Karlsruher Institut für Technologie

Fraunhofer IOSB

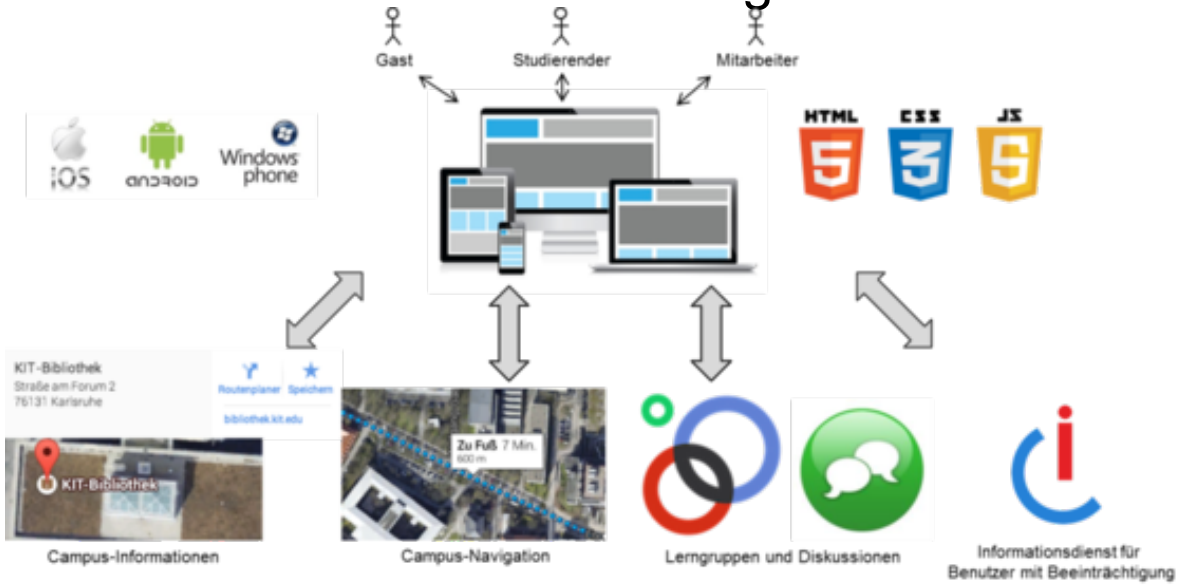
Entwicklung von ressourcenorientierten Services mithilfe des API-Design-First-Prinzips

Pascal Giessler, Reinhard Herzog, Sebastian Abeck

COOPERATION & MANAGEMENT (C&M, PROF. ABECK), INSTITUT FÜR TELEMATIK, FAKULTÄT FÜR INFORMATIK

Beteiligte Partner und SmartCampus

(1) Die KIT-Forschungsgruppe C&M und das Fraunhofer IOSB kooperieren im Bereich der Web-Technologien
   (1) Serviceorientierte Architekturen, RESTful Web-Services, Internet der Dinge, Ontologien
   (2) SmartCampus ist eine aus verschiedenen gemeinsamen Projekten hervorgegangene und kontinuierlich weiterentwickelte serviceorientierte Web-Anwendung



GRUNDLAGEN – Kurz und knapp

(1) API (Application Programming Interface)
  (1) Vertrag zwischen Dienstnehmer und Dienstgeber
  (2) Ermöglicht Zugriff auf Daten oder Funktionalitäten
  (3) Web-API als Ausprägung einer API, welche den Zugriff über das Web ermöglicht

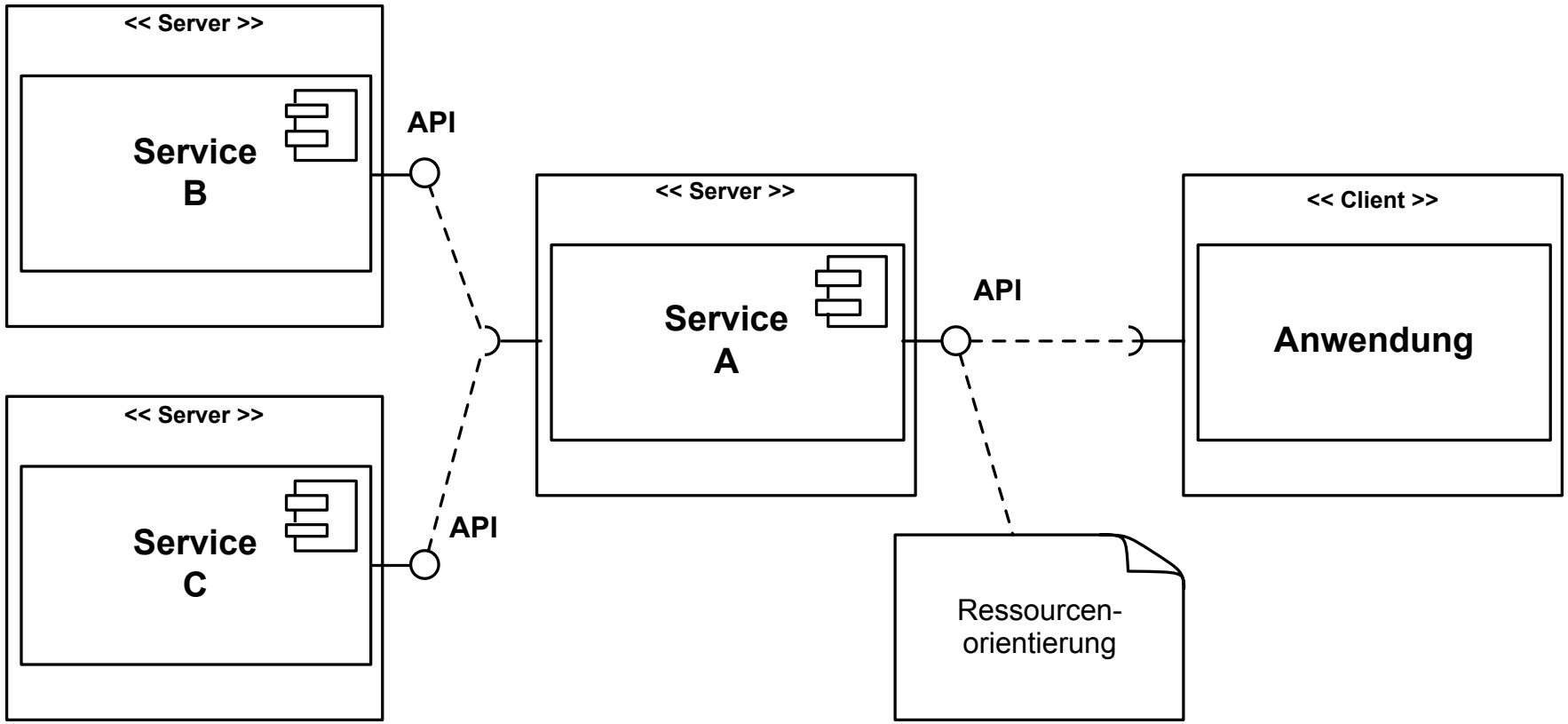
(2) Ressourcenorientierung
  (1) Bestandteil des Architekturstils REST von Fielding
  (2) Zweite Ebene des Richardson Maturity Modells
    (1) Geschäftsentitäten ≈ Ressourcen
    (2) Geschäftsmethoden = HTTP-Methoden

(3) Service
  (1) Bereitstellung von Informationen und Funktionalitäten
  (2) Web-Service als Ausprägung eines Service mit einer Web-API

MOTIVATION

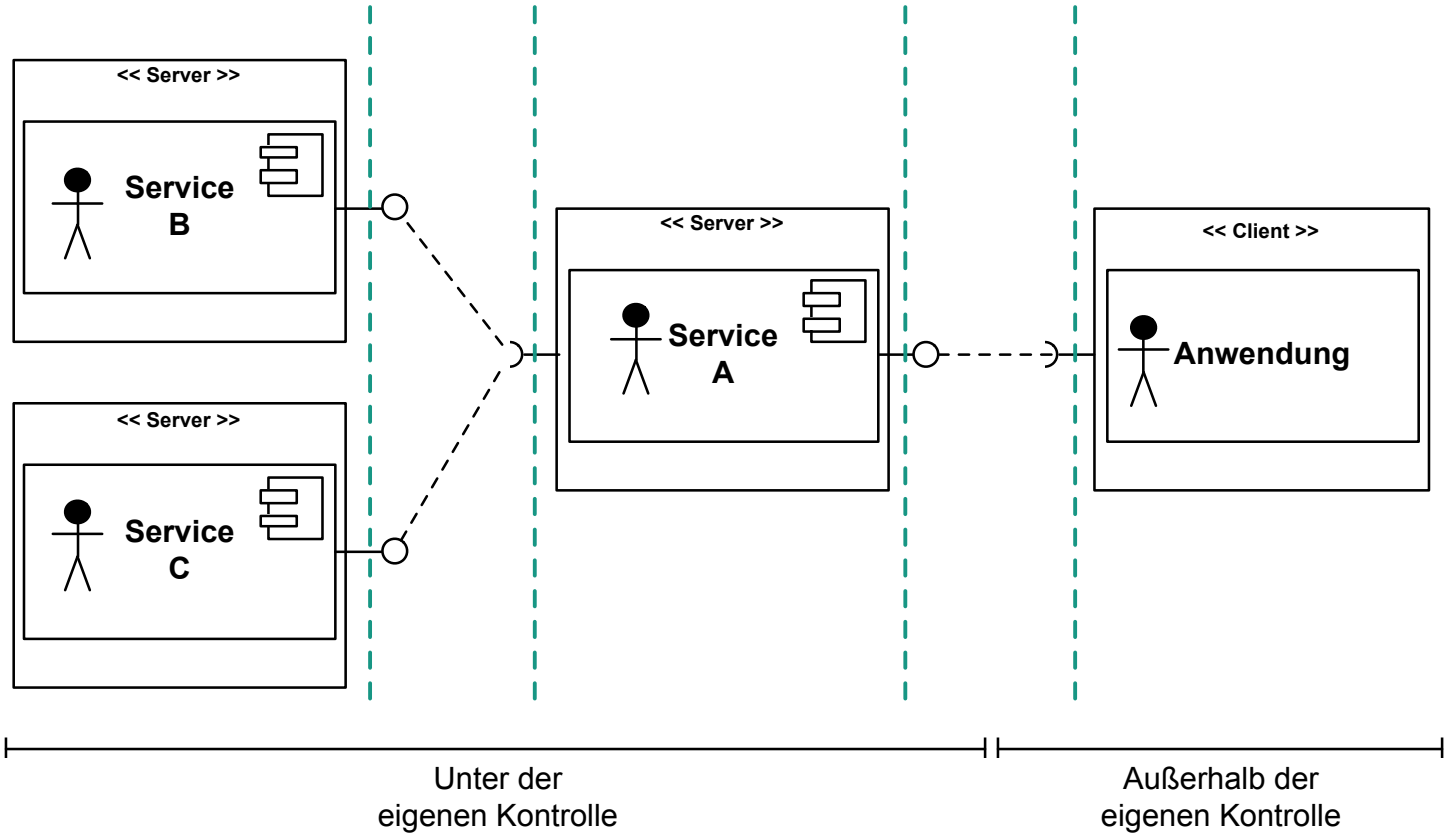
(1) Autonome Entwicklungsteams

(1) Zuständigkeit des Teams gegeben durch Bounded Context



HERAUSFORDERUNGEN

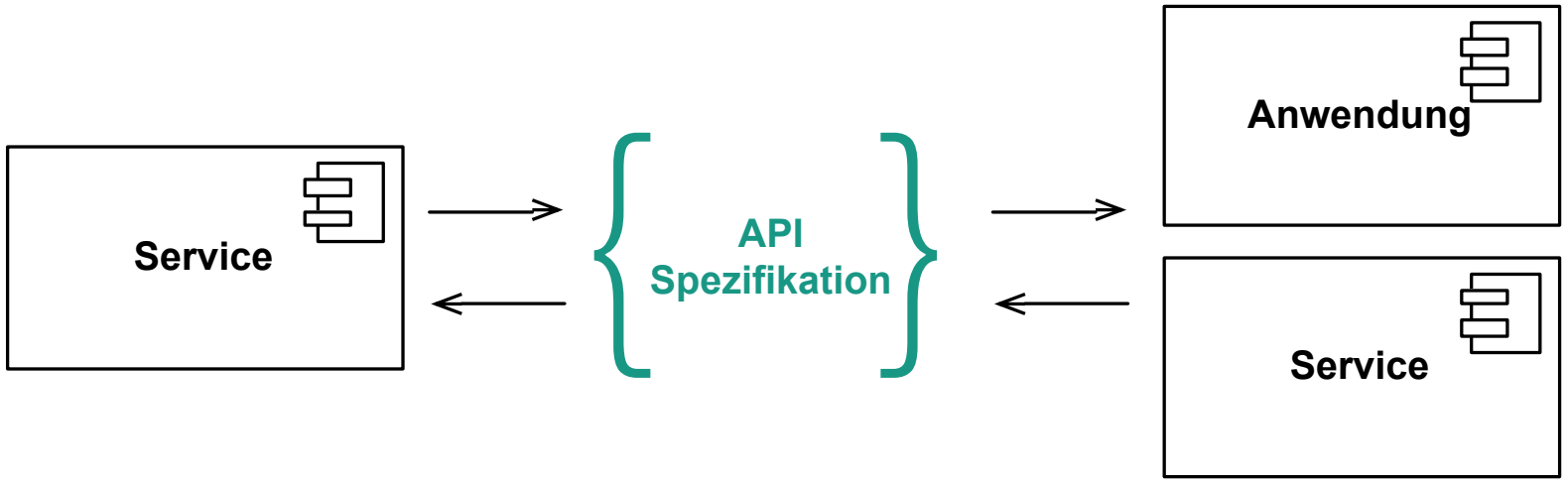
(1) Sicherstellung der Autonomie einzelner Entwicklungsteams
   (1) Reduktion des Koordinationsaufwands
   (2) Reduktion des Kommunikationsaufwands



LÖSUNGSANSATZ

(1) Reduktion des Koordinationsaufwands / Kommunikationsaufwands
  (1) Vertrag für die Kommunikation zwischen Dienstnehmer und –geber
    (1) Fester Bestandteil des Entwurfsprozesses und Grundlage für die Implementierungsphase

(2) Diese Entwicklungsmethodik wird auch als API-First bezeichnet
  (1) API-Spezifikation als zentrales Entwurfsartefakt



Werkzeuge und Sprachen für die API-Spezifikation

(1) Zahlreiche Werkzeuge/ Sprachen zur Spezifikation von APIs
  (1) WADL, WSDL 2.0
  (2) RAML, Swagger (OpenAPI), API Blueprint, apiDoc, slate

(2) Neuere Ansätze unterstützen meist folgende Inhaltstypen:
  (1) YAML, JSON, (Markdown)

(3) Relevante Kriterien für die Auswahl bei C&M
  (1) Spezifikation unabhängig von der eingesetzten Technologie
  (2) Generierung von menschenlesbarer u. verständlicher Dokumentation
  (3) Großes Ökosystem (Generierung von Quellcode, API Konsole u.s.w)
  (4) Open-Source

Open API-Spezifikation (ehemals Swagger)

(1) Herstellerneutraler Standard für die Beschreibung von RESTful APIs

(2) Entwicklung vorangetrieben von der Open API Initiative
   (1) Zusammenschluss einiger Firmen (u.a. Google, Microsoft)
   (2) Unter dem Dach der Linux Foundation

(3) *"The goal […] is to define a standard, language-agnostic interface to REST APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection."*[1]

(4) Hervorgegangen aus *Swagger*
   (1) Formale Spezifikation (Grundlage der Open API Specification)
   (2) Großes Ökosystem aus Werkzeugen und Bibliotheken

[1] OpenAPI Initiative: Website, https://openapis.org

Beispiel: API Spezifikation im Open API-Format

(1) Nutzt YAML(hier) oder JSON

```
1.  swagger: "2.0"
2.  info:
3.    version: "1.0"
4.    title: "Hello World API"
5.  paths:
6.    /hello/{user}:
7.      get:
8.        description: Returns a greeting to the user!
9.        parameters:
10.         - name: user
11.           in: path
12.           type: string
13. description: The name of the user to greet.
14.       responses:
15.         200:
16.           description: Returns the greeting.
17.           schema:
18.             type: string
19.         400:
20.           description: Invalid characters in "user" were provided
```

Anfertigen einer API-Spezifikation

(1) Beispiel: Inkonsistente API-Spezifikationen

```
1.  swagger: "2.0"
2.  info:
3.    version: "1.0"
4.    title: "Sales API"
5.  paths:
6.    /salesOrders/{id}:
7.      get:
8.        description: {…}
9.        parameters:
10.         - name: id
11.           in: path
12.           type: integer
13.   /cart/{id}
14.     post:
15.       description: {…}
16.       parameters:
17.         - name: articleName
18.         - in: body
```

```
1.  swagger: "2.0"
2.  info:
3.    version: "1.1"
4.    title: "Sales API"
5.  paths:
6.    /sales-orders/{uuid}:
7.      get:
8.        description: {…}
9.        parameters:
10.         - name: uuid
11.           in: path
12.           type: string
13.   /carts/{uuid}
14.     post:
15.       description: {…}
16.       parameters:
17.         - name: article_name
18.         - in: body
```

Identifiziertes Problem

(1) Die Spezifikation von APIs erlaubt sehr viele Freiräume
  (1) REST ist ein Architekturstil und keine Schnittstellenspezifikation
  (2) Keine existierenden Standards oder genaue Richtlinien
    (1) Kann zu inkonsistenten APIs führen

(2) Der Entwurf und die Spezifikation von APIs ist Teil eines Entscheidungsprozesses
  (1) Attraktivität und Mehrwert einer API
  (2) Qualitätseigenschaften eines Web-Service

(3) Die Gestaltung von "guten" APIs ist bis heute eine Herausforderung
  (1) "It is very easy to create a bad API and rather difficult to create a good one. Even minor and quite innocent design flaws have a tendency to get magnified out of all proportion because APIs are provided once, but are called many times.", Henning

Festlegen von API-Richtlinien

(1) Befolgen von grundlegenden Eigenschaften einer API
   (1) Einfaches Verständnis
   (2) Technologieunabhängig
   (3) Konsistent in der Benutzung
   (4) Robust gegenüber Evolution

(2) Aufstellen von Richtlinien für die Spezifikation von APIs
   (1) Bewährte Methoden aus Praxis und Forschung

Beispiel: Richtlinie 3.8.2 – Nutzung eines allgemeingültigen Error-Objekts

```
1  {
2   "problem": {
3     "type": "http://httpstatus.es/503",
4     "status": 503,
5     "error_code": 186,
6     "title": "The service is currently under
          maintenance",
7     "detail": "Connection to database timed out",
8     "instance": "http://.../errors/186"
9    }
10 }
```

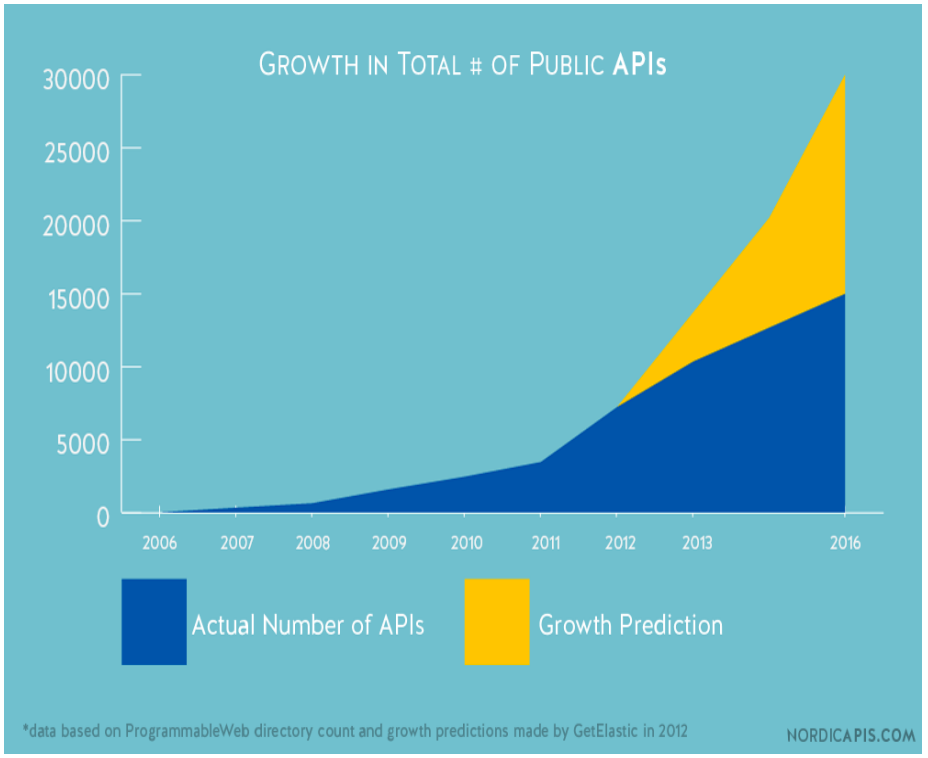
Weitere Vorteile für qualitätsgesicherte APIs

(1) Die Bereitstellung von Funktionalitäten über (öffentliche) APIs kann heutzutage als ein eigenständiges Geschäftsfeld angesehen werden
  (1) Salesforce (50% / Umsatz), Expedia (90% / Umsatz), Ebay (60% / Umsatz)

(2) Nutzungsaufkommen durch APIs pro Jahr um fast das Dreifache

(1) Zentraler Bestandteil der digitalen Transformation

(2) Ermöglichen neue Geschäfts-felder (z.B. FinTech)

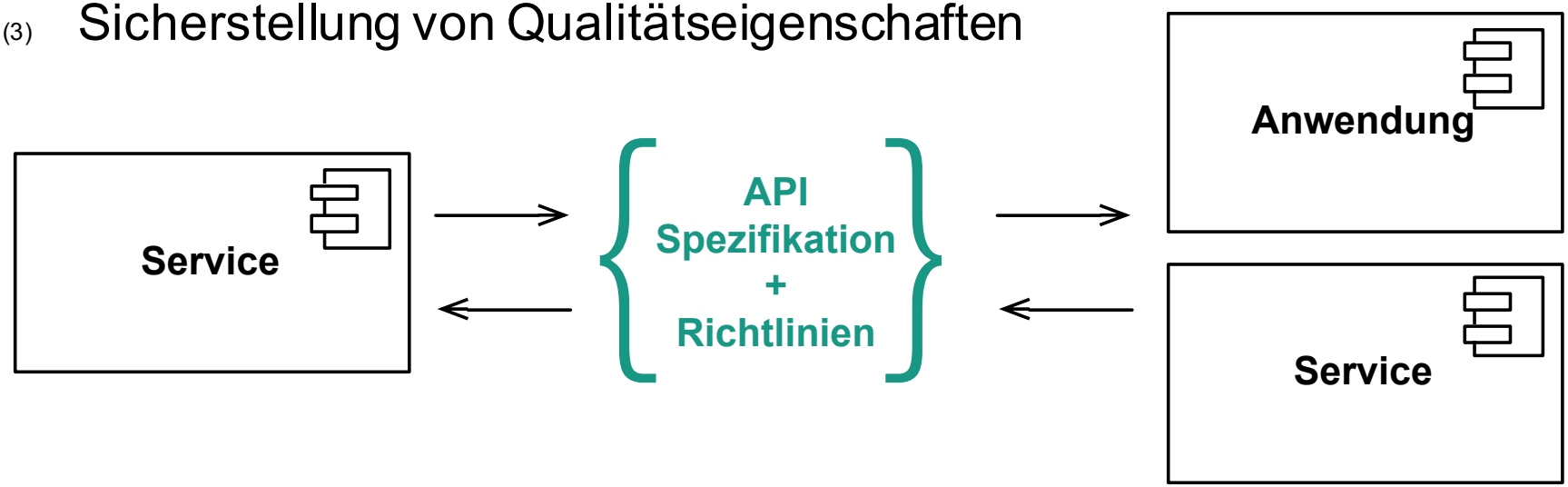


Erweiterter Lösungsansatz

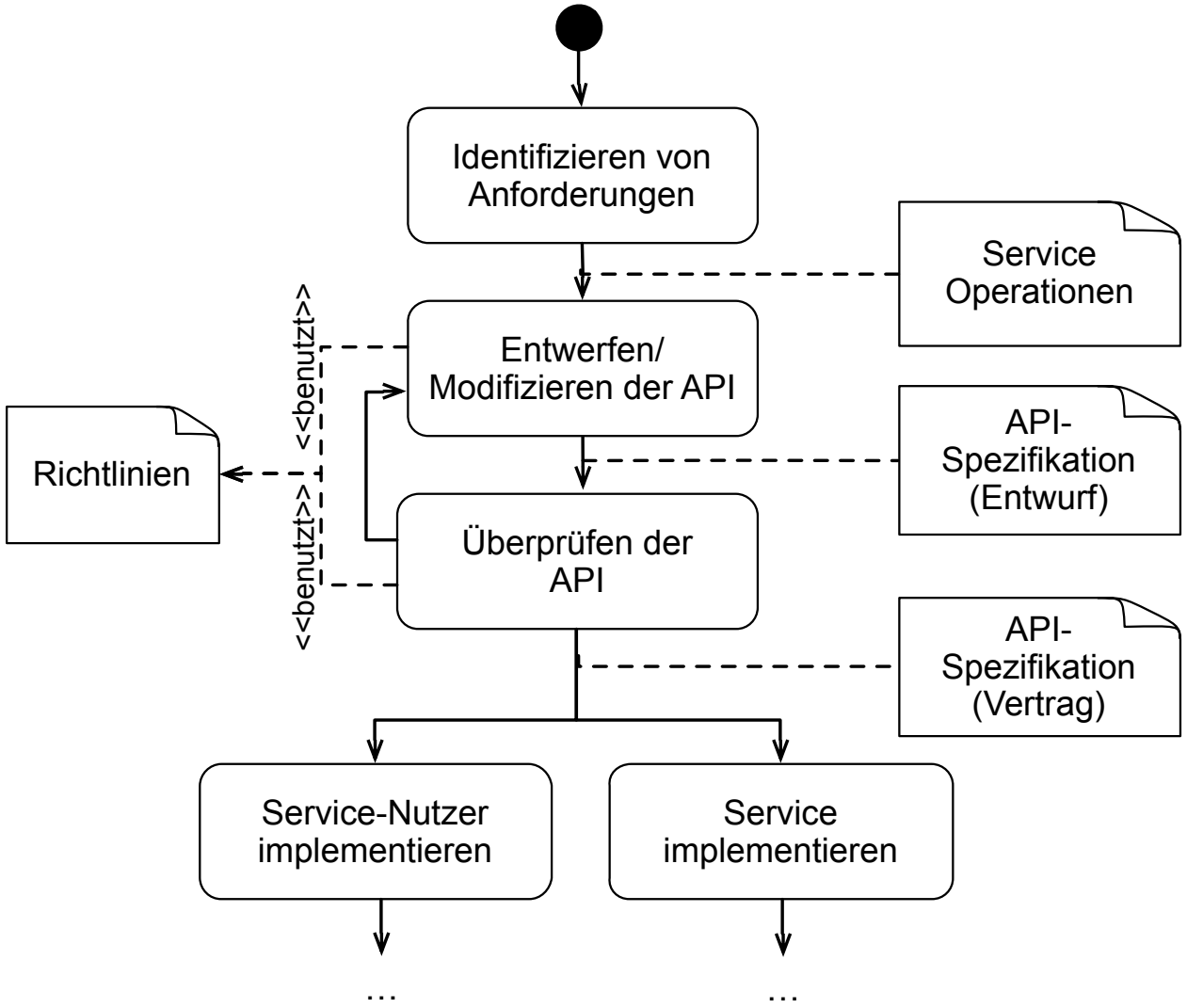
(1) Reduktion des Koordinationsaufwands
   (1) Vertrag zwischen Dienstnehmer und -geber vor der Entwicklung
   (2) Vermeidung von nicht abwärtskompatiblen API-Änderungen

(2) Reduktion des Kommunikationsaufwands
   (1) Verständliche und nachvollziehbare Dokumentation
   (2) Verständliche und klare Fehlermeldungen

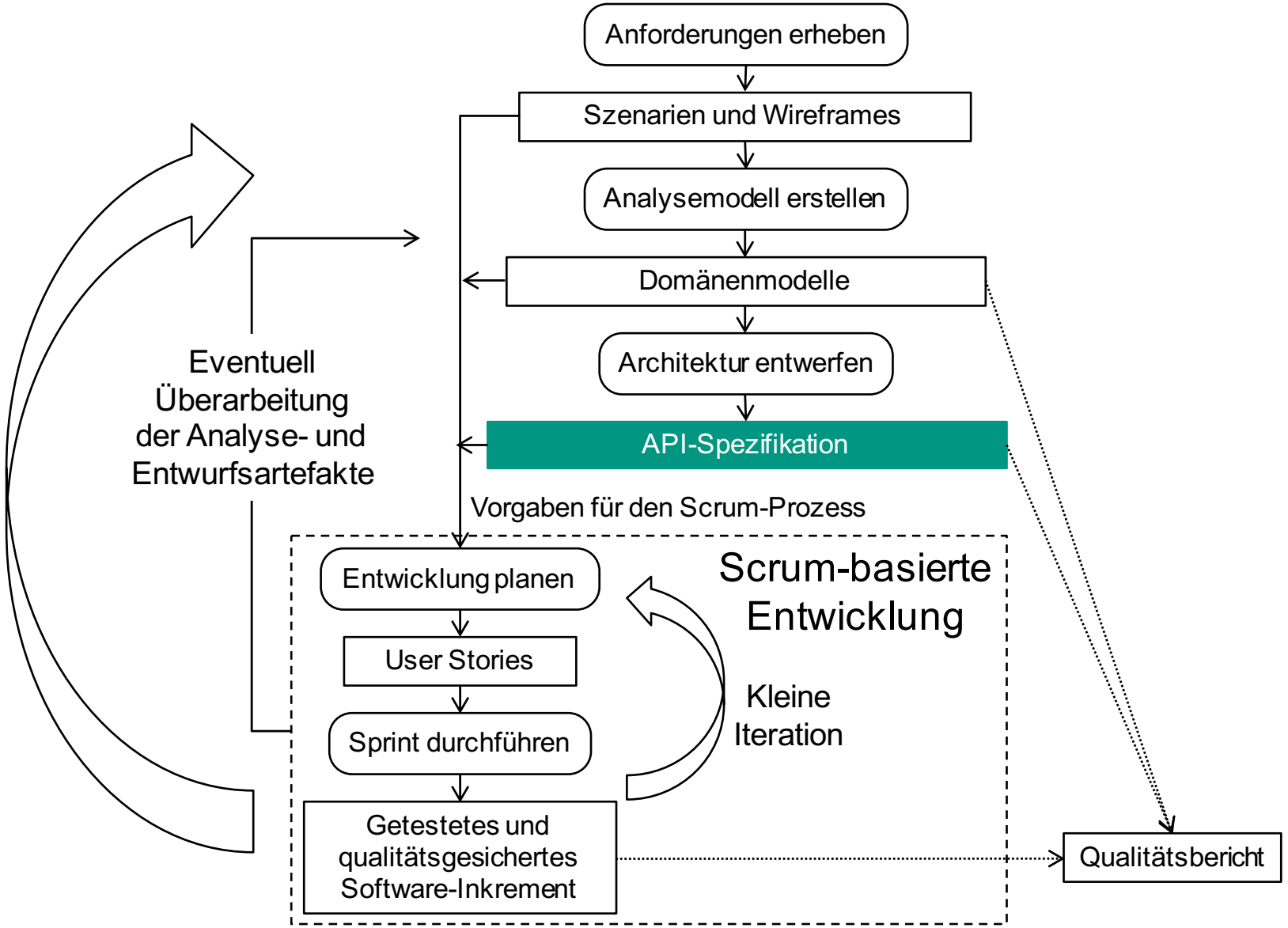
(3) Sicherstellung von Qualitätseigenschaften



Iterativer Prozess für die Anfertigung oder Überarbeitung einer API-Spezifikation



Einordnung in Softwareentwicklung bei C&M



VORZÜGE DURCH API-FIRST UND AUFGESTELLTER RICHTLINIEN

Karlsruher Institut für Technologie

(1) Sicherstellung der Autonomie unabhängiger Entwicklungsteams
    (1) Autonome Abarbeitung möglich
    (2) Klar definierte Anforderungen
    (3) Verständliche und aktuelle Dokumentation

(2) Verkürzung der Ramp-up-Phase von Entwicklungsprojekten
    (1) Einsatz von Modelgetriebener Softwareentwicklung
        (1) Generierung von Applikations- und Service-Schablonen
    (2) Spezifikation möglich, wenn nicht mit Technologie-Stack vertraut

(3) Veröffentlichung von APIs für Drittanbieter möglich
    (1) Neue Geschäftsfelder (FinTechs), z.B. Number26

(4) Qualität einer API steigern und sicherstellen
    (1) Evolvierbarkeit, Performanz, Konsistenz, Kompatibilität, Wartbarkeit

Fraunhofer IOSB

ZUSAMMENFASSUNG

(1) Problemstellung bei der Entwicklung einer Service-Landschaft
  (1) Autonome Entwicklungsteams

(2) Vorstellung eines API-First orientierten Lösungsansatzes
  (1) Reduktion des Koordination- und Kommunikationsaufwands
    (1) API-Spezifikation vor Implementierungsphase
  (2) Sicherstellung von Qualitätseigenschaften
    (1) Notwendigkeit von Richtlinien

(3) Entwicklungsprozess und -werkzeuge für die API-Spezifikation
  (1) Iterativer und inkrementeller Prozess
  (2) Open API und Swagger

(4) Einblick in SmartCampus
  (1) Demonstration und Tragfähigkeitsnachweis

AUSBLICK

(1) Entwicklung von weiteren Services als Teil des SmartCampus

(2) Entwicklung einer Service-Registry und –Discovery-Lösung
   (1) Unterstützung des API-First-Ansatzes
   (2) Wiederfinden von bestehenden Funktionalitäten
      (1) Semantische Ableitung von Informationen möglich
   (3) Unterstützung zur Laufzeit- als auch Entwicklungszeit
   (4) Qualitätssicherung zur Einhaltung von API-Richtlinien

(3) Tragfähigkeitsnachweis bei mehreren Unternehmen geplant