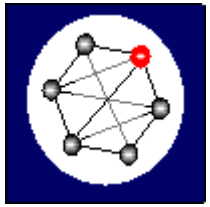


New Technology Trends And Challenges
For The Database Professional
- Challenges For Database Education -

From Flat to Complex Transactions

Tim Lessner, PhD student
Prof. Dr. rer. nat. Fritz Laux

UWS UNIVERSITY OF THE WEST of SCOTLAND

INF
Fakultät
Informatik

Hochschule Reutlingen
Reutlingen University

Questions & Focus

- What is a Transaction?
  (assumed to be known already)
- What is a Flat Transaction?
  (assumed to be known already)
- What makes a transaction complex?

- Transactional Interaction
- Our Research

Outline

I. Preliminaries

II.What makes a transaction complex?

III. Concepts and transaction models

IV. Technologies

V. Our Research

Preliminaries

- ***Why do we need the transaction concept?***
  Ensure always correct data under all circumstances.

- ***What is a transaction?***
  Originally: a transaction is an **ACID *(Atomicity, Consistency, Isolation, Durability)*** unit of work *[Gray, 1993]*
  However, the term has evolved, e.g., Business Process-, Workflow-, User-Transaction.

- ***What is TM?***
  TM encompasses all models, frameworks and physical instances required to enable the successful (*ACID preserving*) execution of concurrent transactions in a computing environment.

Preliminaries

Properties of a transaction:

- **ACID** (Atomicity, Consistency, Isolation, Durability)

System wide Expectations *(not just the database)*:

- **C**onsistency
- **A**vailability
- **P**artitioning Tolerance

[Brewer, 2000], [Gilbert, 2002]

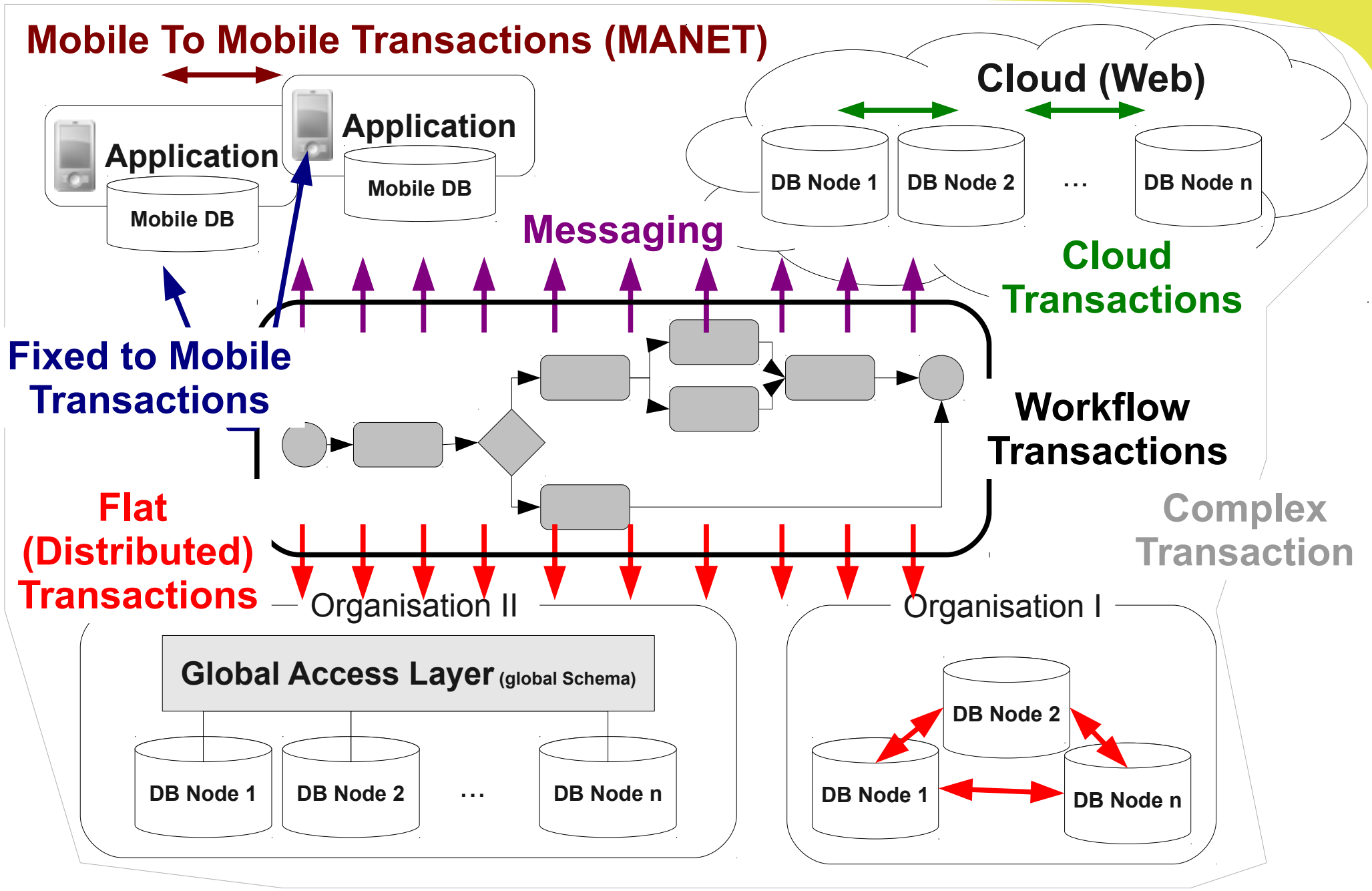
A non formal approach to show a correct and consistent behaviour:

- **Harvest & Yield** *(allows for trade-offs)*

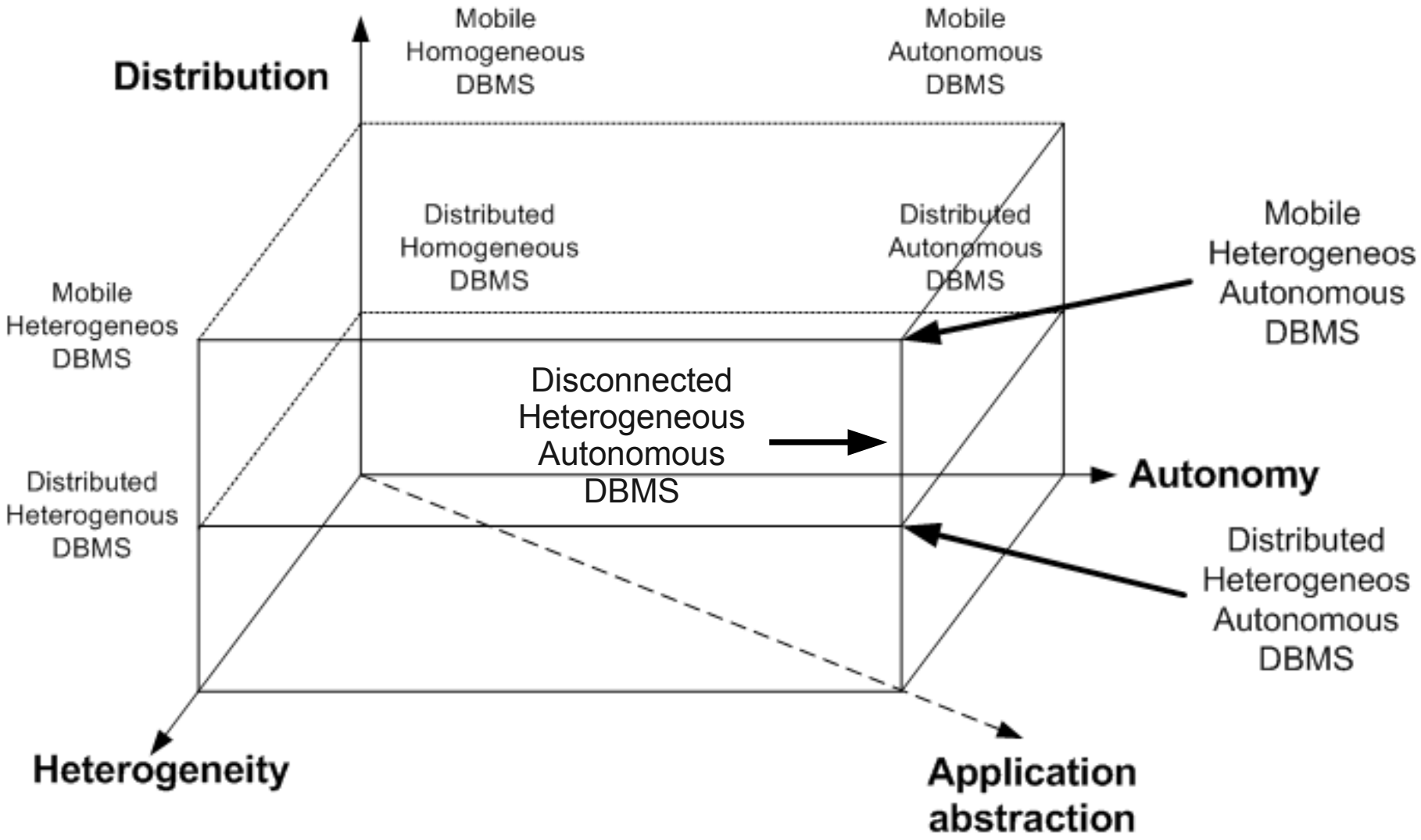
[Fox, 1999]

What makes a transaction complex?

A Scenario

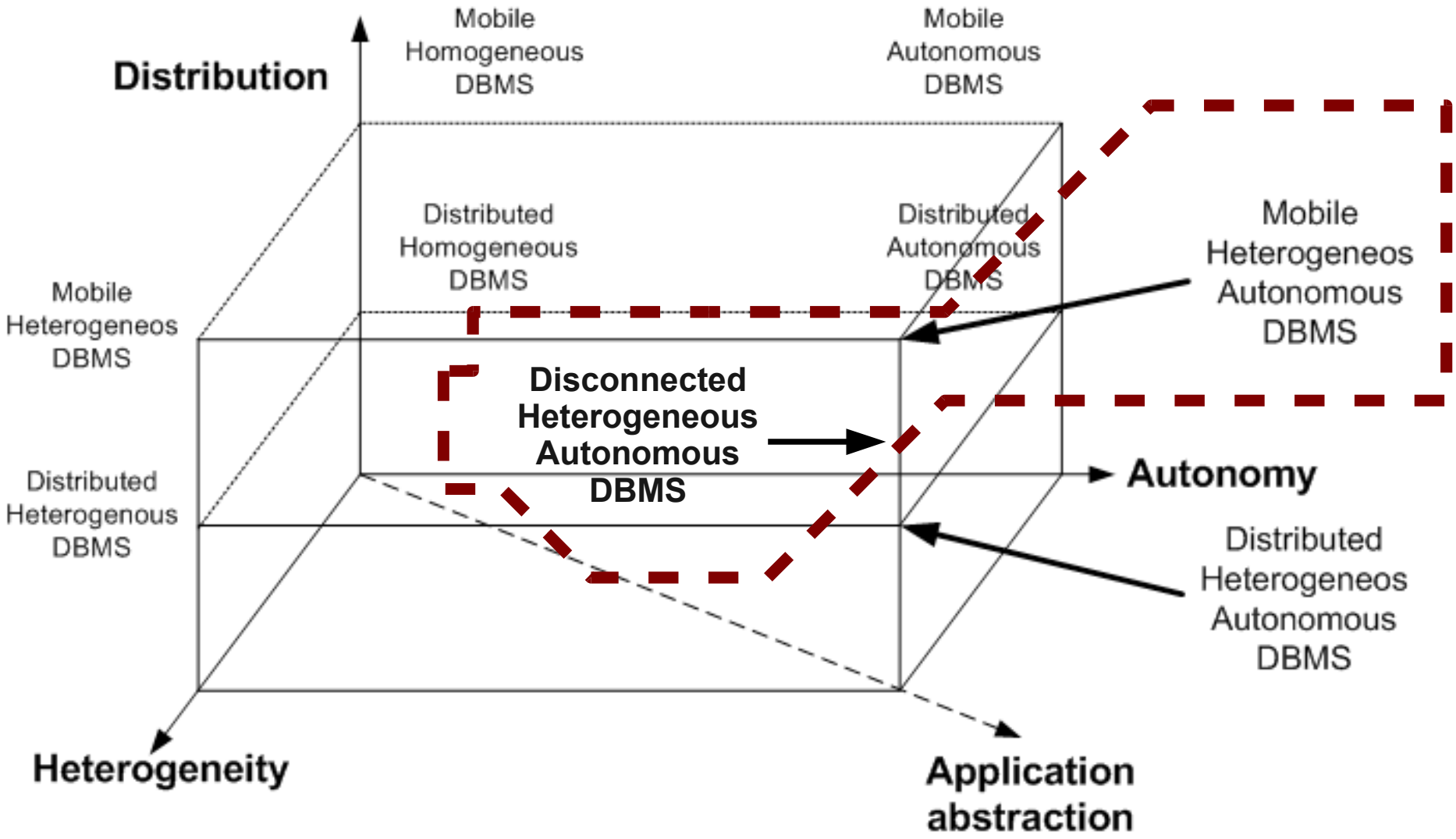

Mobile To Mobile Transactions (MANET)
Application
Mobile DB
Application
Mobile DB
Cloud (Web)
DB Node 1
DB Node 2
...
DB Node n
Messaging
Cloud Transactions
Fixed to Mobile Transactions
Workflow Transactions
Flat (Distributed) Transactions
Complex Transaction
Organisation II
Global Access Layer (global Schema)
DB Node 1
DB Node 2
...
DB Node n
Organisation I
DB Node 2
DB Node 1
DB Node n

Database Layer Dimensions



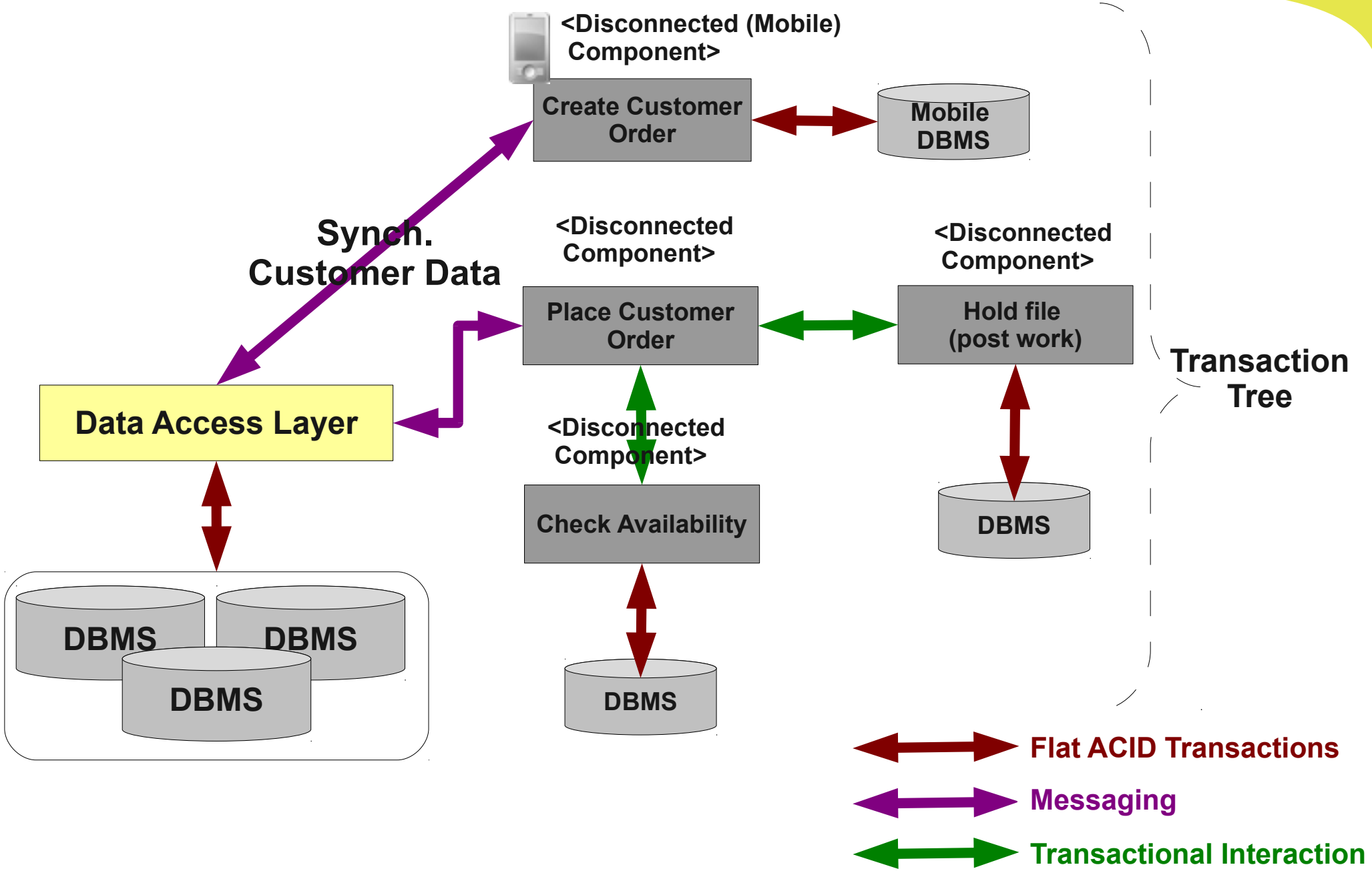
[Dunham, 1997]

Database Layer Dimensions

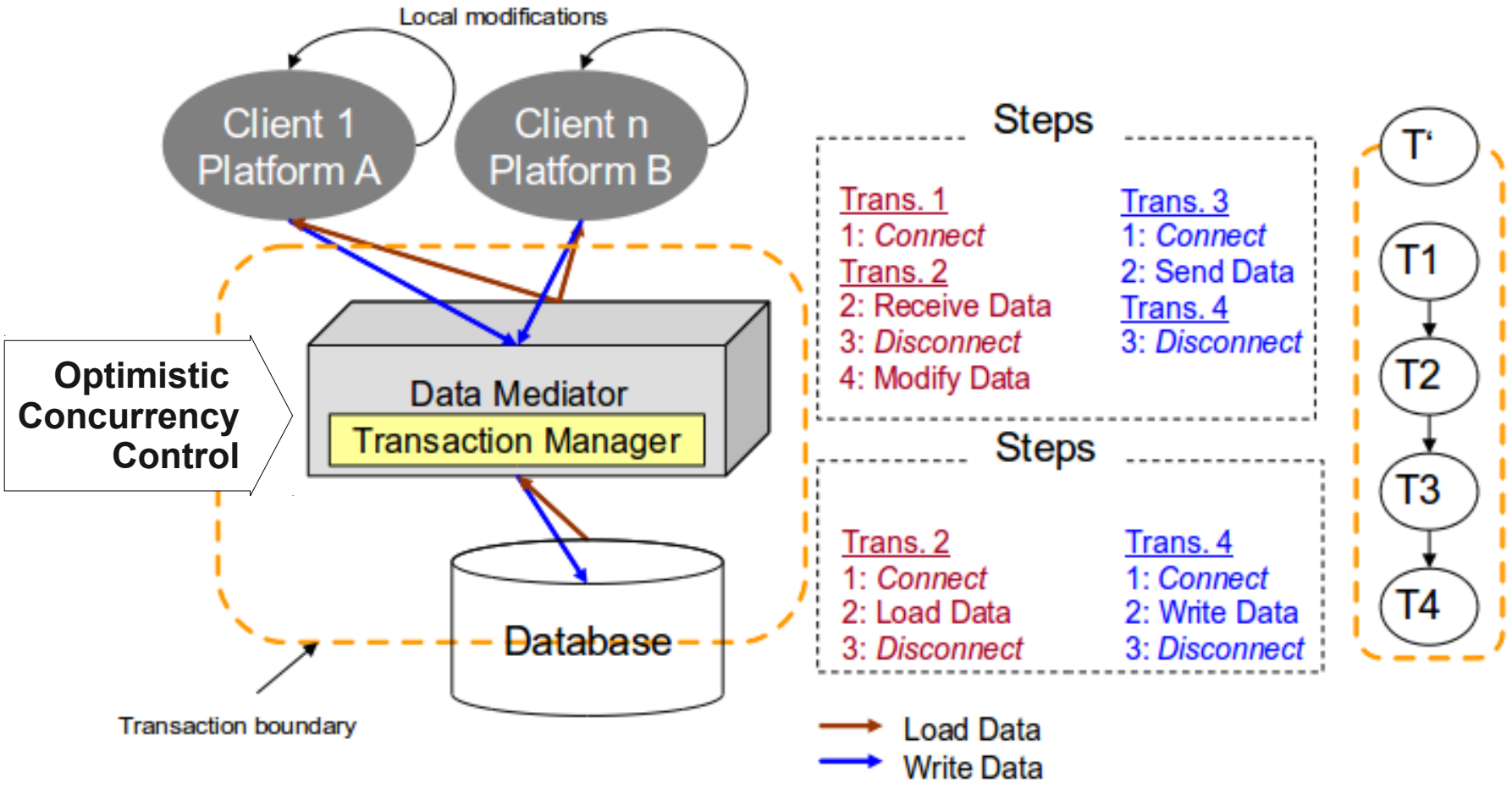


[Dunham, 1997]

Middleware & Application Layer
Disconnected Components

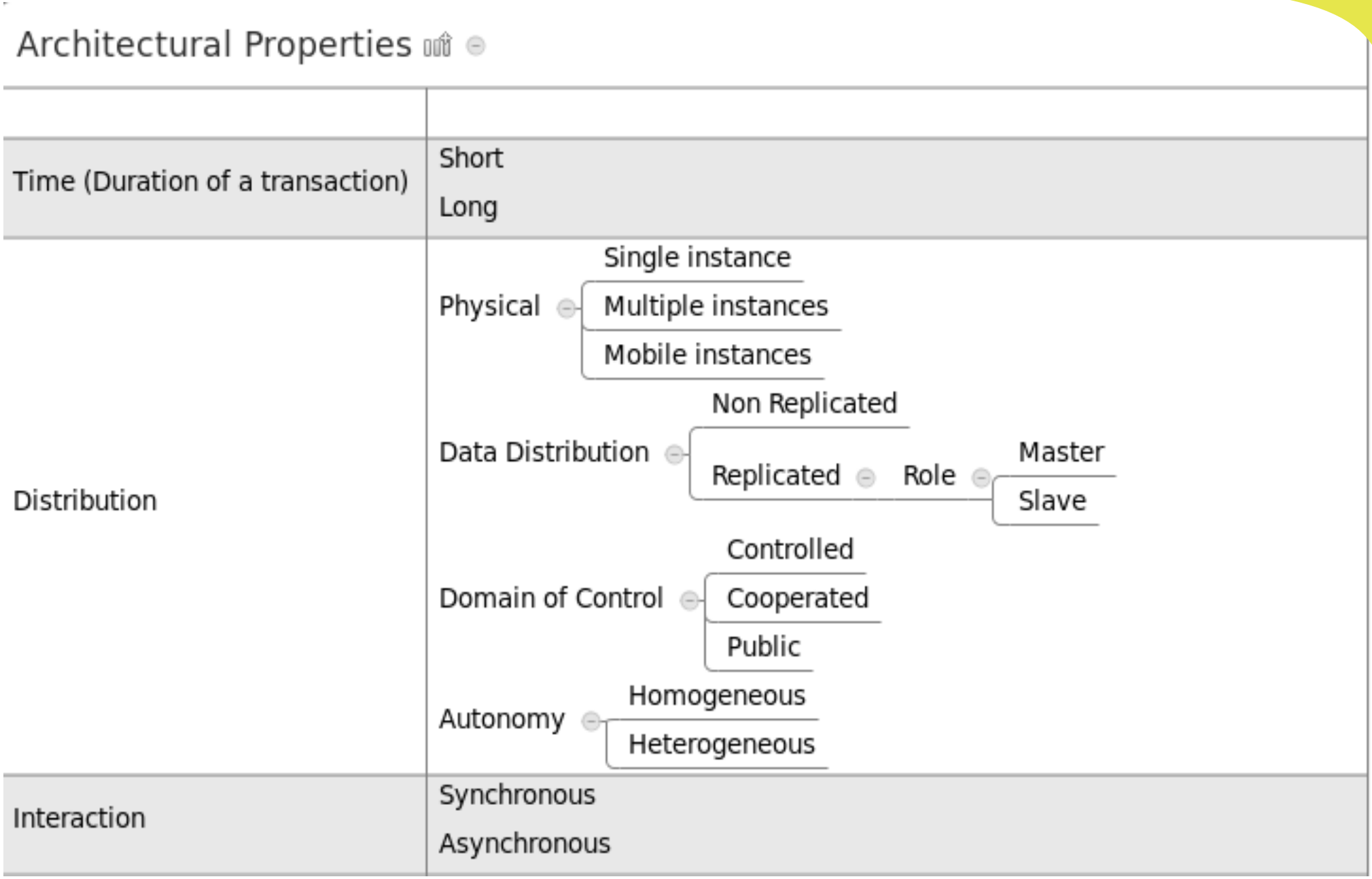


Disconnection

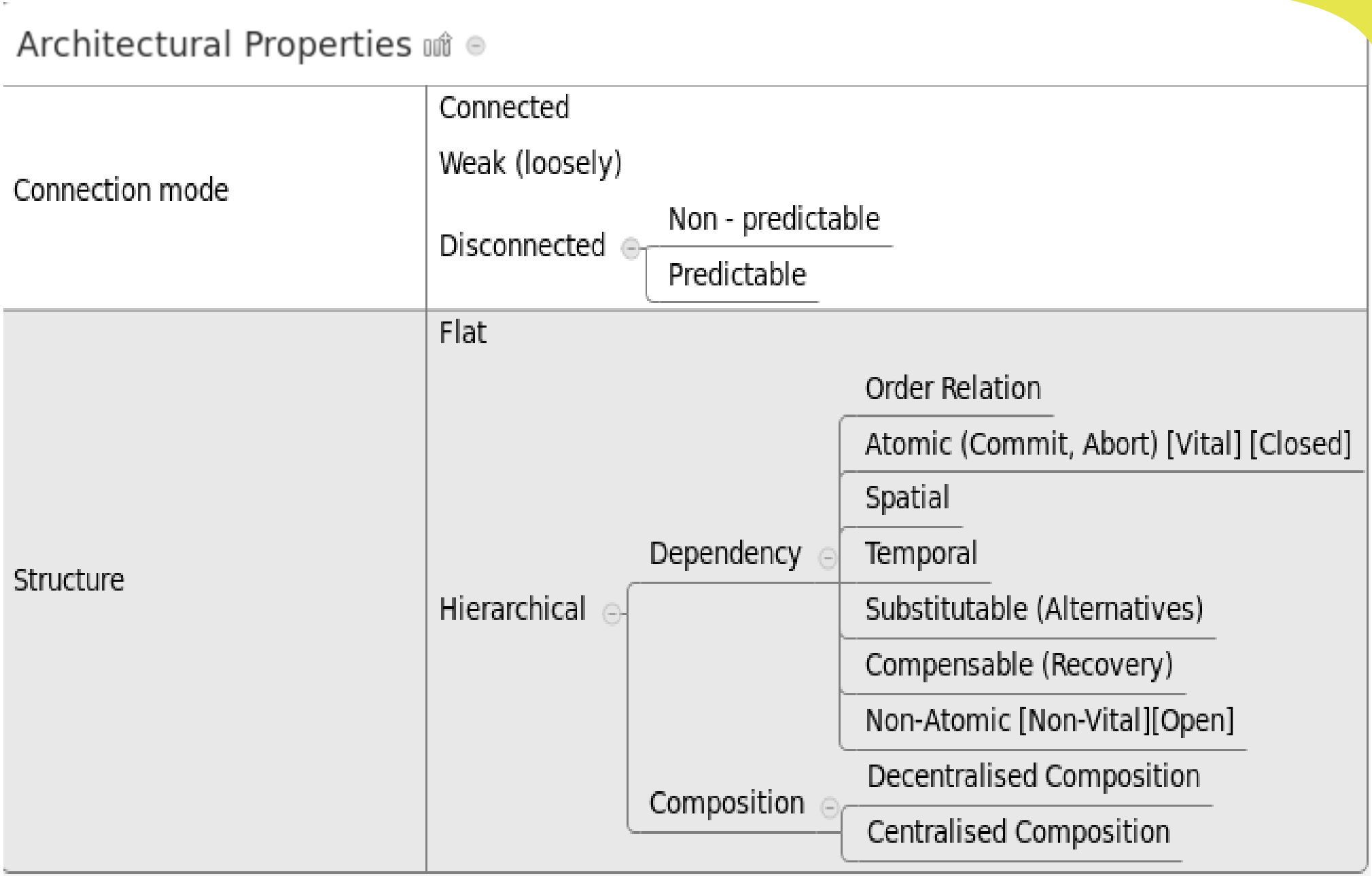


→ *Notice: communication is asynchronous!*

Architectural Properties

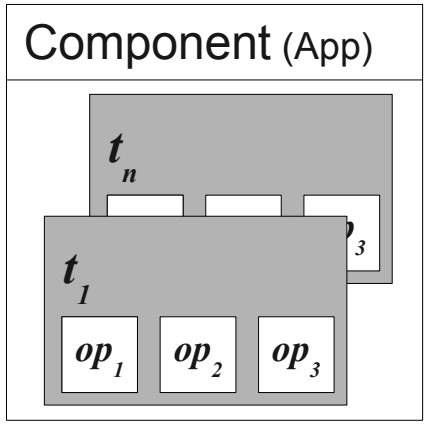


Architectural Properties

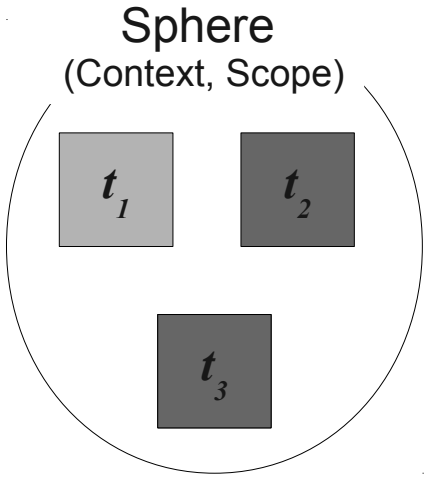


Some concepts and transaction models

Concepts to structure a complex transaction

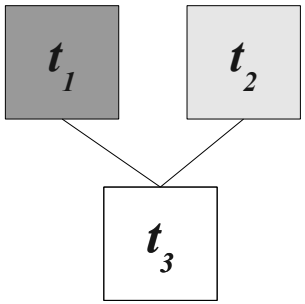


Flat transactions as implemented by a **component**

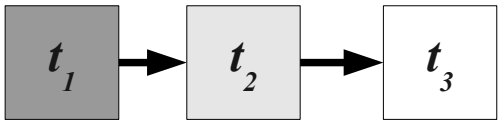


Sphere as an “abstract” construct to structure (group) transactions

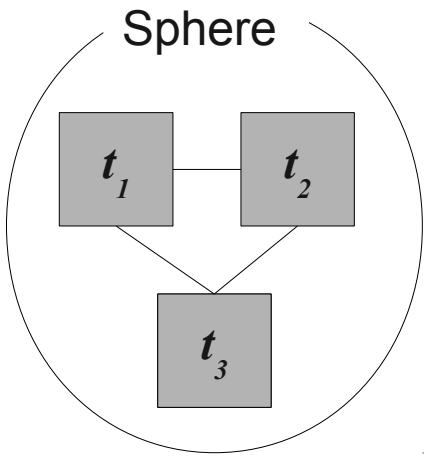
Concepts to structure a complex transaction



Nested structure: Dependencies as a concrete **hierarchical structure**,
e.g., commit dependencies → atomic unit

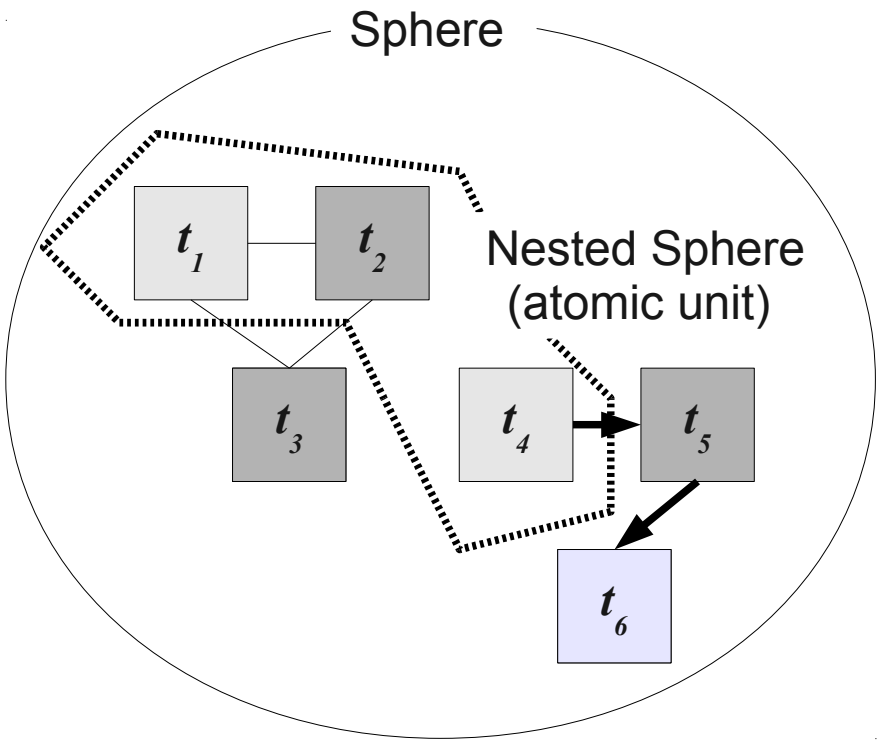


Explicit **execution order** for transactions



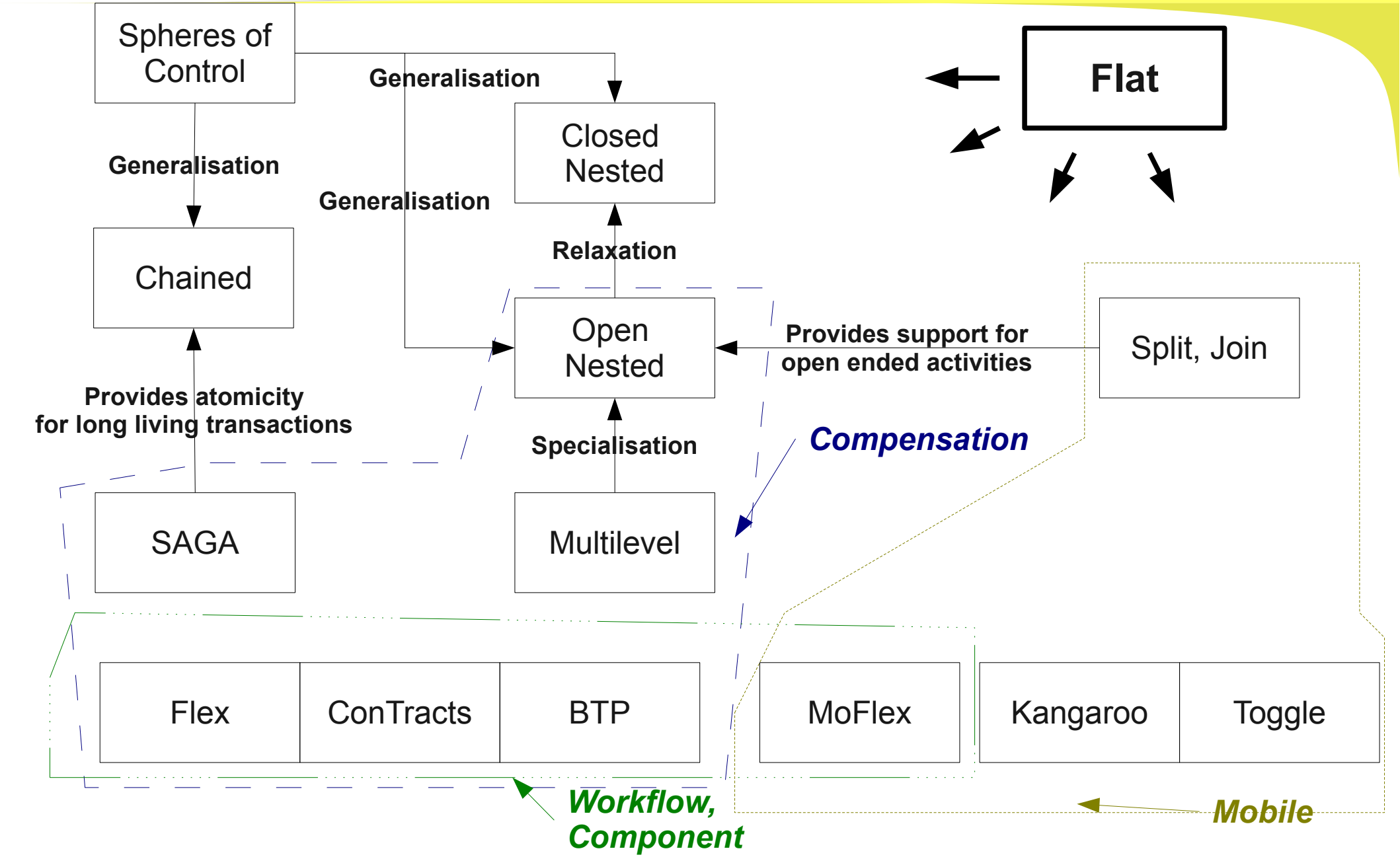
Sphere and dependencies “concrete” and “abstract” structure

Concepts to structure a complex transaction



Nested spheres and hierarchically structured (nested) transactions

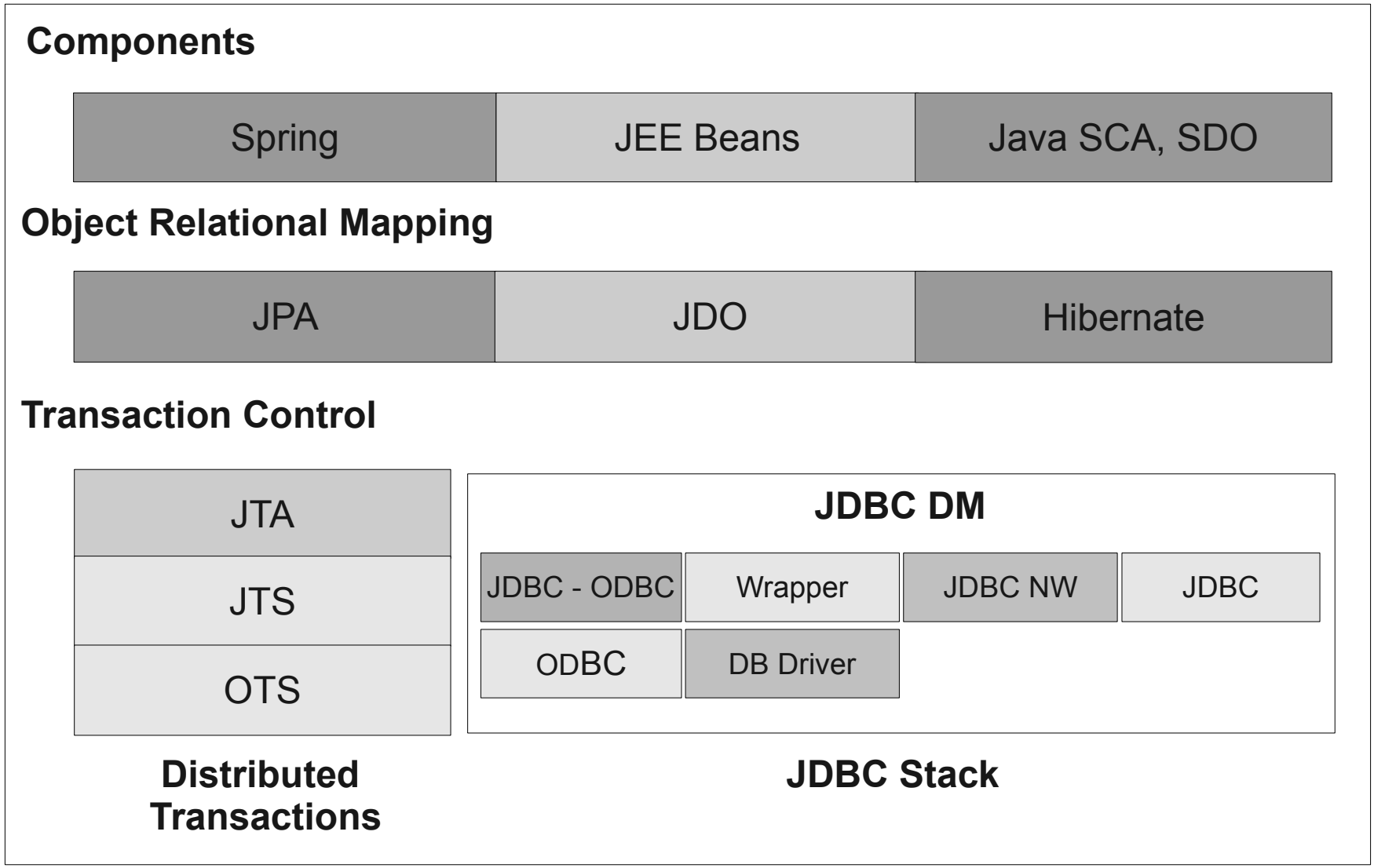
Important Advanced Transaction Models
A rough overview



[Davies, 1978],[Weikum, 1992],[Garcia-Molina, 1987],[Elmagarmid, 1992],[Reuter, 1997],[Leymann, 1995],[Wang, 2008],[Waechter, 1992]

Technologies: Data Access and Transactional Interaction *(Transaction Propagation)*

The Java World
An Extract



[JEE],[Spring],[JTA, 2002],[SCA, 2006]

The Java World
Data Access & Transaction - Control

"Own Code" Example

```
create connection()
begin transaction
...
  update table-a
...
if (condition)
  commit transaction
else if (condition)
   update table-b
   commit
else
   rollback

begin transaction
   update table-c
commit
```

JTA Example (ORM)

```
EntityManager em = createEntityManager();
em.getTransaction().begin();
Employee employee = em.find(Employee.class,
id);
employee.setSalary(employee.getSalary() +
1000);
em.commit();
em.close();
```

Spring Example (ORM, Declarative)

```
@Transactional(readOnly = true)
public class DefaultFooService implements
FooService {
    public Foo getFoo(String fooName) {
        // do something
    }
@Transactional(readOnly = false)
    public void updateFoo(Foo foo) {
        // do something
    }
}
```

[JEE],[Spring],[JTA, 2002]

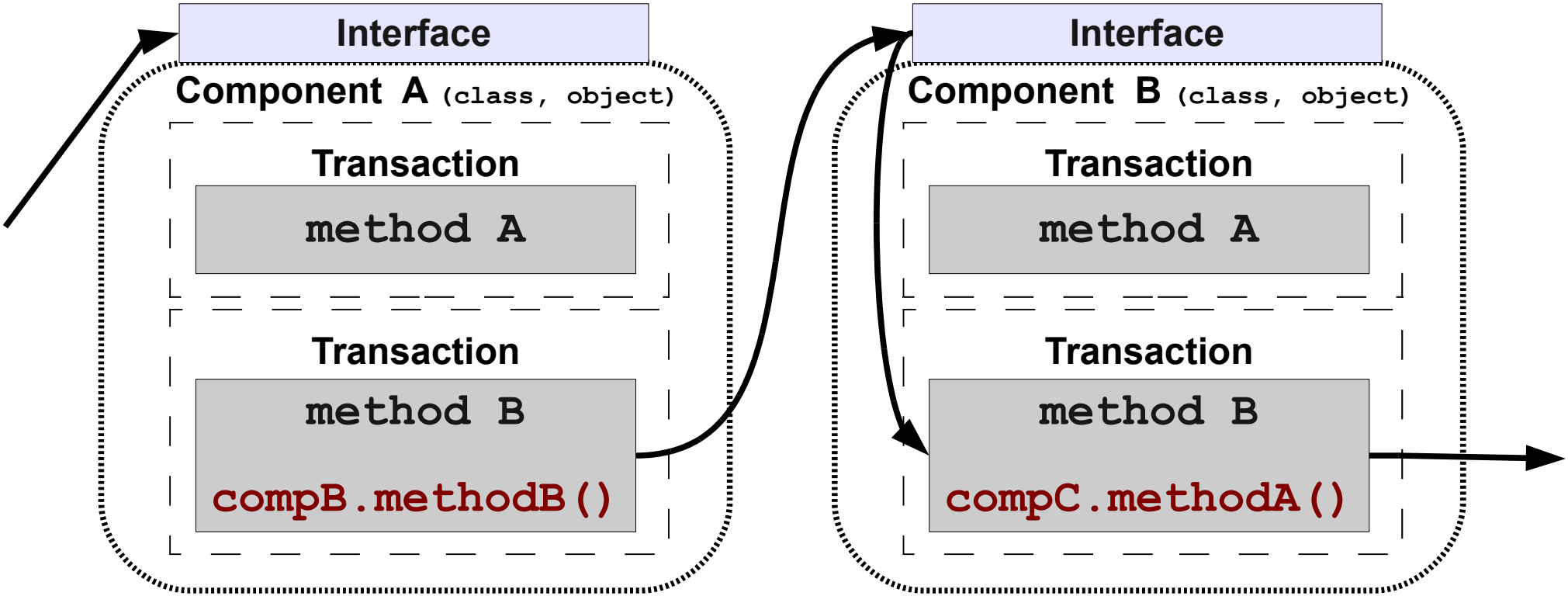
The Java World
Declarative Transaction – Control *(Rollback)*

Spring example (declarative exception and rollback handling)

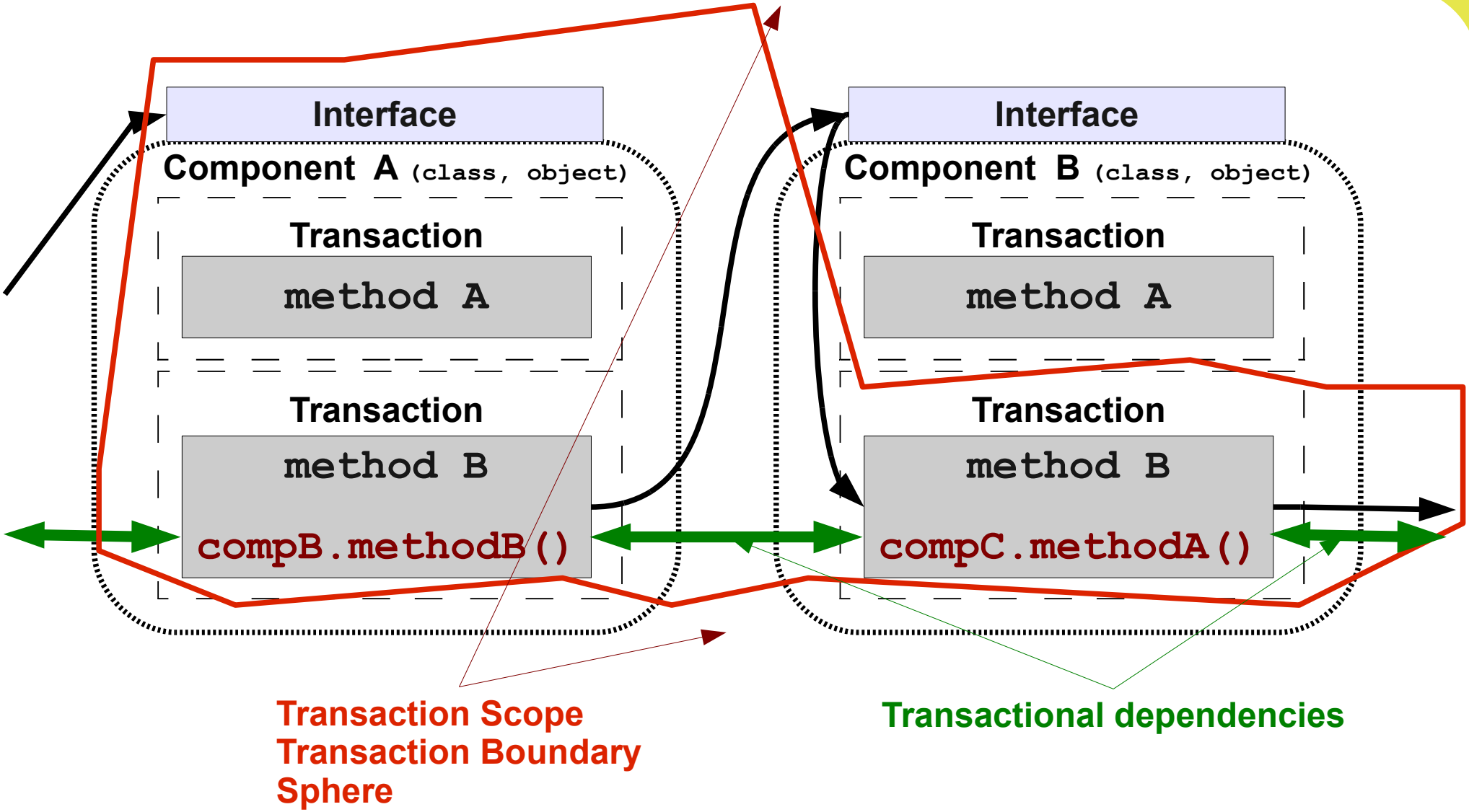
```
<tx:advice id="txAdvice" transaction-manager="txManager">
  <tx:attributes>
        <tx:method name="get*" read-only="false"
        rollback-for="
           NoProductInStockException,
           InvalidProductIDException"/>
        <tx:method name="*"/>
  </tx:attributes>
</tx:advice>
```

[Spring]

Component Composition
Transaction Propagation

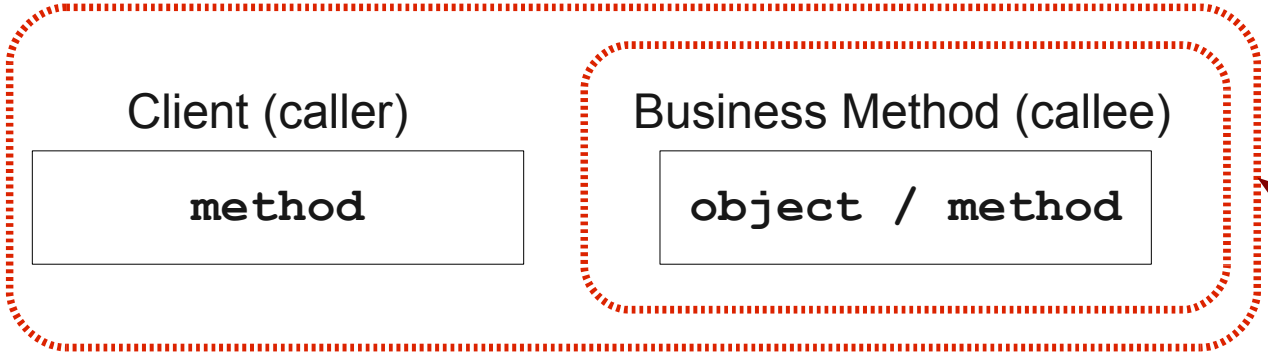


Component Composition
Transaction Propagation



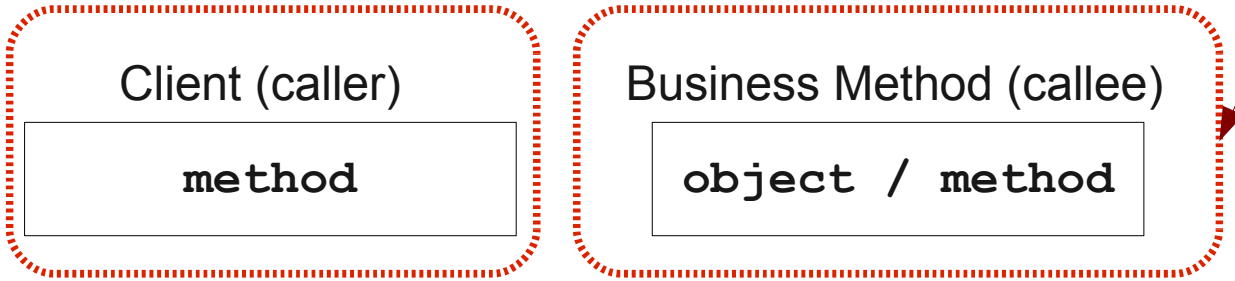
The Java World (JEE)
Declarative Transaction Propagation

Callee: `TransactionAttributeType.REQUIRED`



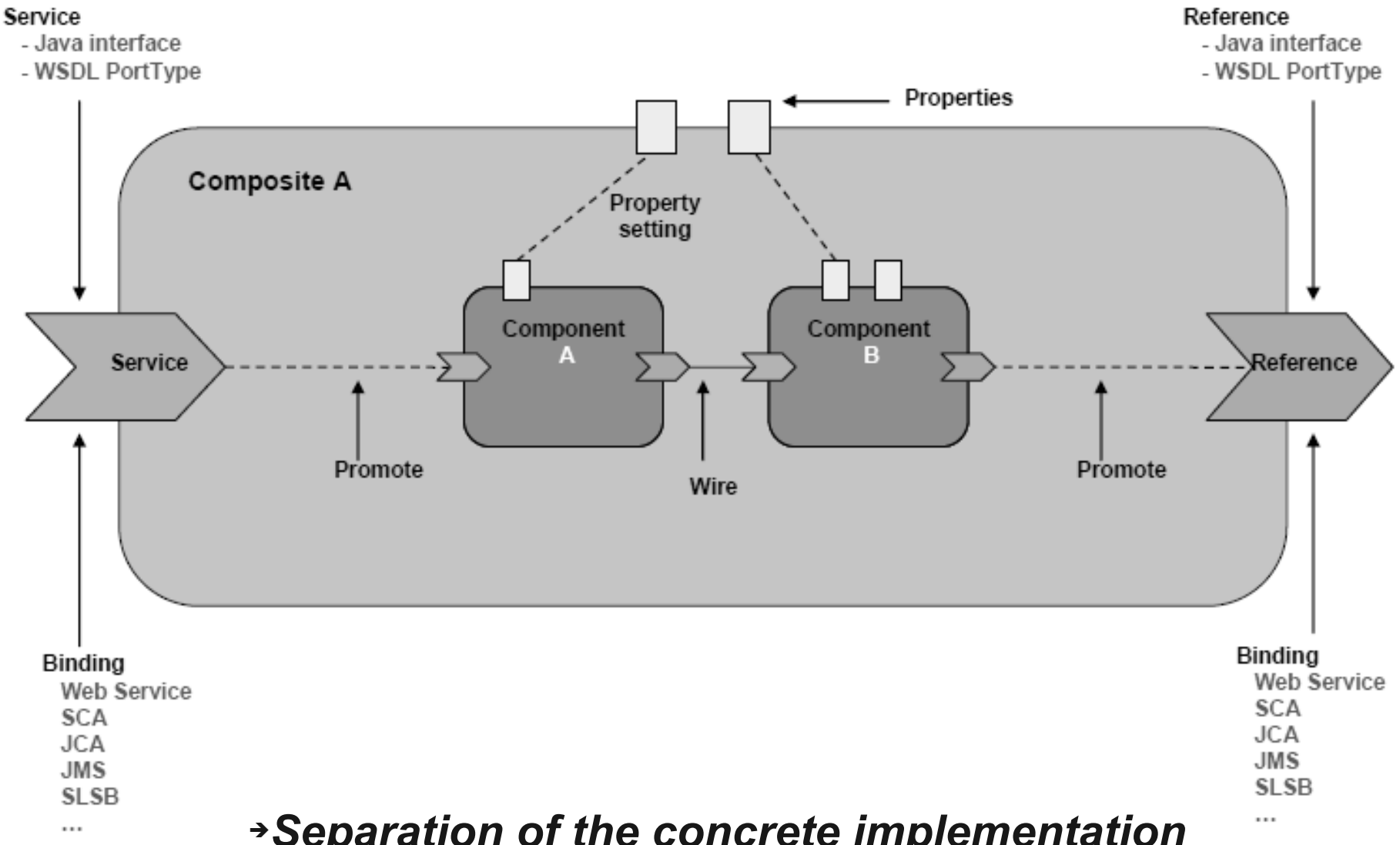
Transaction Scope
Transaction Boundary
Sphere

Callee: `TransactionAttributeType.REQUIRES_NEW`



1. Suspends client transaction (Thread)
2. Starts a new transaction (Thread)
3. Delegate call
4. Resume client transaction (Thread) after the method completes

Service Component Architecture (SCA)

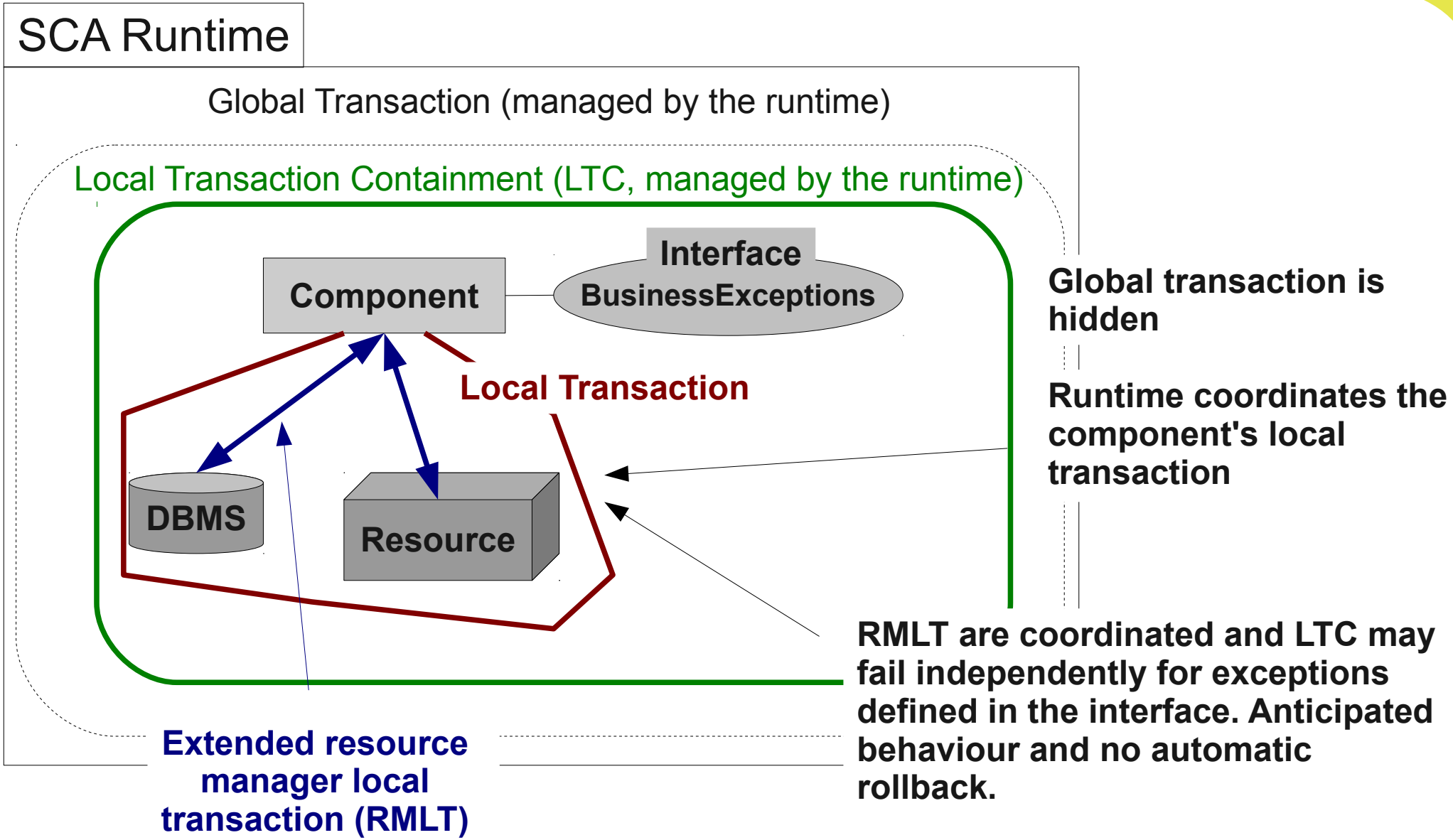


→ *Separation of the concrete implementation*

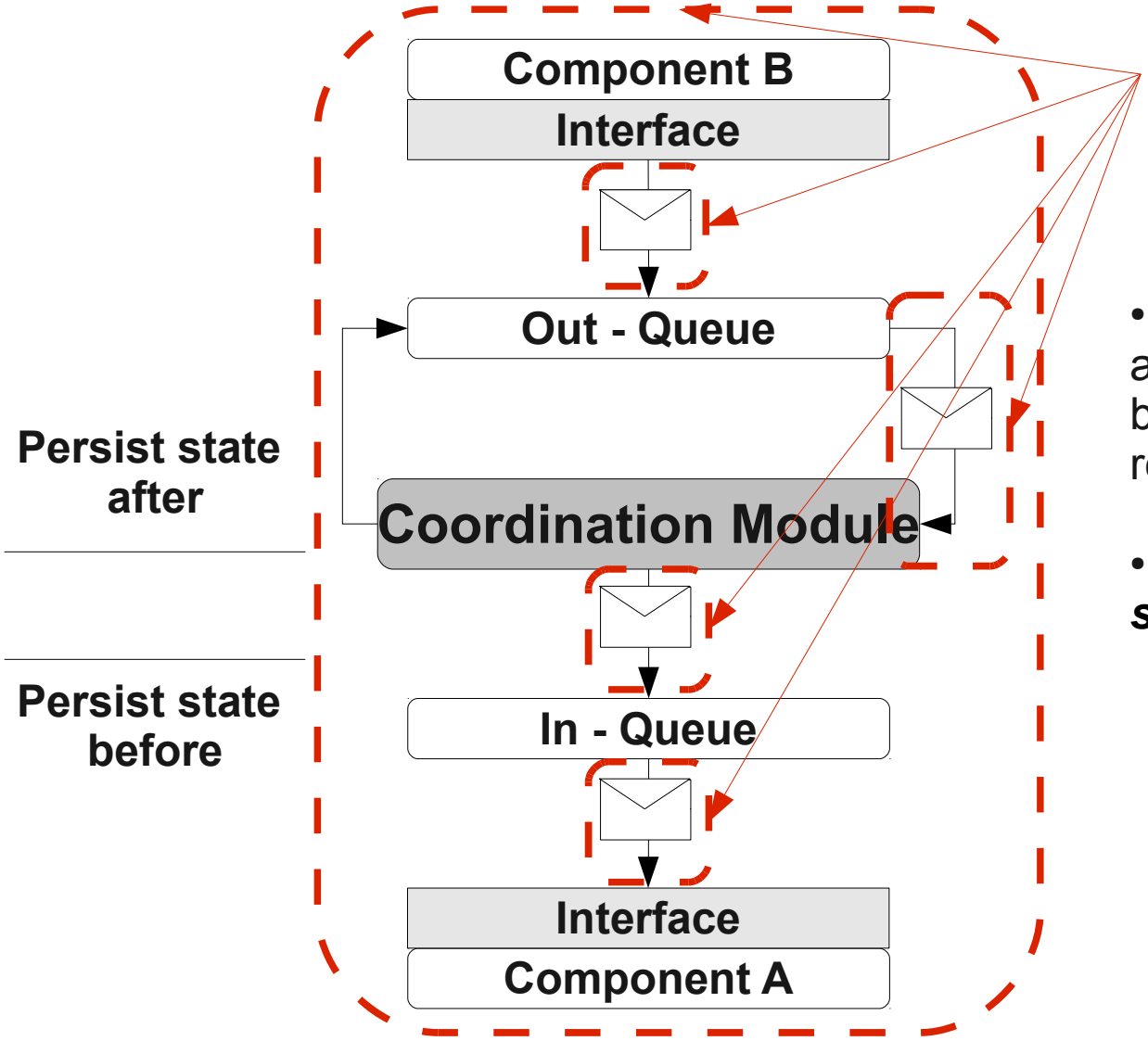
→ *Higher Abstraction*

[SCA, 2006]

SCA Transaction Policy
Example: `managedTransaction.local`



SCA Transaction Policy Example: `transactedOneWay`

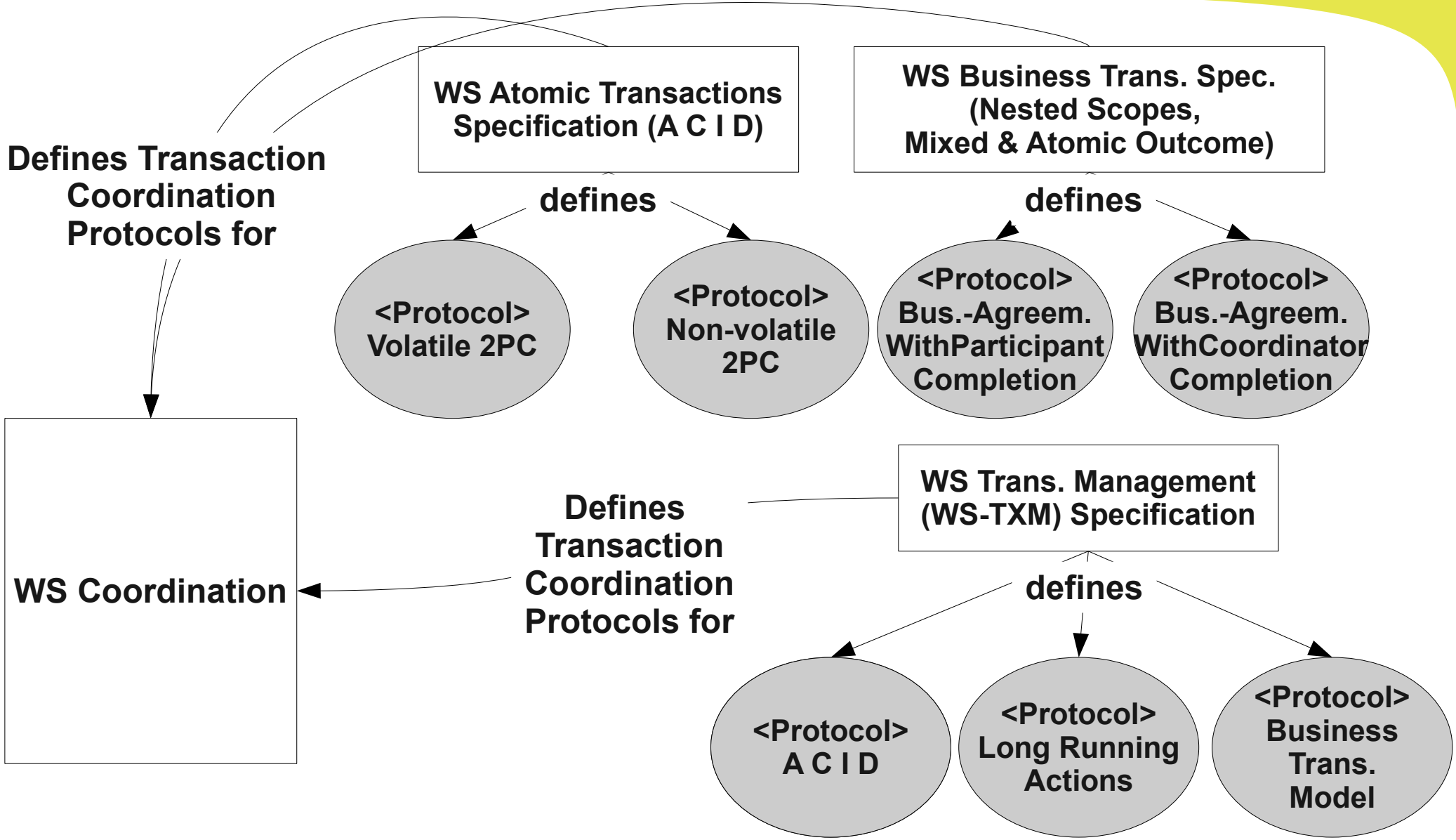


Transactions

- Messaging increases the amount of states that need to be made persistent due to recovery.
- ***In a bad, but safe way 4 safe-points per invocation!***

[Zimmermann, 2007]

Web Service (WS) Transaction Coordination
An extract



[Newcomer, 2007],[Newcomer, 2007a],[Newcomer, 2007b]

Messages so far?

Focus was on the transactional interaction!

Existing technologies abstract the data access *(sufficient? ~ Yes)*

Existing technologies allow to define the transactional interaction behaviour (~Atomicity and Isolation) of components *(sufficient? ~ Yes)*

Messages so far?

Existing technologies do not allow to define consistency requirements on a component base ***(problematic? ~YES)***

Composition of transactional properties is complicated ***(Metrics are required)***

Is a higher level classification of transactions valuable and necessary? ***(~YES)***

Our Research

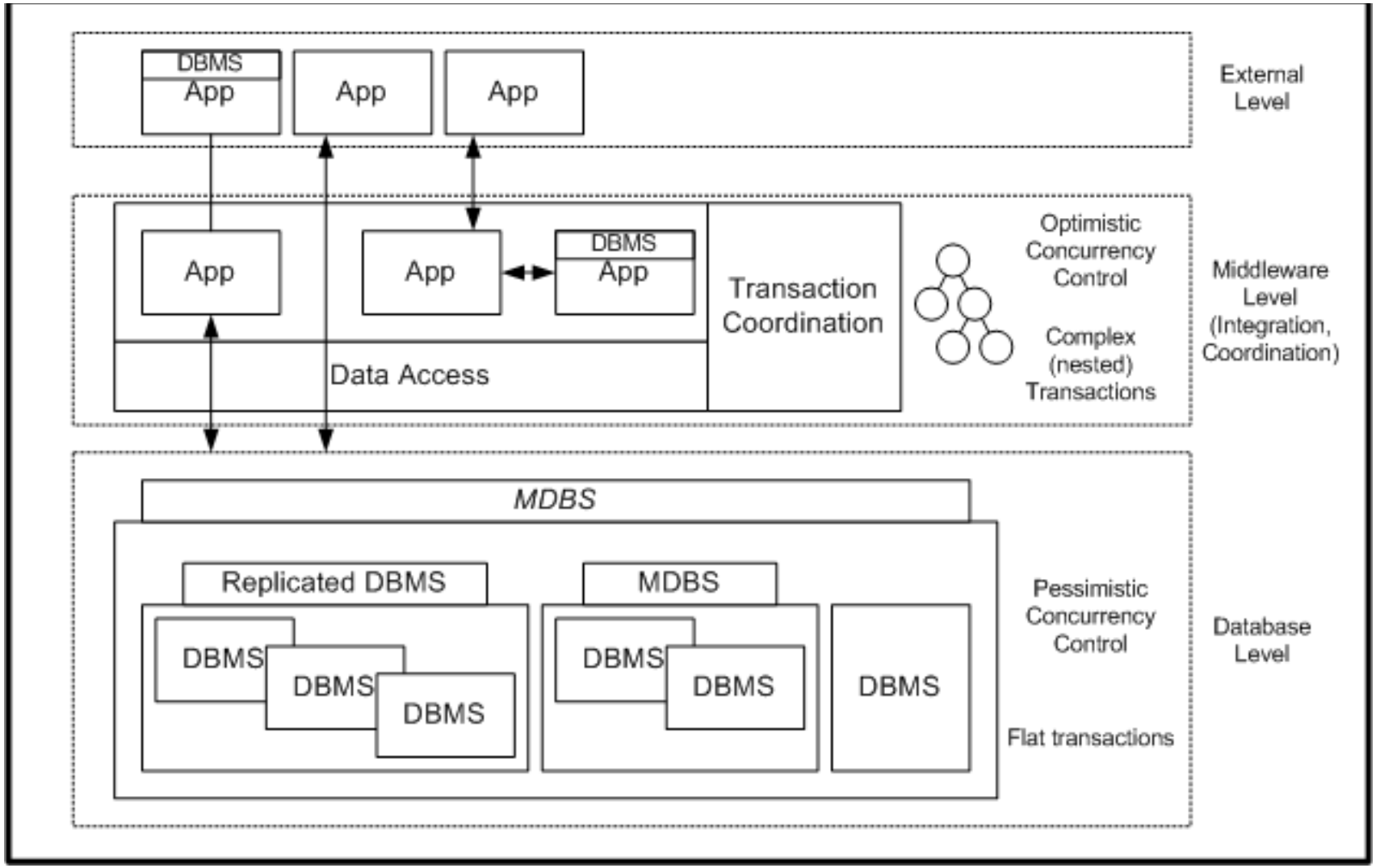
Focus of our research

Scenario based trade-off decisions between

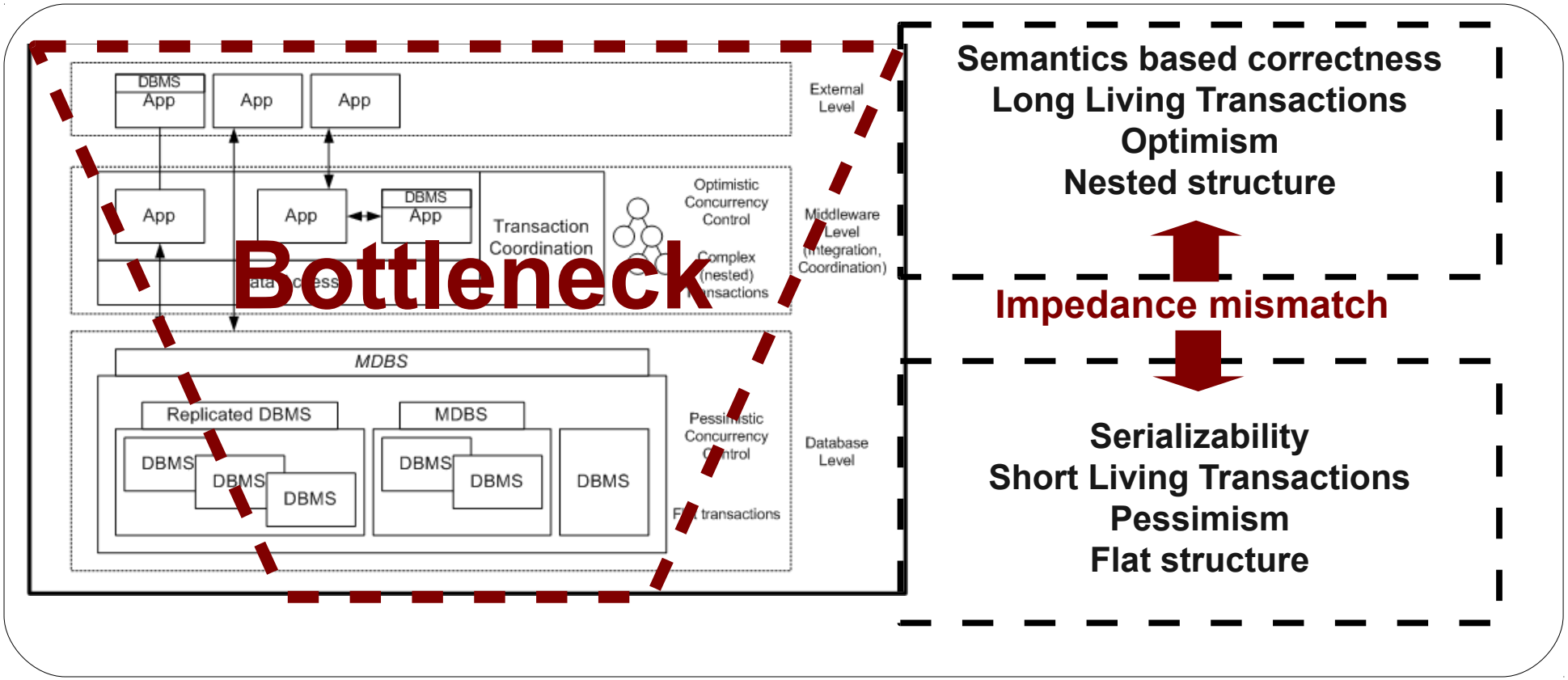
1) consistency,
2) failure-handling, and
3) availability

for the transactional integration of components in heterogeneous disconnected computing (considering all layers).

Transactional Integration

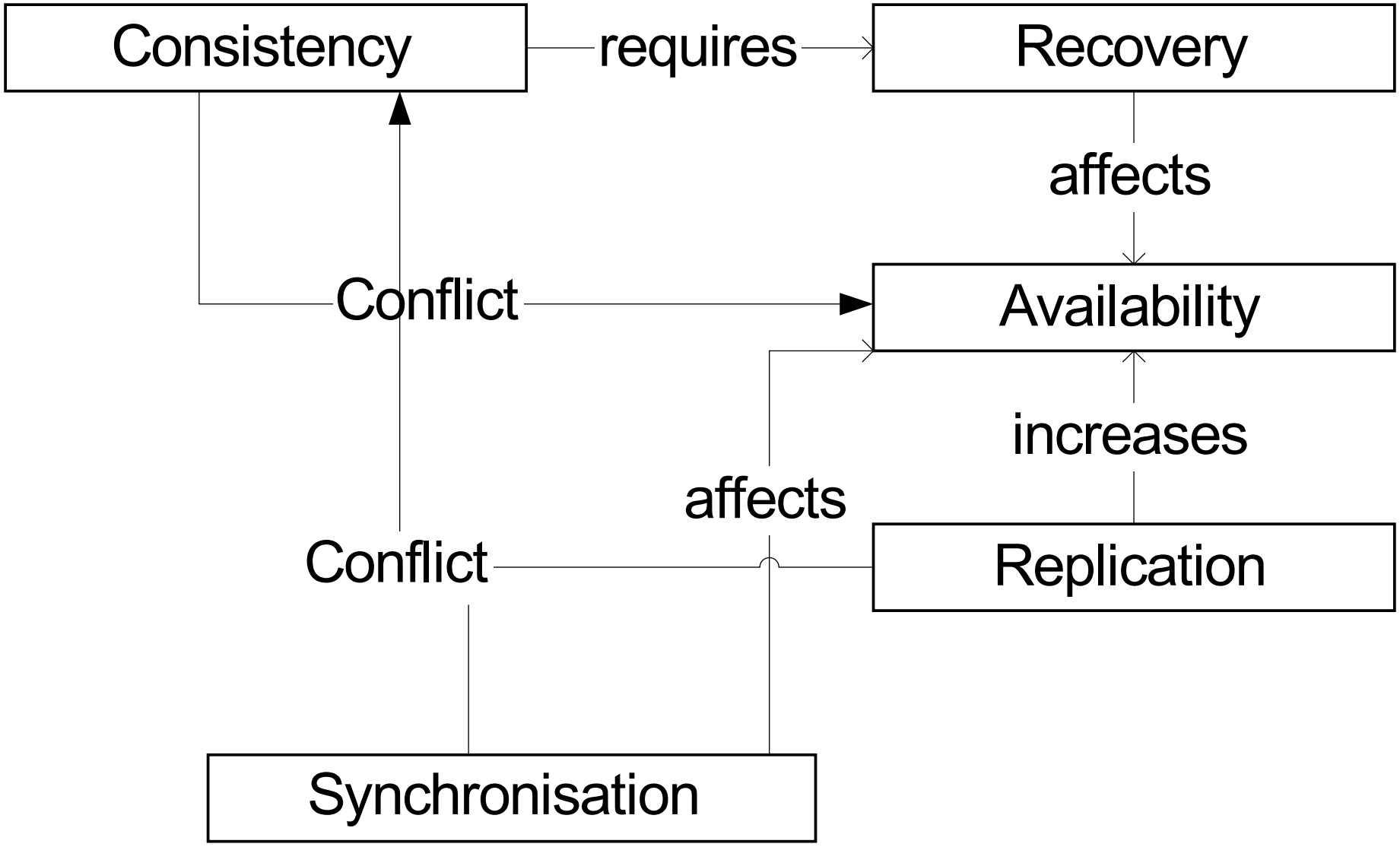


Transactional Integration

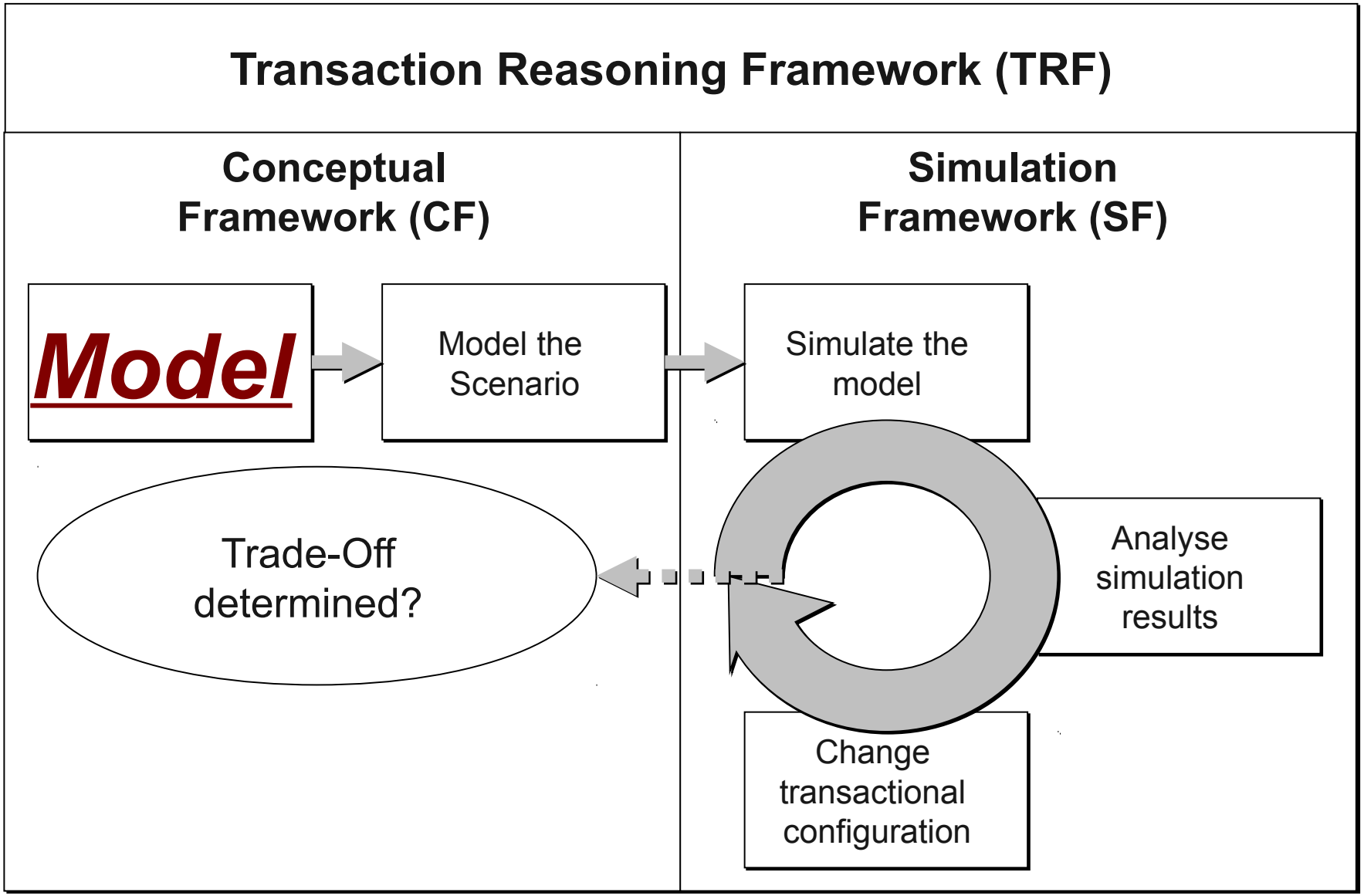


Provide an available (scalable), consistent (ACI), and durable (D) TM despite this bottleneck and the impedance mismatch.

General Problem
-in a nutshell-

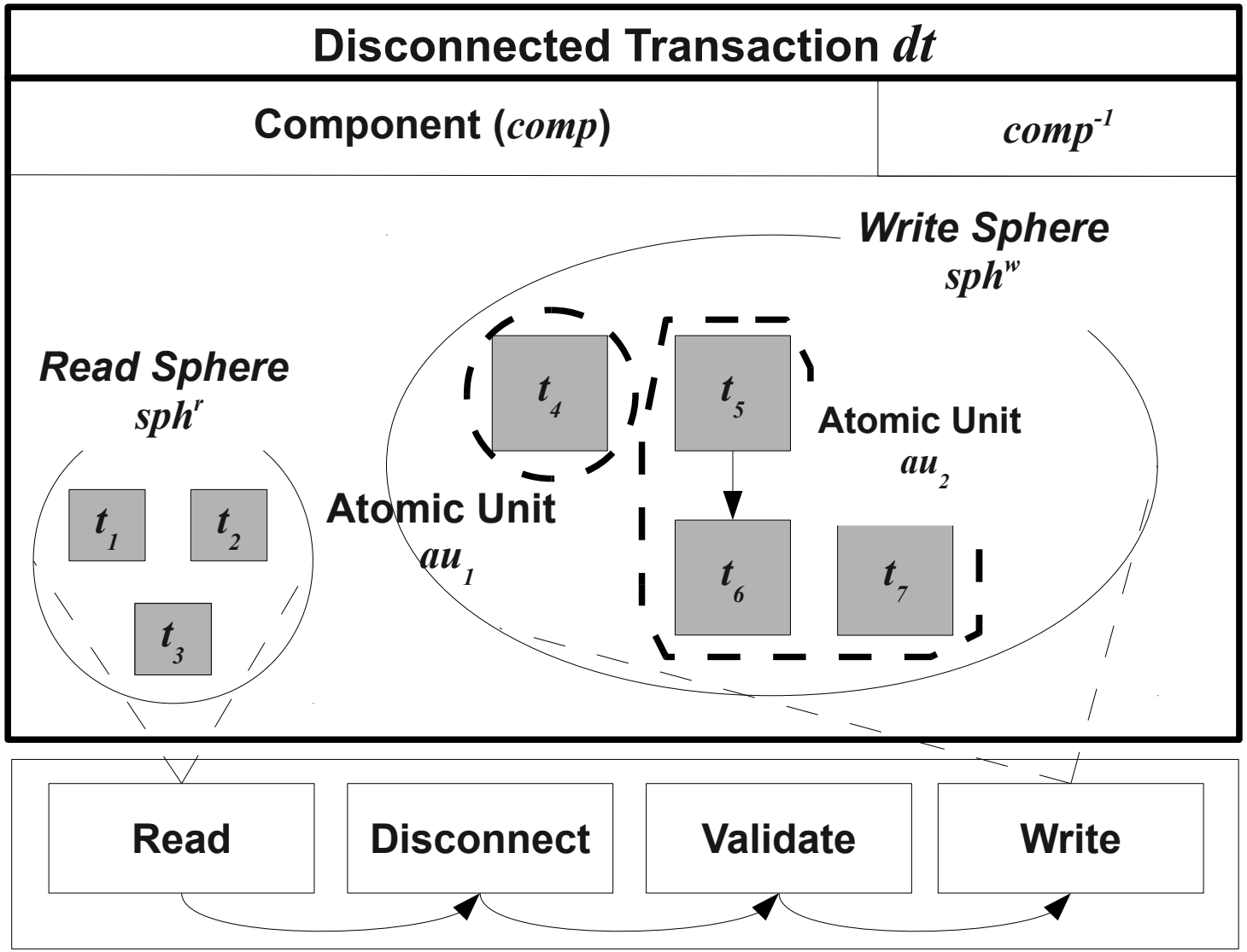


Methodology



Model of a disconnected transaction

$$dt := (sph^r, sph^w, Gdt_i, comp, comp^{-1})$$

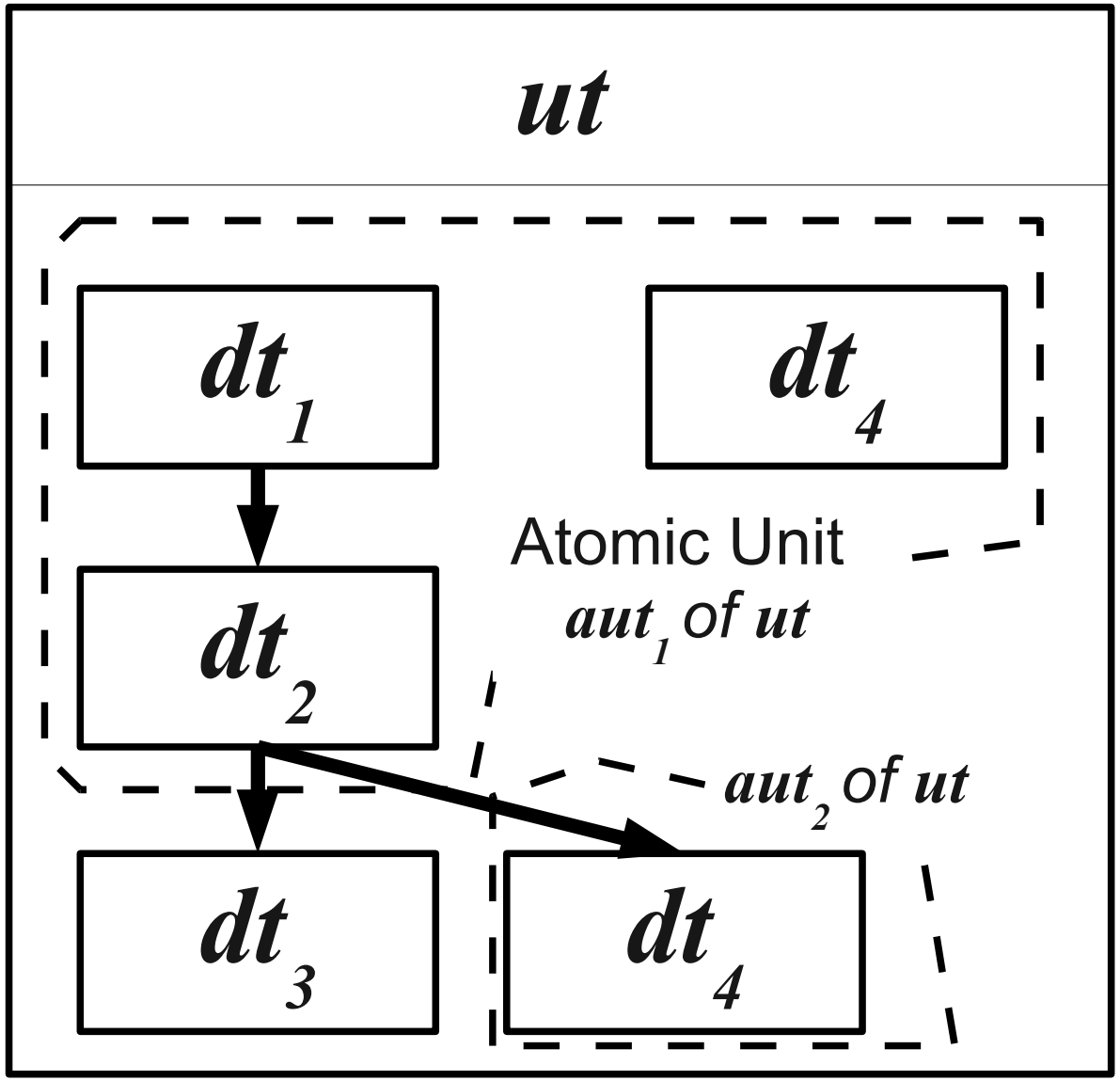


Imposed structure 1: OCC phases within a component *comp*
Imposed structure 2: atomic units of a sph^w
Imposed structure 3: execution order *Gdt* of a sph^w

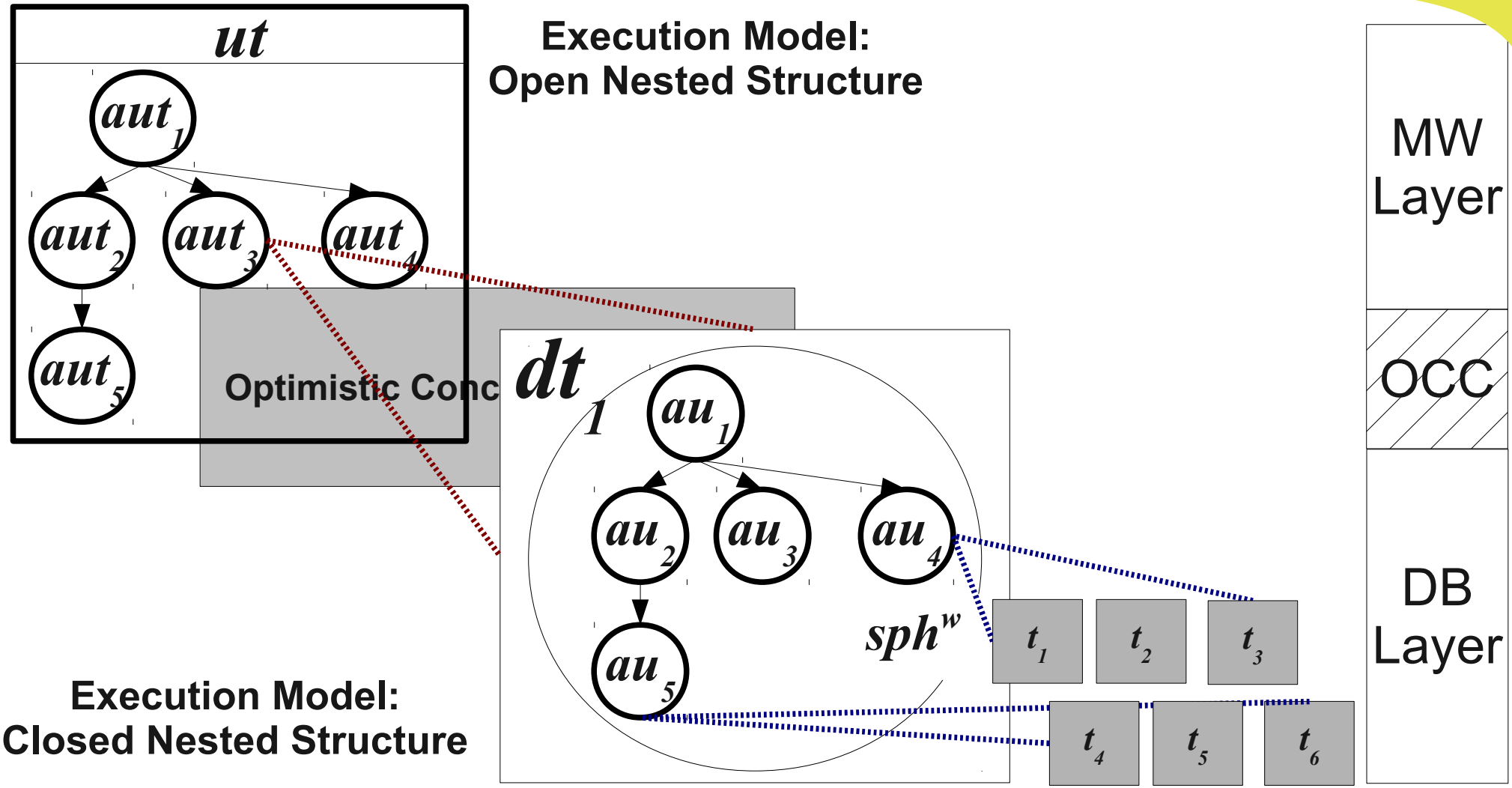
[Lessner et al. ,2011]

Model of a user transaction

$ut := (DT, Gut, AUT)$



Execution Model

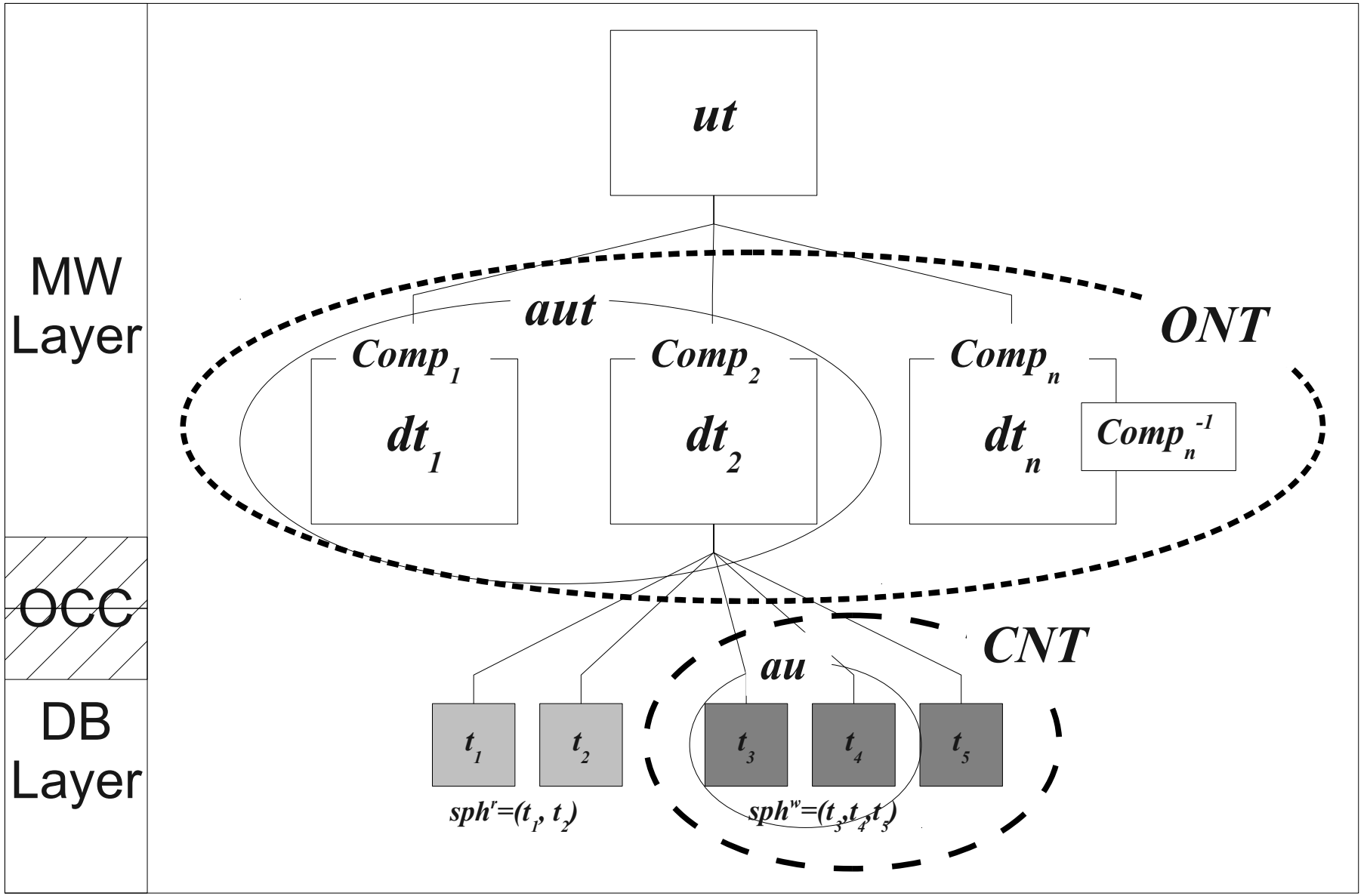


Execution Model:
Open Nested Structure

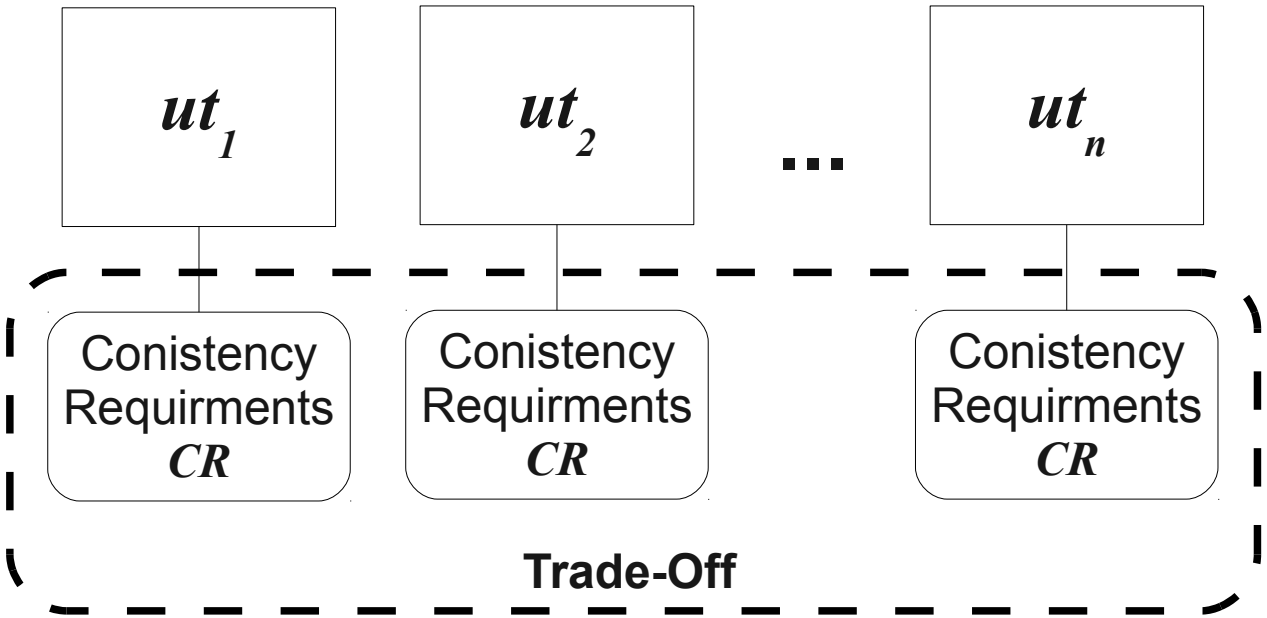
Execution Model:
Closed Nested Structure

Execution Model:
Flat ACID Transactions

Model of a disconnected transaction
The complete model



Trade-Off



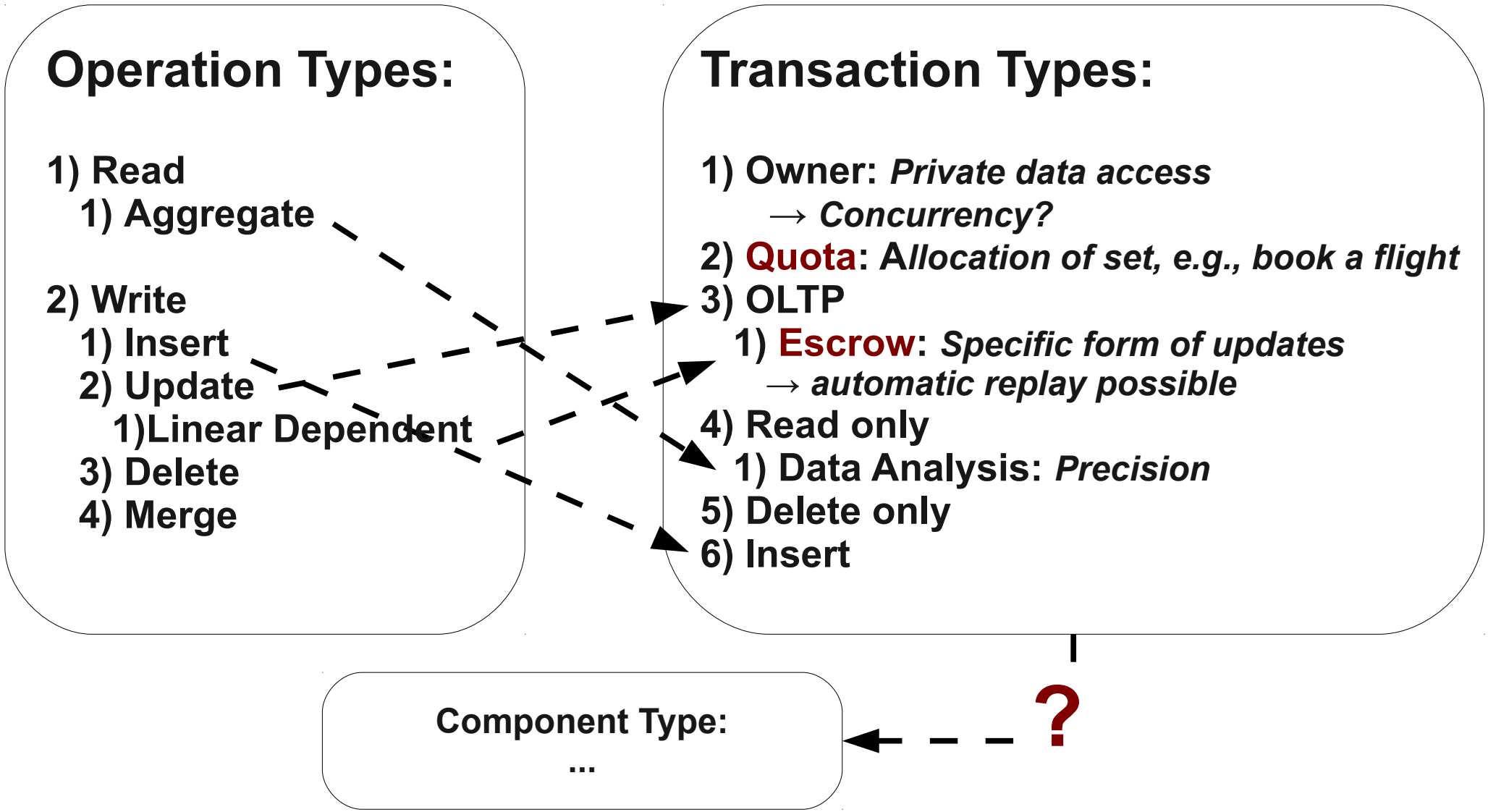
Derive costs functions from *CR*

→ ***Quantification of CR***

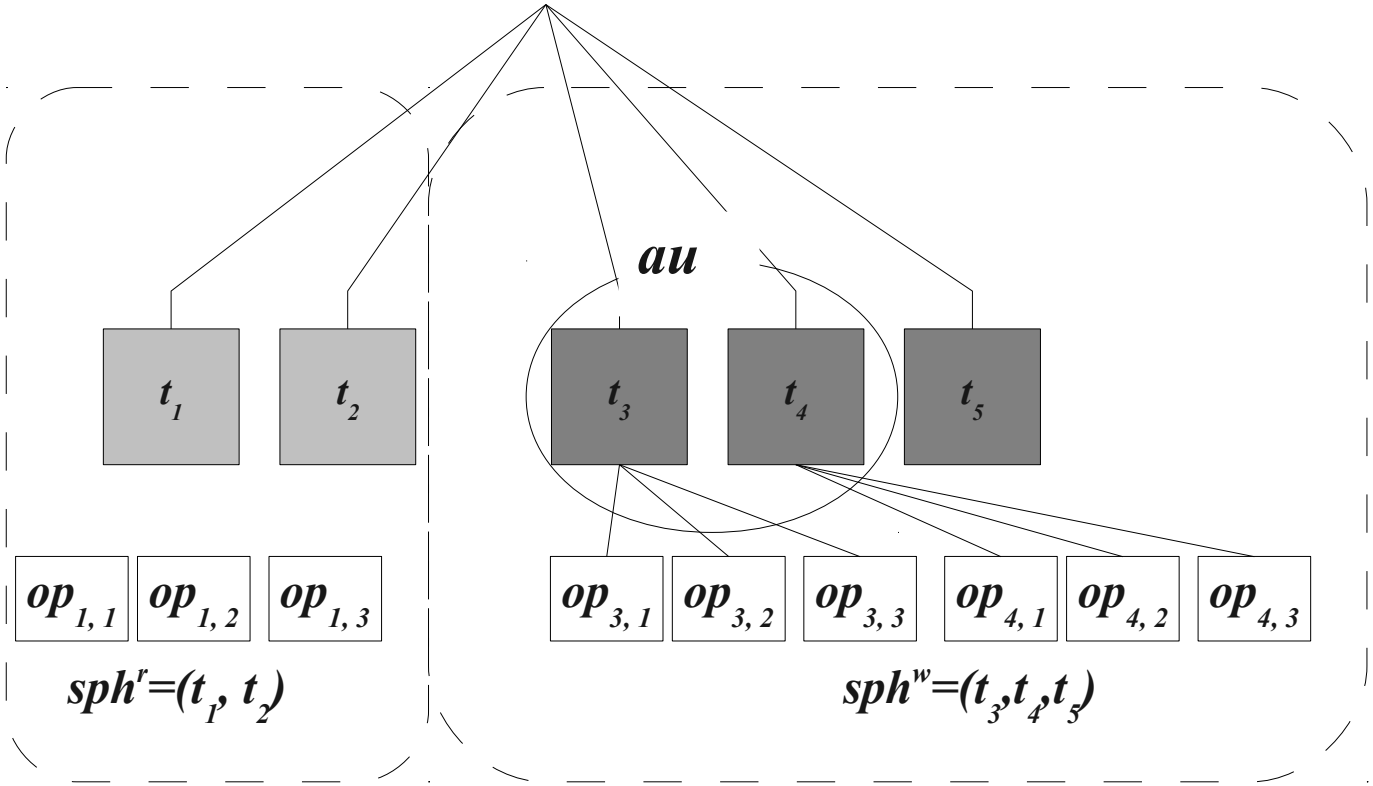
→ ***Harvest & Yield?***

Idea
Derive CR based on a transaction type

Approach:
Operations (t) → Transaction Type ($tType$) → Component Type?



Derive *CR* based on a transaction type
Problem: granularity



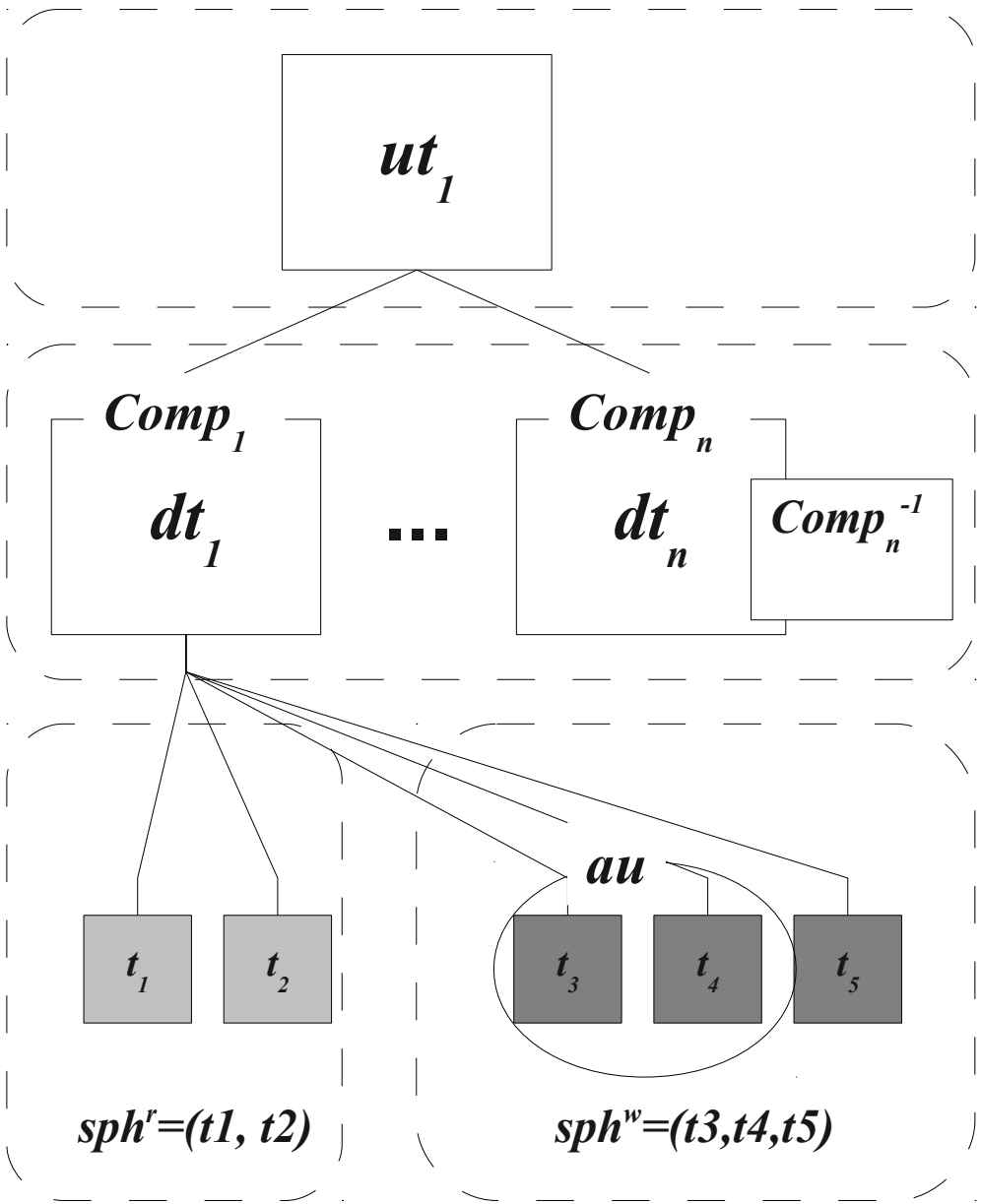
Problem:

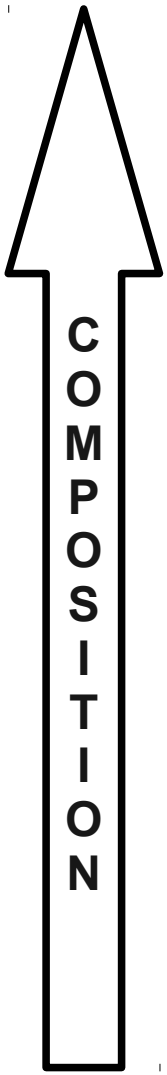
$opType(op_{3,1})=Escrow$

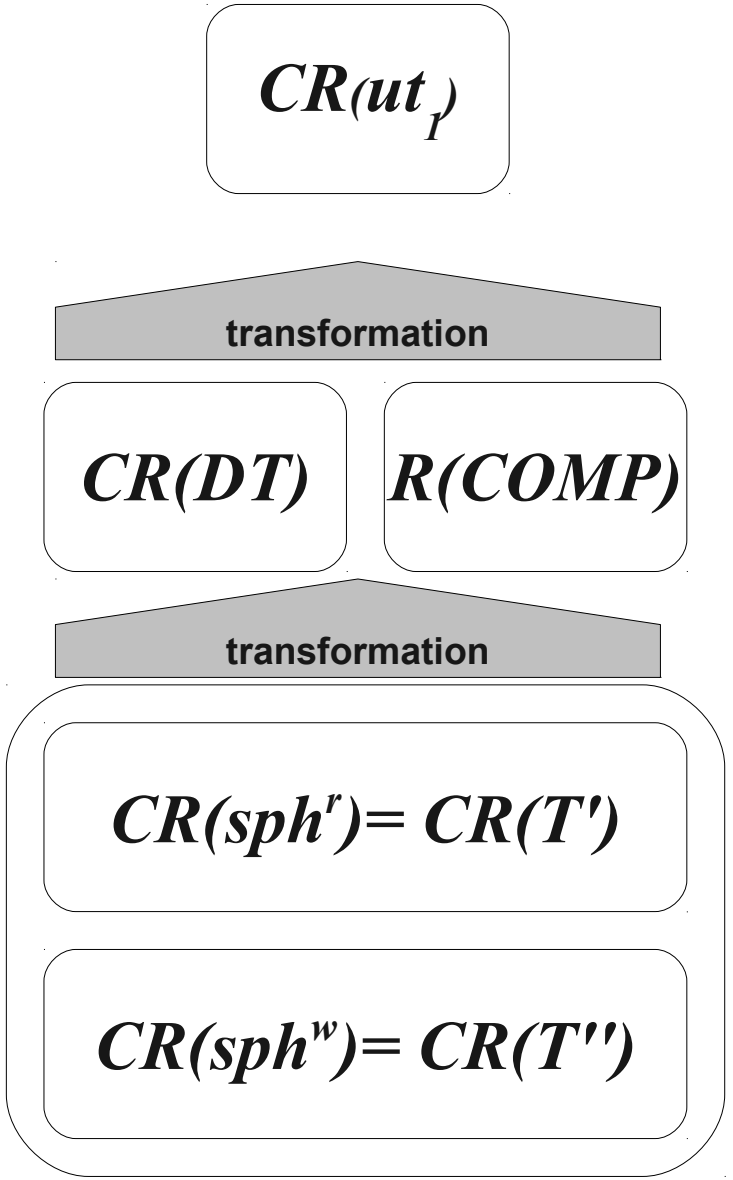
$opType(op_{3,2})=Update$

$\rightarrow tType(t_3)?$

Derive *CR* based on a transaction type
Problem: transformation







An idea to quantify Time (and order)

“Look back” in time:

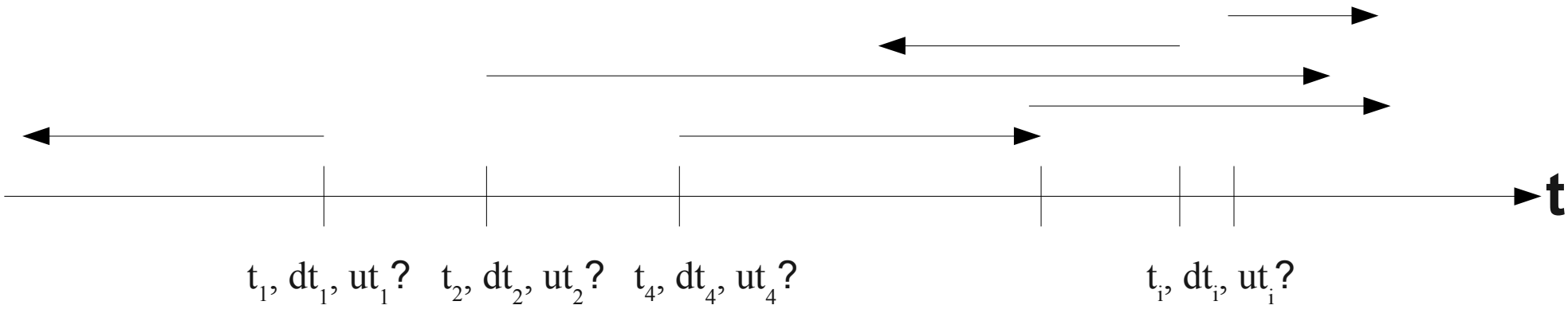
How consistent was a read?
When did the last write occur?
When will the next write occur?
Where did the last write occur?
...

Conflict

“Look forward” in time:

How fast was an update disseminated?
When did the next read occur?
When will the next read occur?
Where will the next read occur?
...

By which t?
By which tType?
By which oType?
By which dt?
...



References

[Dunham, 1997] Margaret H. Dunham, Abdelsalam Helal, and Santosh Balakrishnan. A mobile transaction model that captures both the data and movement behavior. Mobile Networks and Applications, 2:149–162, 1997.

[Gray, 1993] Jim Gray and Andreas Reuter. Transaction Processing: Concepts and Techniques. Morgan Kaufmann, 1993.

[Brewer, 2000] Eric A. Brewer. Towards robust distributed systems (abstract). In PODC ’00: Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing, page 7. ACM, 2000.

[Gilbert, 2002] Seth Gilbert and Nancy Lynch. Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News, 33(2):51–59, 2002.

[Fox, 1999] Armando Fox and Eric A. Brewer. Harvest, Yield, and Scalable Tolerant Systems. Hot Topics in Operating Systems, 174 - 178, 1999.

[Zimmermann, 2007] Olaf Zimmermann, Jonas Grundler, Stefan Tai, and Frank Leymann. Architectural Decisions and Patterns for Transactional Workflows in SOA. In ICSOC ’07: Proceedings of the 5th international conference on Service-Oriented Computing, pages 81–93. Springer-Verlag, 2007.

[Davies, 1978] C. T. Davies. Data processing spheres of control. IBM Syst. J., 17(2):179–198, 1978.

[Moss, 1985] John Eliot Blakeslee Moss. Nested transactions: An approach to reliable distributed computing. MIT Press series in information systems. MIT Press, Cambridge, Mass., 1985.

[Weikum, 1992] Gerhard Weikum and Hans-Joerg Schek. Concepts and Applications of Multilevel Transactions and Open Nested Transactions. In Database Transaction Models for Advanced Applications, pages 515–553. Morgan Kaufmann, 1992.

[Garcia-Molina, 1987] Hector Garcia-Molina and Kenneth Salem. Sagas. In SIGMOD ’87: Proceedings of the 1987 ACM SIGMOD international conference on Management of data, pages 249–259. ACM, 1987.

[Elmagarmid, 1992] Ahmed K. Elmagarmid, editor. Database Transaction Models for Advanced Applications. Morgan Kaufmann, 1992.

[Waechter, 1992] Helmut Waechter and Andreas Reuter. The ConTract Model. In Database Transaction Models for Advanced Applications, pages 219–263. Morgan Kaufmann, 1992.

[Reuter, 1997] Andreas Reuter and Kerstin Schneider and Friedemann Schwenkreis. ConTracts Revisited. In Sushil Jajodia and Larry Kerschberg, editors, Advanced Transaction Models and Architectures. 1997.

[Leymann, 1995] Frank Leymann. Supporting Business Transactions Via Partial Backward Recovery In Workflow Management Systems. In Georg Lausen, editor, Datenbanksysteme in B¨uro, Technik und Wissenschaft (BTW'95), Informatik aktuell, pages 51–70., Springer-Verlag, 1995.

[SCA, 2006] Open Service Oriented Architecture (OSOA) Consortium. Service Data Objects For Java Specification, Version 2.1.0, November 2006.

[Wang, 2008] Ting Wang, Jochem Vonk, Benedikt Kratz, and Paul Grefen. A survey on the history of transaction management: from flat to grid transactions. Distrib. Parallel Databases, 23(3):235–270, 2008.

[JEE] The Java EE Tutorial, Chapter Transactions, http://docs.sun.com/app/docs/doc/819-3669/bncih?l=en&a=view, 23.11.2010

[Spring] The Spring Documentation, Chapter Transaction Management, http://static.springsource.org/spring/docs/2.0.x/reference/transaction.html, 23.11.2010

[JTA, 2002] Susan Cheung & Vlada Matena, The Java Transaction Specification, Version 1.1, 2002

[Newcomer, 2007] Newcomer, E., Robinson, I., Feingold, M., Jeyaraman, R.: Web services coordination (WSCoordination) version 1.1. OASIS, April 2007. http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-os.pdf, 23.11.2010

[Newcomer, 2007a] Newcomer, E., Robinson, I., Freund, T., Little, M.: Web services business activity framework (WSBusinessActivity) version 1.1. OASIS, April 2007. http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec-os.pdf, 23.11.2010

[Newcomer, 2007b] Newcomer, E., Robinson, I., Little, M., Wilkinson, A.: Web services atomic transaction (WSAtomicTransaction) version 1.1. OASIS, April 2007. http://docs.oasis-open.org/ws-tx/wstx-wsat-1.1-spec-os.pdf, 23.11.2010

[Lessner, 2011] Tim Lessner, Fritz Laux, Thomas Connolly, Cherfi Branki, Malcolm Crowe, and Martti Laiho. An Optimistic Transaction Model for a Disconnected Integration Architecture, The Third International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA 2011), ACCEPTED

Thank you!