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
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

![Diagram showing different ways to isolate concurrent transactions](image-url)
Modernization Revisited

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

Impact on system performance? → PCM
Transactions with the PCM – Status Quo

**RDSEFF reserveSeat:**

```
<<AcquireAction>>
LockReservation [Resource: ReservationTable]

<<InternalAction>>
MakeReservation
ResourceDemands
100 [CPU]
FailureOccurrenceDescriptions
InfrastructureCallsCompartment

<<ReleaseAction>>
UnlockReservation [Resource: ReservationTable]
```
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

**Application**

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

**Data Access**

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

**Database**

- **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

```
Entity Type Repository

Person
  Entity Type

Address
  Entity Type

...  

components require entity types (rather than tables)

Schema Definition

Person+Address Table

Person
  Entity Type

Address
  Entity Type

Component Repository

Hotel Booking

Flight Booking

Application

Data Access

Database
```

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

### Diagram Description

**Schema Definition**

- **EntityType**
  - name: EString
  - has types
    - **Table**
      - name: EString
      - rows: EInt
      - deployed Table

- **DataRepository**
  - 1..* - database

- **Database**
  - name: EString
  - isolation: TransactionIsolation
  - timeout: EInt

**Table Deployment**

- **BasicComponent**
  - requires

- **ResourceType**
  - provides

- **ResourceInterface**
  - provides (through deployed tables)

- **ResourceContainer**
  - Stereotype

**Application**

- **Data Access**

- **Database**

---

**References**

Merkle, Knoche: Modernizing Transactional Software Towards Service-Oriented Application

**Timeline**

- 2015-11-05

---

**Institute for Program Structures and Data Organization**
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-uniformly distributed data accesses
- Sharding (horizontal partitioning of tables)
References


PCM Resource Interfaces