Karlsruhe Institute of Technology

Christian-Albrechts-Universität zu Kiel

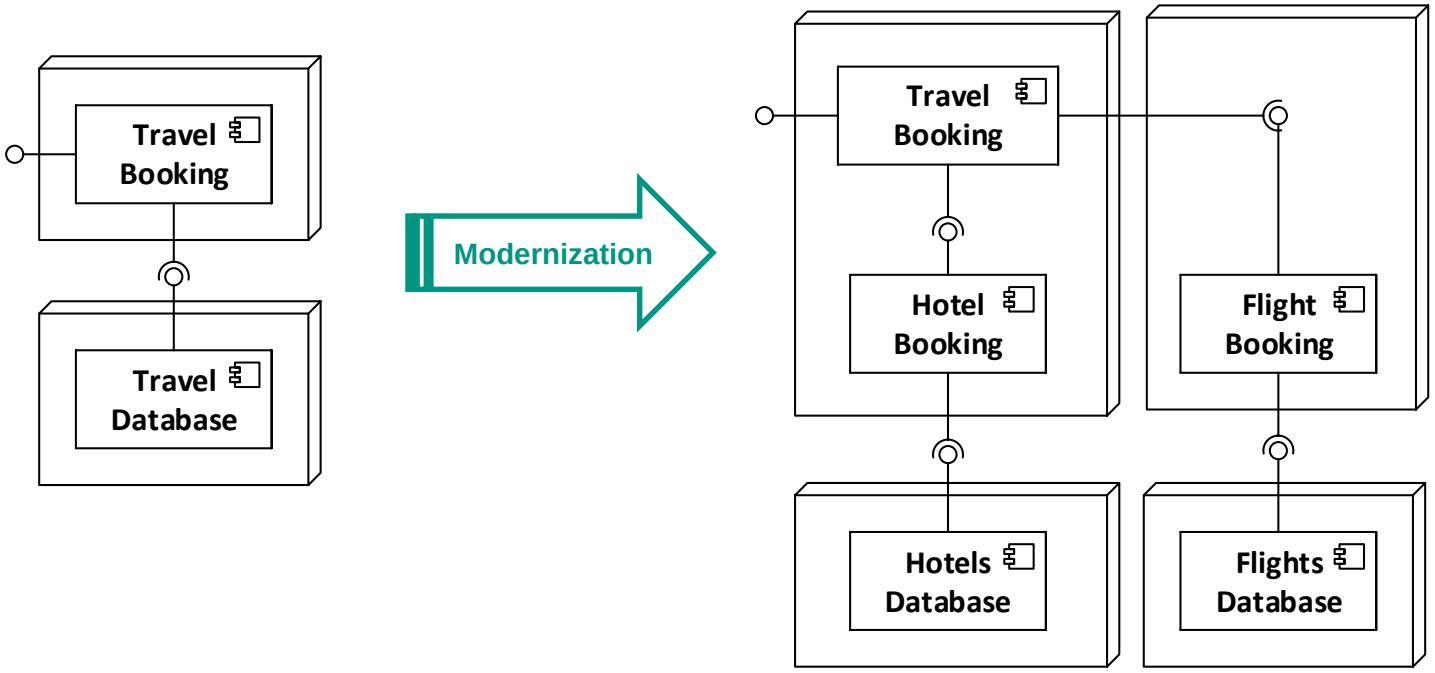
Extending the Palladio Component Model to Analyze Data Contention for Modernizing Transactional Software Towards Service-Orientation

Philipp Merkle (KIT, Karlsruhe),
Holger Knoche (CAU, Kiel)

| November 5th 2015 @ SSP 2015

SOFTWARE DESIGN AND QUALITY GROUP
INSTITUTE FOR PROGRAM STRUCTURES AND DATA ORGANIZATION, FACULTY OF INFORMATICS

Modernizing towards Service-Orientation



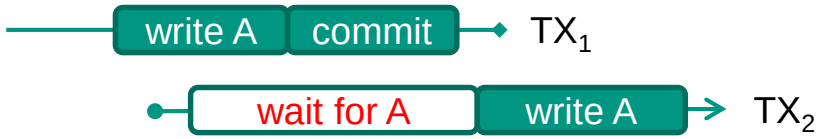
Monolithic application

Component-based, service-oriented application

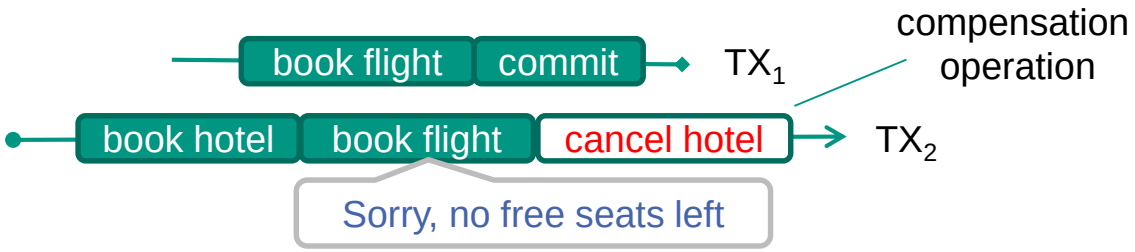
Different Ways to Isolate Concurrent Transactions

- Isolation prevents concurrent transactions from mutually overriding their data

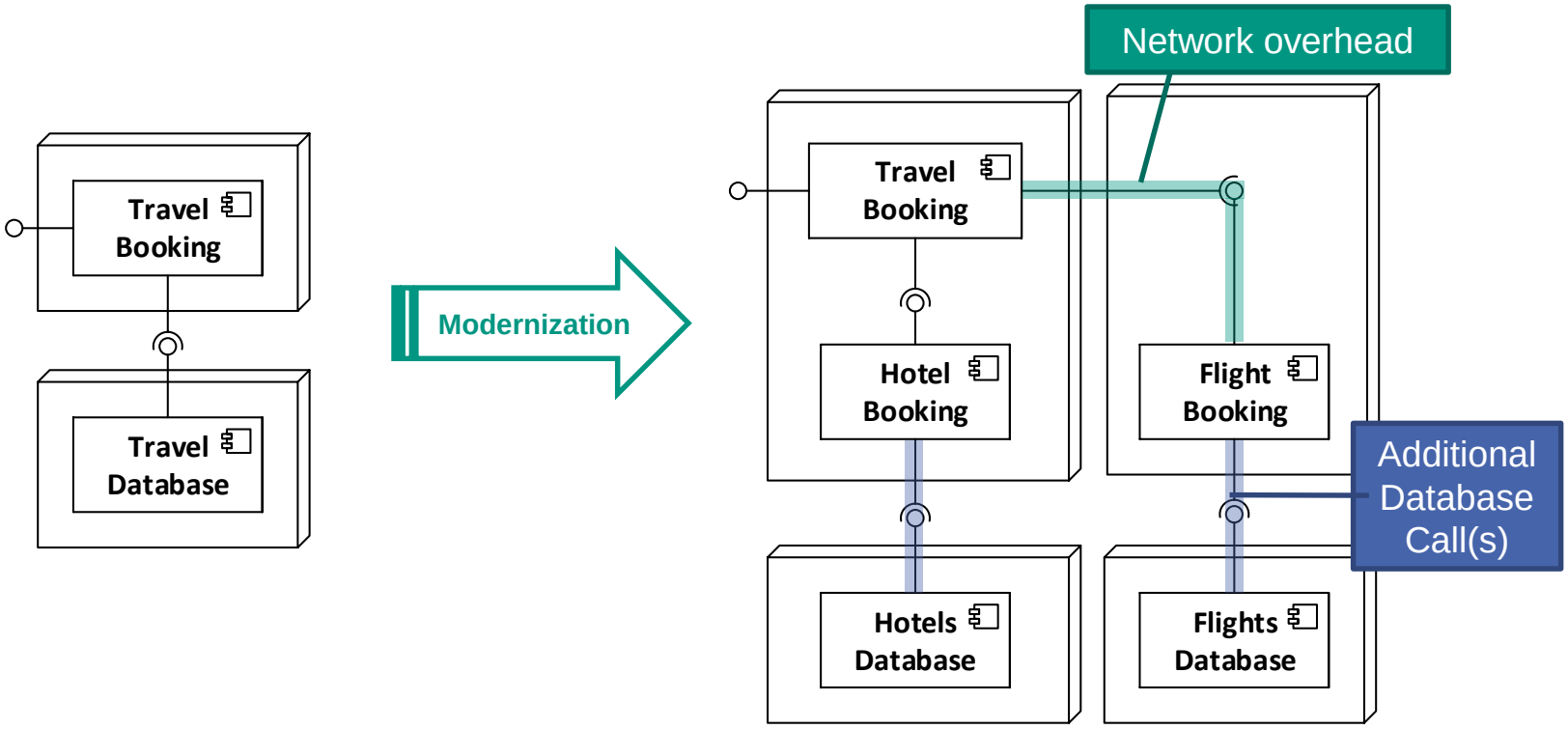
- **Classical „pessimistic“ way**
  - Lock data before reading/writing
  - Waiting time impacts performance



- **Service-oriented „optimistic“ way**
  - Pretend there is no concurrent data access…
  - …until access conflict actually happens → abort + retry
  - Abort often uses **compensation**
  - Try-Cancel-Confirm



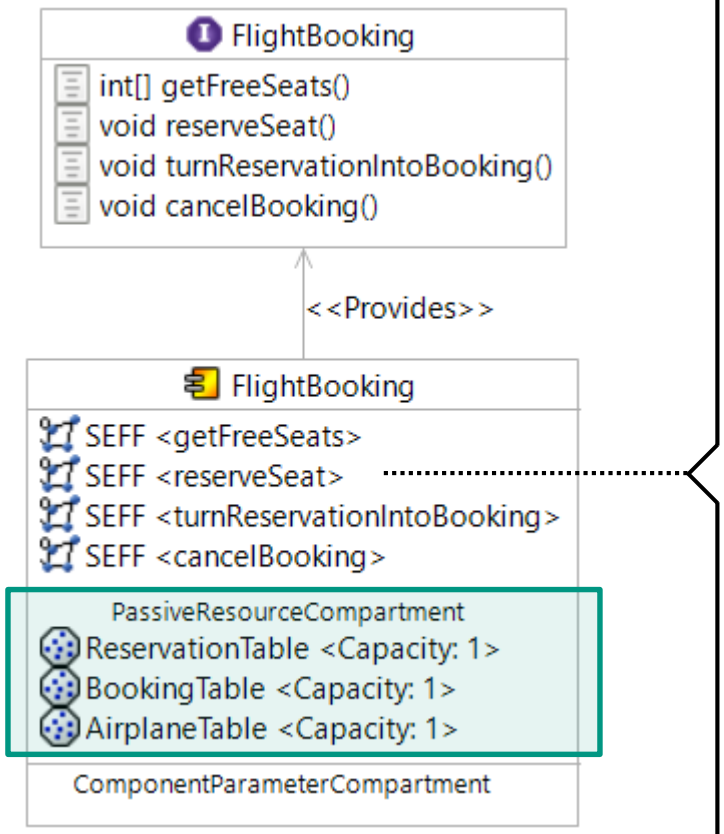
Modernization Revisited



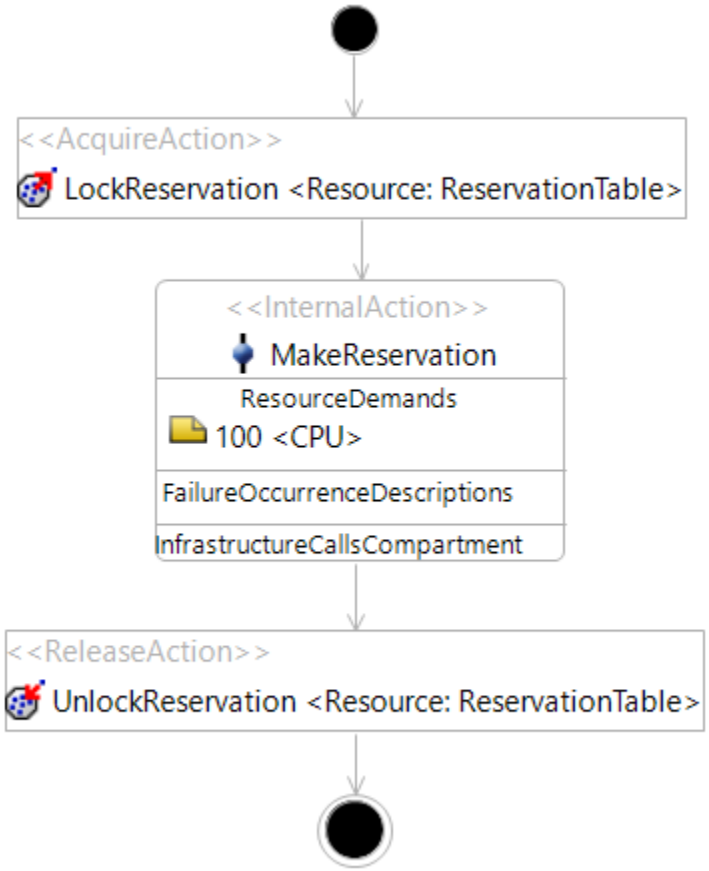
- Transaction duration increased by network + database overheads
  - Locks held longer (Pessimistic)
  - Higher abort risk (Optimistic)

Increased Data contention
Impact on system performance? → PCM

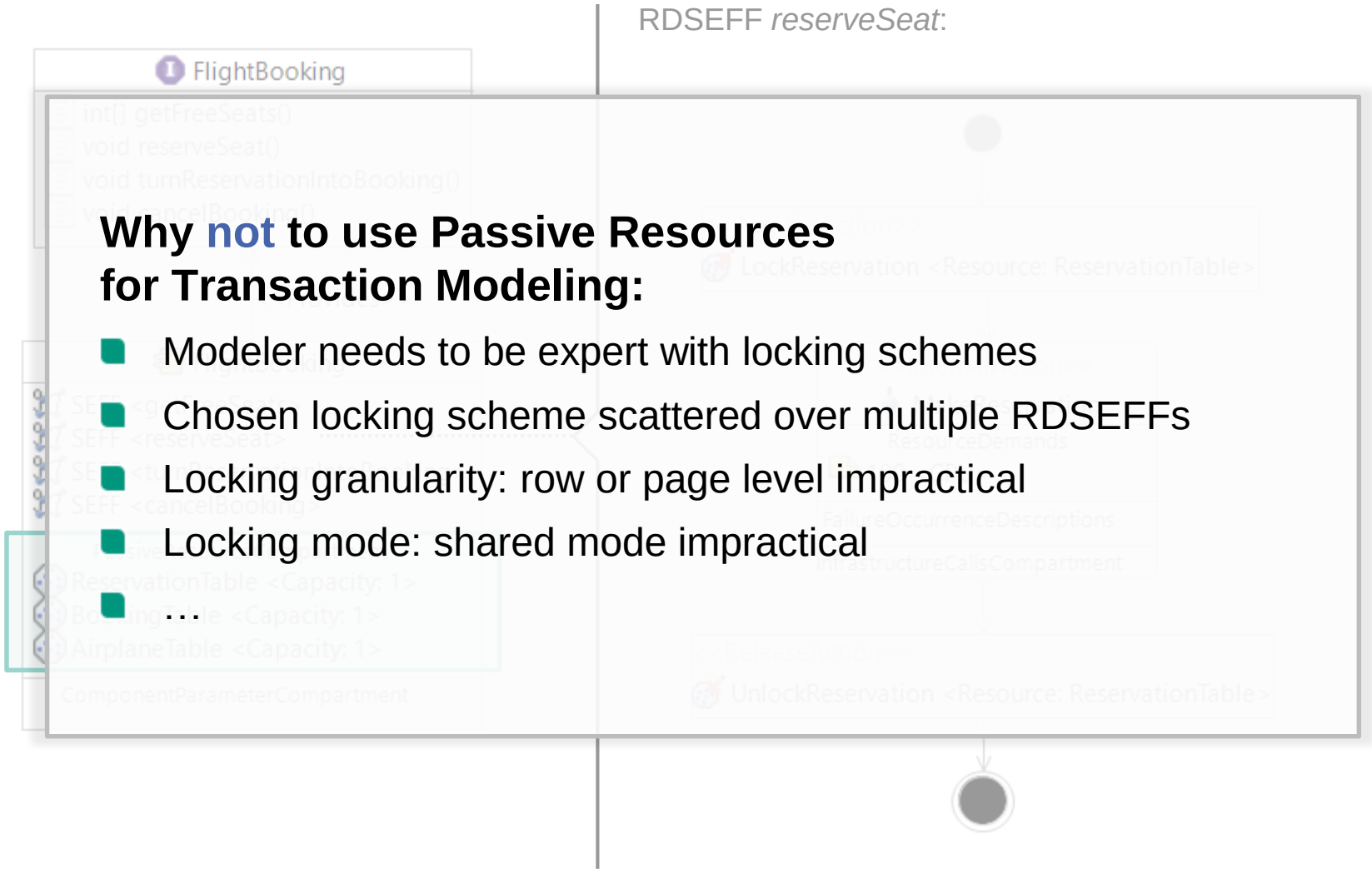
Transactions with the PCM – Status Quo



RDSEFF *reserveSeat*:



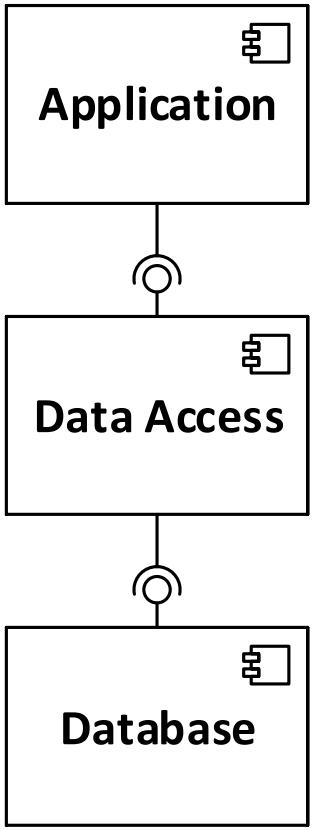
Transactions with the PCM – Status Quo



Why not to use Passive Resources for Transaction Modeling:

- Modeler needs to be expert with locking schemes
- Chosen locking scheme scattered over multiple RDSEFFs
- Locking granularity: row or page level impractical
- Locking mode: shared mode impractical
- …

PCM.TX Overview



- **Transaction** Boundaries
- Compensation Behaviour
- „Try-Cancel-Confirm“ Behaviour

- **Queries**
- (Transaction Boundaries)
- (Compensation Behaviour)

- **Schema** Definition
- Table Deployment

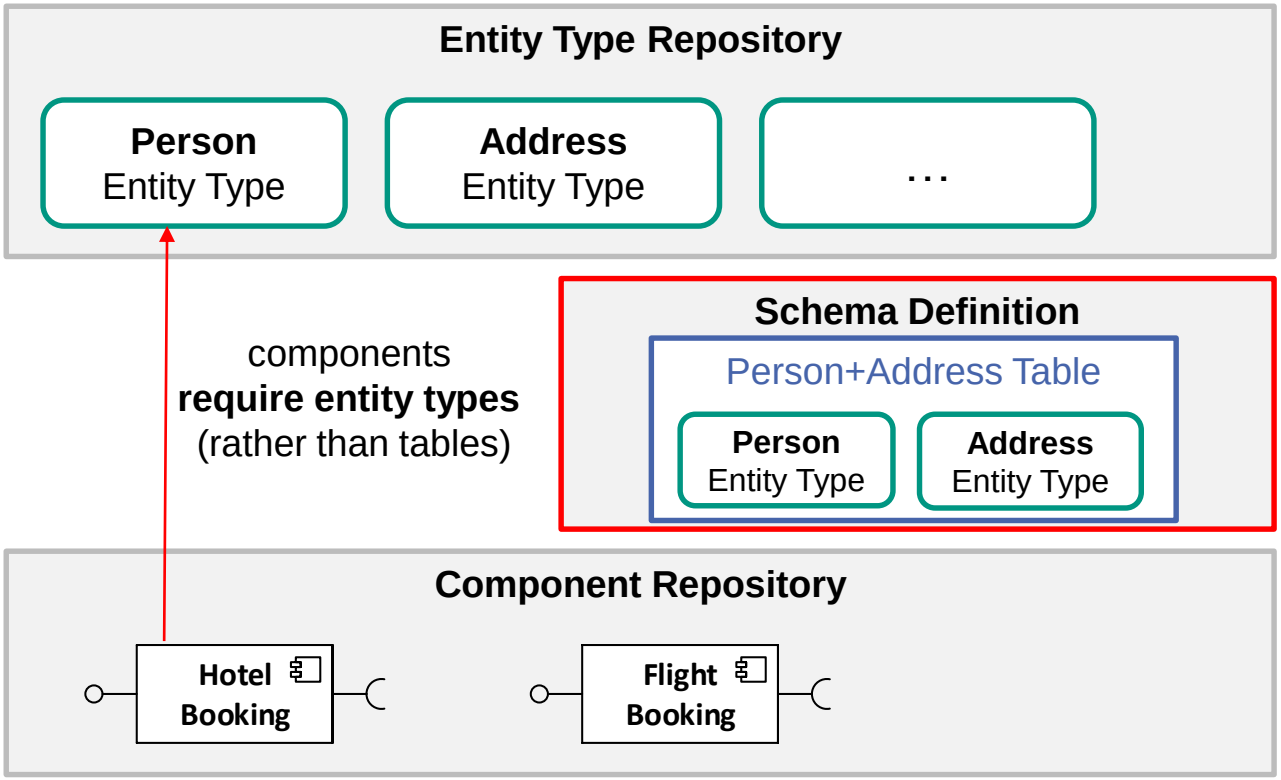
Cross-Cutting:

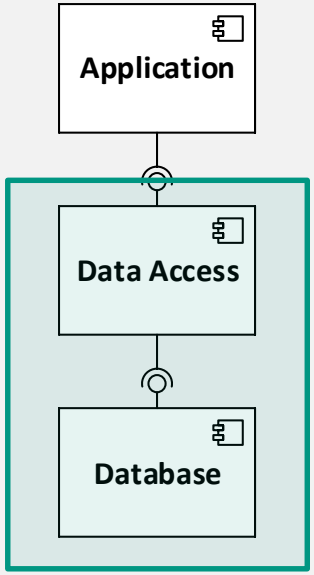
Explicit Context (Database) Dependencies

Separate Schema Specification from Component Specification

Separate Schema Specification from Component Specification

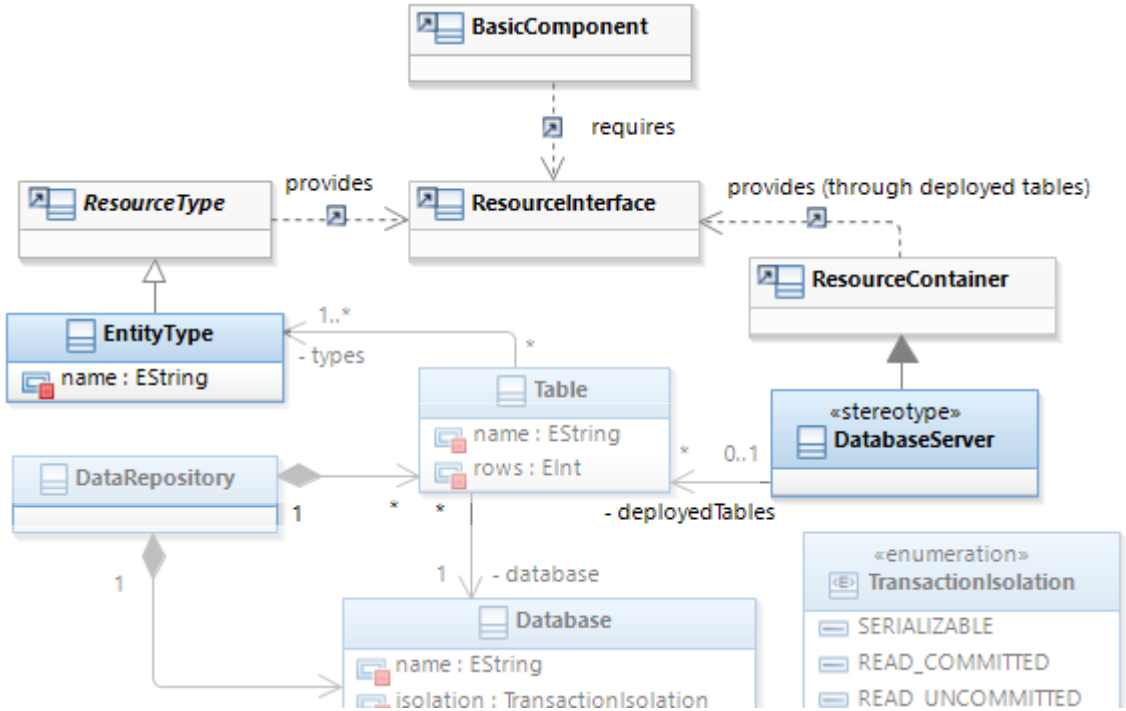
- Problem: If components refer to tables directly, they need modification whenever the schema changes!
- Solution: **Decouple** schema and component specification

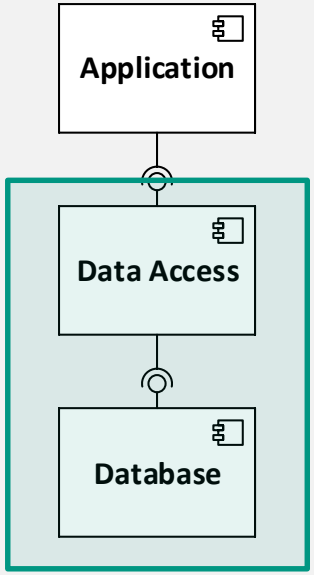




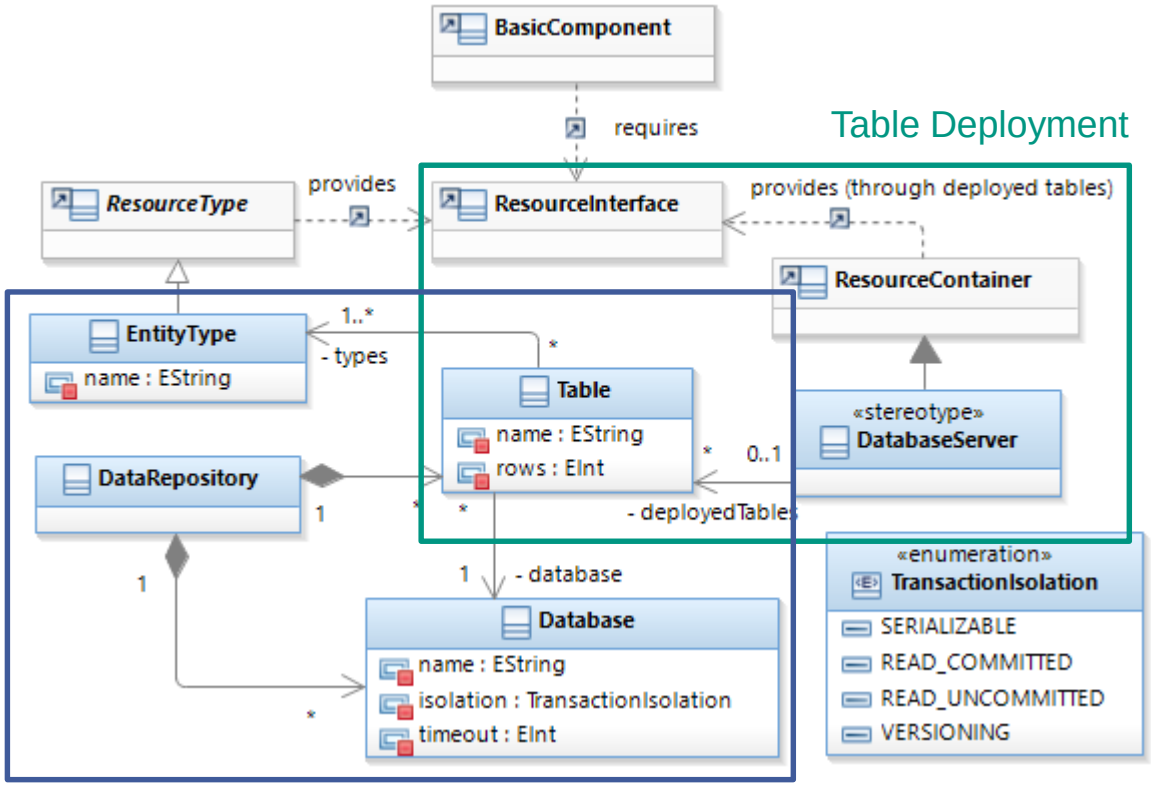
Explicit Context (Database) Dependencies

- "*A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only*" [Szyperski 2002
- Facilitates Reuse of Component Specifications
- Component's dependence upon entity types is a context dependency
- Approach: Reuse **PCM Resource Interfaces** [Hauck et al. 2009]

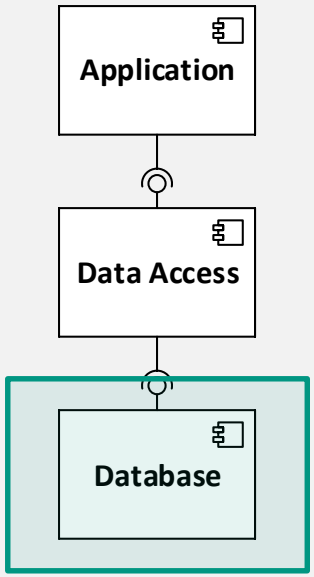




Schema Definition & Table Deployment

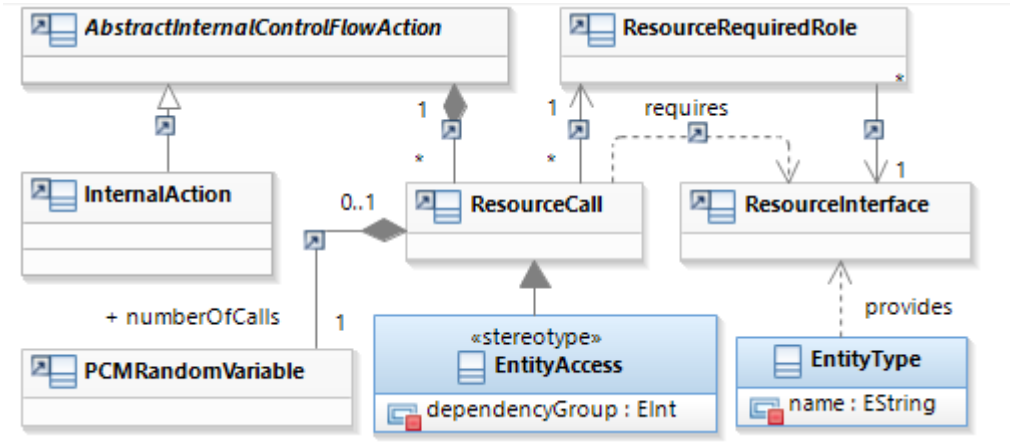


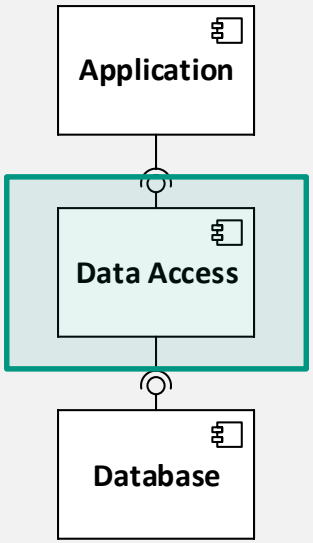
Schema Definition



Queries

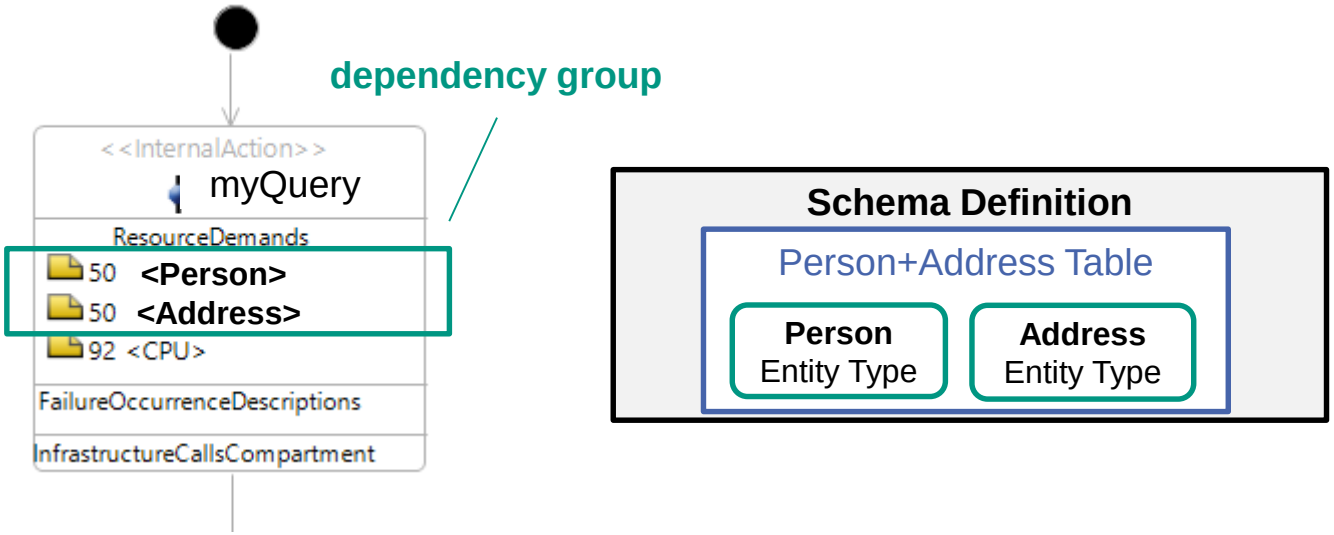
- So far: InternalActions require processing resources (CPU, HDD)
- PCM.TX: InternalActions may additionally **require entities**

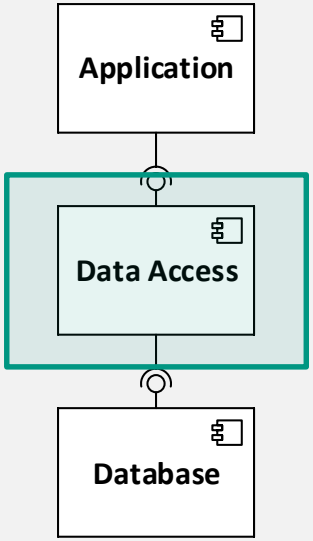




- *Number of entities* described as PCMRandomVariable, hence a…
  - Constant
  - Probability distribution function
  - Function of (call/component) parameters

Queries – Dependency Groups





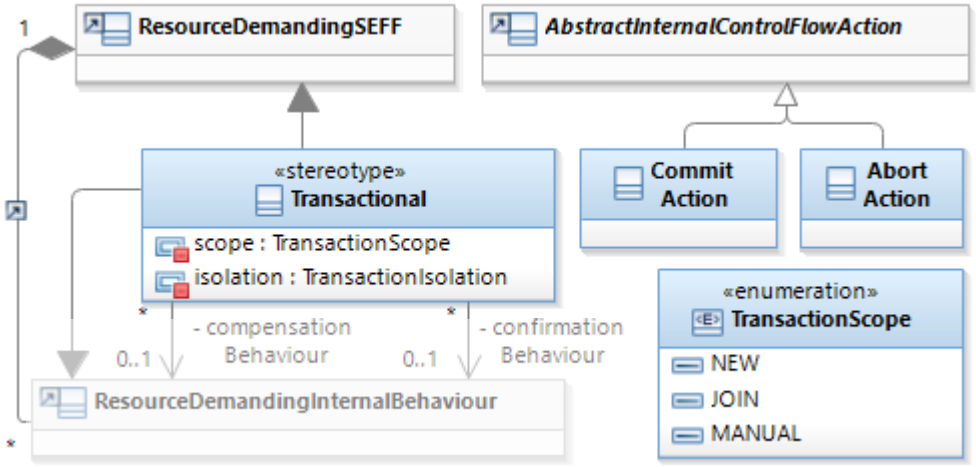
- How many rows from the Person+Address Table are accessed?
  - Case 1: 50 rows, if each address **belongs to** one of the 50 persons

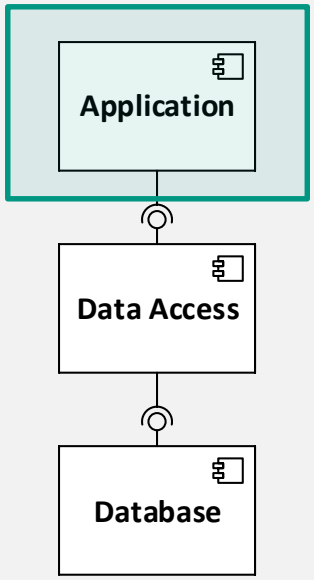
    → assign entity accesses to same dependency group
  - Case 2: 100 rows, if there is **no correspondence** between the addresses and persons

    → no dependency (default case)

Transaction Boundaries

- Annotate transactional SEFFs („declarative“ approach)
- Scope determines what happens when SEFF S1 calls S2
  - Case 1: Two separate transactions, one for S1, one for S2 (NEW)
  - Case 2: A joint transaction (JOIN)

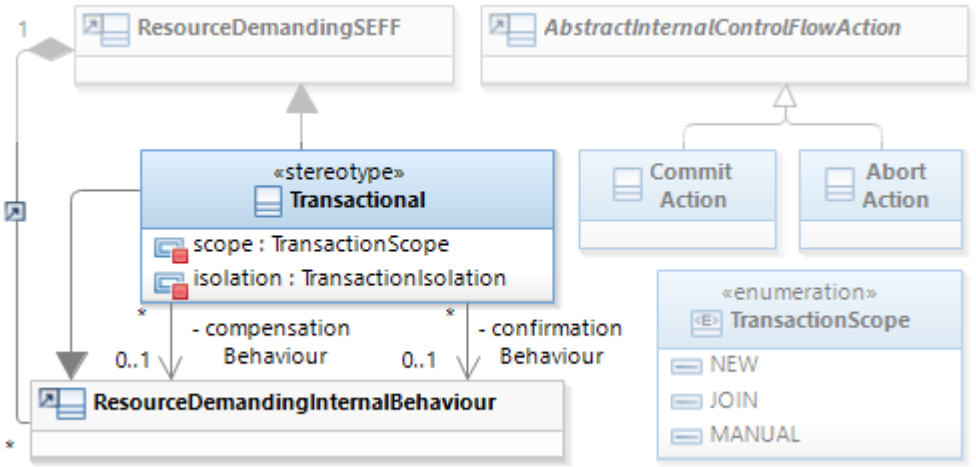


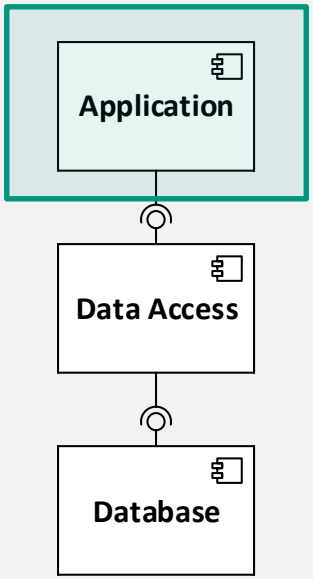


- Scope MANUAL for fine-grained transaction demarcation, together with Commit/Abort

Compensation,Try-Cancel-Confirm

- Compensation behaviour executes when its corresponding transaction **aborts**





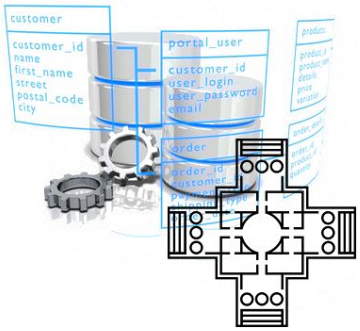
- Confirmation behaviour executes when its corresponding transaction **succeeds**

Conclusion and Future Work

- PCM.TX adds transaction and database modeling to the PCM
- Allows to evaluate design decisions specific to transactional information systems

  → **evaluate impact of migration towards service-orientation**

- Future Work
  - Simulation of PCM.TX models in EventSim
  - Non-uniformely distributed data accesses
  - Sharding (horizontal partitioning of tables)



References

- **[Szyperski 2002]** C. Szyperski, D. Gruntz, and S. Murer. *Component Software: Beyond Object-Oriented Programming*. ACM Press and Addison-Wesley, New York, 2 edition, 2002.

- **[Hauck et al. 2009]** M. Hauck, M. Kuperberg, K. Krogmann, and R. Reussner. Modelling Layered Component Execution Environments for Performance Prediction. In Proceedings of the 12th International Symposium on Component Based Software Engineering (CBSE 2009), number 5582 in LNCS, pages 191{208. Springer, 2009.

Backup Slides

PCM Resource Interfaces