SDODR – CLOUD MODELS

Version 05.09.2018 – OIH Architecture Call

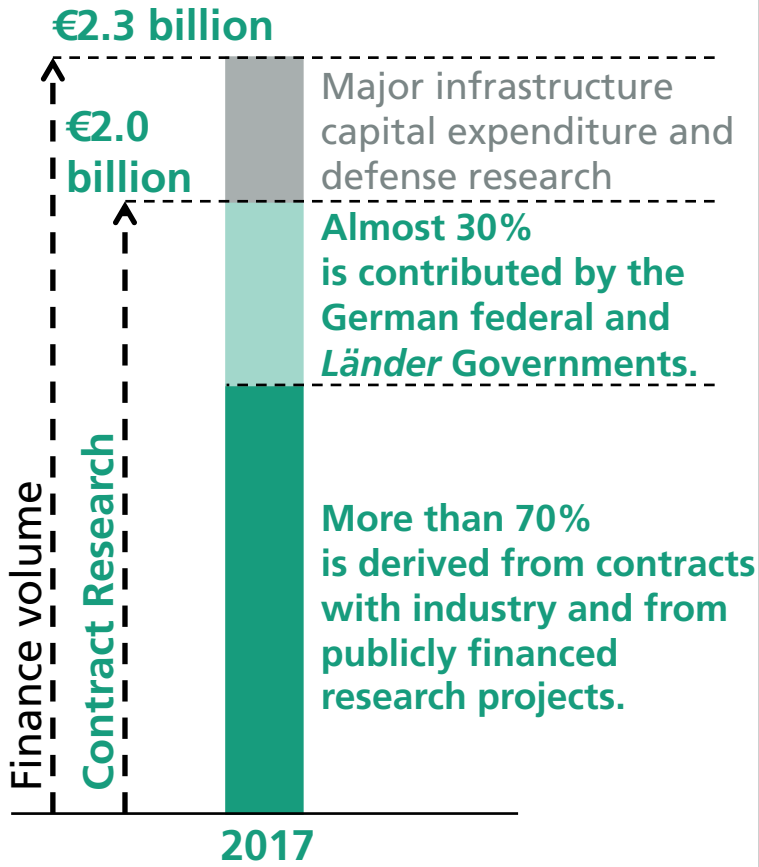
Susanne Braun

The Fraunhofer-Gesellschaft at a Glance

The Fraunhofer-Gesellschaft undertakes applied research of direct utility to private and public enterprise and of wide benefit to society.

25,527 staff

72 institutes and research units



Fraunhofer IESE

Fraunhofer IESE
The institute for software and systems engineering methods

- Founded in 1996, headquartered in Kaiserslautern
- Over 155 full-time equivalents (FTEs)
- Our solutions can be scaled flexibly and are suitable for companies of any size

- Our most important business areas:
  - Autonomous & Cyber-Physical Systems
  - Digital Services
  - Defense & eGovernment

Our Competencies – for Your Benefit

Fraunhofer IESE

Industry 4.0

Smart Farming

Smart Mobility

Smart Health

SMART Ecosystems

Smart Energy

Smart Rural Areas

Smart X

Project Roots

Industry Cooperation

Fraunhofer IESE

Cloud Models

Cloud Models

Based on
Domain Driven Design

Cloud Models
Replicated
Eventually Consistent
Built-in Conflict Management

Synchronization of High-Level
Service Operations

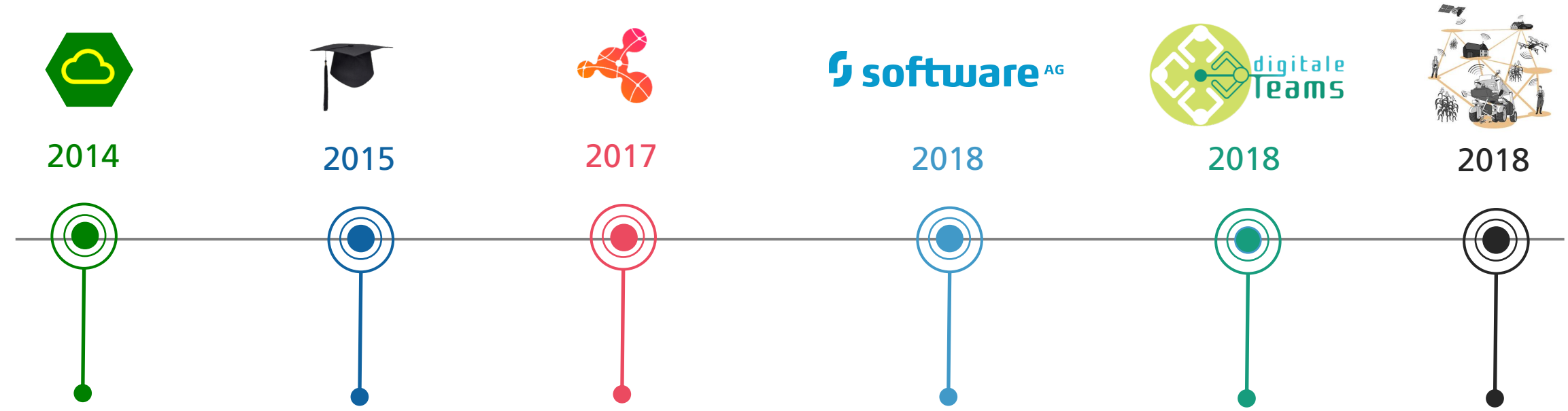
Cross-Platform
Model-Driven Approach

Multi-Level Transaction
Support

Fraunhofer IESE

Cloud Models Funding

SPONSORED BY THE
Federal Ministry of Education and Research
Supported by:
Federal Ministry for Economic Affairs and Energy
on the basis of a decision by the German Bundestag



Fraunhofer IESE

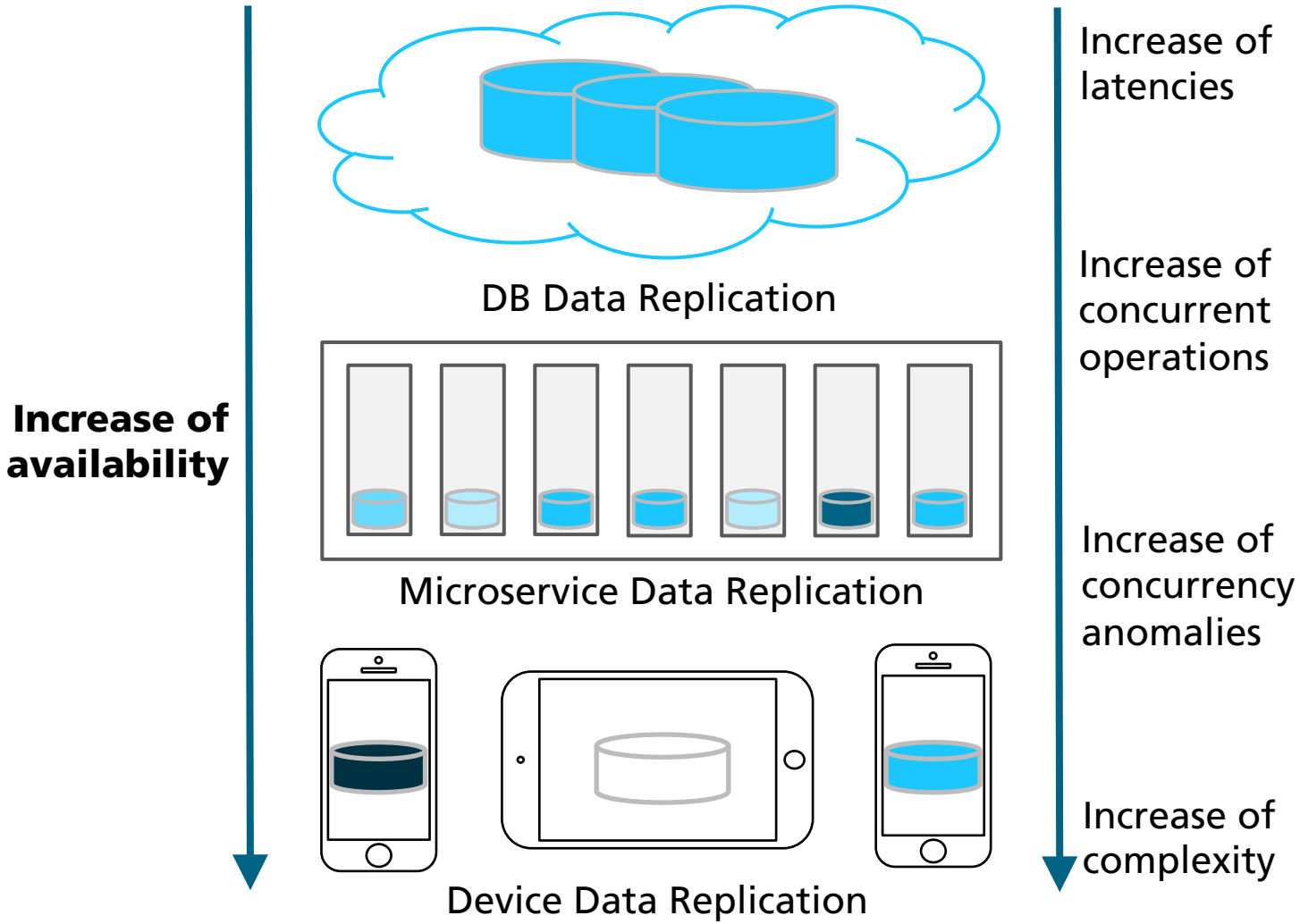
Key Requirements Replication in Ecosystems

- Offline capability of mobile apps
- Data synchronization of **core shared domain data** as platform service
- Support for standardized data models of **core shared domains**
  - No canonical data models

Key Requirements Replication in Ecosystems

- Modularity
- Horizontal Scalability
- Loose coupling of apps and services of the ecosystem
- High availability of apps and platform services, in particular data synchronization services
- Resilience of apps and platform services, in particular data synchronization services

Relevance for Modern Software Architectures



Goals for Cloud Models in Open Integration Hub

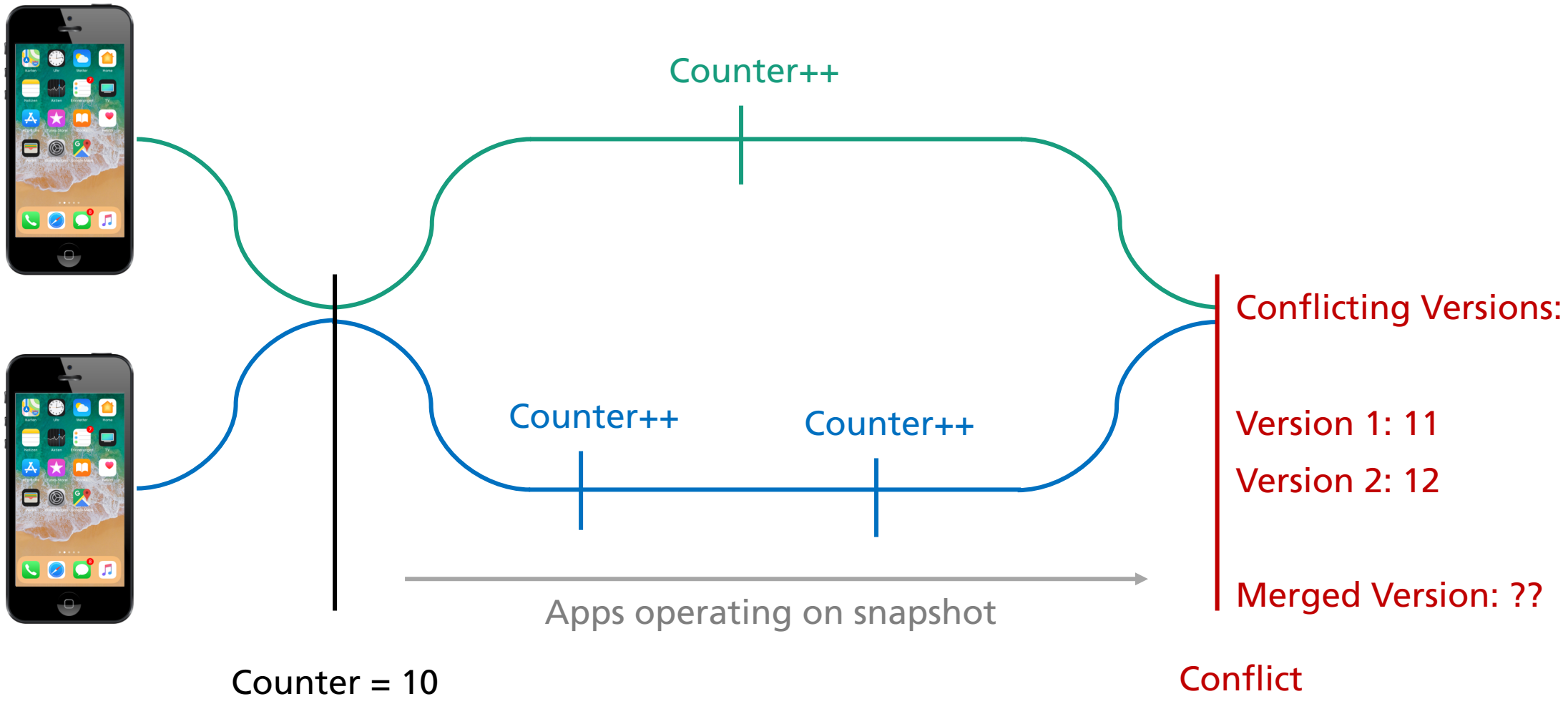
- Compare IESE approach with OIH approach
- Make small evaluation
- Publish results at scientific conferences
  - Wissenschaftliche Verwertung!

Problems

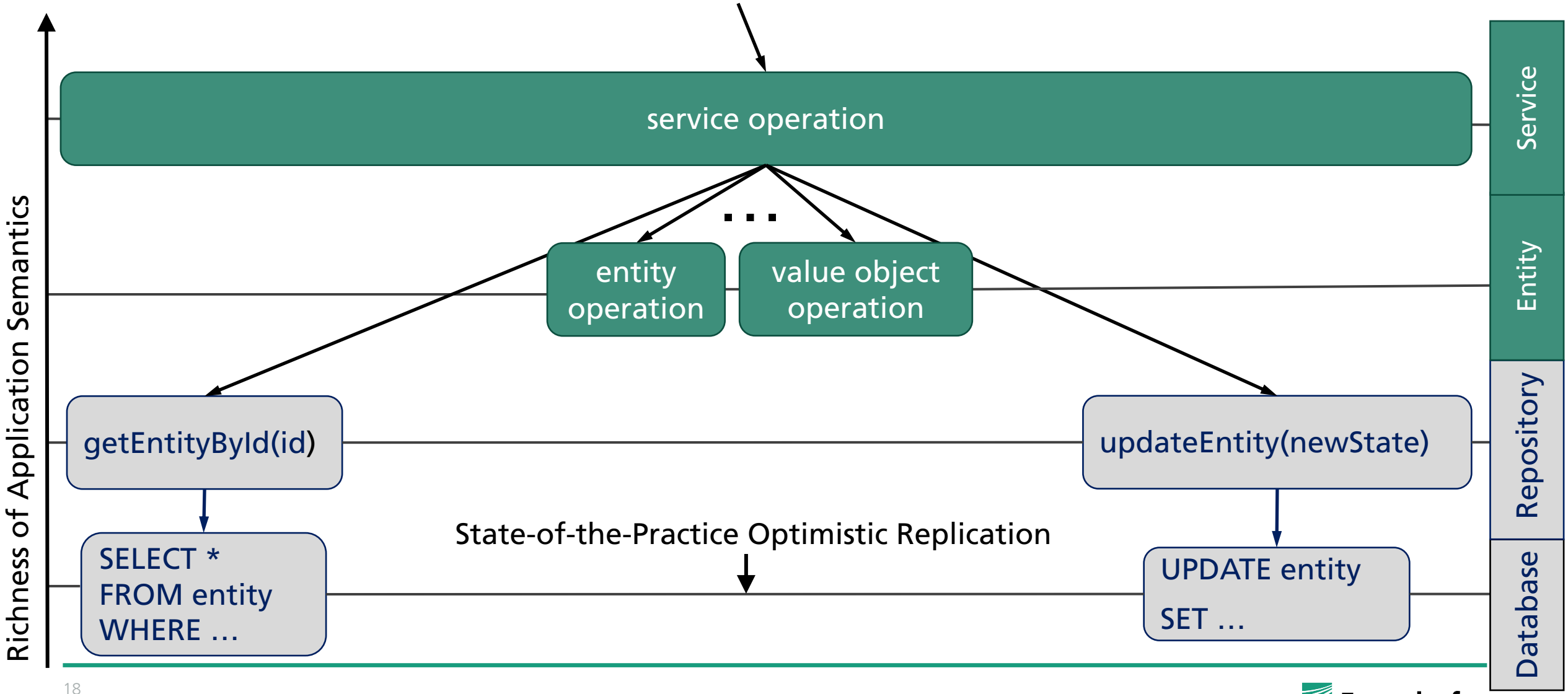
Eventual Consistency

- **Updates are propagated asynchronously (high latencies possible)**
  - Drivers: Availability, Scalability, Loose Coupling, Resilience
- Stale Data
- Conflicts
- **Merge of conflicting versions**
  - Unintuitive and error prone
  - Huge source of human error

Shared Counter Example



State-of-the-Practice Optimistic Replication



Fraunhofer IESE

Issues in Practice

- Conflicts that cannot be resolved correctly
  - ➡ Ultimately resulting in lost updates
- Conflicts on the semantic level that cannot be detected syntactically
  - ➡ Ultimately resulting in constraint violations
- Pseudo-conflicts detected syntactically that do not conflict on the semantic level but have to be reconciled
  - ➡ Thereby adversely impacting performance

Recap Concurrency Control in Relational DBs

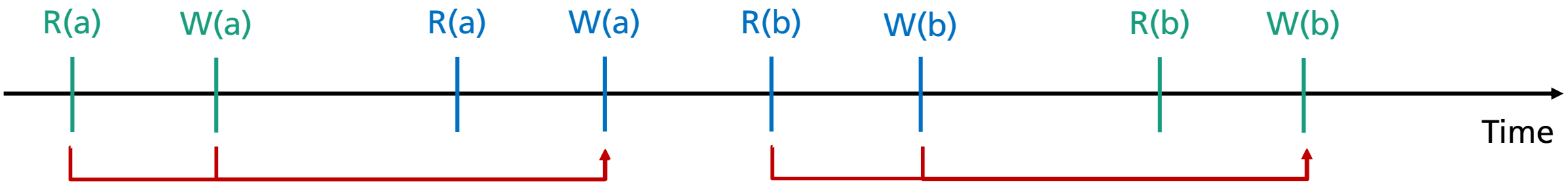
Transaction Interleaving Approach - Conflict Serializability

- Serial execution of transactions prevents occurrence of concurrency anomalies
- Conflict Serializability
  - R(x) is compatible with R(x)
  - R(x) is in **conflict** with W(x)
  - W(x) is in **conflict** with W(x)
- A schedule of concurrent transactions is conflict-serializable iff the conflict serialization graph is acyclic

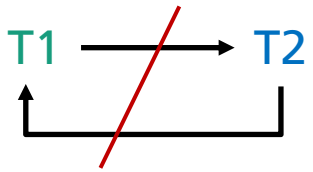
Example Conflict Serializability

- A schedule of concurrent transactions is conflict-serializable iff the conflict graph is acyclic and compatible with the execution order of the conflicting operations

Transactions T1, T2:



Conflict Graph:



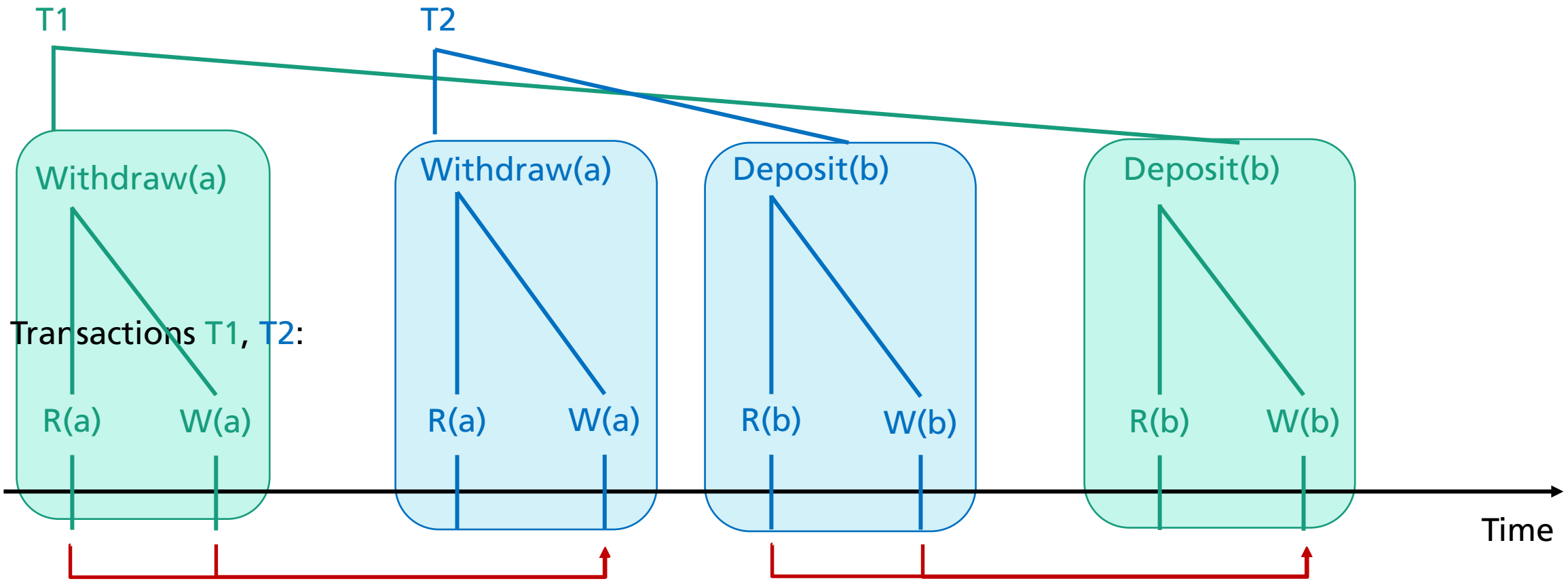
Conflict Serialization Graph is cyclic

⇢ No conflict serializability

⇢ Schedule would be rejected

Fraunhofer IESE

Banking Example



Fraunhofer IESE

Multilevel Transactions

- **Key Idea**: **Exploit the semantics of operations in level-specific conflict relations that reflect the commutativity or compatibility of operations to increase concurrency**

- Transactions are decomposed into operations and the operations again into sub-operations on multiple levels
  - Transactions, Business operations, Low-level read and write operations
- At each level a conflict relationship is defined
  - read-write conflicts and write-write conflicts on the same data item conflict at the lowest level
  - Non-commutative operations are conflicting on the level of business operations

- If at each level the conflict serialization graph is acyclic then the multilevel schedule is in total multilevel serializable

Banking Example

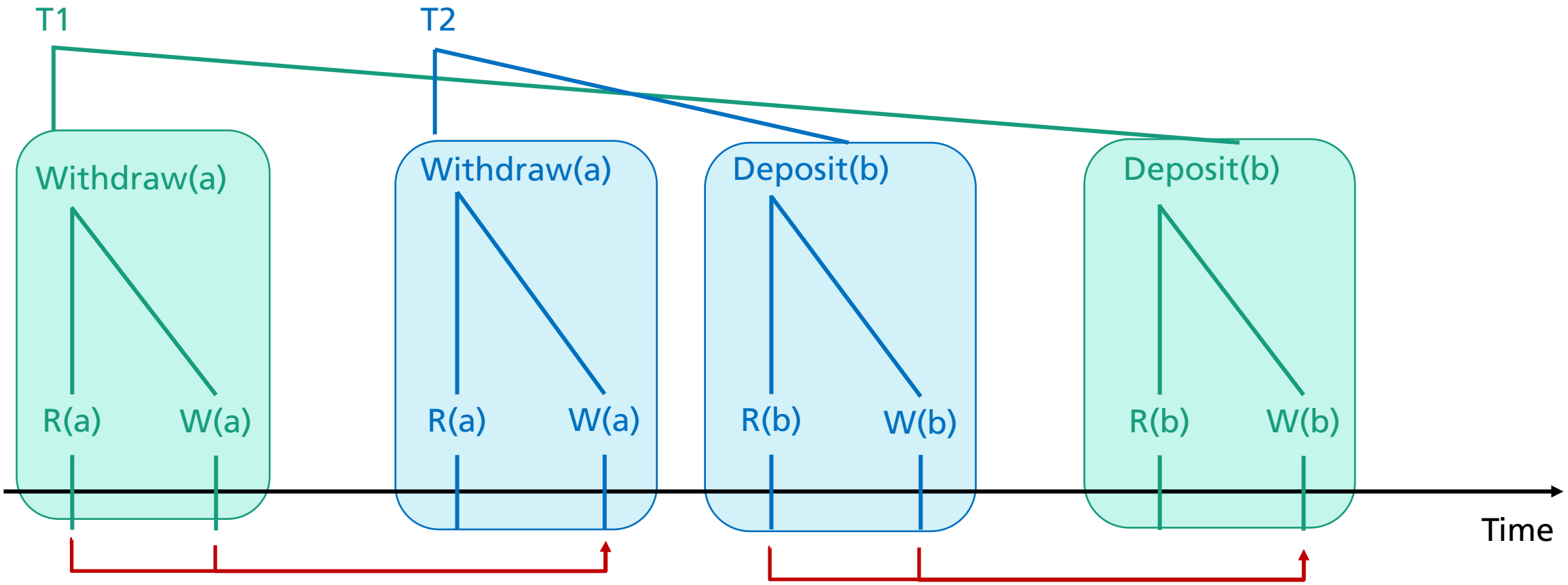
Conflict graph at level of business operations:

Withdraw(a) → Withdraw(a)

Deposit(b) → Deposit(b)

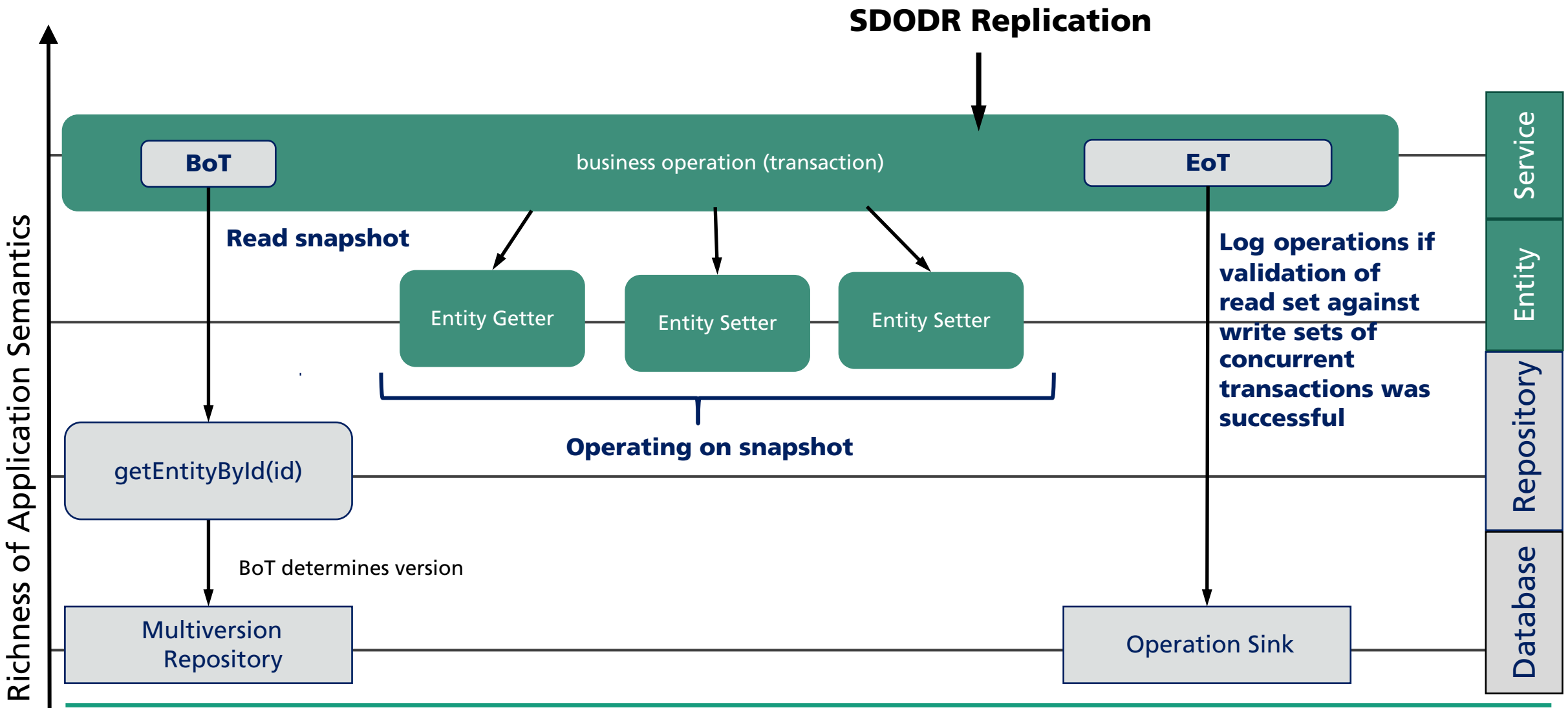
Conflict graph at level of transactions:

∅



Fraunhofer IESE

Level of SDODR Replication



Fraunhofer IESE

Idea

Exploit application semantics to increase number of operations that can run concurrently and conflict-free on different replication nodes

Semantics-Driven Optimistic Data Replication (SDODR)

Method

Design method based on **Domain Driven Design**

Best Practices and Design Patterns

Design for maximum compatibility of business operations

Technical Framework: Cloud Models

A cross-platform data replication framework with support for propagation and synchronization of high-level business operations and built-in conflict management

Operation approach with support for high-level business operations

Enables exploitation of semantics of high-level business operations for increased concurrency and minimization of conflicts and replication anomalies

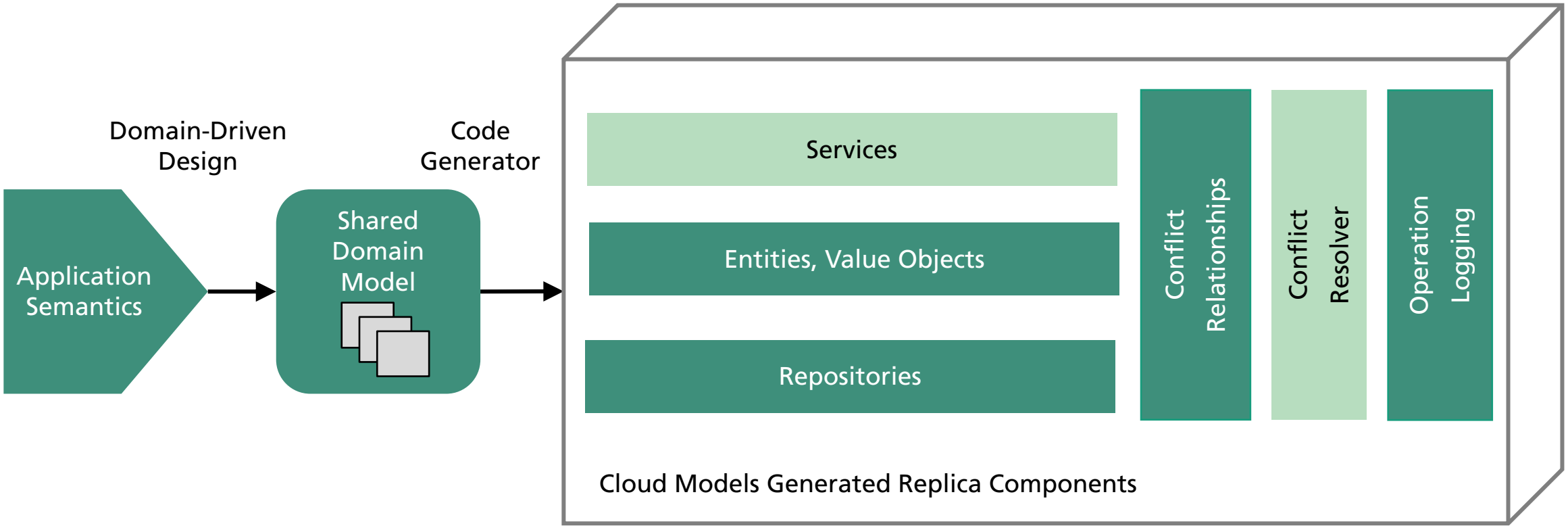
Flexible Conflict Management

- Support for management of conflict relationships of business operations
- Support for semantic reconciliation
- No lost updates

Cross-Platform Technology

Model-driven approach with code generation

How to provide a framework that can exploit application semantics for concurrency control?

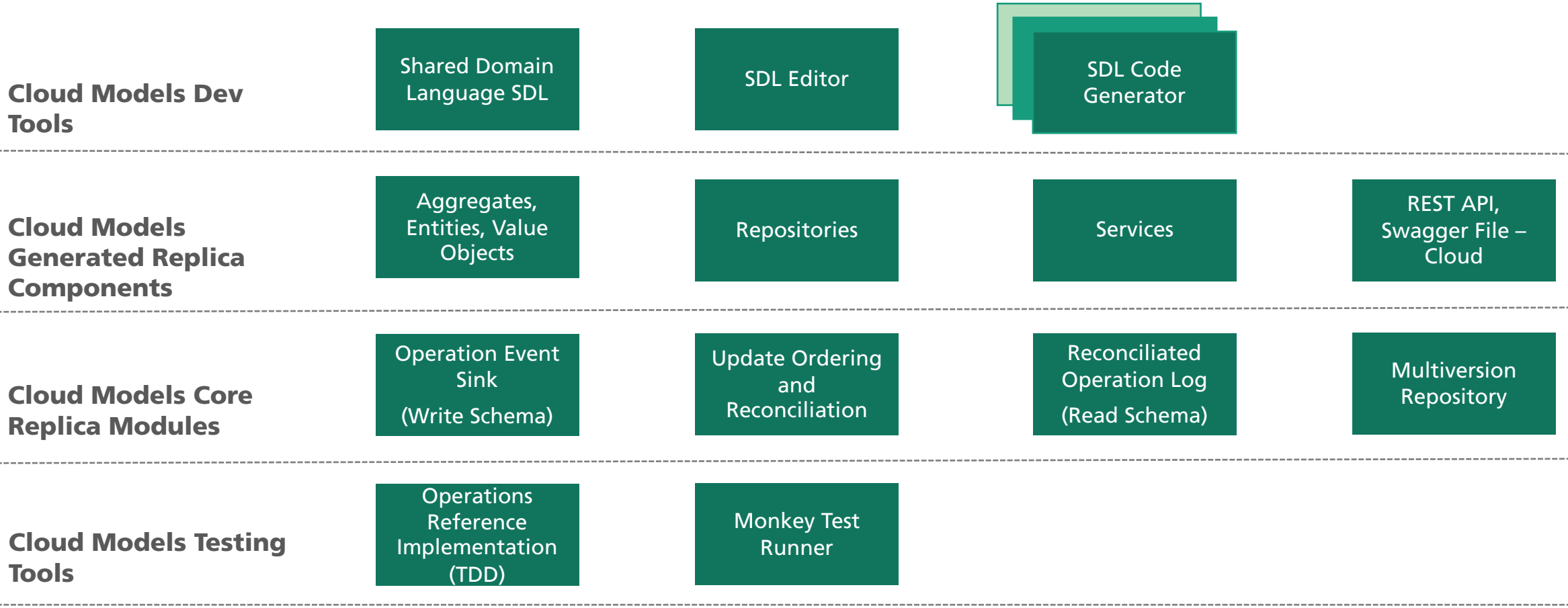


Shared Domain Model Example

```
Namespace de.fraunhofer.iese.oih
Shared Domain Contacts
Version 1.0

Entity Contact {
   person: Person
   organization: Organization
   phoneNumbers: List of PhoneNumber
}
Entity Person {
   firstName: String
   lastName: String
}
Entity Organization {
   name: String
   address: Address
}
Value Object Address {
   street: Street
   city: City
}
ValueObject PhoneNumber {
…
```

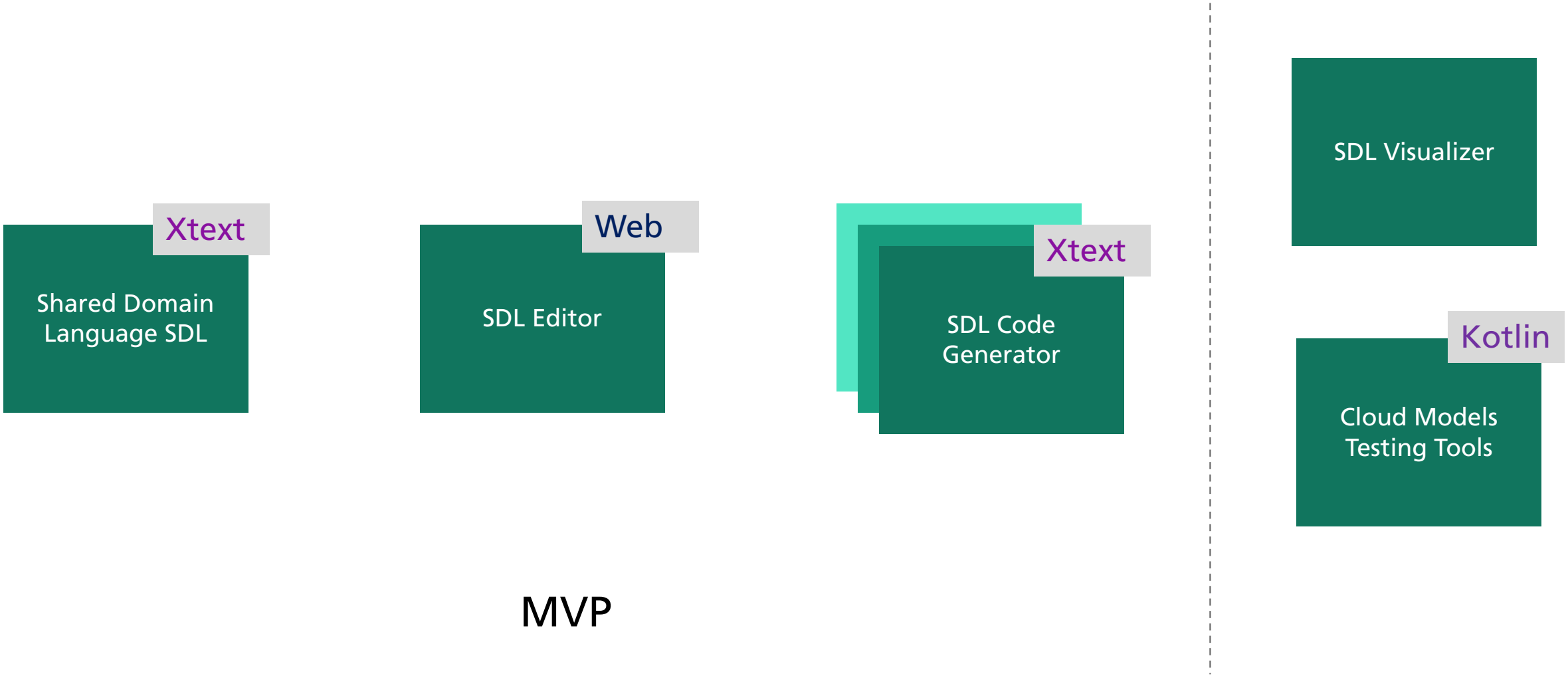
```
…

Serivce ContactsService {

   Operation deletePhoneNumber(Contact contact, Integer phoneNumberIndex): Contact
      conflicts with changePositionOfPhoneNumber
      resolve with resolveConflictOnListOfPhoneNumbers

   Operation changePositionOfPhoneNumber(Contact contact, Integer newPhoneNumberIndex): Contact
      conflicts with deletePhoneNumber
      resolve with resolveConflictOnListOfPhoneNumbers


    ConflictHandler resolveConflictOnListOfPhoneNumbers
}

…
```

Software Architecture – Major Building Blocks



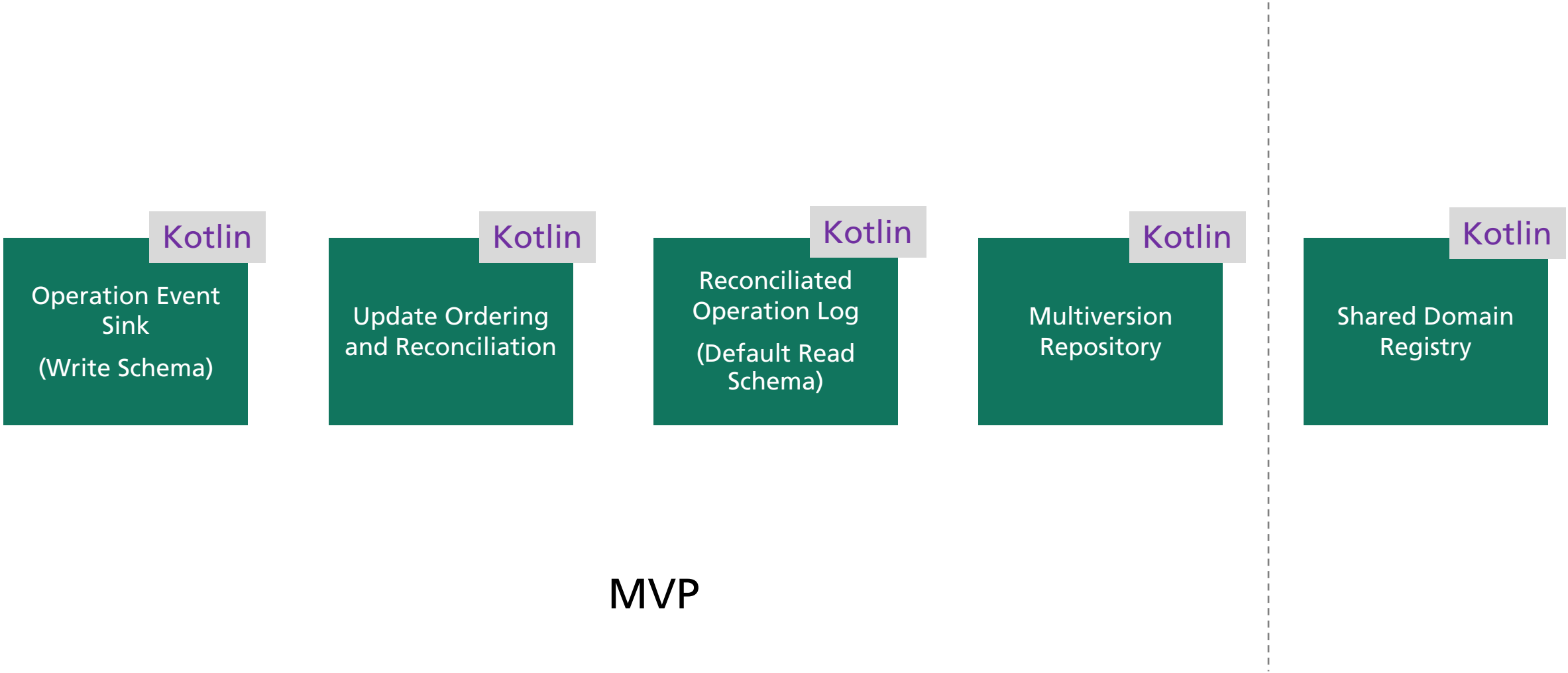
Fraunhofer IESE

Cloud Models Dev Tools



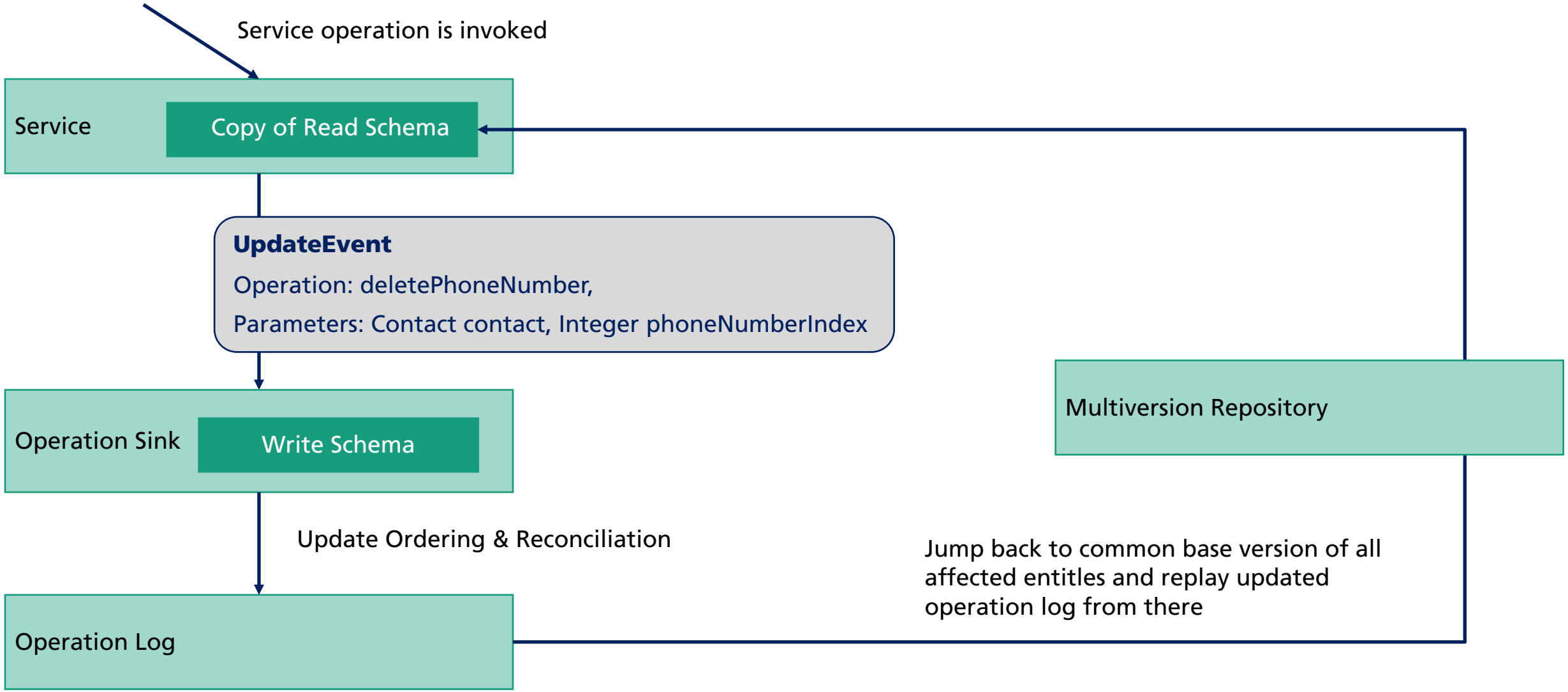
Fraunhofer IESE

Cloud Models Core Modules (Cloud and Local)



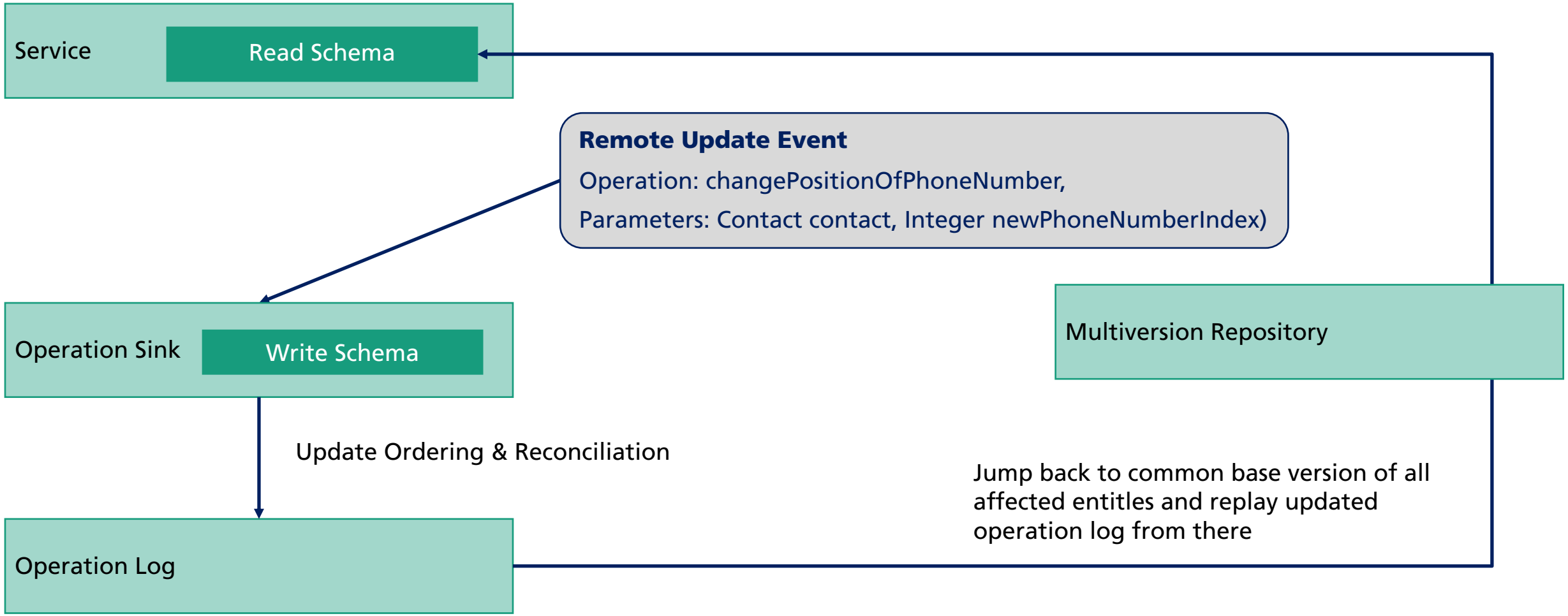
Fraunhofer IESE

Distributed CQRS With Local Update Event



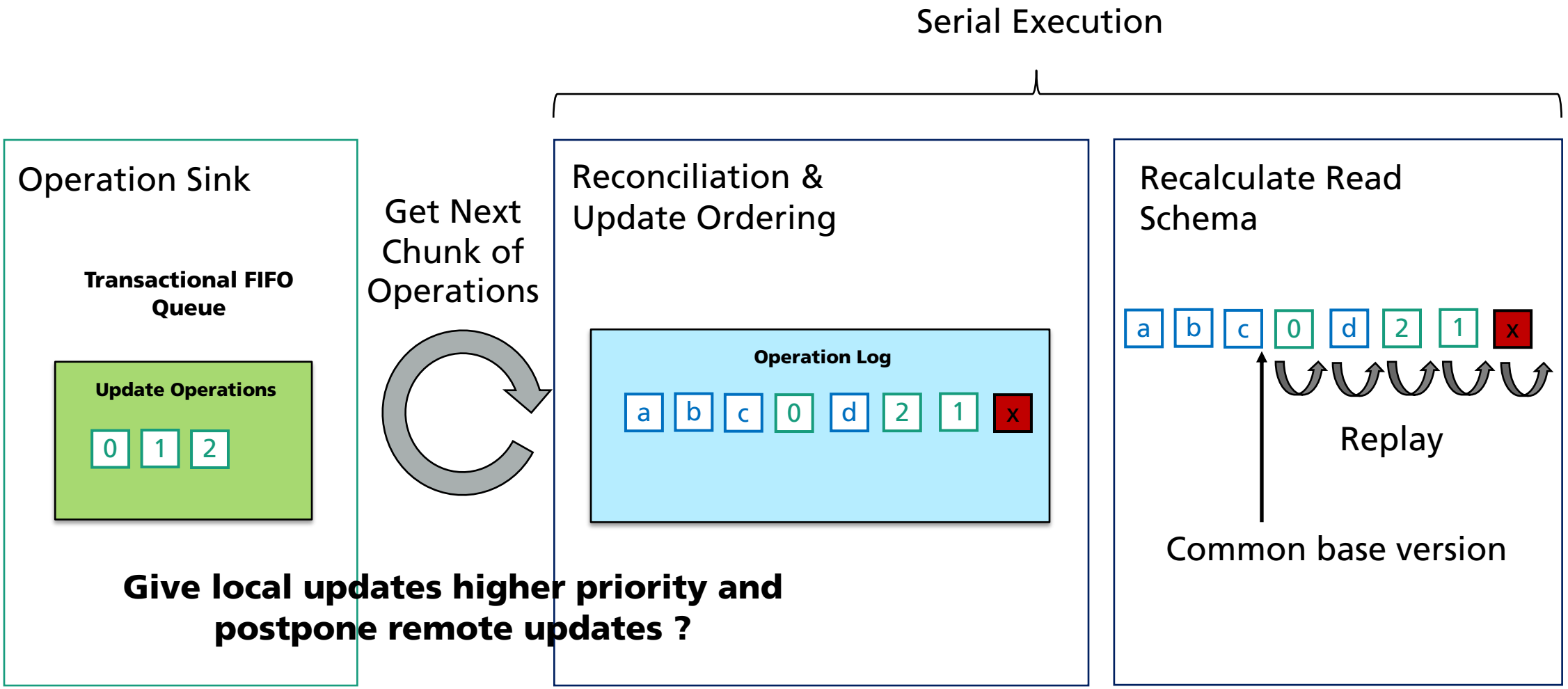
Fraunhofer IESE

Distributed CQRS with Remote Updates



Fraunhofer IESE

CQRS Performance Considerations (App / Microservice)



Fraunhofer IESE

Operation Event Sink and Global Operation Log in the Cloud

- Transactional Queue System (e.g. Kafka) shared by different services (that scales well horizontally)
  - One “operation event sink” queue **per** “Master Data Model” (**shared by all cloud replicas**)
  - Updates can be ”submitted” via generated REST API (similar to the one I proposed)
    - Example: POST to /contacts/update-events
  - Updates can be polled via the same generated REST API (similar to the one I proposed)
    - Example: GET /contacts/update-events
    - services maintain their own pointer in the transactional log (compare to how Linkedin does it, Kappa Architecture)
  - REST endpoints can easily be generated
  - Each service has its own “global operation log” queue
  - services / global operation logs being eventually consistent

Response of GET /contacts/update-events?snapshot=<causal history>

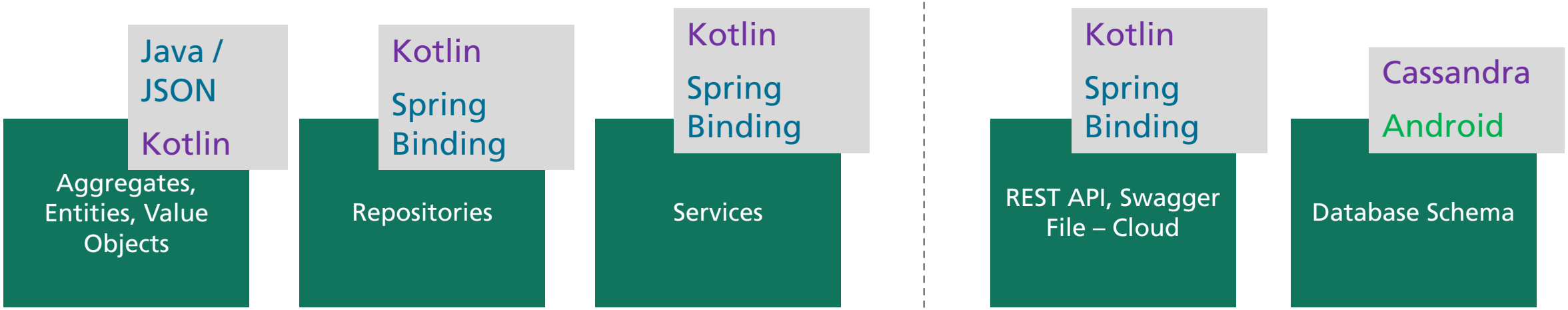
```
[
  {
    "id": "d10b642c-6e76-11e8-adc0-fa7ae01bbebc",
    "readVersion": [ecfe3992-b0e4-11e8-96f8-529269fb1459, 080c2a0a-b0e5-11e8-96f8-529269fb1459, ...],

    "operation": "de.fraunhofer.iese.oih.Contact.delete()",
    "operationTime": "2007-04-05T12:30-02:00",
    "operationOriginAppUuid": "com.snazzycontacts.SnazzyContacts",

    "securityUserUuid": "bc9c46fe-238b-11e8-b467-0ed5f89f718b",
    "securityUserRole": "some-role",

    "parameters": []
  },
  {
    "id": "7d2817e6-6e77-11e8-adc0-fa7ae01bbebc",
    "readVersion": [ecfe3992-b0e4-11e8-96f8-529269fb1459, 080c2a0a-b0e5-11e8-96f8-529269fb1459, ...],

    "operation": "de.fraunhofer.iese.oih.Contact.update(de.fraunhofer.iese.oih.Contact)",
    "operationTime": "2007-04-05T12:30-02:00",
    "operationOriginAppUuid": "com.snazzycontacts.SnazzyContacts",

    "securityUserUuid": "bc9c46fe-238b-11e8-b467-0ed5f89f718b",
    "securityUserRole": "some-role",

    "parameters": [
      {
        "firstName": "Torsten",
        "lastName": "Kuhn"
      }
    ]
  }
]
```

Causal Histories

Operation Event Sink and Global Operation Log – Mobile Sync Scenario

- One (horizontally scalable) Central Service Replica in the Cloud functioning as a hub
  - designed similar to services of the cloud scenario from the previous slide
  - mobile apps use it to poll remote updates
  - mobile apps use it to transmit local updates
  - -> API of the central replica can again be generated (same approach as in cloud scenario)
- Apps have operation event sink and global operation log on the mobile device

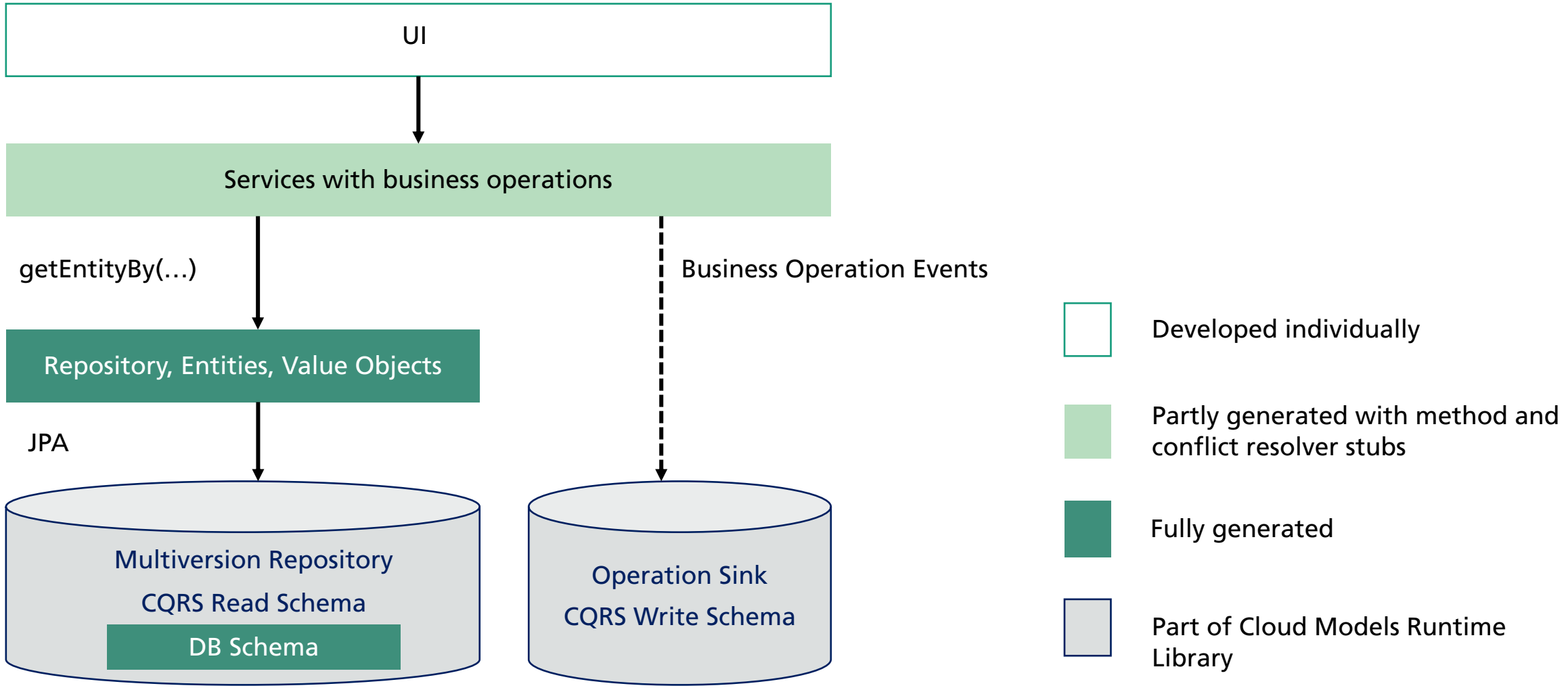
Cloud Models Generated Replica Components



MVP

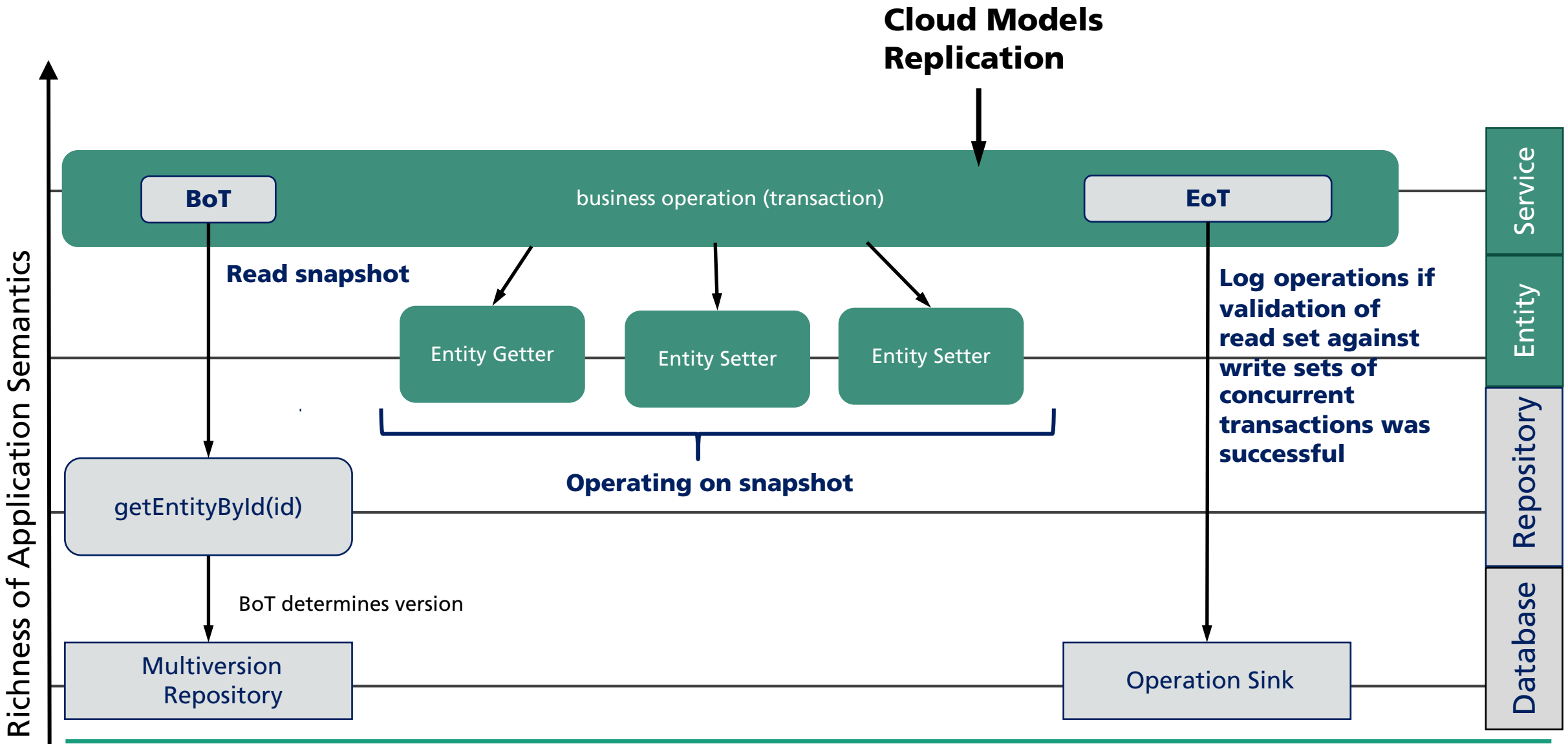
Fraunhofer IESE

Layered Architecture of Microservices / Mobile Apps



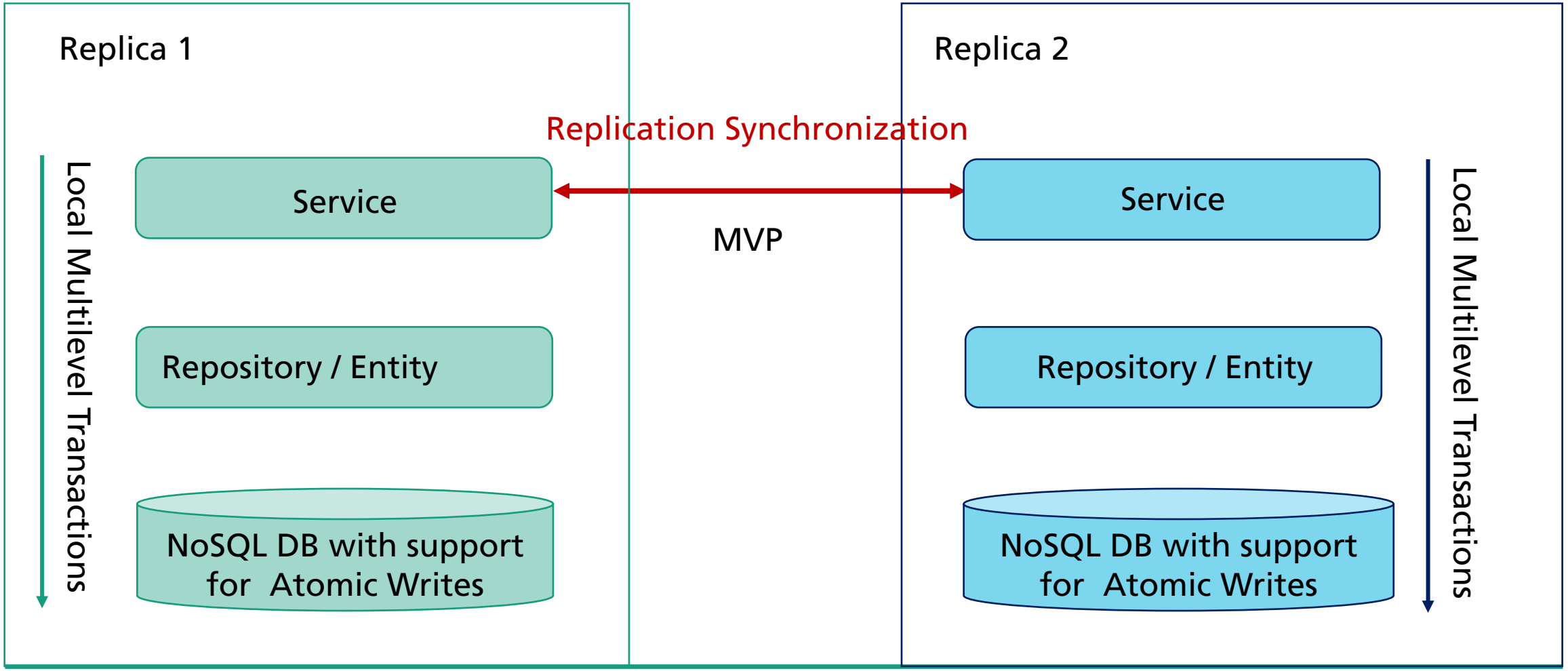
Fraunhofer IESE

Level of Cloud Models Replication



Fraunhofer IESE

Outlook: Cloud Models Multilevel Transactions



Fraunhofer IESE

Cloud Models Testing Tools

Cloud Models Roadmap

- Ecosystem Data Sync: Lessons Learned, Best Practices, First Ideas & State of the Art Analysis -> John Deere Cooperation
- Cloud Models Core Library & Code Generator -> Software Campus Project (cloud-models.de), Digitale Teams Project (digitale-teams.de)
- Methodology & Design Guidelines -> PhD
- **OIH Connector for Cloud Model Apps -> OIH Project**

Integration Issues & Challenges OIH Connector

- OIH /data REST endpoint can be used and data can be taken directly from the multi-version repository of the cloud-models framework
  - How is versioning treated in OIH ?
  - -> Connector will maintain pending operations that need to be scheduled for synchronization with OIH

- Remote updates polled from /webhooks endpoint
  - Problem: No operations can be queried via /webhooks endpoint
  - Again: How is versioning treated in OIH ?
  - CRUD operations need to be derived by the connector
    - **how to determine the identity of a (derived) CRUD operation ?**
      - **what happens if a update-events are delivered more than once?**
      - **how to prevent "update cycles"?**
      - remote updates events need to be treated differently, than locale ones that are scheduled for propagation
  - Cloud Models Replication can only run on level of basic CRUD operations. No support for high-level service operations. No sophisticated conflict handling.
    - **Adaption of the cloud models framework necessary**, s.t. replication level is configurable **(basic CRUD operations vs. service operations (preferred))**
    - **Conflicts will be resolved by the replicas with the deterministic conflict resolution algorithm of cloud models framework (based on the conflict relationships of basic read-and-write operations)**
    - **cloud model apps will be able to register conflict resolvers via DSL of the cloud models framework**
    - **Consequence: cloud models apps might not be eventually consistent with OIH apps, in case OIH applies a different reconciliation algorithm.**

Susanne Braun

Tel.: +49 631 6800 2138

susanne.braun@iese.fraunhofer.de

Twitter: @susannebraun