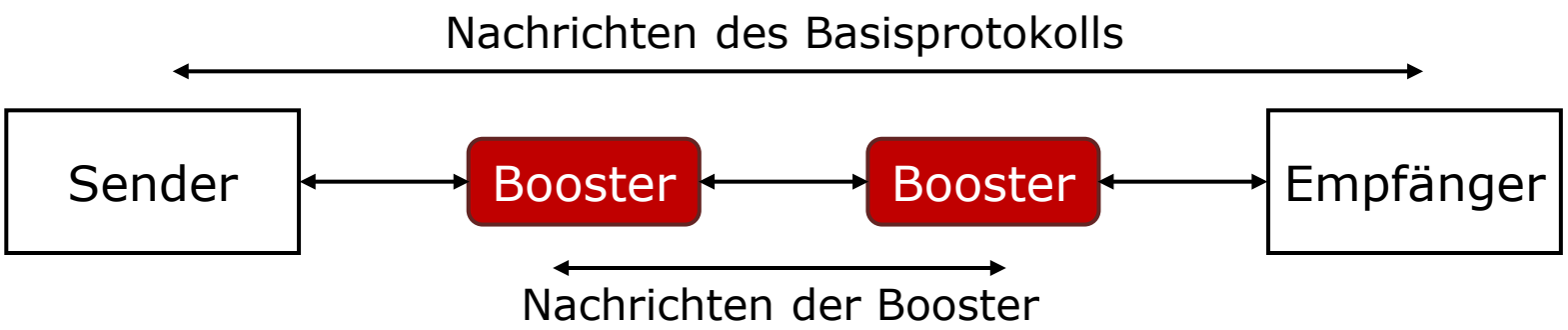
Pipes-und-Filter-Pattern

- Architekturmuster
- Ziel: flexible Konfiguration und Rekonfiguration der Verarbeitung
- Verarbeitung zerlegt in autonome, in Filtern eingebettete Schritte
- lose Kopplung von Filtern – flexible Auswahl der Filtersequenz
- erhöht Wiederverwendbarkeit
- Systembestandteile
  - Datenquelle, Filterkette, Datensenke
  - Filter enthalten typisierte Schnittstellen für Lese-/Schreib-Operationen
  - Pipes verbinden Filter über Schnittstellen kompatiblen Typs
  - Push- oder Pull-Mechanismus zum Datenaustausch



Protokoll-Verstärker (Booster)

- Kombination von Adaptionsmechanismen für Kommunikationsprotokolle
- Booster
  - transparent in Protokollverarbeitung injiziertes Hardware- oder Software-Modul
  - arbeitet auf bestimmtem Basisprotokoll (z.B. TCP oder UDP)
  - Basisprotokoll bleibt unverändert (transparente Injektion)
  - einzelner Booster: Kumulation von TCP-ACK, Erzeugung von duplizierten TCP-ACK, Implementierung des Snoop-Protokolls
  - Booster-Paar: Mechanismen für Vorwärtsfehlerkorrektur, Verschlüsselung, Kompression



Aufgaben

- Überlegen Sie gemeinsam mit Ihren Kommilitonen Beispiele für die Kombination unterschiedlicher Adaptionsmechanismen.
- Welche Mechanismen sind sinnvoll zur Datenmengenreduktion? Welche eher zur Beibehaltung der Qualität (echte sowie wahrgenommene)?
- Diskutieren Sie mit Ihren Kommilitonen welche Adaptionsmechanismen echtzeitfähig sind. Welche sind es nicht? Kann man pauschal eine sinnvolle Aussage dazu treffen? Warum ja/nein?
- Diskutieren Sie mit Ihren Kommilitonen welche Probleme durch die einzelnen Adaptionsmechanismen auftreten können.
  → Gibt es Möglichkeiten, die Auswirkungen auf das Nutzererlebnis zu minimieren?
- Erweitern Sie die Tabelle (Folien 99-102) um die Diskussionsergebnisse!

Datenreduktions- und -ersetzungsmechanismen

Mechanismus	Effekt	Beispiel	Platzierung	Komponen-tenanzahl
Filtern	reduziert Daten- und Informationsmenge; entfernt nicht unter-stützte Typen und Formate	E-Mail-Anhang; ver-linkter Inhalt einer Web-Seite	Proxy oder Server (vor der Übertragung)	1
Konversion	reduziert Daten- und Informationsmenge; adaptiert Dateninhalt (typspezifisch)	Bildskalierung; Bild-ratenreduktion; Ab-tastratenänderung	Proxy oder Server (vor der Übertragung)	1
Selektion	wählt eine alternative Repräsentation aus mehreren	multiple Versionen eines Videos im video-Tag einer Web-Seite	Proxy oder Server	1
Extraktion	Extrahiert alternative Daten; kann Datentyp verändern	Einzelbild aus Video; Dateiname eines entfernten Anhangs	Proxy oder Server	1
Generierung	erzeugt alternative Daten mit gleichem oder reduzierten Infor-mationsgehalt; kann Datentyp verändern	Sprache-zu-Text; Vektorrepräsentation einer Bit-Ebenen-grafik	Proxy oder Server	1

Datentransformationsmechanismen

Mechanismus	Effekt	Beispiel	Platzierung	Komponen-tenanzahl
Kodierungs-transformation	ändert Datenkodierung; reduziert Datenmenge (Kompression)	ASCII mit HTML-Entitäten zu UTF-8; Huffman-Kodierung eines Textes	mobiles Gerät und Proxy oder Server	2
Format-transformation	ändert Datenformat	TIFF zu PNG	mobiles Gerät und Proxy oder Server	2
Struktur-transformation	ändert Datenstruktur	De-/Rekomposition einer E-Mail-Nachricht; Web-Seiten-Parser (DOM→XML)	mobiles Gerät und Proxy oder Server	2

Datenübertragungsmechanismen

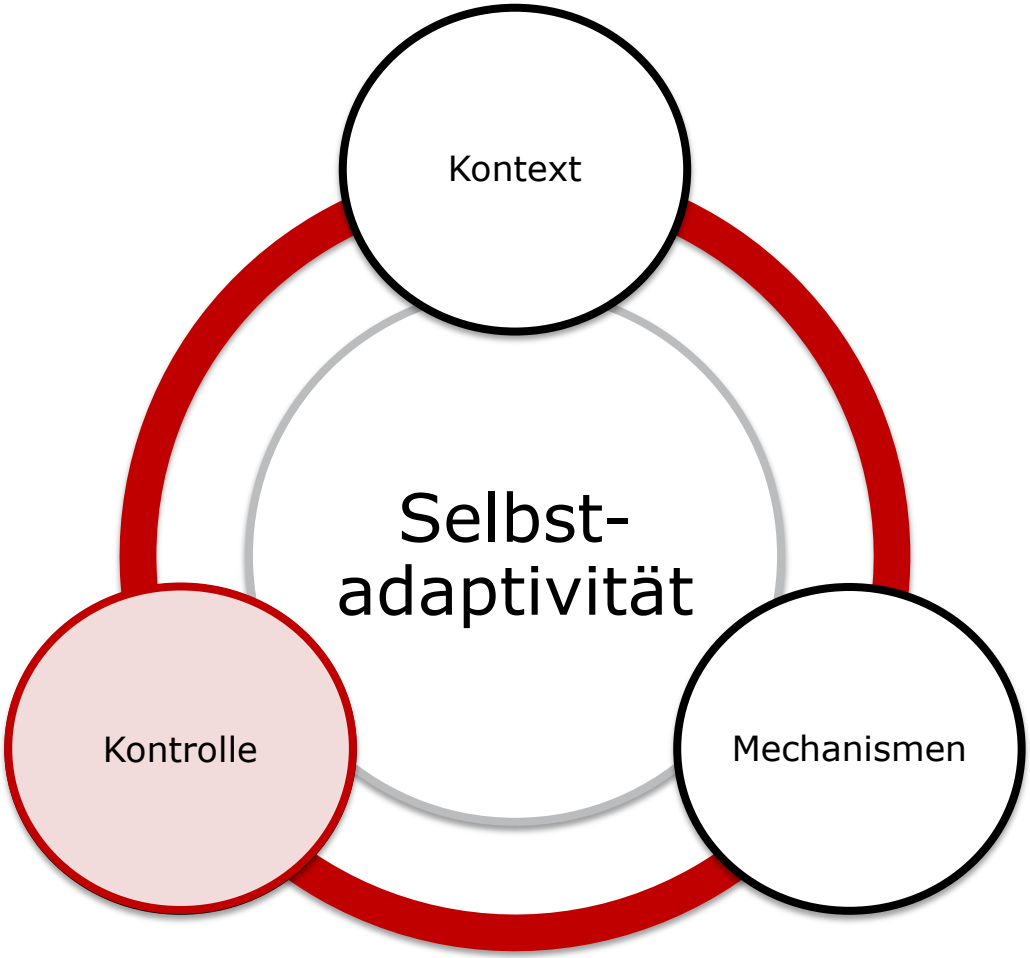
Mechanismus	Effekt	Beispiel	Platzierung	Komponentenanzahl
Anpassung der Protokollparameter	verbessert Bandbreitenausnutzung; verringert Antwortzeit; vermeidet Überlast	Time-out; Fenstergröße; Paketgröße; verringerte Paket-Header	Kommunikationspartner	2
Vorwärtsfehlerkorrektur	kommt ohne Antwortkanal aus; keine Wiederholungsübertragungen	in Kombination: Optimierung der Balance zwischen Redundanz und Wiederholungsübertragungen (bzw. deren Verzögerung)	Kommunikationspartner	2
fehlerkorrigierende automatische Wiederholungsanfrage (ARQ)	vollzieht Wiederholungsübertragungen; keine Redundanz		Kommunikationspartner	2
Datenpriorisierung	multiple Dienstklassen; optimierte Verbindungsnutzung	Priorisierung von Web-Inhalten; Nachrichten unterschiedlicher Anwendungen	Sender (und Empfänger)	1 (2)

Zugriffszeit- und Nachrichtenzustellungsadaption

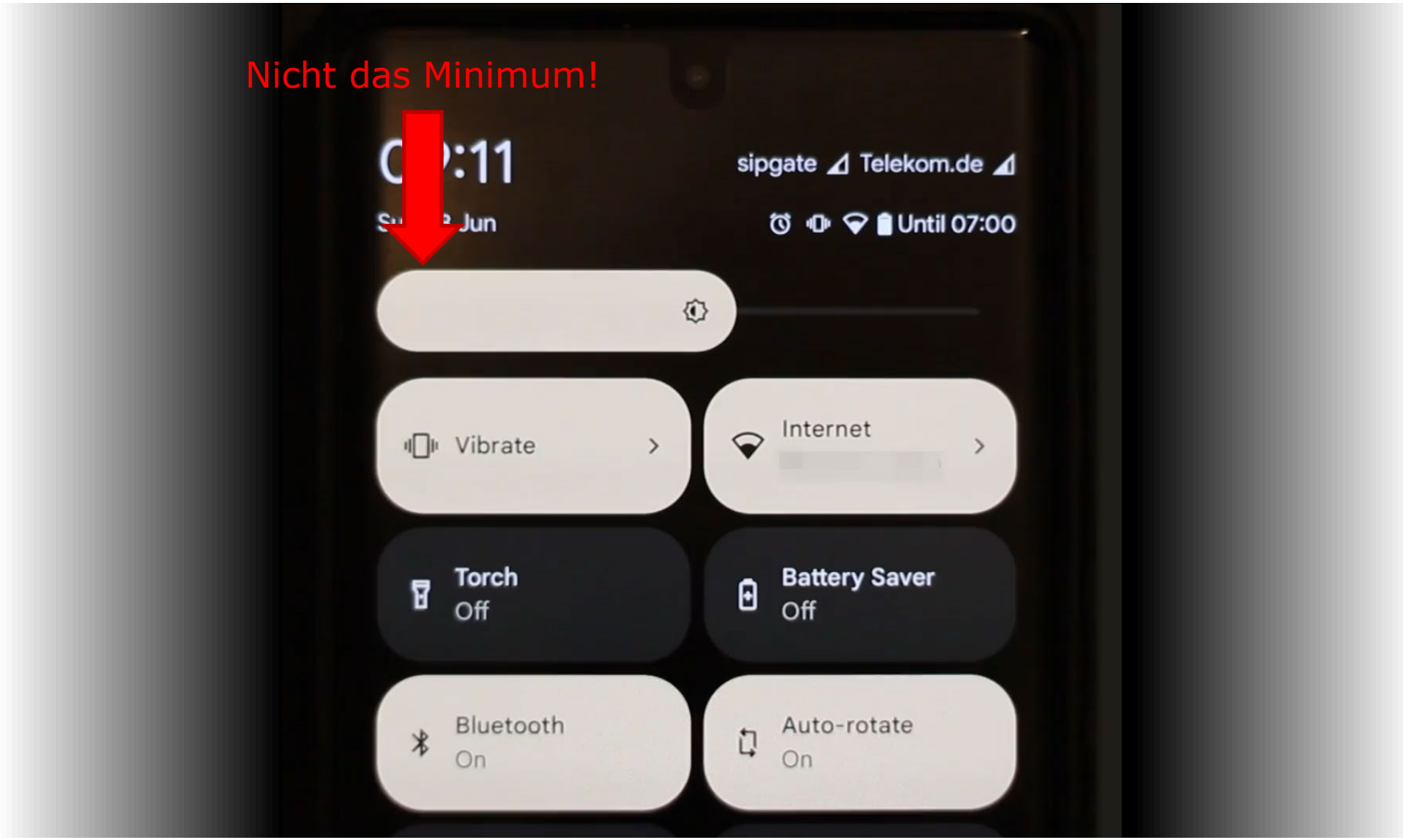
Mechanismus	Effekt	Beispiel	Platzierung	Komponentenanzahl
Caching	reduziert Antwortzeit und Datenmenge; erhöht Verfügbarkeit	CODA (Diconnected Operations); Web-Cache	mobiles Gerät oder Proxy	1
Bulking	Erhöht Durchsatz (und Verzögerung)	Zusammenführen multipler RPC-Nachrichten	mobiles Gerät und Server	2
Queuing	erhöhe Unabhängigkeit; erhöht Antwortzeit; Zwischenspeicher	Nachrichtengetriebene Middleware; XMPP	mobiles Gerät und Server	1 oder 2
Prefetching	reduziert Antwortzeit; erhöht Verfügbarkeit; verschwendet Bandbreite	Web-Inhalte; HTTP/2-Server	mobiles Gerät oder Server	1 oder 2
Lazy Evaluation	reduziert zu übertragende Datenmenge	Aufriss von Web-Inhalten; Google Maps	Mobiles Gerät und Proxy oder Server	2 oder 1
Delayed Write-back	reduziert Daten- und Nachrichtenmenge; verteilt Netzwerklast über Zeit; erhöht Konfliktpotential	CODA (Diconnected Operations); Schreib/Lese-Cache	Mobiles Gerät und Proxy oder Server	2 oder 1

Selbstadaptivität: Kontrolle

Kapitelübersicht

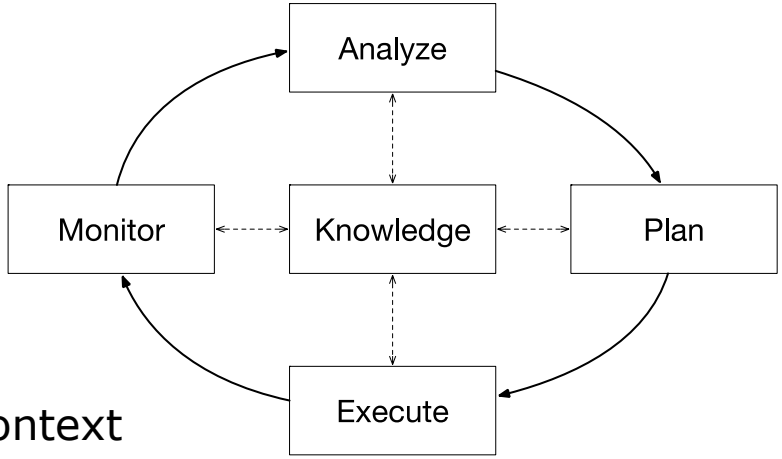


Rückblick: Beispiel 2



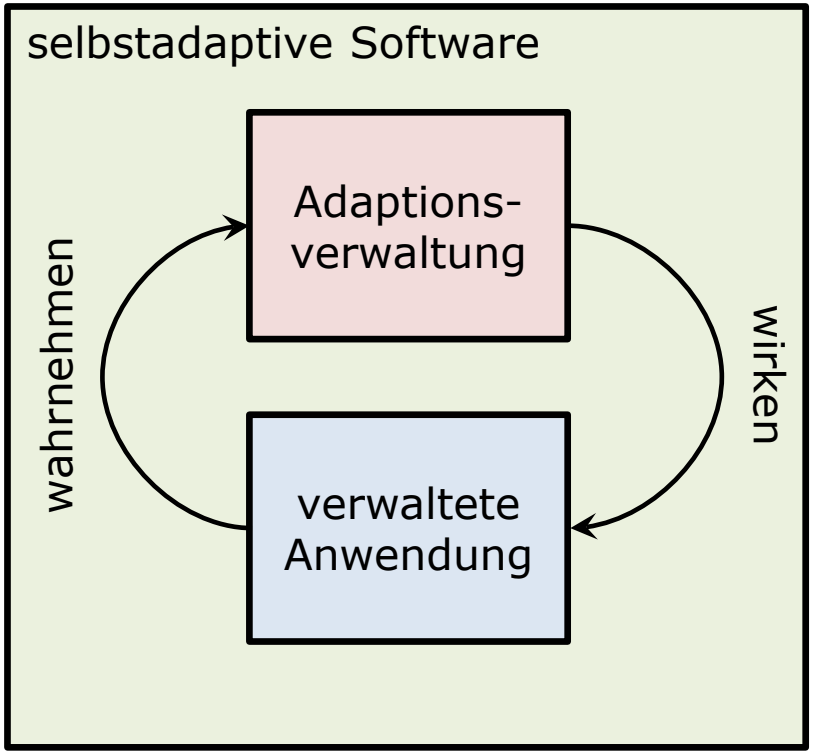
Adaptionskontrolle mit MAPE-K [Kephart et al.]

MAPE-K-Regelkreis als Architekturreferenz zum Entwurf selbstadaptiver Software-Systeme



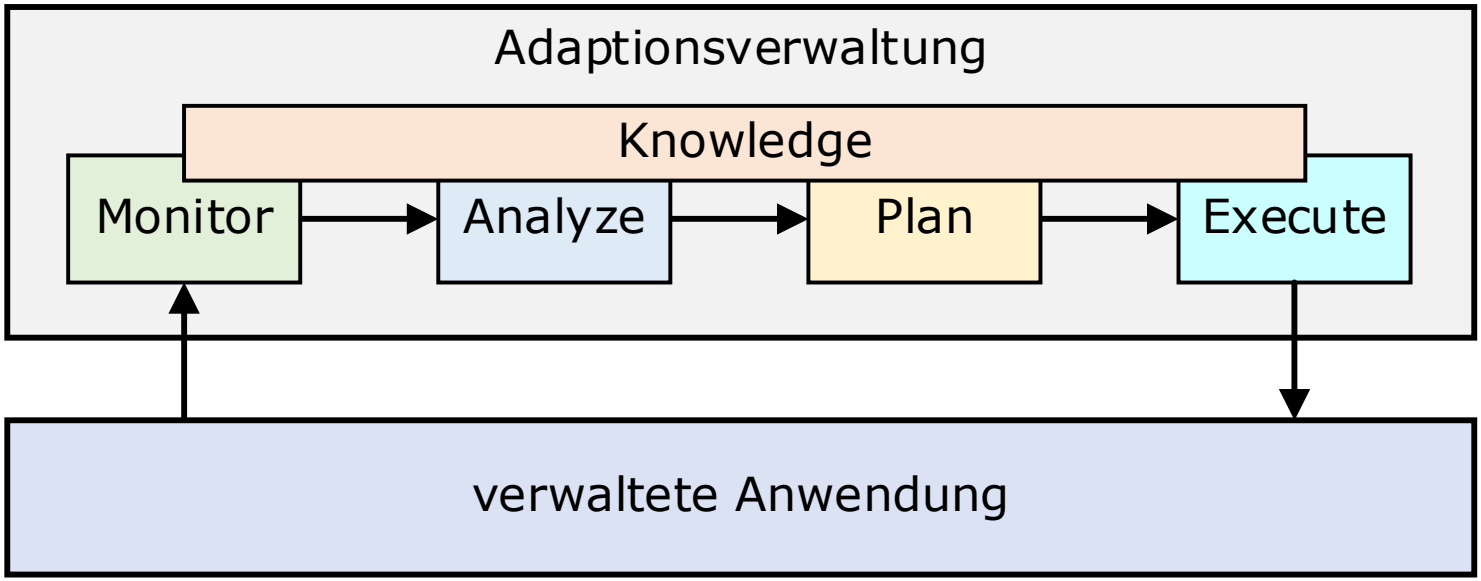
- **M**onitor
  - Sammeln von Daten aus Sensoren und anderen Datenquellen
  - Korrelation von Messwerten zu höherem Kontext
- **A**nalyze
  - Evaluation erfasster Daten und Adaptionsnotwendigkeit bestimmen
- **P**lan
  - Reflektieren des aktuellen Systemzustands und Identifikation angemessener Änderungen
  - Definition von Adaptionsplänen
- **E**xecute
  - Ausführen der Adaptionsentscheidungen auf dem Zielsystem
  - Ändern der Systemkonfiguration
- **K**nowledge
  - Speichern der Informationen über die verwalteten Elemente

Interne und externe Adaptionskontrolle



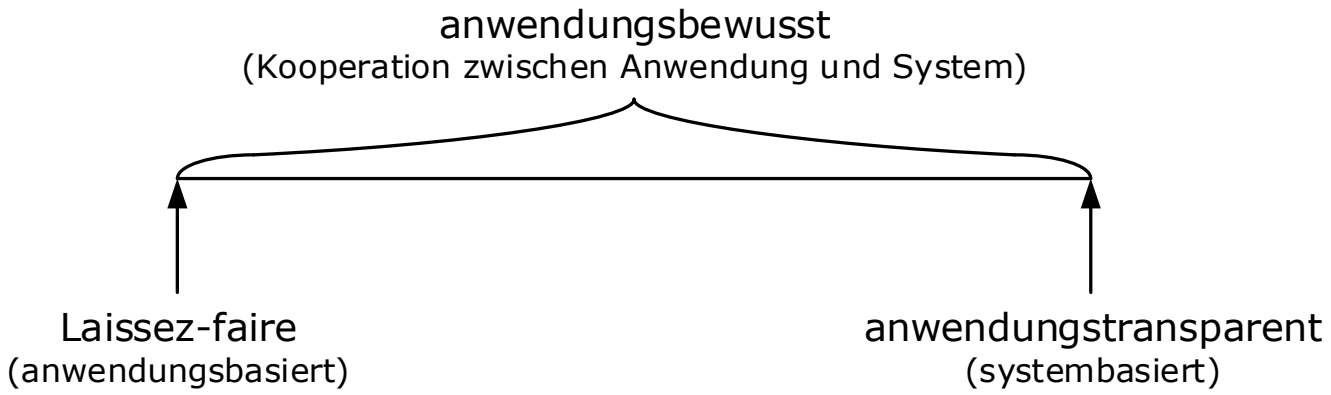
Externe Adaptionskontrolle

- Nutzen einer externen Adaptionsfunktionseinheit (Adaptation Engine)
- klare Trennung der Zuständigkeiten



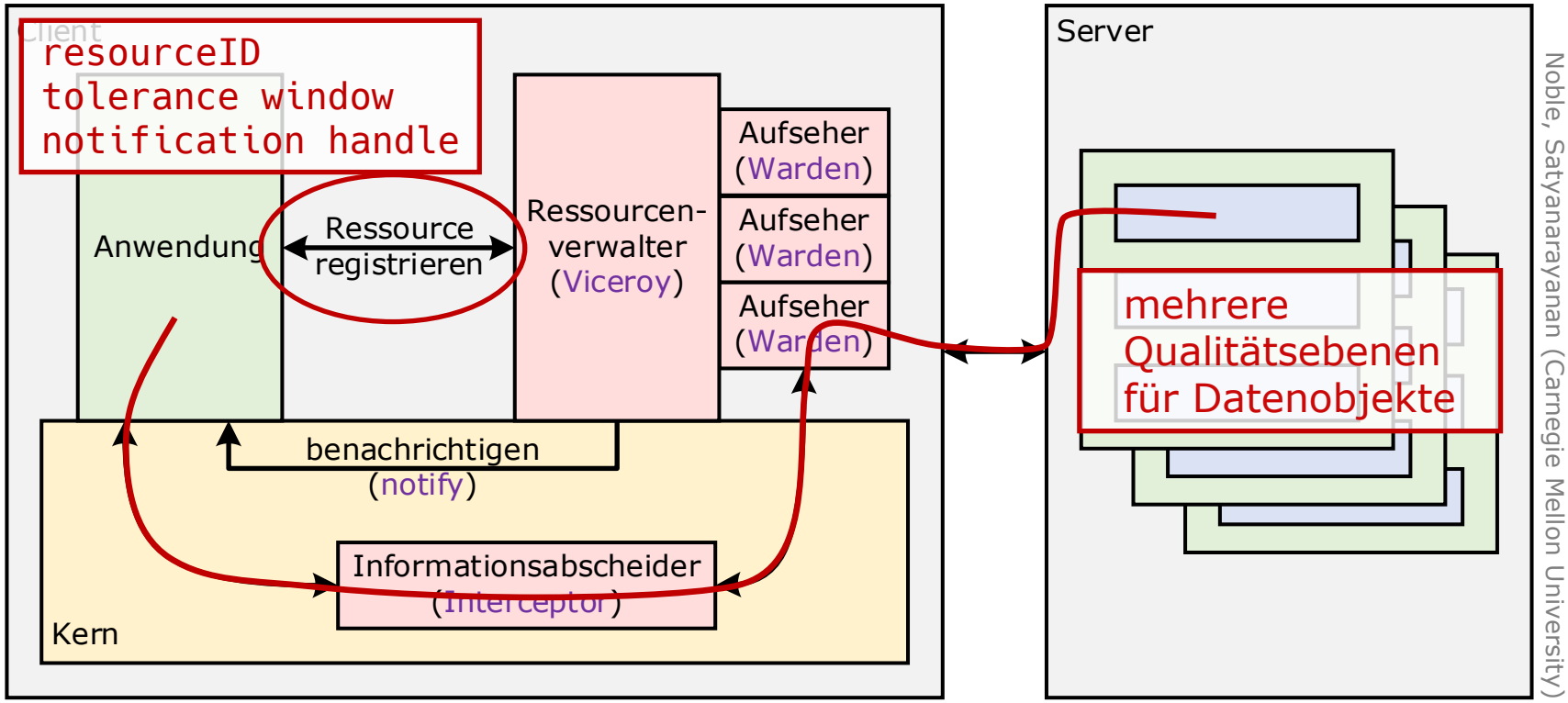
Adaptionsgrad

- Freilauf (Laissez-faire)
  - keine Systemunterstützung bei der Adaption
  - Anwendungen müssen Adaptionsfunktionalität explizit implementieren
  - Anwendungen kontrollieren Ressourcenzuteilung und Adaption
  - weder Optimierung der Ressourcenzuteilung noch der Adaption konkurrierender, nebenläufiger Anwendungen
- anwendungstransparent
  - keine Änderungen an Anwendung notwendig
  - zentralisierte Adaptionskontrolle und Ressourcenzuteilung für konkurrierende, nebenläufige Anwendungen durch das System
  - Adaption ohne Informationen der Anwendungsebene
  - keine anwendungsspezifische Adaptionsfunktionalität



ANWENDUNGSBEWUSSTE ADAPTION

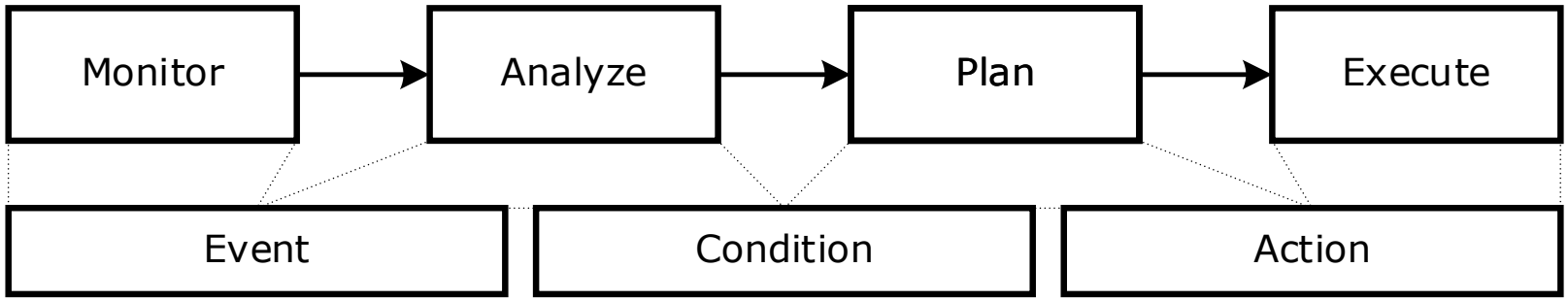
- Kooperation zwischen System und Anwendung
- Kombination der Vorteile
  - anwendungsspezifische Adaptionsoperationen
  - Informationen der Anwendungsebene für Adaption zugänglich
  - zentrale Koordination der Ressourcenzuteilung und Adaption



ECA-Regeln zur Adaptionskontrolle

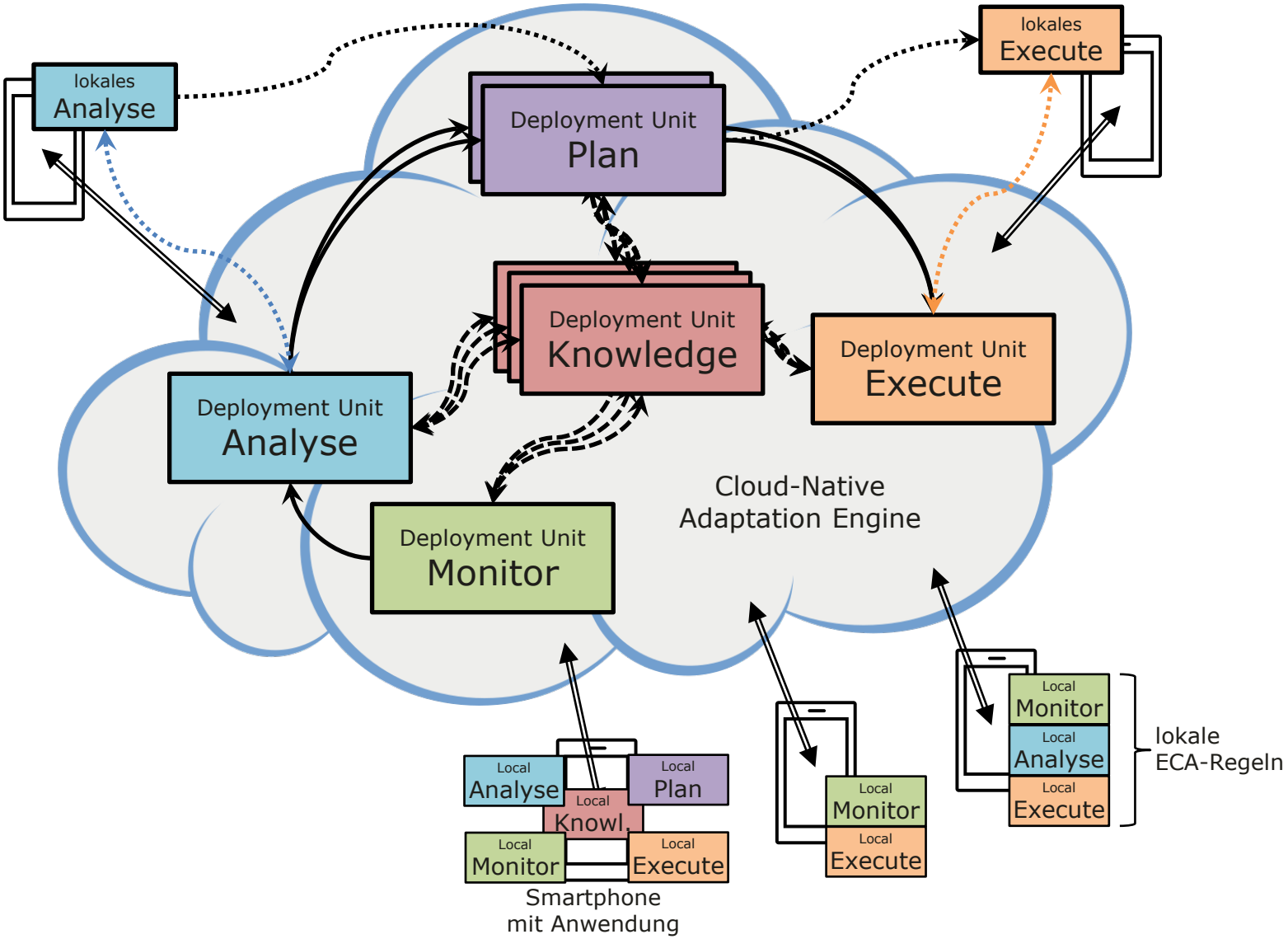
- ECA-Regeln: Event-Condition-Action Rules
  → Regeln zur Definition von Ereignissen, Bedingungen und Aktionen
- reaktive Regeln v.d.F. `onEvent - if - doAction`
  (auf Ereignis – falls Bedingung – mache Aktion)
- sobald ein Ereignis eintritt wird die Bedingung evaluiert und, falls die Bedingung erfüllt ist, die Aktion ausgeführt
- Beispiel:
  - Ereignis: „Joggen"-Aktivität wird gestartet
  - Bedingung: (Netzwerk verfügbar) &&
    (Nutzer ist Mitglied der Gruppe „Training")
  - Aktion: sende Aktivitätsinformationen an alle Mitglieder der Gruppe „Training", die nicht der Nutzer selbst sind

Abbildung von MAPE-K auf ECA-Regeln



→ Wissen (Knowledge) lässt sich in ECA nicht darstellen.

MAPE-K als Cloud-native-Anwendung



Aufgaben

- Diskutieren Sie mit Ihren Kommilitonen weshalb der MAPE-K eine gute Architekturreferenz zum Design selbstadaptiver Systeme ist.
- Diskutieren Sie mit Ihren Kommilitonen wie interne Adaptionskontrolle mittels MAPE-K realisiert werden könnte.
- Erstellen Sie mit Ihren Kommilitonen eine Menge von ECA-Regeln für die beispielhafte Fitness-App.
- Diskutieren Sie mit Ihren Kommilitonen wie anwendungsbewusste Adaption bei der (serverseitigen) Koordinierung paralleler App-Zugriffe helfen kann.
- Erstellen Sie ein konkretes Beispiel anhand dessen Sie Kontext, Adaptionsmechanismen und Adaptionskontrolle erklären können. Diskutieren Sie im Anschluss Ihr Beispiel mit Ihren Kommilitonen.

Zusammenfassung

Zusammenfassung

- Adaption ist ein Schlüsselprinzip der Entwicklung mobiler Anwendungen
- die Mechanismen sind in unterschiedlichen Kontexten anwendbar
- große Auswahl an Adaptionsmechanismen
  - Daten
  - Kommunikation
  - Anwendungsstruktur
- Adaptionsmechanismen sind kombinierbar
- Adaptionskontrolle basierend auf Regelkreisen wie MAPE-K

Referenzen

Loke, S.: Context-Aware Pervasive Systems – Architectures for a New Breed of Applications. Auerbach Publications, ISBN 978-0-8493-7255-1, 2006

Dey, A.K.: Understanding and Using Context. *Personal Ubiquitous Computing,* 2001*, 5*, 4-7

Schilit, B.N.; Adams, N. & Want, R. Context-Aware Computing Applications *Proc. of the 1st Int. Workshop on Mobile Computing Systems and Applications, IEEE,* 1999, 5-90

Winograd, T.: Architectures for Context. *Human-Computer Interaction,* 2001*, 16*, 401-419

Gu, T.; Pung, H. & Zhang, D.: Toward an OSGi-based infrastructure for context-aware applications. *Pervasive Computing, IEEE,* 2004*, 3*, 66-74

Strang, T. & Linnhoff-popien, C.: A Context Modeling Survey *In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England,* 2004

Referenzen

Lieberman, H. & Selker, T.: Out of context: Computer systems that adapt to, and learn from, context. *IBM Systems Journal,* 2000*, 39*, 617

Schmidt, A.; Aidoo, K.A.; Takaluoma, A.; Tuomela, U.; Laerhoven, K.V. & de Velde, W.V.: Advanced interaction in context. *Proceedings of First International Symposium on Handheld and Ubiquitous Computing, HUC'99, Springer Verlag,* 1999, 89-101

T. R. Gruber: A Translation Approach to Portable Ontologies. Knowledge Acquisition, 5(2):199–220, 1993.

T. Springer, K. Kadner, F. Steuer and M. Yin: Middleware Support for Context-Awareness in 4G Environments. World of Wireless, Mobile and Multimedia Networks, 2006. WoWMoM 2006. International Symposium on a, , 203-211 , 2006

Korpipää, P.; Mäntyjärvi, J.; Kela, J.; Kernen, H. & Malm, E. Managing Context Information in Mobile Devices. *IEEE Pervasive Computing,* 2003*, 2*, 42-51

Referenzen

LUTHER, M.; FUKAZAWA, Y.; WAGNER, M. & KURAKAKE, S.: Situational reasoning for task-oriented mobile service recommendation. *The Knowledge Engineering Review,* 2008*, 23*, 7-19

E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X.Zheng, and A. T. Campbell: Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe application. In Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys '08), 2008.

P. Demeester and K. Luyten and B. Dhoedt and S. Dauwe and F. D. Turck and O. V. Laere and M. Strobbe and F. Ongenae: Novel Applications Integrate Location and Context Information, IEEE Pervasive Computing 11 64-73 (2011)

T. Springer and A.-Y. Turhan: Employing Description Logics in Ambient Intelligence for Modeling and Reasoning about Complex Situations. Journal of Ambient Intelligence and Smart Environments (JAISE), 235 - 259 , 2009

Referenzen

Vermeulen, A., Beged-Dov, G., Thompson, P.: *The Pipeline Design Pattern.* Workshop on Design Patterns for Concurrent, Parallel and Distributed Object-Oriented Systems OOPSLA 95. 1995.

Gamma, E., Helm, R., Johnson, R. E.: *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley, ISBN-13: 978-0201633610, 1995.

Watson, T.: *Effective wireless communication through application partitioning*. Proceedings of the Fifth Workshop on Hot Topics in Operating Systems (HotOS-V), S. 24-27, 1995.

Noble, B. D., Satyanarayanan, M, Narayanan, D., Tilton, J. E., Flinn, J., Walker, K. R.: *Agile Application-Aware Adaptation for Mobility.* In: proceedings of the 16 ACM Symposium on Operating System Principles, 1997

Gribble, S. D., Welsh, M., von Behren, R., Brewer, E. A., Culler, D., Borisov, N., Czerwinski, S., Gummadi, R., Hill, J., Josheph, A., Katz, R.H., Mao, Z. M., Ross, S., Zhao, B.: *The Ninja Architecture for Robust Internet-Scale Systems and Services. Special Issue on Pervasive Computing, Vol. 35, No. 4,* 2001.

Referenzen

Marcus, W. S., Hadzic, I., McAuley, A. J., Smith, J. M.: *Protocol Boosters: Applying Programmability to Network Infrastructures. IEEE Communications* Magazine Vol. 36, No. 10, S. 79-83. 1998.

P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf, "An architecture-based approach to self-adaptive software," *IEEE Intell. Syst.*, vol. 14, no. 3, pp. 54–62, May 1999.

Jeffrey O Kephart and David M Chess. "The Vision of Autonomic Computing." In:Computer 36.1 (Jan. 2003), pp. 41–50.

Springer, T.: Ein komponentenbasiertes Meta-Modell kontextabhängiger Adaptionsgraphen für mobile und ubiquitäre Anwendungen; Dissertation, TU Dresden, 2004