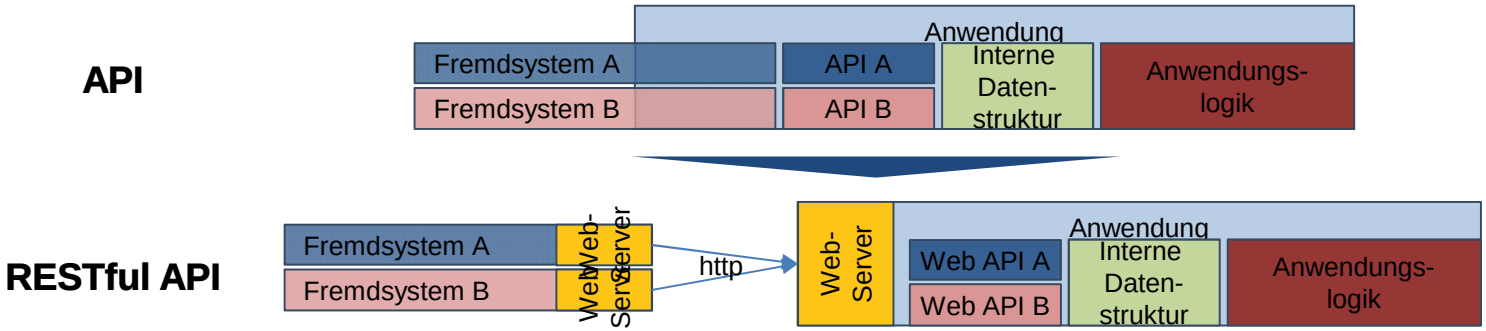
Kompositionssprache für vernetzte VT-Systeme

Andreas Harth, Felix Leif Keppmann, KIT

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung
DLR Projektträger

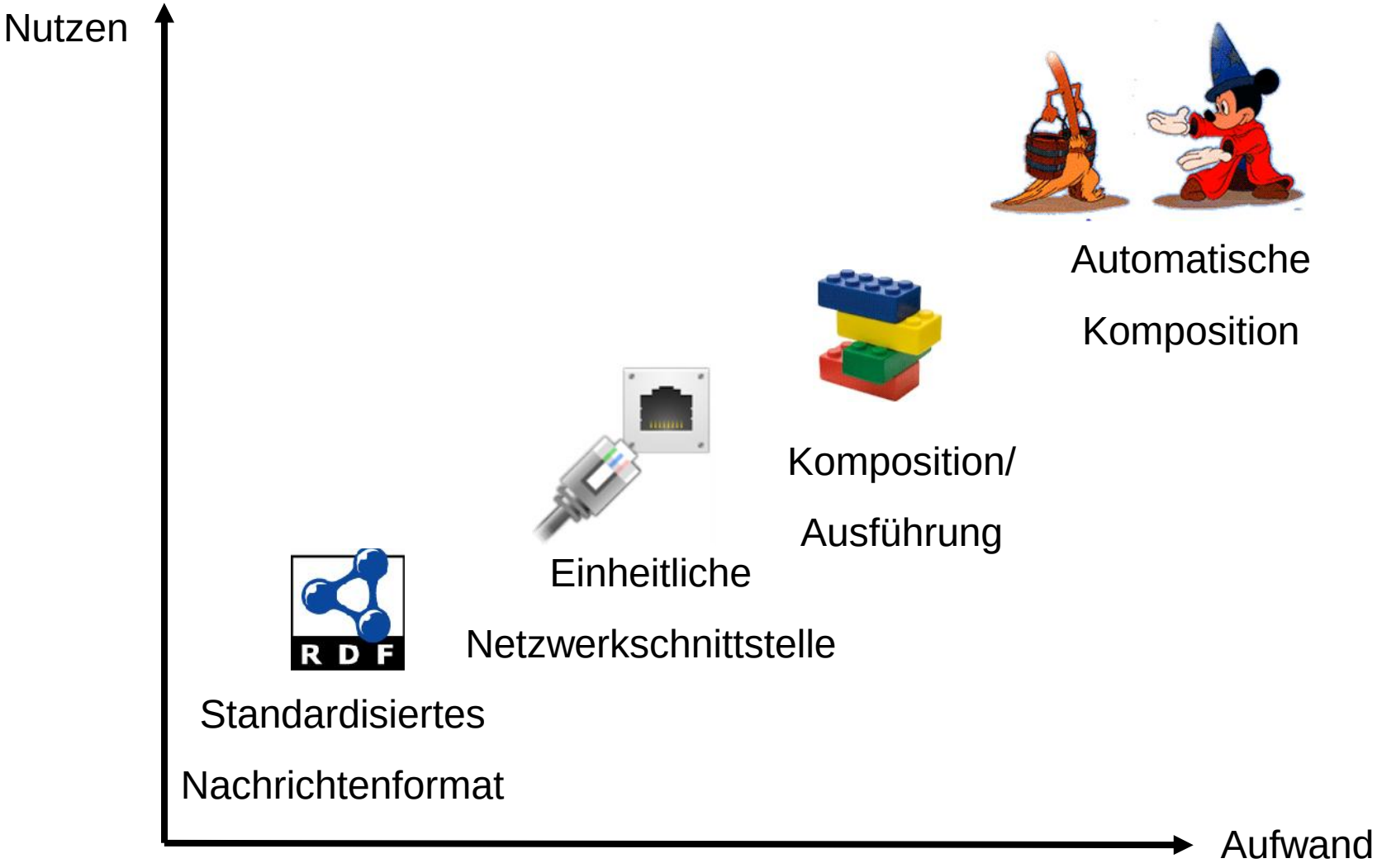
Ausgangslage

- Ziel von ARVIDA: Aufbrechen bestehender monolithischer VT-Systeme, damit Komponenten einfach wiederverwendet werden können



- Strategie ARVIDA Referenzarchitektur:
  - Komponenten mit standardisierter Datenmodellierung und einheitlichen Netzwerk-Schnittstellen
  - Erstellung von Anwendungen („Anwendungslogik“) durch Zusammenschalten der Komponenten
    - Mittels regulärer Programmiersprachen
    - Mittels Kompositionssprache

Grobe Aufwand-/Nutzenabschätzung



Gliederung des Vortrags

Nicht behandelt im Vortrag:

- Standardisiertes Nachrichtenformat
  - hat in ARVIDA gut funktioniert (siehe auch andere Präsentationen)
  - Modellierung in RDFS, OWL und SHACL
  - Festlegung von Content-Types für Binärformate
- Automatische Komposition
  - haben wir im Antrag ausgeschlossen, aktuell zu ambitioniert für industriellen Einsatz

Weitere Themen des Vortrags:

- Einheitliche Netzwerkschnittstelle (Request/Response-Muster für Kommunikation) und
- Komposition (regelbasierte ausführbare Spezifikation der Anwendungslogik)

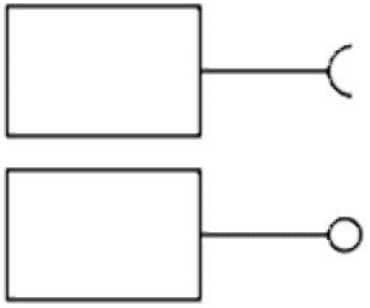
Einheitliche Netzwerkschnittstelle

Komponenten A und B, Datenfluss von A nach B

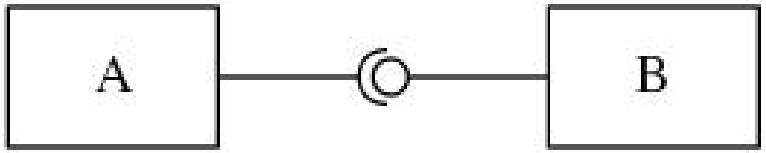
Quelle/Produzent Senke/Konsument

REST schreibt Request/Response Kommunikationsmuster zwischen Client und Server vor

- Clients emittieren HTTP Requests
- Server antworten auf eingehende HTTP Requests



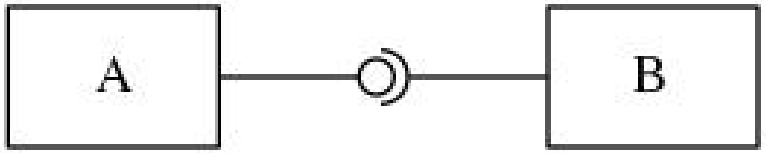
Einheitliche Netzwerkschnittstelle



A ist Client, B ist Server

A emittiert PUT Request

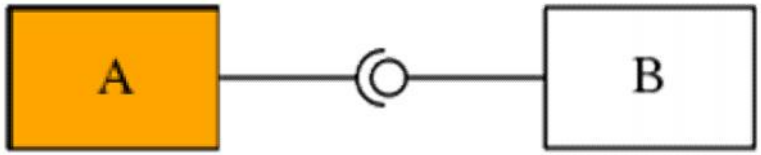
```
B.put(wert)
```



A ist Server, B ist Client

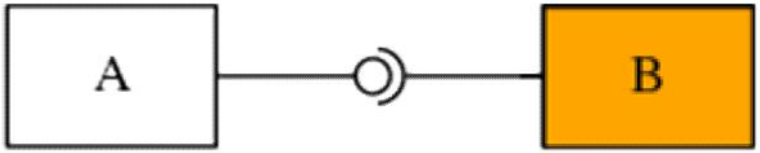
B emittiert GET Request

```
wert = A.get()
```



Schleife bei A (PUT zum Schreiben):

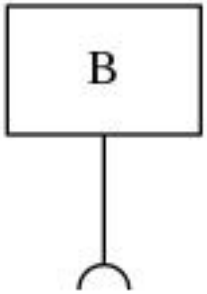
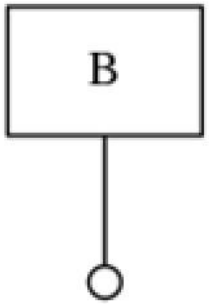
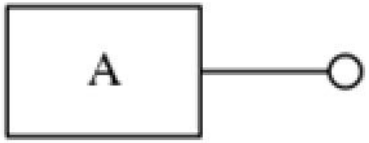
```
while true:
  B.put(wert)
```



Schleife bei B (GET zum Lesen):

```
while true:
  wert = A.get()
```

„Architectural Mismatch“

Konsument/ Senke / Produzent/ Quelle	 B ist Client	 B ist Server
 A ist Client		Streaming
 A ist Server	Polling	

D. Garlan, R. Allen, and J. Ockerbloom, “Architectural Mismatch: Why Reuse Is So Hard,” IEEE Software, Nov./Dec. 1995, pp. 17–26. 2.

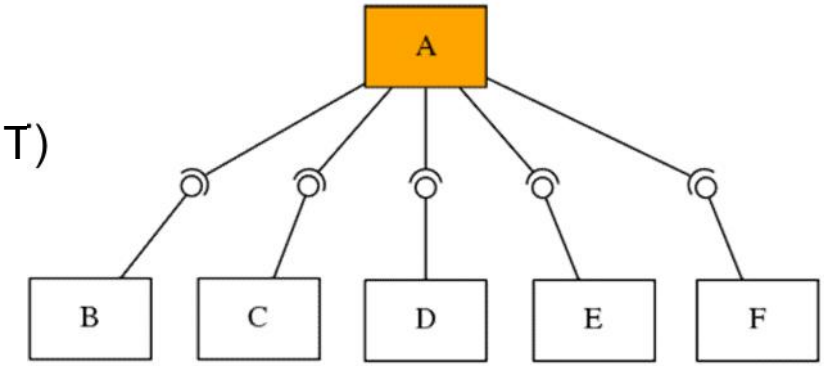
Kompositionssprache für Anwendungslogik

Primitive für Datenzugriff (GET) und -manipulation (PUT)

Wenn-Dann Regeln auf Weltzustand in RDF

Regelmäßige Ausführung (Schleife), 1 Client

1 Client-Komponente (1 Schleife)



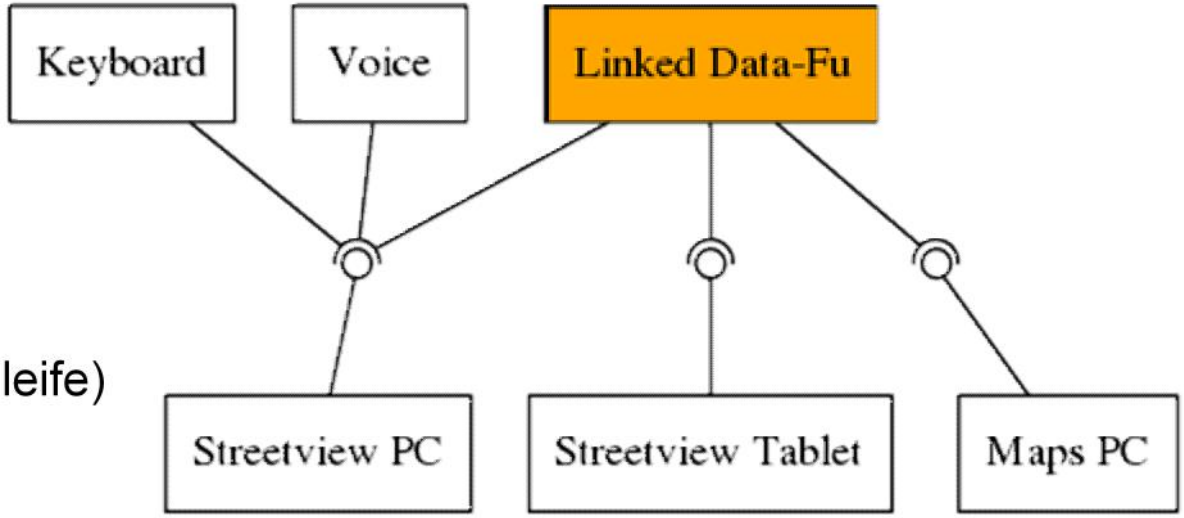
n Server-Komponenten

```
while true:
        GET B, C, D, E, F
        Wenn Bedingung Dann PUT F { Wert }
```

Vorteil: Einfache Serverschnittstelle zu Diensten, einfache Spezifikation der Anwendung

Nachteil: Polling kostet Ressourcen, mögliche Latenz bei Nutzereingaben

Demonstrator



Clients:

- Tastatur (Nutzereingabe)
- Sprache (Nutzereingabe)
- Linked Data-Fu (Koordination, Schleife)

Servers:

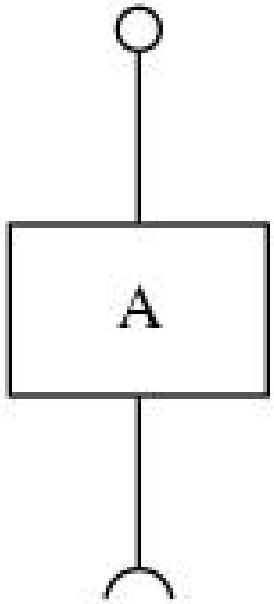
- PC mit Streetview (Visualisierung)
- Tablet mit Streetview (Visualisierung)
- PC mit Maps (Visualisierung)

Demonstration der deklarativen Spezifikation von Anwendungslogik.

Komponenten sind über das Internet miteinander verbunden und werden mittels Linked Data-Fu koordiniert.

Smart Components

- Je nach Anwendungsszenario ist pull bzw. push besser
- Zentrale Komponente ist nicht immer erwünscht
- Anforderungen eines konkreten Szenarios an eine Komponente ist zur Entwicklungszeit nicht bekannt



- Wie kann das Verhalten einer Komponente nach der Entwicklungszeit (zur Deployzeit/zur Laufzeit) festgelegt werden?
- Kernfunktionalität bleibt, Entscheidung der Festlegung der Interaktion, Datentransformation (Vokabulare), einfache Berechnungen und Entscheidungen, Delegation
- Komponenten sind sowohl Clients als auch Server
- Abhängig von der Mächtigkeit der zugrundeliegenden Logik

Zusammenfassung und Fazit

- Gemeinsames Datenformat (RDF bzw. Binärdaten) ist Konsens
- Abwägung zwischen Client- vs. Server-Konnektoren (pull/push) in Komponenten
- Kompositionssprache und Ausführungsumgebung (mit Client-Konnektor) verfügbar
  - Fokus ist auf Ausführung der Spezifikationen
  - Einsatz in ersten industriellen Demonstratoren

- Publikationen zum Thema Komposition als Client- und Server-Konnektoren
- Breite Einsatzmöglichkeiten

- Mögliche zukünftige Arbeiten betreffen
  - Beschreibung von komplexem Verhalten (Automatisierung)
  - Verifikation von Regelprogrammen
  - Automatische Komposition