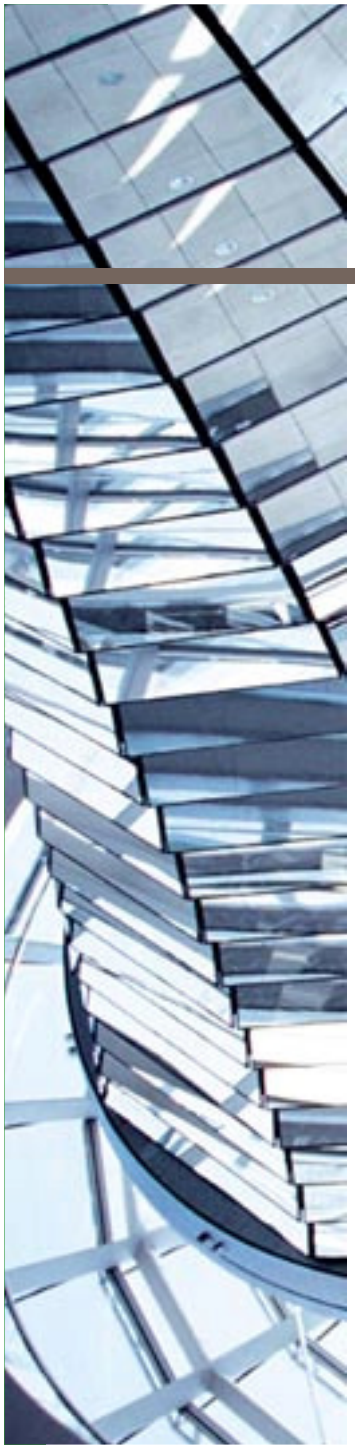


4. Persistence and Transaction

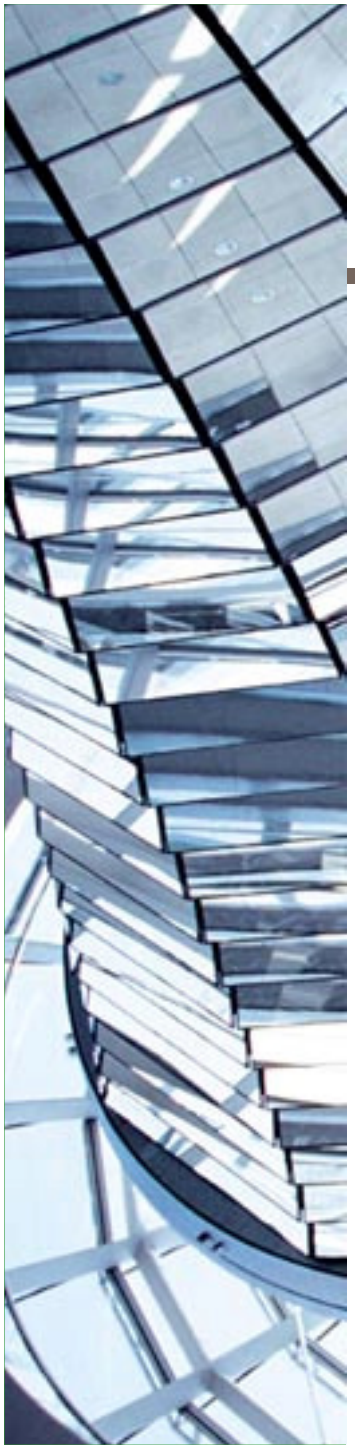
Reference Architectures and Patterns

Winter Semester 2008 / 2009
Prof. Dr. Bernhard Humm
Darmstadt University of Applied Sciences
Department of Computer Science

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

The lecture in the context of the entire course

1. Introduction
2. A reference architecture for business information systems
3. Application kernel
4. Persistence and transaction
5. Authorization
6. Client architecture
7. Other reference architectures: SOA, BI, systems integration, …



Agenda

➔ **Reference architecture**

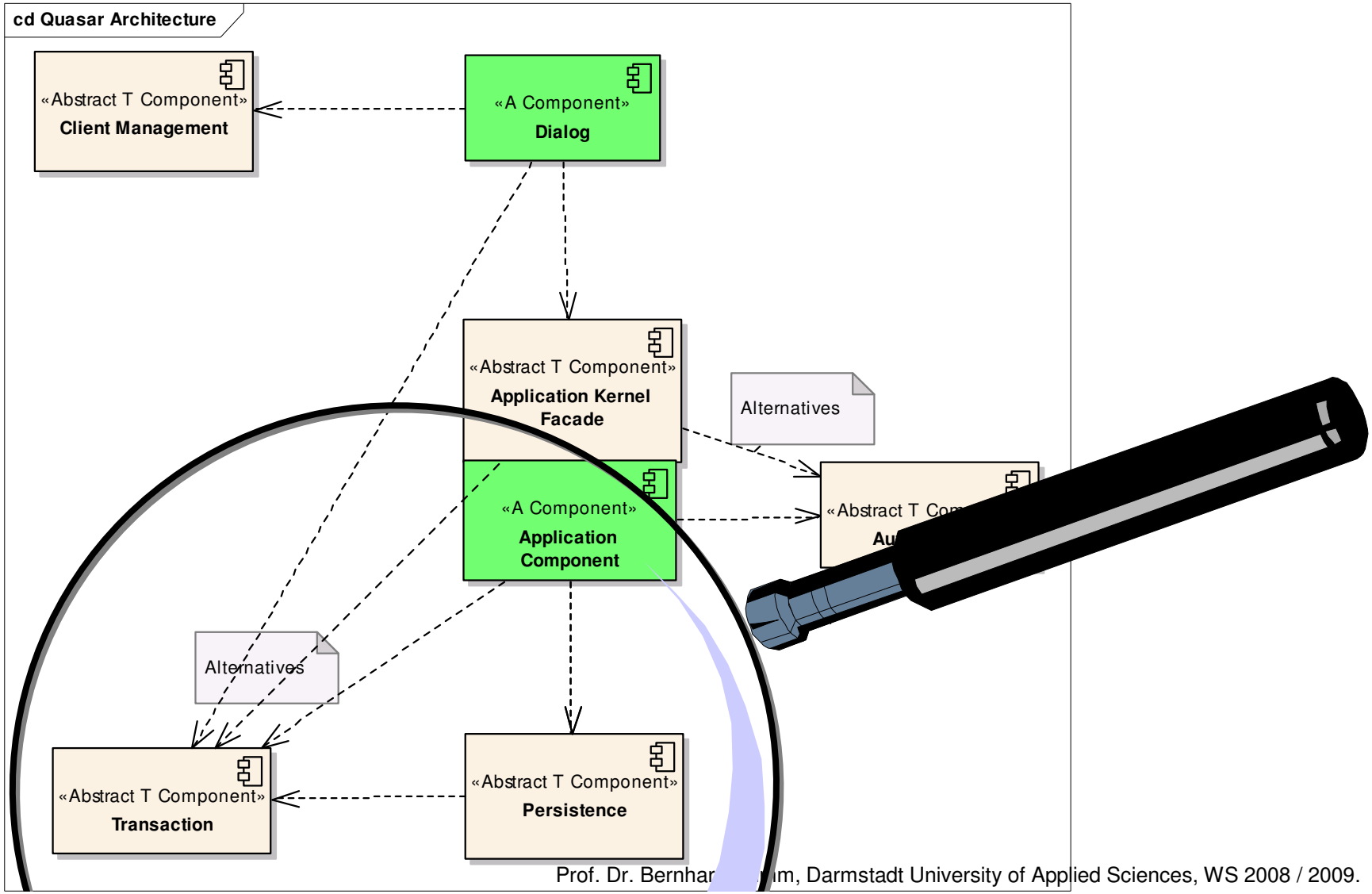
Persistence

Transaction

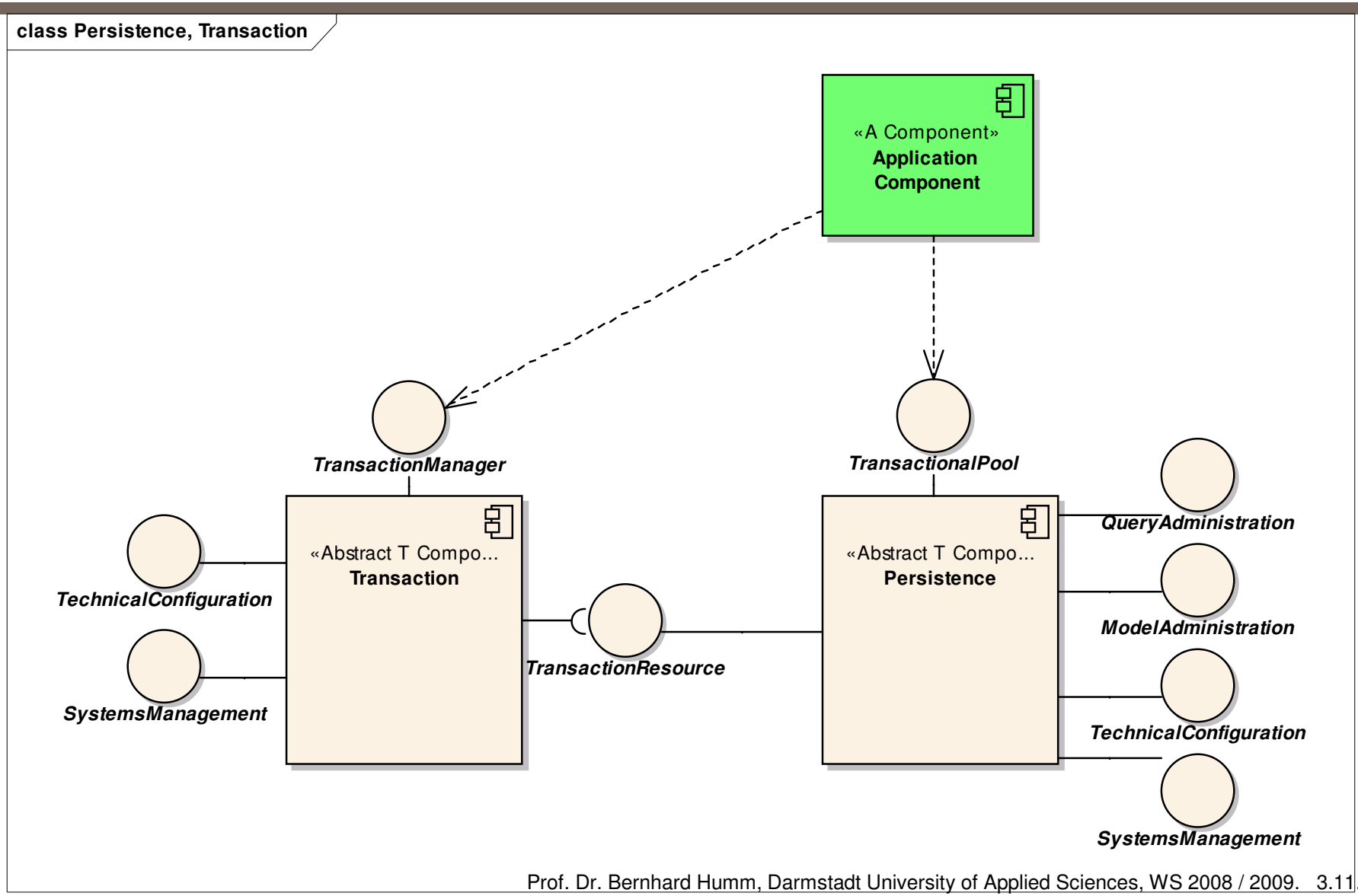
Sequences

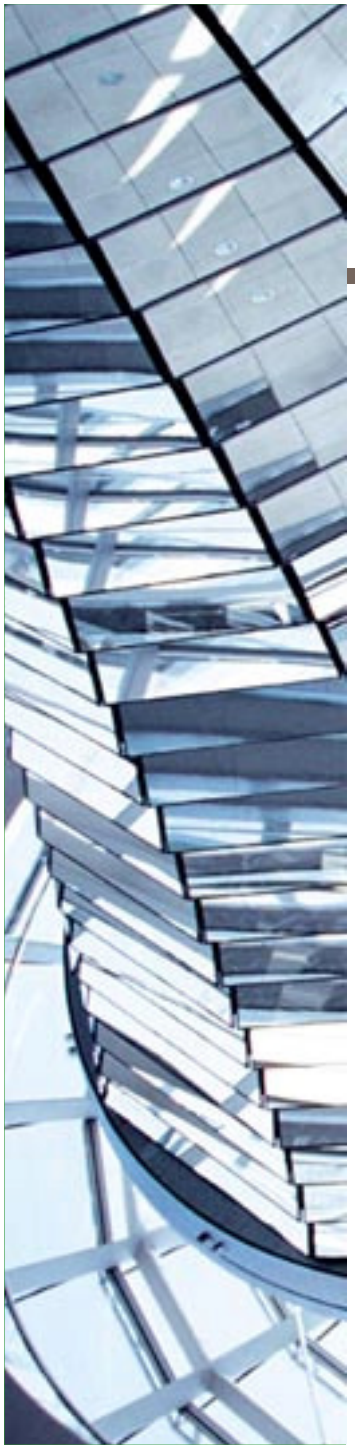
Literature

Reference architecture for business information systems Quasar (Quality Software Architecture)



Persistence and transaction reference interfaces





Agenda

Reference architecture

➔ **Persistence**

Transaction

Sequences

Literature

The Pool reference interface

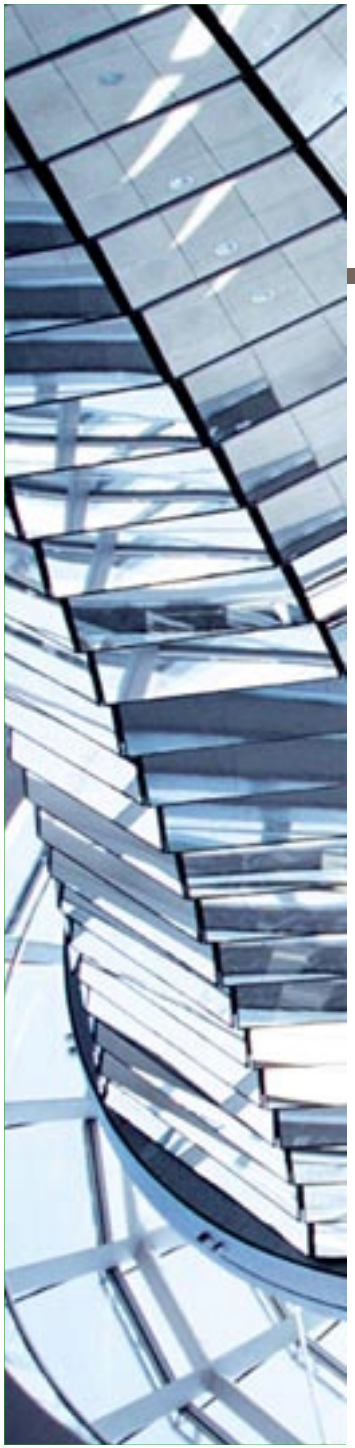
Method Summary	
java.util.Set<java.lang.Object>	**executeQuery**(Query query, java.util.List<java.lang.Object> arguments) [basicQuery] Executes query on this pool and returns the result set
java.lang.Object	**fetch**(UID uid) [basicQuery] Returns the object stored in the Pool under the unique identifier uid.
UID	**getUID**(java.lang.Object object) [basicQuery] Returns the unique identifier of object if object is stored in the pool, error objectNotFound otherwise.
UID	**insert**(java.lang.Object object) [command] Inserts object to this pool and returns the unique identifier
void	**remove**(java.lang.Object object) [command] Removes object from this pool

Example realization of the Pool reference interface: javax.persistence.EntityManager (EJB 3.0)

Method Summary	
javax.persistence.Query	createQuery(String ejbqlString)
<A> A	find(Class<A> entityClass, Object primaryKey)
void	flush()
void	lock(Object entity, javax.persistence.LockModeType lockMode)
protected void	markAsRollback()
<A> A	merge(A entity)
void	persist(Object entity)
protected void	postInit()
void	refresh(Object entity)
void	remove(Object entity)

Source: http://www.hibernate.org/hib_docs/ejb3-api/

Correlation between reference interface and realization



Agenda

Reference architecture

Persistence

➔ **Transaction**

Sequences

Literature

The Transaction reference interface

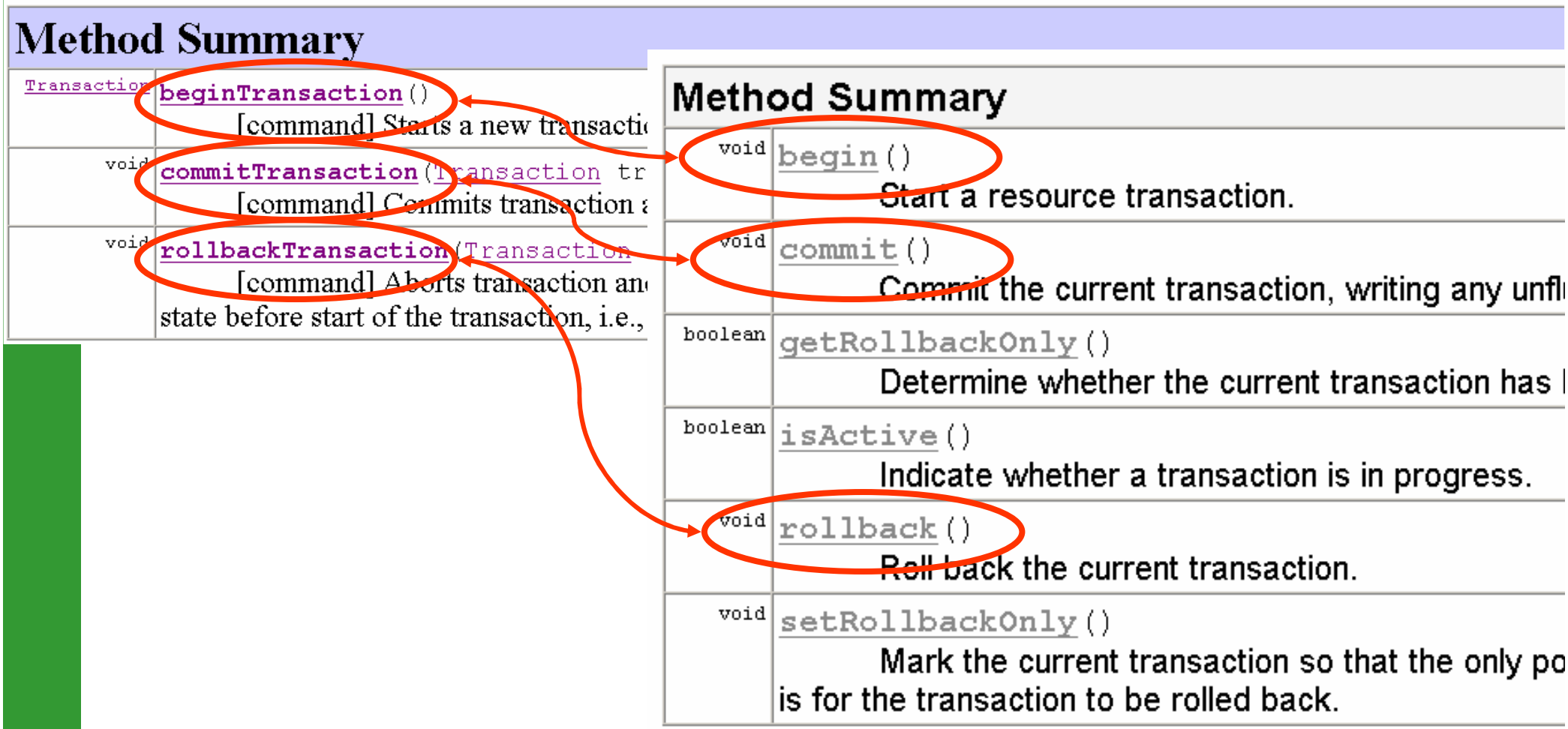
Method Summary	
Transaction	**beginTransaction**() [command] Starts a new transaction and returns the object representing the transaction
void	**commitTransaction**(Transaction transaction) [command] Commits transaction and thus ensures all ACID properties
void	**rollbackTransaction**(Transaction transaction) [command] Aborts transaction and rolls back all resources that have been modified by transaction to the state before start of the transaction, i.e., ensures consistency

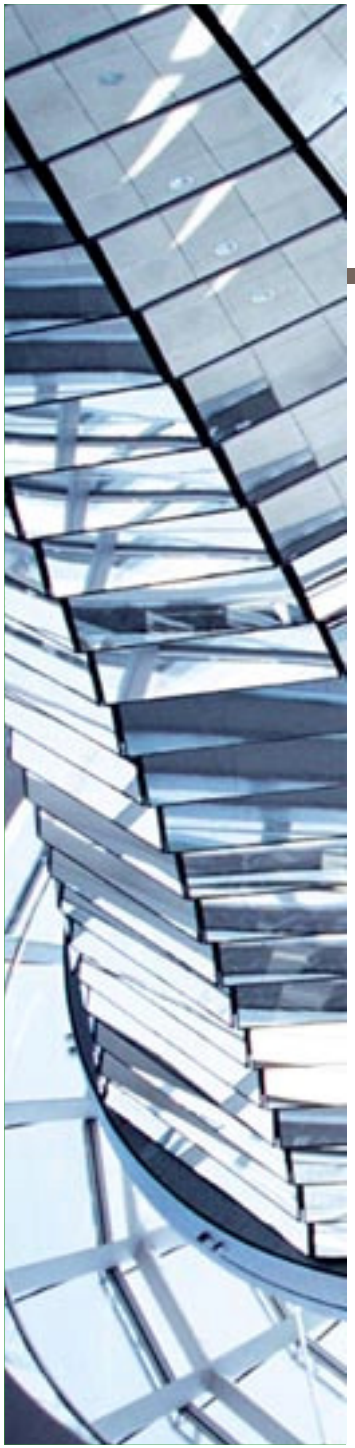
Example realization of the Transaction reference i/f: javax.persistence.EntityTransaction (EJB 3.0)

Method Summary	
void	`begin()` Start a resource transaction.
void	`commit()` Commit the current transaction, writing any unflushed changes to the database.
boolean	`getRollbackOnly()` Determine whether the current transaction has been marked for rollback.
boolean	`isActive()` Indicate whether a transaction is in progress.
void	`rollback()` Roll back the current transaction.
void	`setRollbackOnly()` Mark the current transaction so that the only possible outcome of the transaction is for the transaction to be rolled back.

Source: http://www.hibernate.org/hib_docs/ejb3-api/

Correlation between reference interface and realization





Agenda

Reference architecture

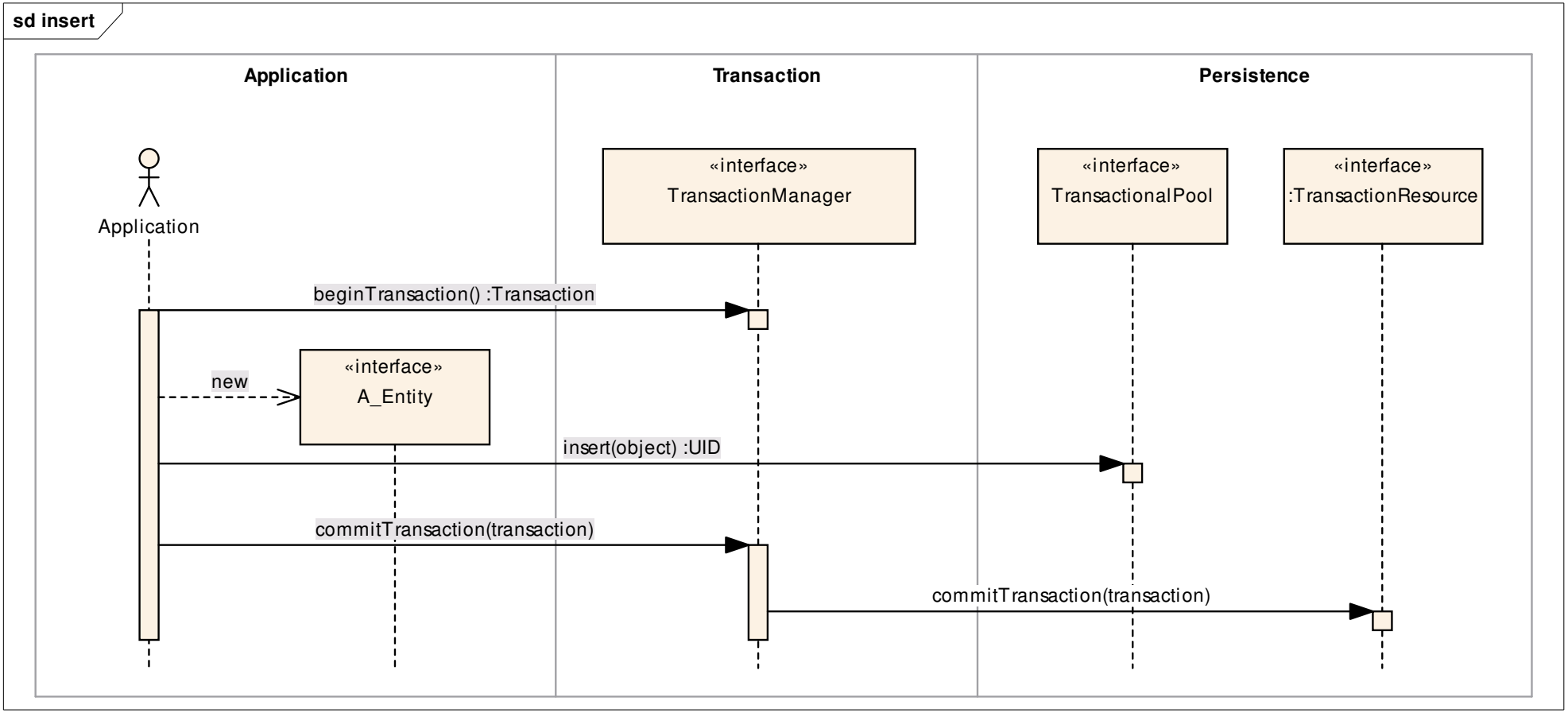
Persistence

Transaction

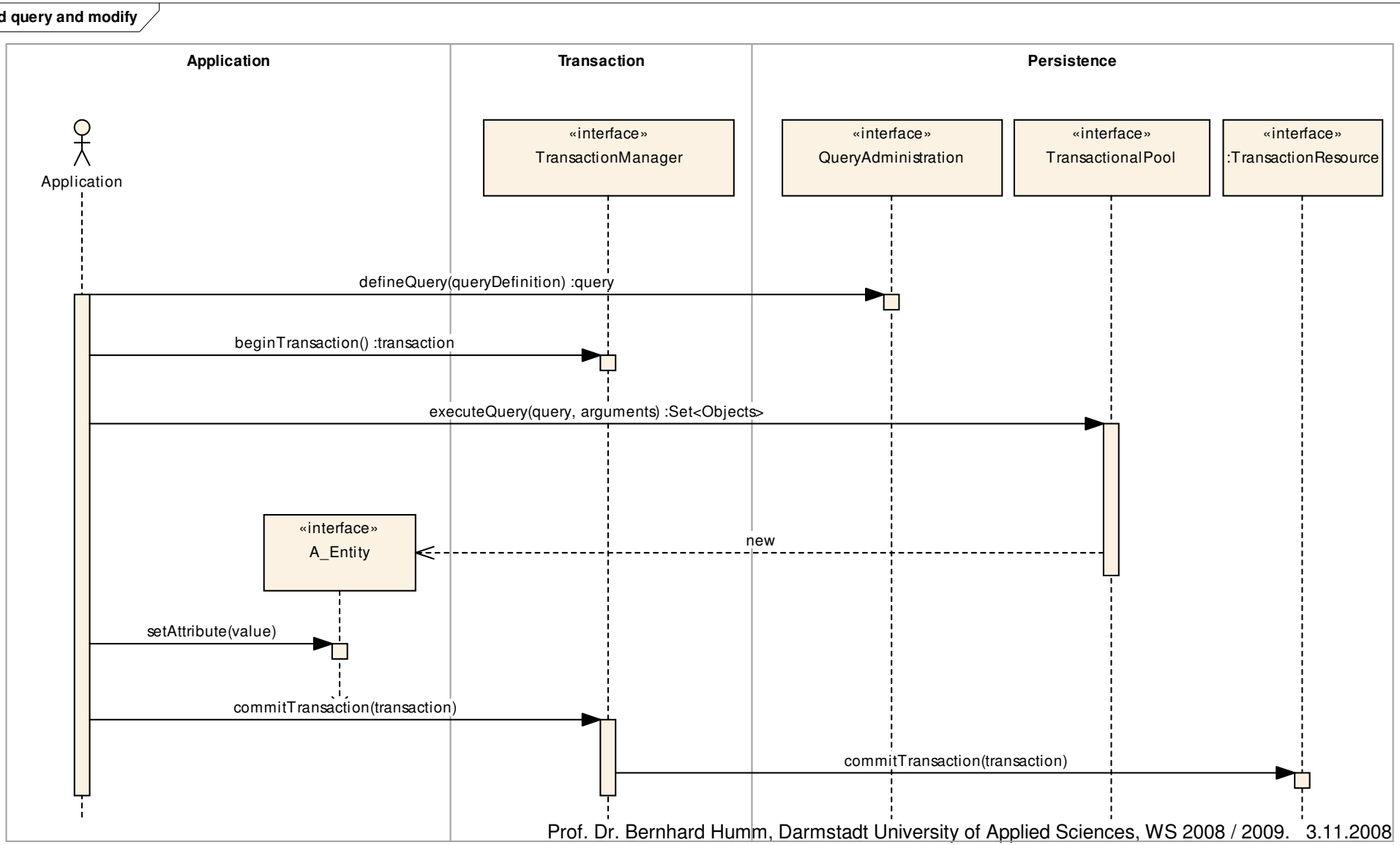
➔ **Sequences**

Literature

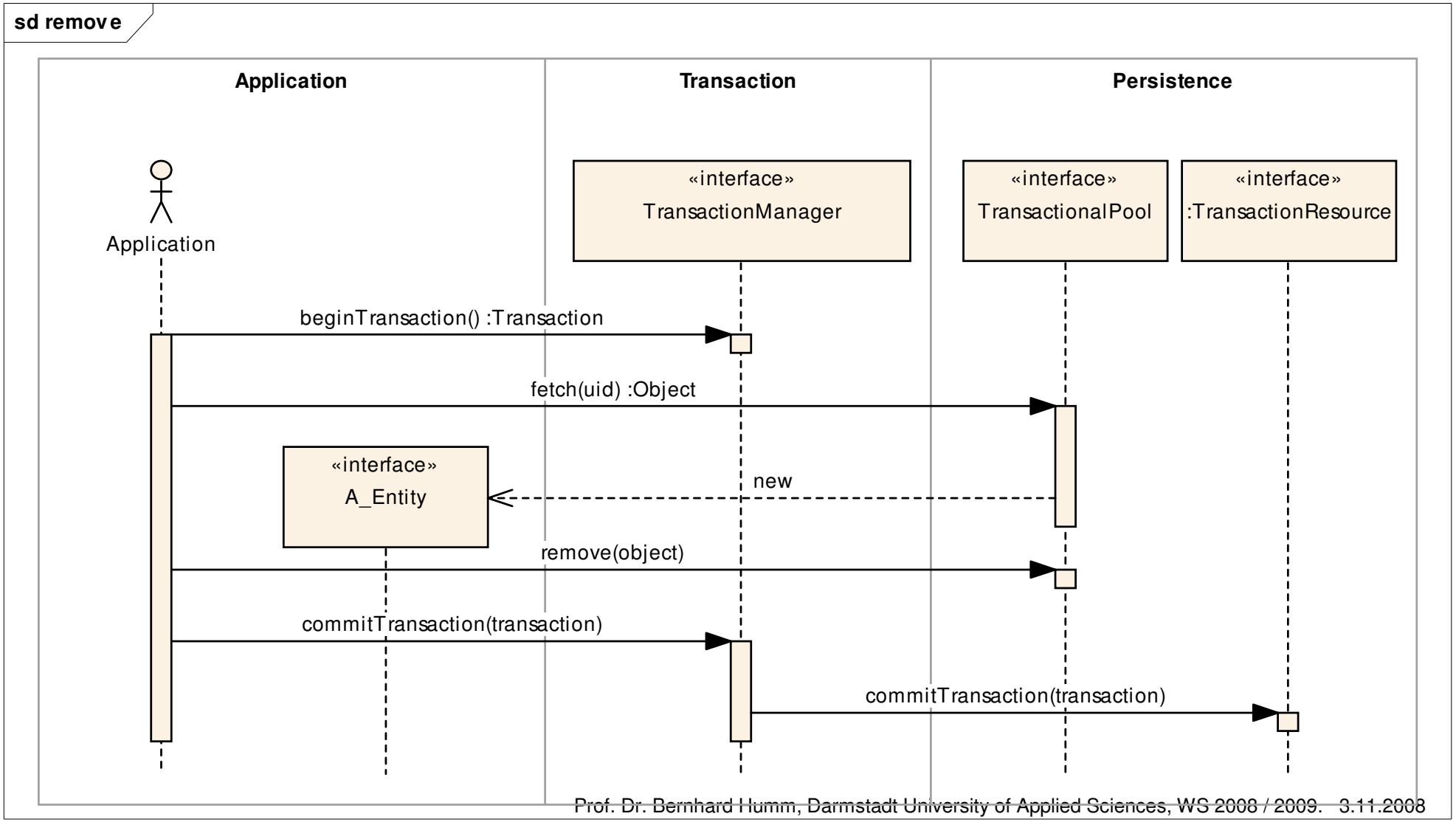
Sequence: insert

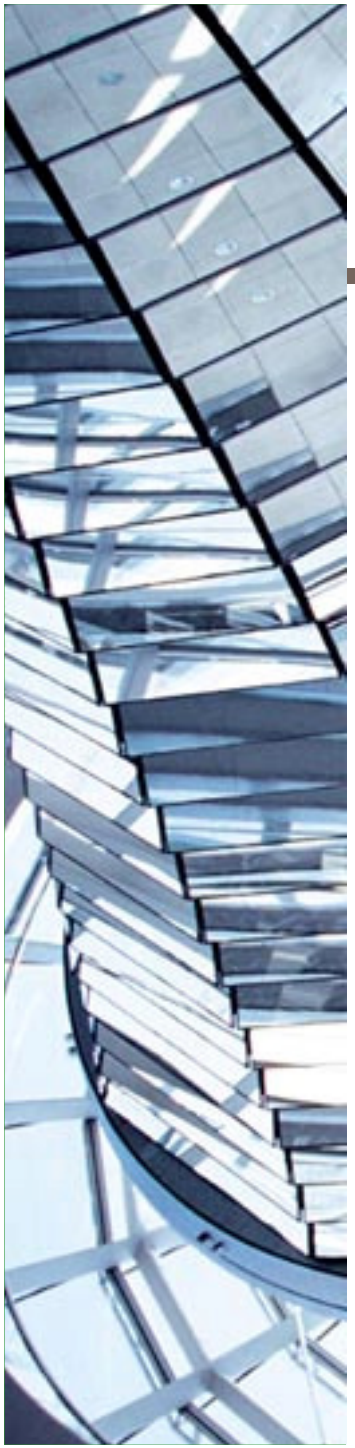


Sequence: query and modify



Sequence: Remove





Agenda

Reference architecture

Persistence

Transaction

Sequences

➔ **Literature**

References for this lecture

Sections of „Teil 2“ dealing with today‘s lecture:

- 5. Transaktionen
- 6. Persistenz