Analyse abstrakter Architekturmodelle in verteilten Systemen

Tom Schöner

Informatik Master, HAW Hamburg

Gliederung

1. Motivation
2. Einleitung
   - Verteilte Systeme
   - Softwarearchitektur
3. Modelle
4. Paper: *Building Dynamic, Long-Running Systems*
5. Ausblick für das Masterstudium
6. Konferenzen
7. Referenzen

Motivation

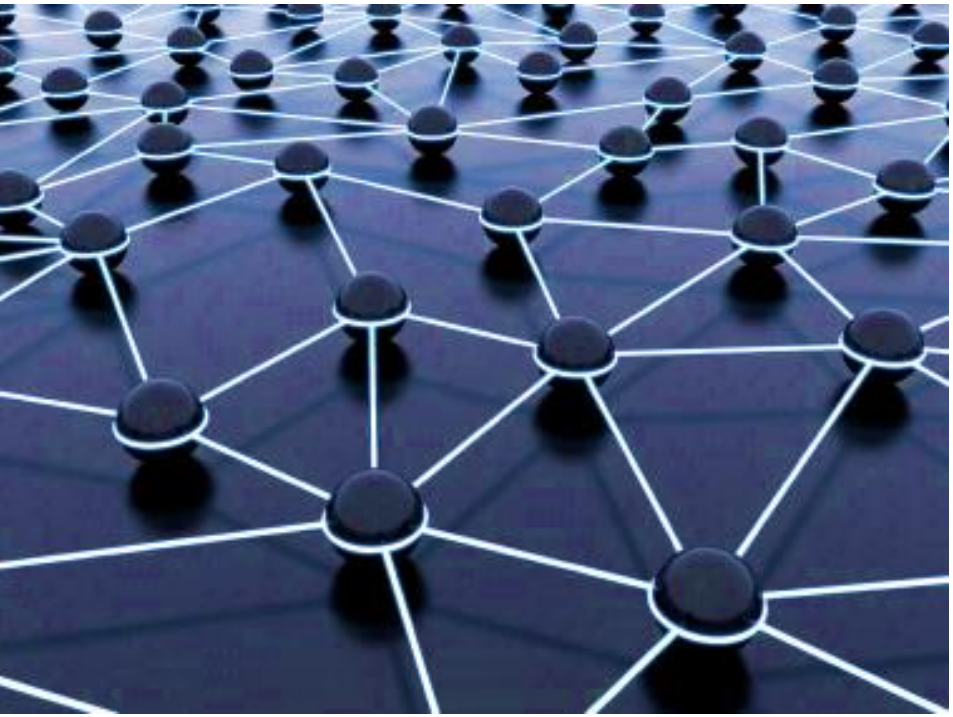


Abb 1: Netz

Interessen

- *Container orchestration*
- *Scalable applications*
- *High-level* Interaktion zwischen Komponenten
- Dezentrale/ Verteilte Systemarchitektur
- …

Motivation

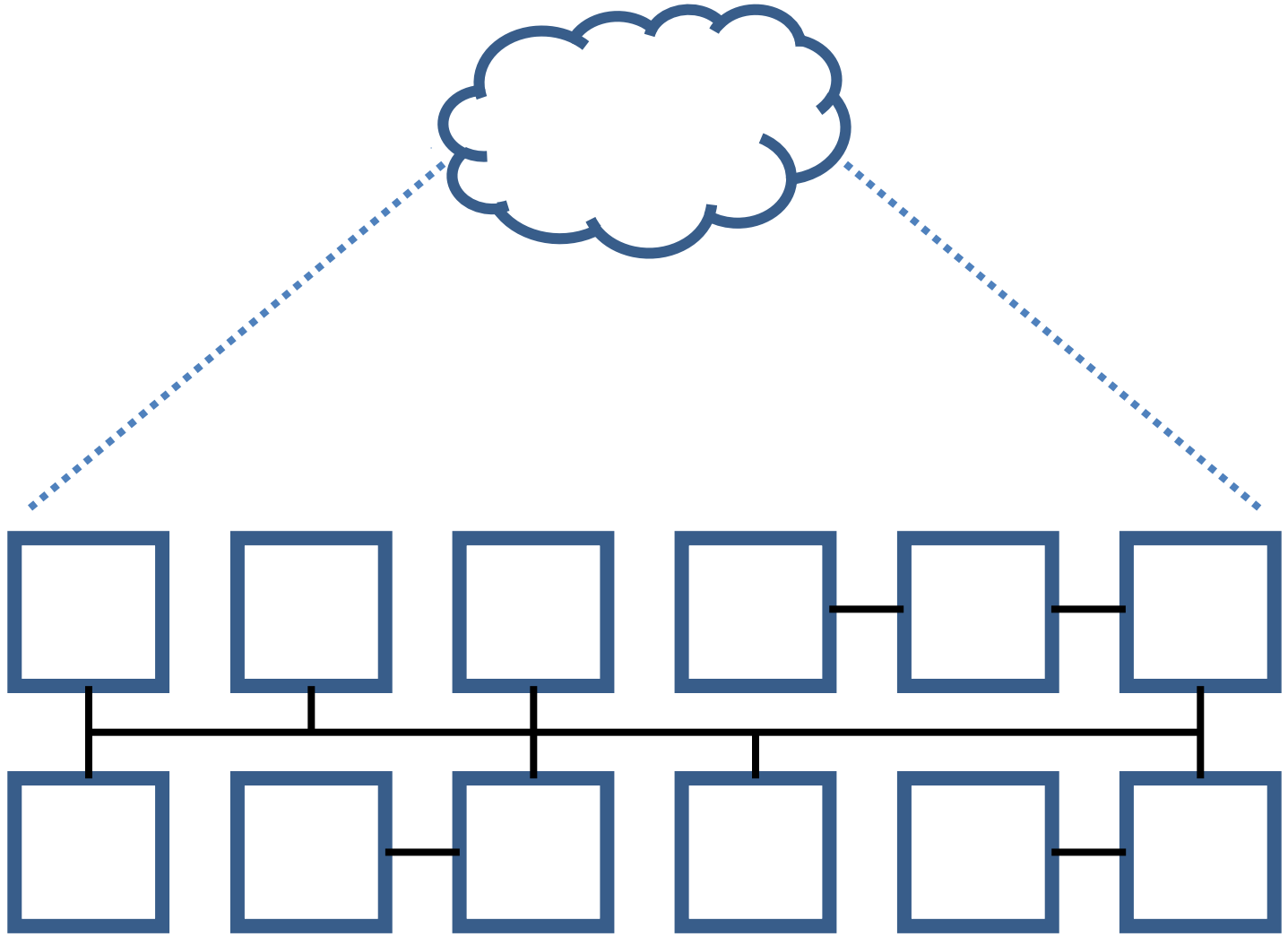


Fig 1: Abstrahierte „Cloud“

Verteilte Systeme

- Unterscheidung zwischen Low- und High-Level
  → Verteilte Algorithmen
- **Typen:** Client-Server, Cluster, Grid, peer-to-peer, …
- **Vorteile:**
  - Skalierbarkeit (Performanz)
  - Erreichbarkeit
- **Nachteile:**
  - Komplexität
  - Latenz

Softwarearchitektur

„Die Software-Architektur ist die grundlegende Organisation eines Systems, dargestellt durch dessen **Komponenten**, deren **Beziehungen zueinander und zur Umgebung** sowie die Prinzipien, die den **Entwurf und die Evolution** des Systems bestimmen.“ [Ralf Reussner, Willhelm Hasselbring - Handbuch der Softwarearchitektur, Seite 1 (Abgeleitet vom IEEE-Standard 1471-2000, ISO06)]

→ Im Allgemeinen steigt die Komplexität eines System über seinen Entwicklungszeitraum an und erschwert die Wartbarkeit

- Projektbasis
- Problemorientiert

- Spätere Änderungen sind aufwendig
- Analyse komplexer Architekturen nicht mehr trivial

Software- / Systemarchitektur

- Abstraktionsmodell
- Beschrieben durch die Kriterien der Softwarequalität (Performance, Wartbarkeit, …)
- Früher Entscheidungsprozess
- Architekturbeschreibungssprachen (ADLs), zB xADL
  http://isr.uci.edu/projects/xarchuci

Architekturmodelle

Loose coupling

Datenabstraktion

Stateless components

Stateful components

Compliant Data Replication

Tenant-isolated component

...

Loose coupling

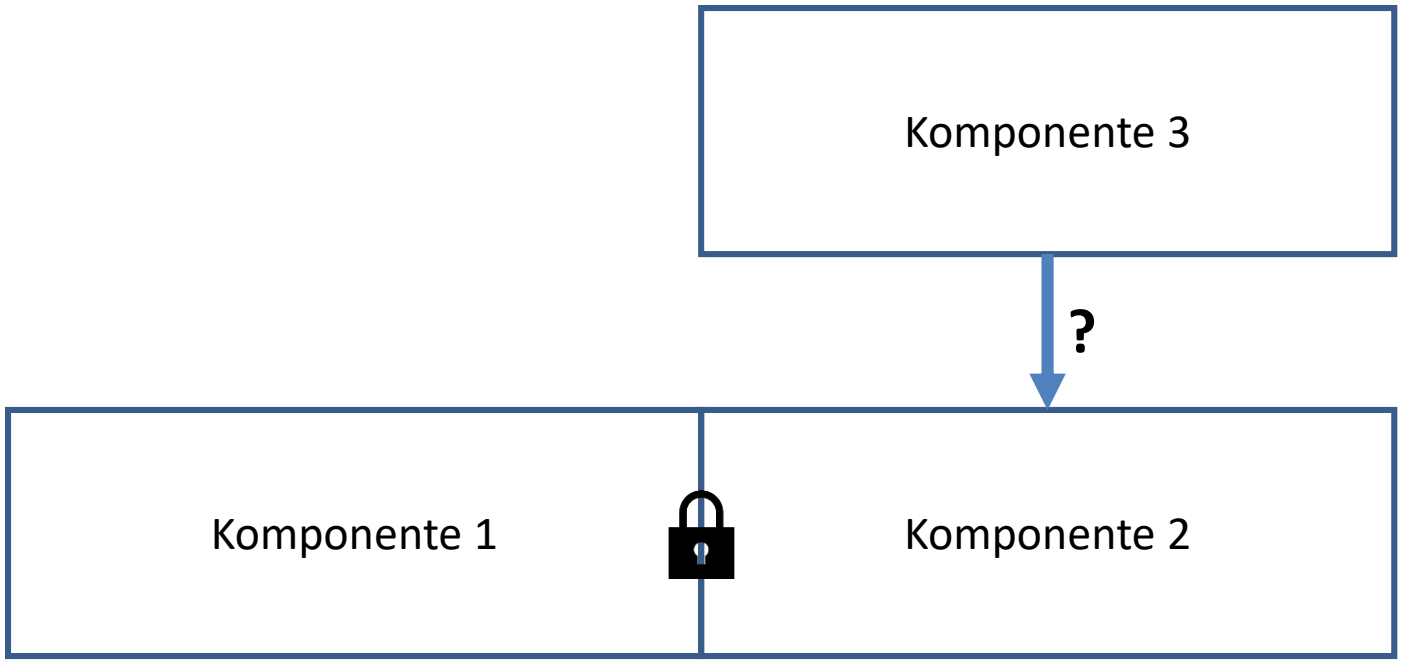


Fig 2: streng gekoppelte Komponenten

Loose coupling

- **Platform** Atonomy
- **Reference** Atonomy
- **Time** Atonomy
- **Format** Atonomy

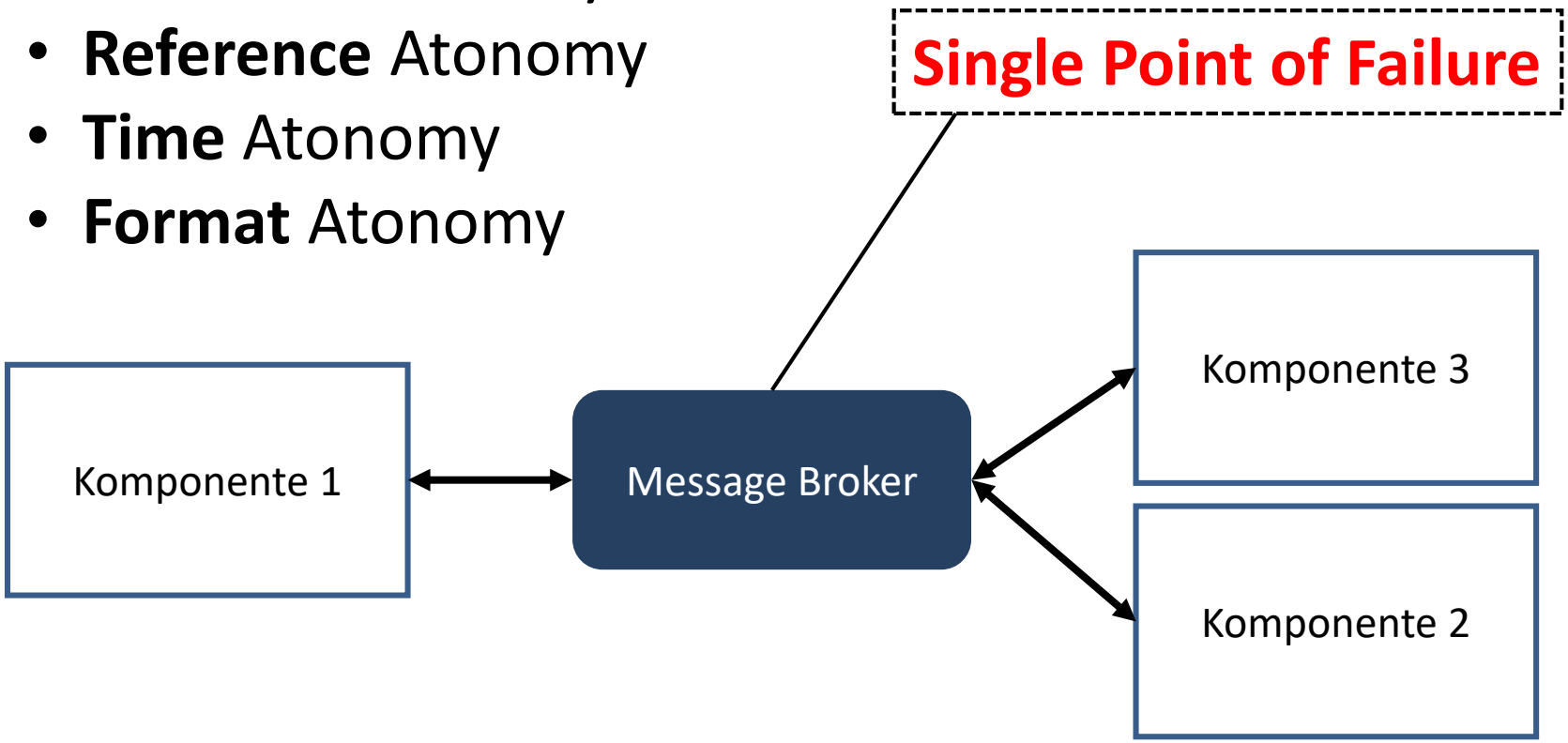


Fig 3: aufgelöste Abhängigkeiten der Komponenten

Loose coupling – CPS

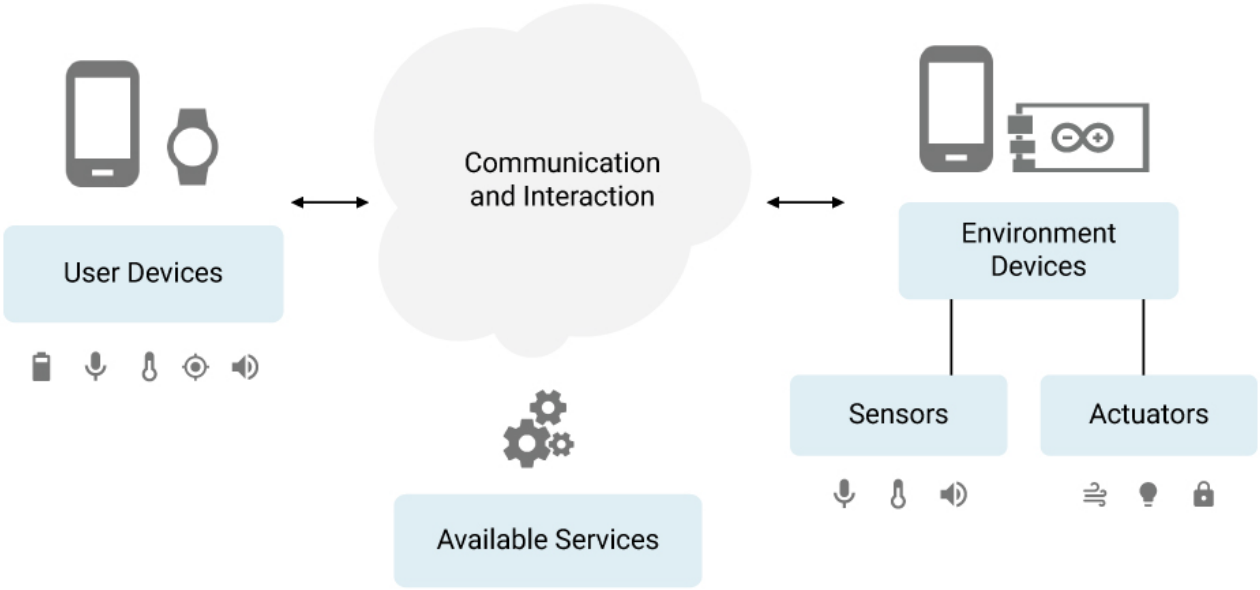


Abb. 2: Towards a Component Infrastructure for Cyber-Physical Systems – Überblick eines CPS

Loose coupling – CPS

> *"A lightweight and interoperable mechanism to specify sensors and actuators, and an* ***uncoupled communication and coordination mechanism for cyber-physical entities.****"* [Towards a Component Infrastructure for Cyber-Physical Systems, Seite 1]

- Technologie unabhängige Schnittstellen
  - Sensoren <-> Applikationslogik
- Dynamische Umgebung
- Modular und Erweiterbar
- Kommunikation durch einen *distributed tuple space*

Loose coupling – CPS *Tuple space*

- **Tuple**: { *key, arg1, [arg2, …]* }
- **Space**: Ansammlung von Tuples, unterstützt:
  - **write**
  - **read**
  - **take**
  - scan, count, waitToTake, waitToRead
- Unterstützt parallele Aufrufe
- Emuliert gemeinsamen Speicher

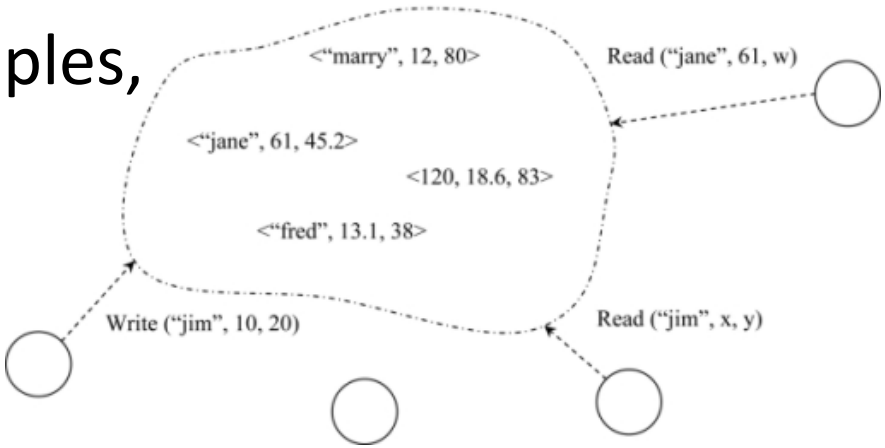


Abb. 3: Tuple Space

Datenabstraktion

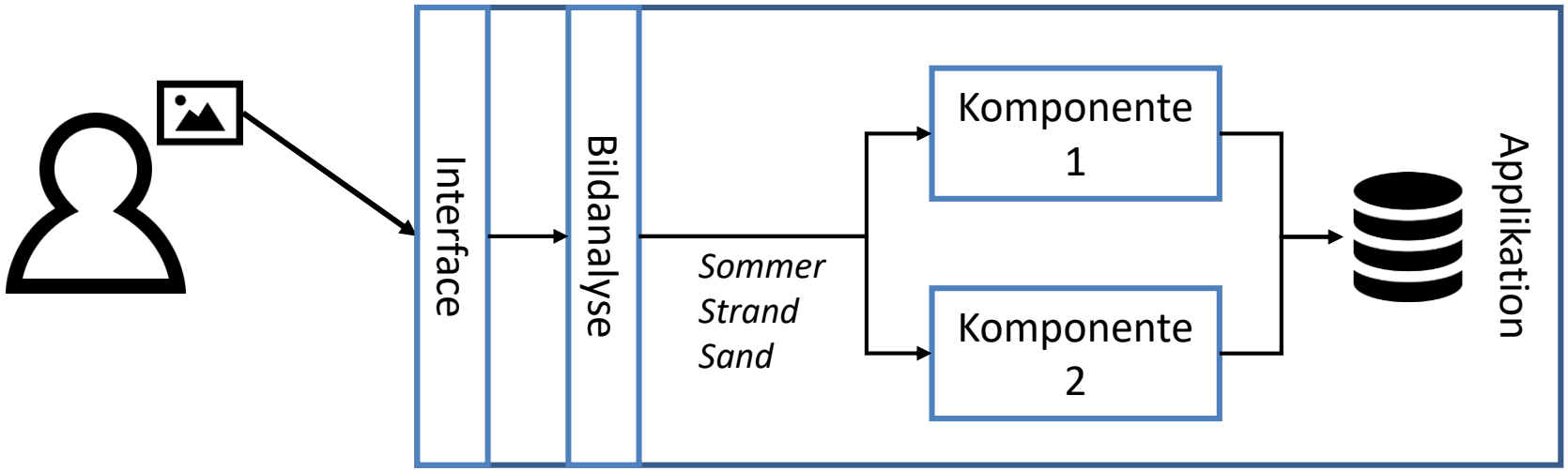


Fig 4: Datenabstraktionsmodell

- Datenrepräsentation den Anforderungen anpassen
- Approximierte Daten(mengen):
  - CAP Theorem: Hohe **C**onsistency und **A**vailability

Datenabstraktion

“It is impossible in the asynchronous network model to implement a read/write data object that guarantees the following properties:
- Availability, in all fair executions,
- Atomic consistency, in fair executions in which no messages are lost.”

[Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services, Seth Gilbert and Nancy Lynch]

2016 4th International Workshop on Software Engineering for Systems-of-Systems

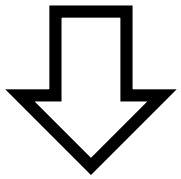
Building Dynamic, Long-Running Systems

Steven P. Reiss
Department of Computer Science
Brown University
Providence, RI. 02912 USA
spr@cs.brown.edu

Qi Xin
Department of Computer Science
Brown University
Providence, RI 02912 USA
qx5@cs.brown.edu

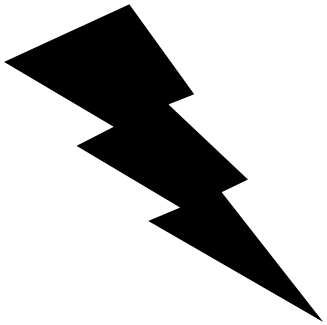
Building Dynamic, Long-Running Systems

- Ältere Systeme in **geschlossenen, spezialisierten und stabilen** Umgebungen *(zB Flugzeuge)*



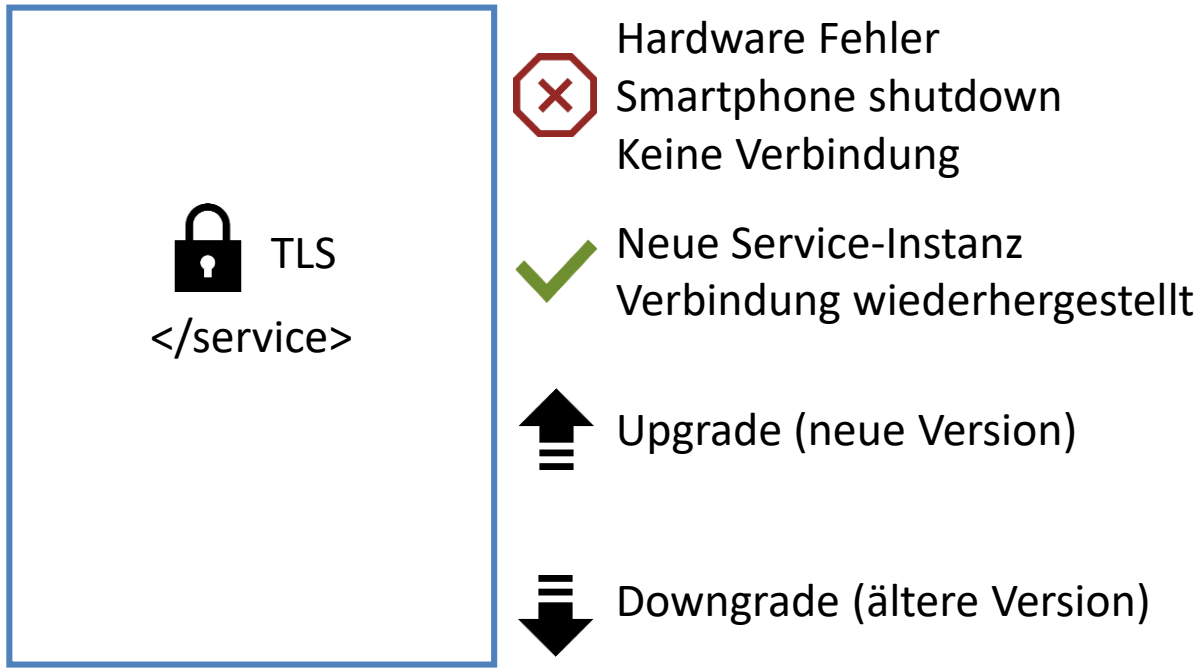
- Neuere Systeme in **offenen, dynamischen** Umgebungen *(zB Smartphones)*

Building Dynamic, Long-Running Systems



1. Aufwendige Entwicklung
2. Austausch oder Neustart
3. Unsichere Netzwerke
4. Instabile Netzwerke
5. Variierende Auslastung
6. Skalierung
7. Fehler-Recovery

Building Dynamic, Long-Running Systems



Building Dynamic, Long-Running Systems

- Hierarchisches Komponenten-basiertes Modell
- Komponenten sind Implementierungen der Interfaces:
  - Semantische Informationen
  - Interfaces benötigen systemabhängige oder kontextbezogene Metadaten (Sicherheit, Bewertungskriterien wie Performanz, …)

Building Dynamic, Long-Running Systems

```
dataface edu.brown.cs.taiga.location.GPSData {

    description {{ location and speed information }}

    location { double latitude, double longitude };
    double accuracy;
    double speed default 0;
    double heading;

    restricts {
        -90 <= latitude <= 90,
        -180 < longitude <= 180,
        0 <= speed < 10000,
        0 <= heading < 360
    }

    units {
        accuracy : meters;
        latitude : degrees;
        longitude : degrees;
        heading : degrees;
        speed: meters / second;
    }

    aggregations {
        location : within;
        timestamp : within;
        speed : within, mean, median, mode, variance;
    }

    filters {
        location { * }
        speed { $ <= speed < $ }
        timestamp { $ <= timestamp < $ }
    }

}   // end of dataface GPSData
```

Fig 5: Dataface Implementierung

Ausblick für das Masterstudium

Grundseminar	➢ Genauere Themenfindung ➢ Erforschung des Themenraums
Hauptseminar	➢ Thema vertiefen ➢ Beispielapplikation entwickeln
Masterarbeit	➢ *Offen (Themenabhängig)*

Konferenzen

- **ICDCS 2017**: **I**nternational **C**onference on **D**istributed **C**omputing **S**ystems http://icdcs2017.gatech.edu/
  05. – 08. Juni 2017 in Atlanta, USA
  - Distributed Big Data Systems & Analytics
  - Distributed Algorithms and Theory
  - Distributed Operating Systems & Middleware
  - Internet of Things & Cyber-Physical Systems
  - …

- **DevOpsCon:** The Conference for Continuous Delivery, Microservices, Docker, Clouds and Lean Business https://devopsconference.de/
  12. – 15. Juni 2017 in Berlin
  - Deploying and scaling containerized Microservices with Docker and Swarm
  - Machine Learning for DevOps using Python
  - Application Load Testing with Open Source and the Cloud
  - …

Referenzen

- http://www.cloudcomputingpatterns.org *(Aufruf: 12.05.2017)*
- https://www.techopedia.com/definition/28553/tuple-space *(Aufruf: 12.05.2017)*
- http://isr.uci.edu/projects/xarchuci *(Aufruf: 10.05.2017)*

- http://icdcs2017.gatech.edu/ (Aufruf 01.05.2017)
- https://devopsconference.de/ (Aufruf 01.05.2017)

Literaturangaben (Papers)

- Marcio E. F. Maia, Rossana M. C. Andrade, and Windson Viana. 2016. **Towards a component infrastructure for cyber-physical systems**. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing* (SAC '16). ACM, New York, NY, USA, 626-628. DOI: https://doi.org/10.1145/2851613.2851934
- Steven P. Reiss and Qi Xin. 2016. **Building dynamic, long-running systems**. In *Proceedings of the 4th International Workshop on Software Engineering for Systems-of-Systems* (SESoS '16). ACM, New York, NY, USA, 19-24. DOI: http://dx.doi.org/10.1145/2897829.2897831

Literaturangaben (Papers)

- Ms.Kamble A.L, Ms.Shinde.T.K, Ms Kothiwale N, Khot.S.S 2013. **Real Time And Distributed Computing Systems.** Second International Conference on Emerging Trends in Engineering (SICETE) 53 | Page Dr.J.J.Magdum College of Engineering, Jaysingpur https://www.semanticscholar.org/paper/Real-Time-and-Distributed-Computing-Systems-I-intr-Kamble-Shinde/295037ee46ac281590f67435380cebc385ac9749

- Seth Gilbert and Nancy Lynch. 2002. **Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services**. *SIGACT News* 33, 2 (June 2002), 51-59. DOI: https://doi.org/10.1145/564585.564601

Literaturangaben

- Carola Lilienthal, **Langlebige Softwarearchitektur** – 1. Auflage, dpunkt Verlag (2016)
- Ralf Reussner, Wilhelm Hasselbring (Hrsg.), **Handbuch der Softwarearchitektur** – 2. Auflage, dpunkt Verlag (2009)

Bilderverzeichnis

- http://www.techferry.com/articles/si/img/distributed_artificial_intelligence.jpg - Abb. 1: Netz (*Aufruf 12.05.2017*)
- https://doi.org/10.1145/2851613.2851934 Abb. 2: CPS
- https://openi.nlm.nih.gov/imgs/512/123/3274288/PMC3274288_sensors-11-10343f5.png Abb. 3: Tuple Space (Aufruf 13.05.2017)