建立过程数据和持久数据的对应关系

Bridging Persistent Data and Process Data

Jianwen Su (苏建文)
University of California at Santa Barbara

Outline

- Activity → data-centricity → artifact
- Lessons from practice
- BP as a Service
- Extending the artifact concept:
  Help from data integration? (or not)
- Cross reference paths
- The updatability requirement
- Isolation of process "footprints" or dataprints
- Many challenges ahead
- Conclusions

Traditional BP Modeling

- Activity-centric, focusing on control flow (e.g. BPMN)
  - Mainly aiming at business management in general (instead of software design/development)
    e.g., resource planning, logistics
- Missing data is a key reason for hindering software design and management,

  many miserable stories including
  - Hangzhou Housing Management Beauru
  - Kingfore Corporation (KFC, Beijing)
  - RuiJing hospital (Shanghai) & Cottage hospital (Santa Barbara, CA)
  - IBM Global Financing (IGF)

Four Kinds of Data

- Business data: essential for business logic
  - Examples: *items, shipping addresses*
- Enactment status: the current execution snapshot
  - Examples: *order sent, shipping request made*
- Resource usage and state needed for service execution
  - Examples: *cargo space reserved, truck schedule to be determined*
- Correlation between processes instances
  - Example: *3 warehouse fulfillment process instances for Jane's order*
- Need models that include both activities and data

Four Classes of BP Models

- Data agnostic models: data mostly absent
  - WF (Petri) nets, BPMN, UML Activity Diagrams, …
- Data-aware models: data present (as variables), but storage and management hidden
  - BPEL, YAWL, …
- Storage-aware models: schemas for persistent stores, mappings to/from data in BPs defined and managed manually
  - jBPM, …
- Data encapsulting models: logical data modeling, automated modeling other 3 types, data-storage mapping
  - Business objects, artifact-centric models

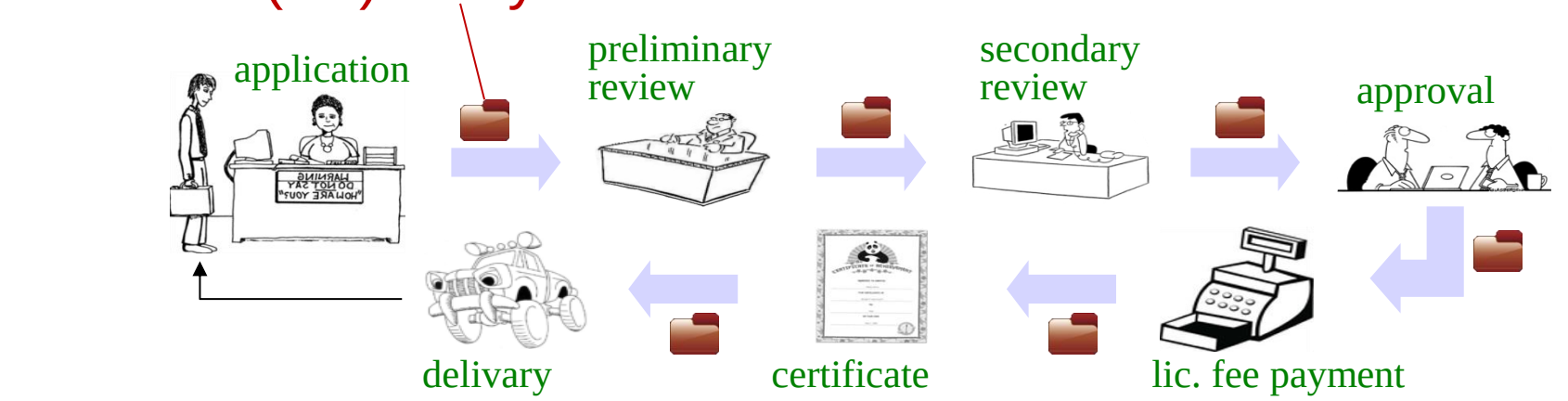
Artifact = Biz Process

- A business artifact is a key conceptual business element that is used in guiding the operation of the business
  - *fedex package delivery, patient visit, application form, insurance claim, order, financial deal, registration, ...*
  - Consists of a business entity and a lifecycle

[Nigum-Caswell IBM Sys J 03]

- Very natural to business managers and BP modelers
- For this talk : artifact is a synonym of BP

(practically beneficial)



Outline

- Activity → data-centricity → artifact
- Lessons from practice
- BP as a Service
- Extending the artifact concept:
  Help from data integration? (or not)
- Cross reference paths
- The updatability requirement
- Isolation of process "footprints" or dataprints
- Many challenges ahead
- Conclusions

Story 1: Toy Application Systems

- Development of application systems in DB a course Last Winter: a bank system
  - Accounts, clients, transactions; a small number of typical transactions; teller & management: monthly statements, tax reports
- Typical development approach: Entity-Relationship modeling → Java classes/modules → Java & JDBC code
- Most frequent mistakes:
  - Mismatch of data designs in Java and in ER: omissions, incompatible semantics

Too bad: this is the best available to teach

The two sides of the coin are indeed separated

Story 2: The Kingfore System

- Heating repair workflow, consisting of *reporting problems*, *assign service persons*, *onsite repair*, and *post-repair review visits*
  - 3-month development contracted to BUPT
- Their problem:
  - Mid-way requirement change including, in particular, adding an activity to the repair workflow:
    demands rewriting a lot of code
  - Artifact BP helps conceptualizaing changes, but…
  - A close look: rewritten code mostly involve DB accesses

Database Design & Biz Entity Design

KINGFORE 金房

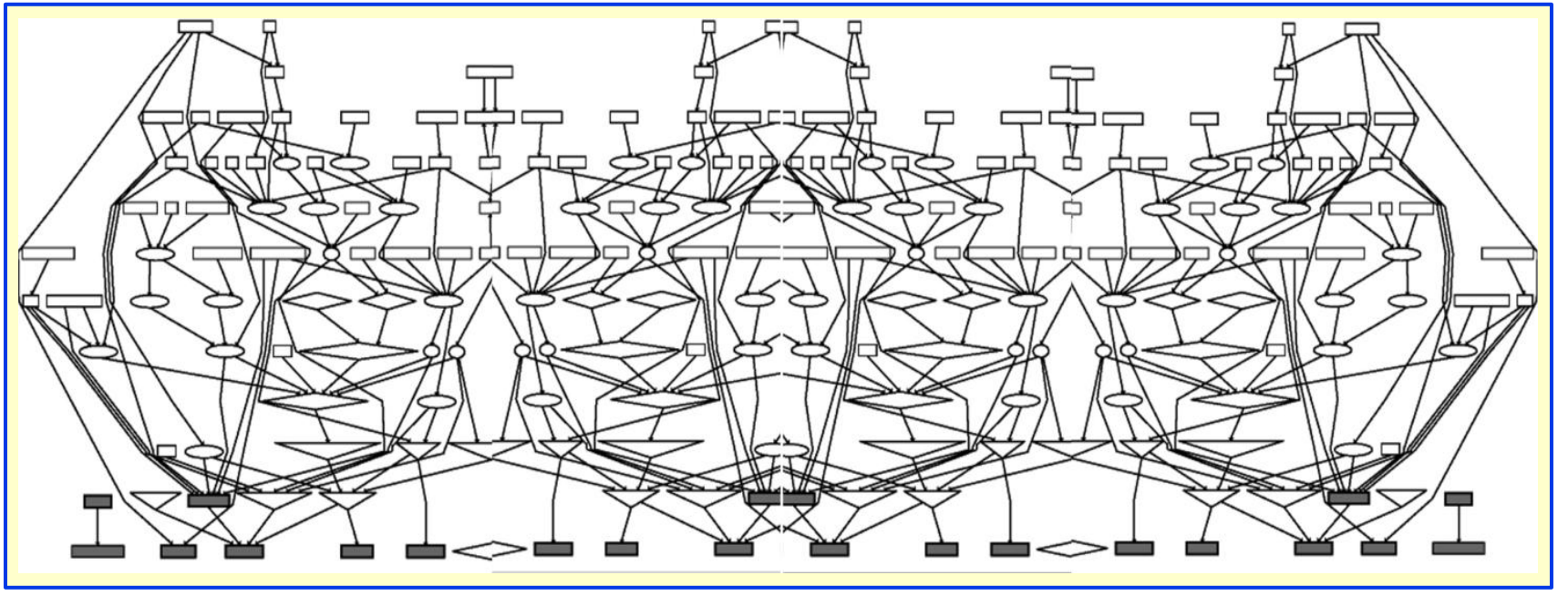
- Typical development steps:
  - Enterprise database design
  - The repair workflow modeled in XPDL (BPMN)
  - Each activity in the workflow coded, "biz entity" never designed but just coded as needed
  - Developers made isolated decisions to "link" biz entity to database (via SQL) (contrast to BP model)
- Elevating to the conceptual level [Sun-S.-Wu-Yang 2013]
  - Biz entity → artifact info model
  - Link → database-entity mappings

  could enable automating coding db accesses

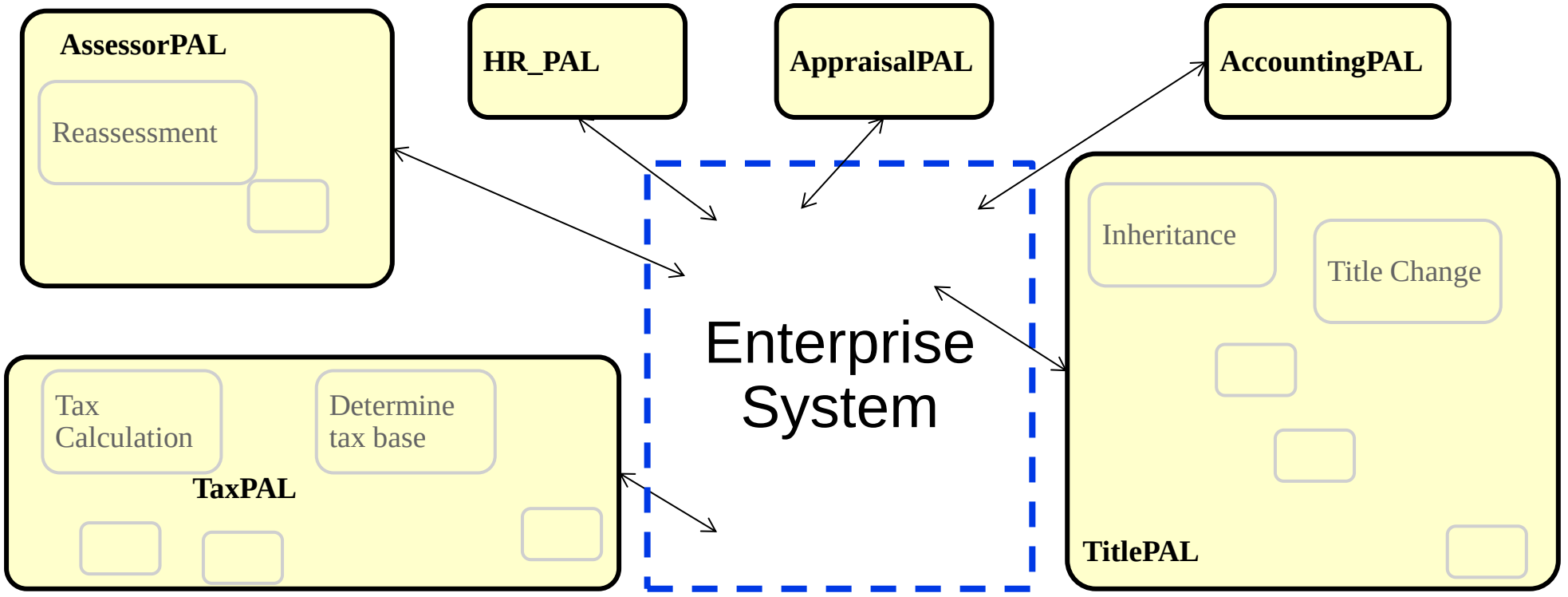
Integrating the two sides helps application development

An XXX Application System

- Ad hoc design, developed over time, patches, multiple technologies, … a typical legacy system
- Problems:
  - Embedded business logic, hard to learn
  - hard to maintain, costly to add new functionality
  - hard to change/evolve



The LEGO Fantasy



Towards a goal of

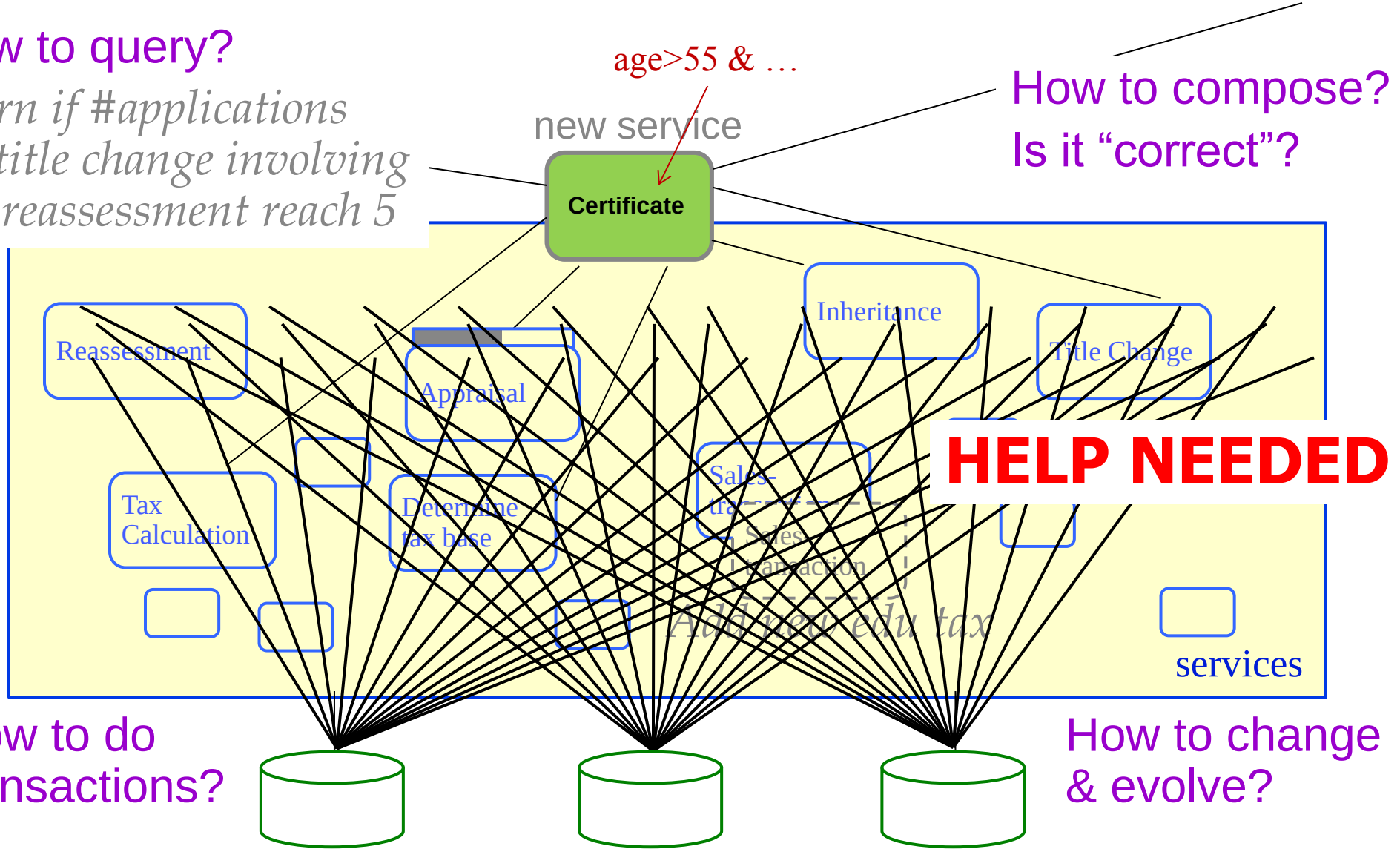
- Business Process as a Service (BPaaS)
- Enterprises may run virtual IT systems

How do we do it?

Service Programming is an Art

How to query?

Warn if #applications for title change involving tax reassessment reach 5

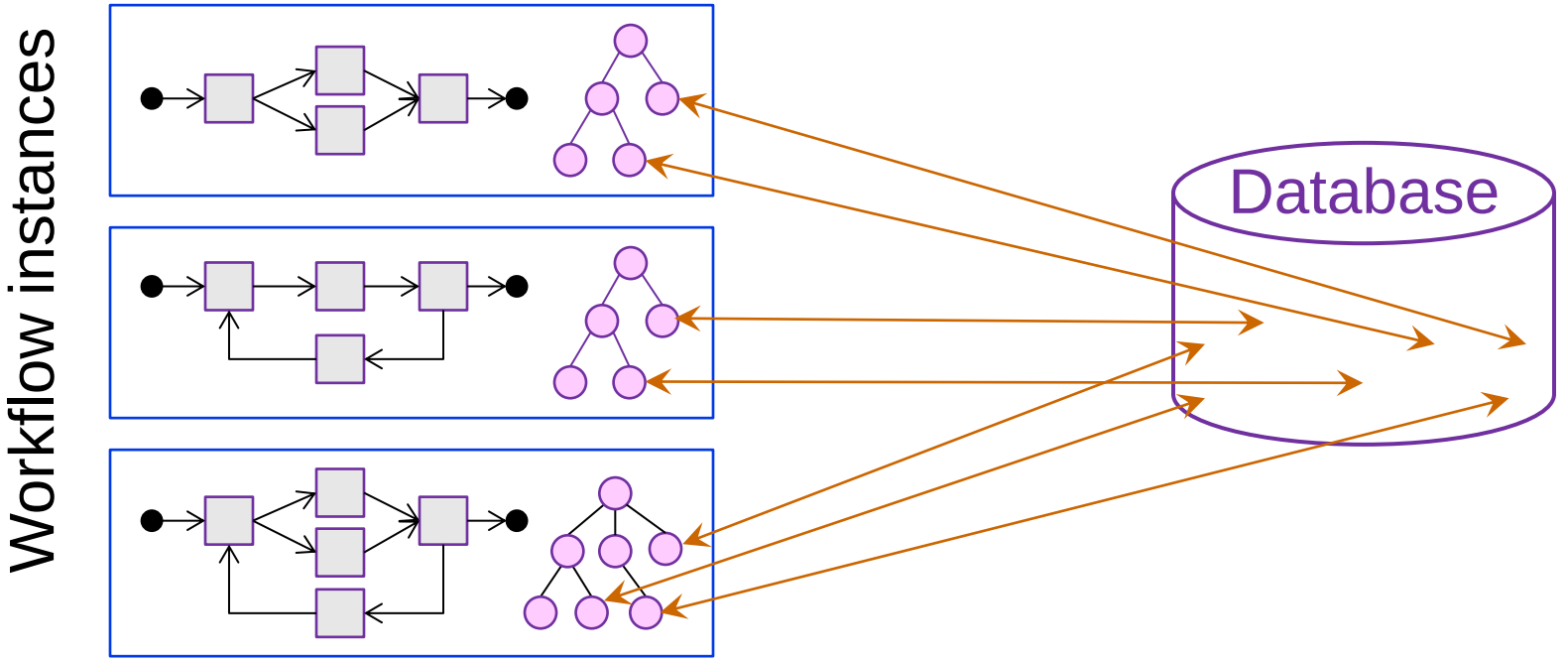


The real world is not very kind

Outline

- Activity → data-centricity → artifact
- Lessons from practice
- BP as a Service
- Extending the artifact concept:
  Help from data integration? (or not)
- Cross reference paths
- The updatability requirement
- Isolation of process "footprints" or dataprints
- Many challenges ahead
- Conclusions

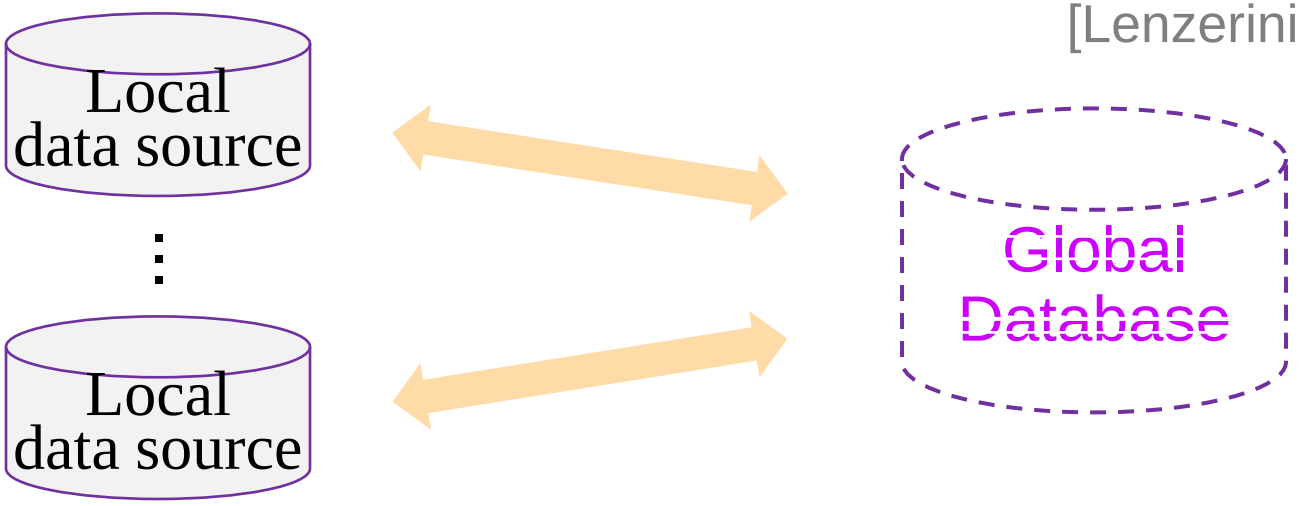
Conceptualizing Running Workflows



- Each workflow (BP) instance consists of a biz entity and a lifecycle
- Data mappings are ad hoc

Data Integration: A Bird's View

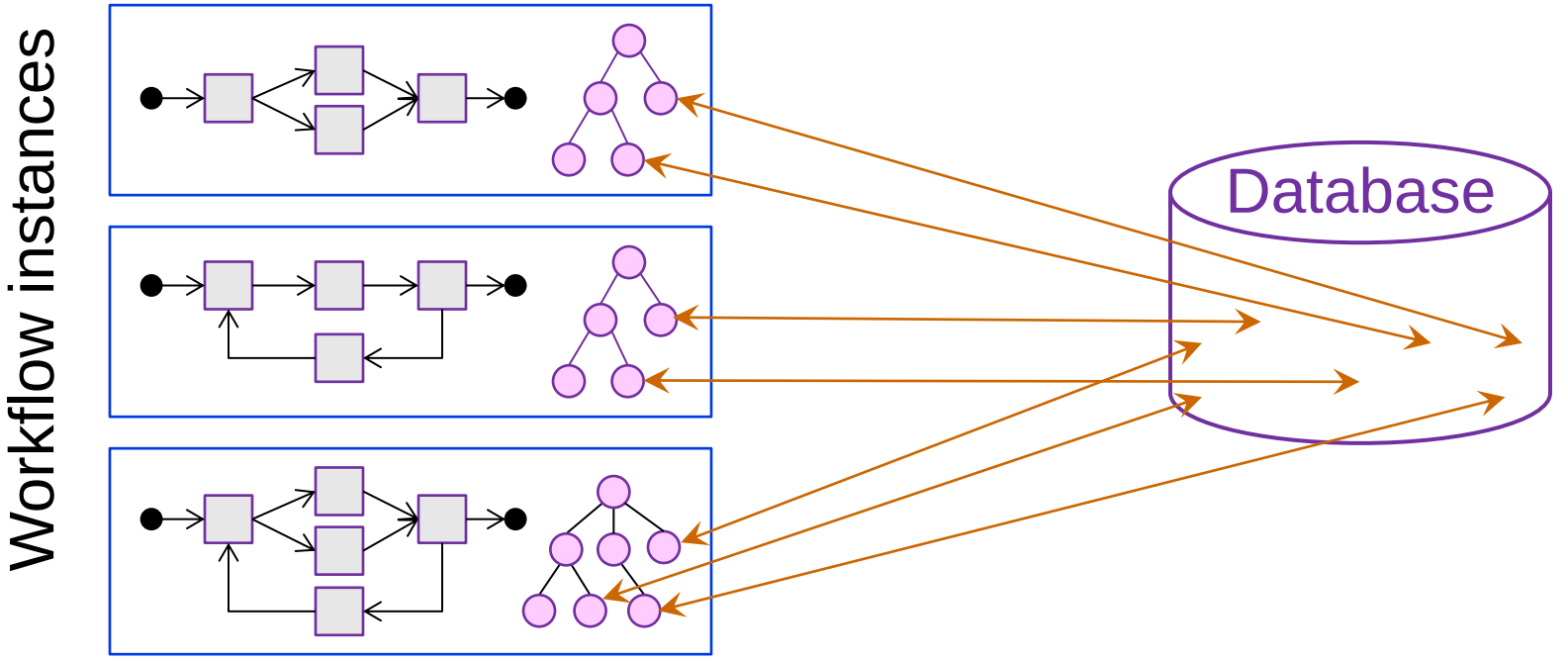
[Lenzerini PODS 02]



- Global as View (GAV): The global database is a view (result of a query) on local data sources
- Local as View (LAV): each local data source stores the result of view on the virtual global database
- Research focused on query evaluation
- Schema mapping (e.g., Clio) focused on computing general target databases

[Popa et al VLDB 02] [Fagin et al ICDT 03]

Data Integration for Workflows?



- GAV is not suitable:
  - Data not stored in workflow instances
  - The number of instances changes at runtime
- LAV?
  - Data not stored in workflow instances

Soundness and Completeness

- A local view is [Lenzerini PODS'02]
  - sound: only contains (part of) results of the view
  - complete: contains all results of the view

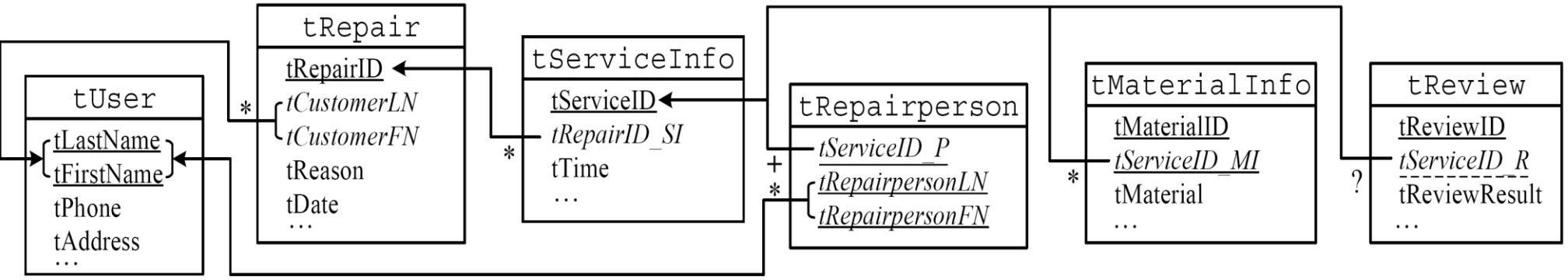
- Workflow data mappings?
  - Must be exact, i.e., both sound and complete

- Open problem:
  demands a better understanding of data mappings

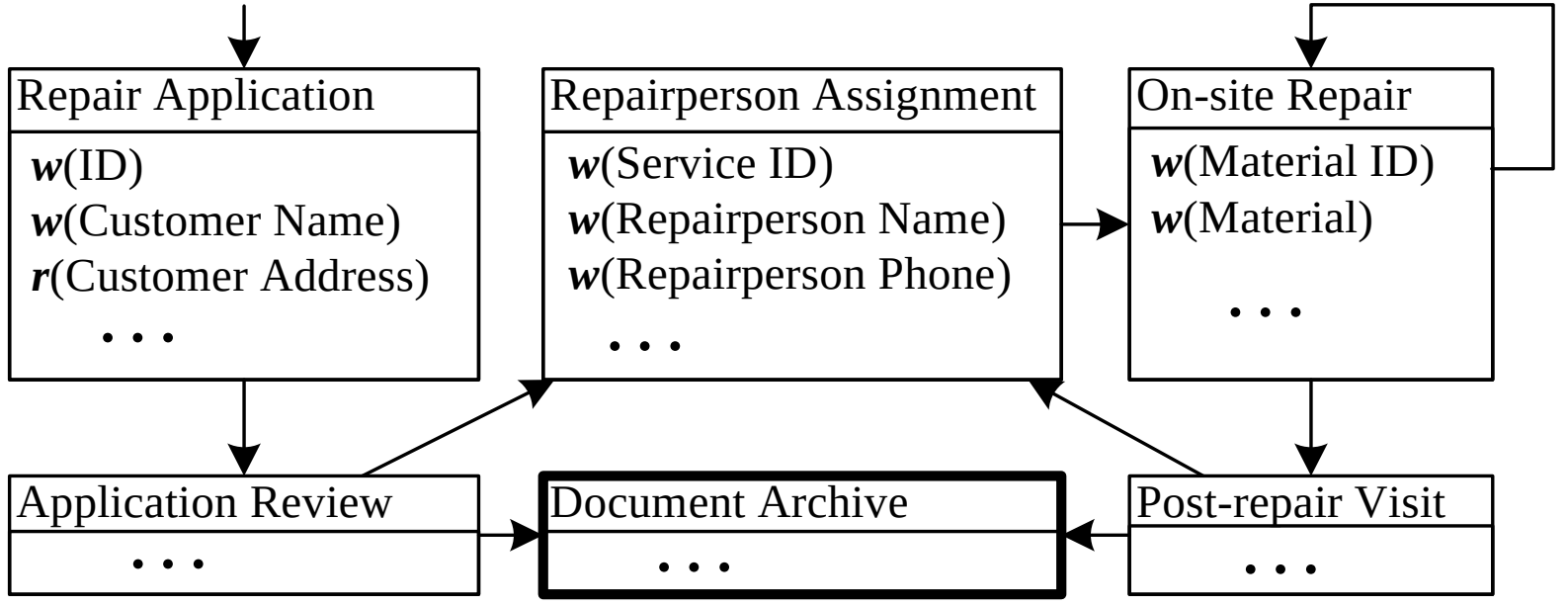
Outline

- Activity → data-centricity → artifact
- Lessons from practice
- BP as a Service
- Extending the artifact concept:
  Help from data integration? (or not)
- Cross reference paths
- The updatability requirement
- Isolation of process "footprints" or dataprints
- Many challenges ahead
- Conclusions

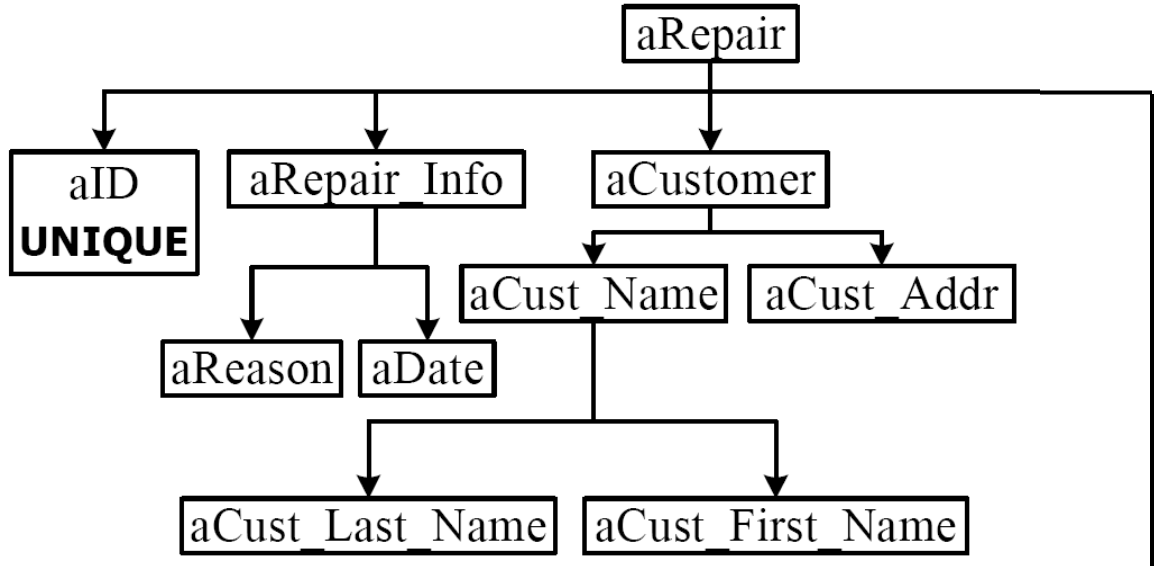
Example: The Database (& Lifecycle)



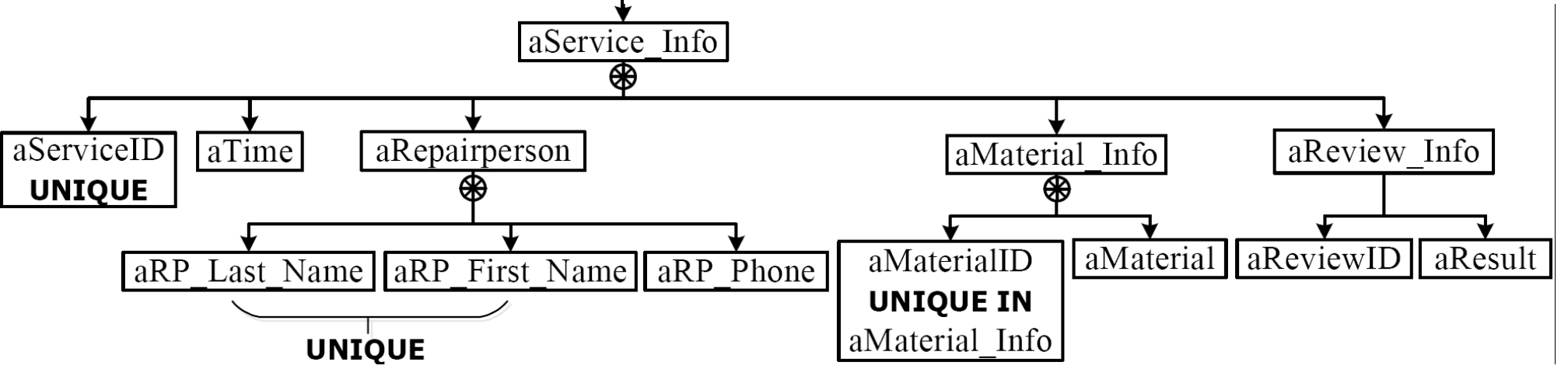
- Includes keys, foreign keys, and a cardinality specification on each foreign key



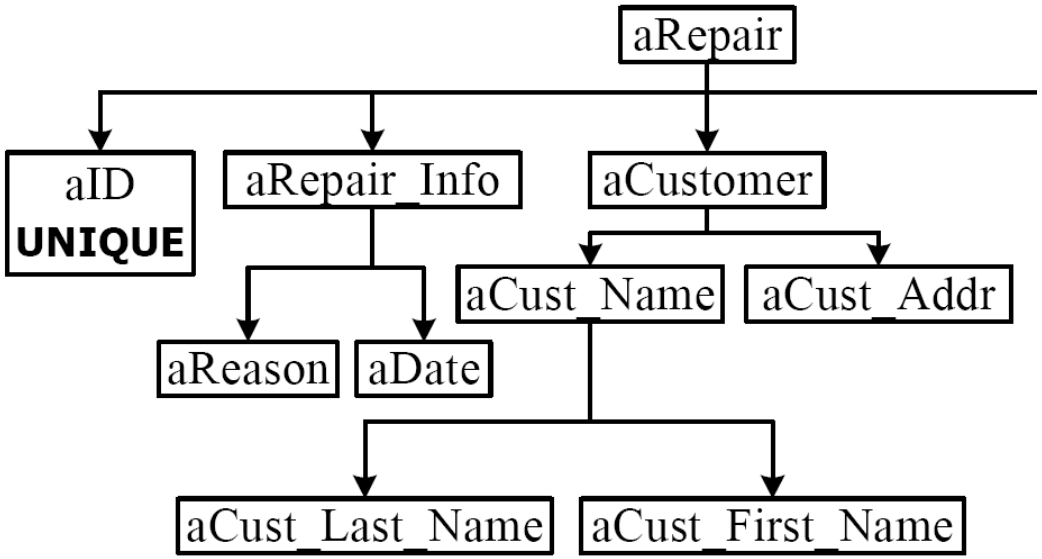
Example: The Biz Entity

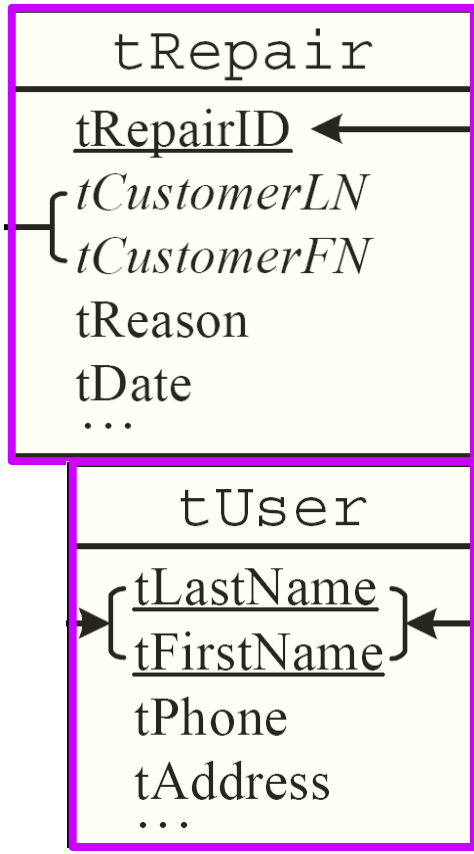


Tuple and
(nested) set constructs

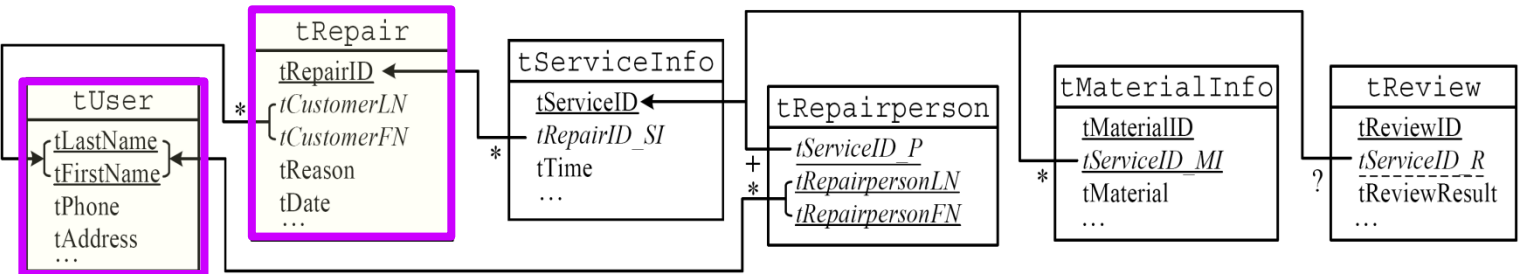


Example: Cross Reference Paths



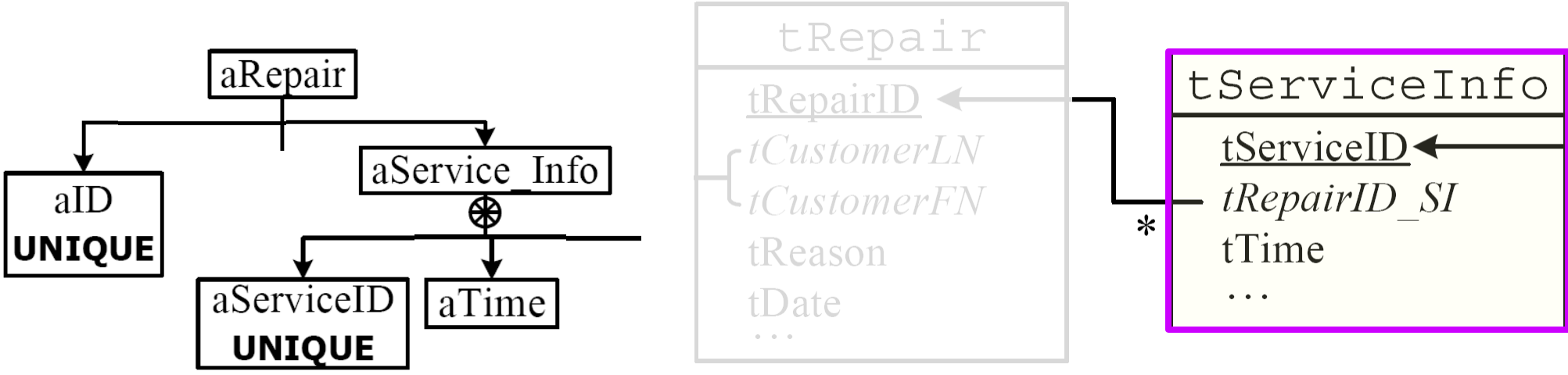


- aID : tRepair.tRepairID
- aReason = aReason.aRepair_Info.aID@tRepair(tRepairID).tReason
- aCust_Addr = aCust_Addr.aCust_Name.[aCust_Last_Name, aCust_First_Name]@tUser(tLastName, tFirstName).tAddress



More Cross Reference Paths

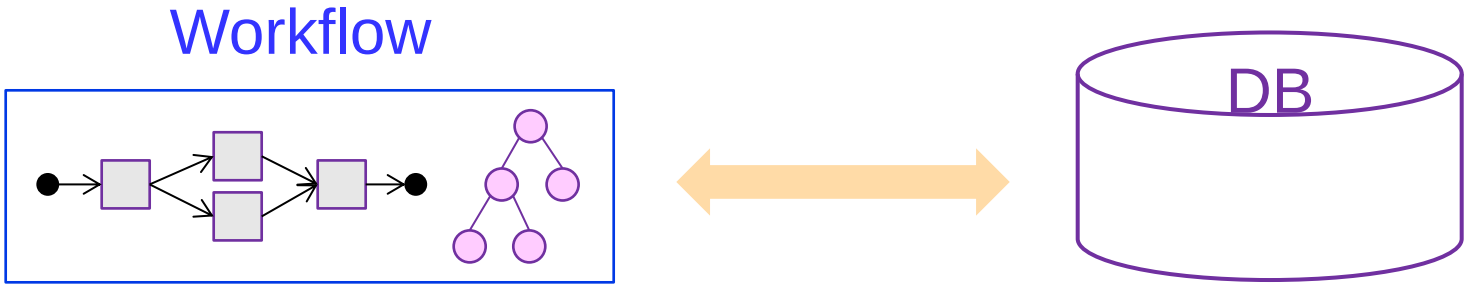
- aServiceID : tServiceInfo.tServiceID **when**
  aServiceID.aService_Info.aID = tServiceInfo.tRepairID_SI
- aTime = aTime.aServiceID@tServiceInfo(tServiceID).tTime



- In summary, two kinds of mapping rules:
  - Key mapping rule — existentially quantified
  - Non-key mapping rules —access path with equality

Entity-Database Cover

- ED cover consists of one mapping rule for each primitive attribute in biz entity
  - Key attributes use key mapping rules
  - Non-key attributes use equality access rules

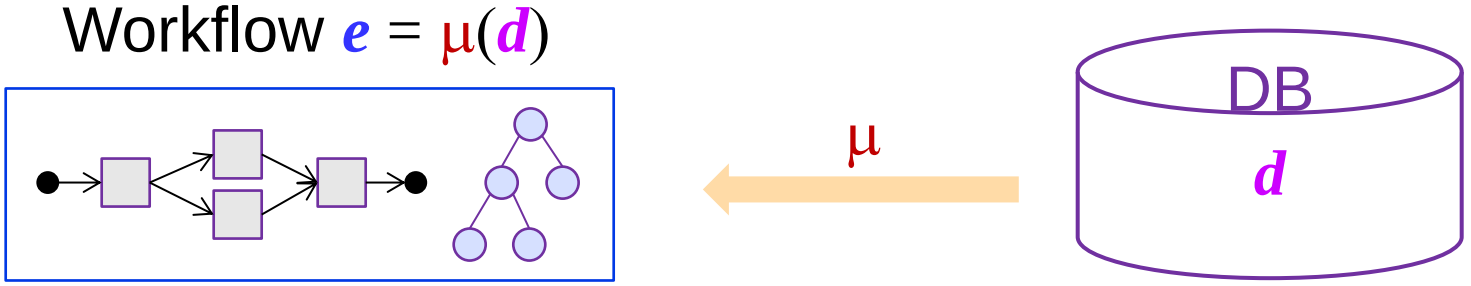


- Great news: DB accessed can be auto-generated
  - Workflow modifies its entity, DB hidden
- Every update on DB can be propogated to entity?
- Every update on entity can be propogated to DB?

Outline

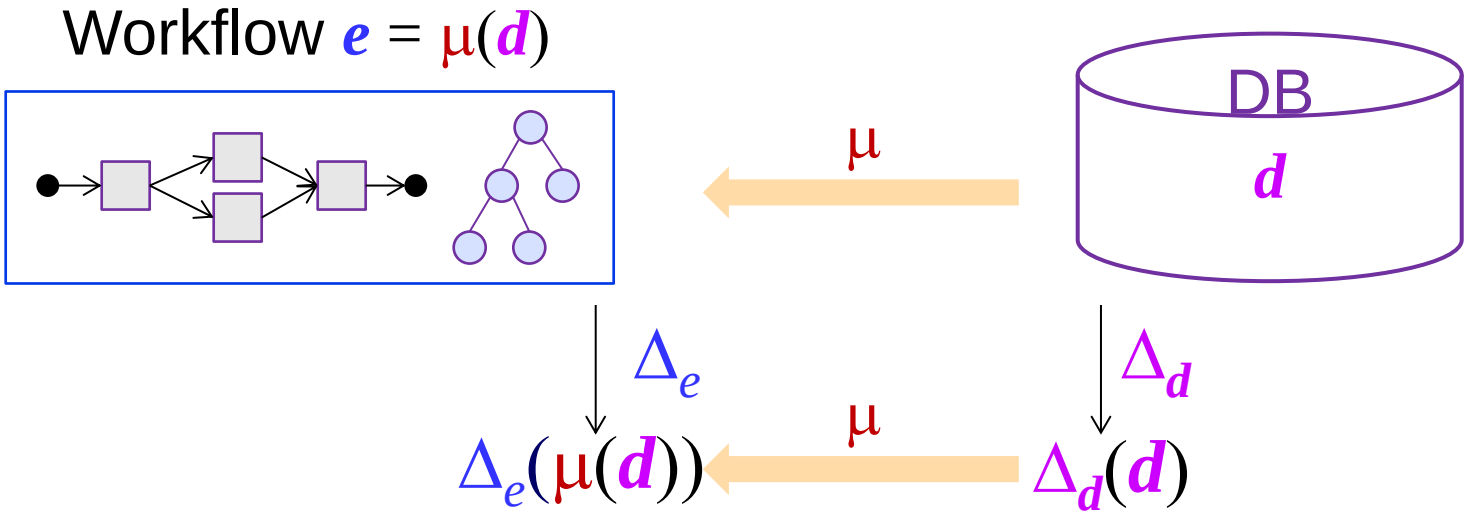
- Activity → data-centricity → artifact
- Lessons from practice
- BP as a Service
- Extending the artifact concept:
  Help from data integration? (or not)
- Cross reference paths
- The updatability requirement
- Isolation of process "footprints" or dataprints
- Many challenges ahead
- Conclusions

Updatability



- Database updability:
  for each update Δ_d on d,
  there is an e' such that $e' = \mu(\Delta_d(d))$
- Entity updability:
  for each update Δ_e on $e = \mu(d)$,
  there is a d' such that $\mu(d') = \Delta_e(e)$

Updatability



- Database updability:
  for each update Δ_d on d,
  there is an update Δ_e such that $\Delta_e(\mu(d)) = \mu(\Delta_d(d))$
- Entity updability:
  for each update Δ_e on $\mu(d)$,
  there is an update Δ_d such that $\mu(\Delta_d(d)) = \Delta_e(\mu(d))$

Entity Update & View Update

- Database updatability: forward, can always be done
- Entity updatability: backward, often not possible

- Very closely related to database view update problem

  [Bancilhon-Spyratos TODS 81]

  - View complement [BS81] [Lechtenbörger et al PODS 03]
  - Clean source [Dayal-Bernstein TODS 82][Wang et al DKE 06]

- Fortunate here:

  Theorem: Every non-overlaping ED cover is entity updatable

  [Sun-S.-Wu-Yang ICDE 14]

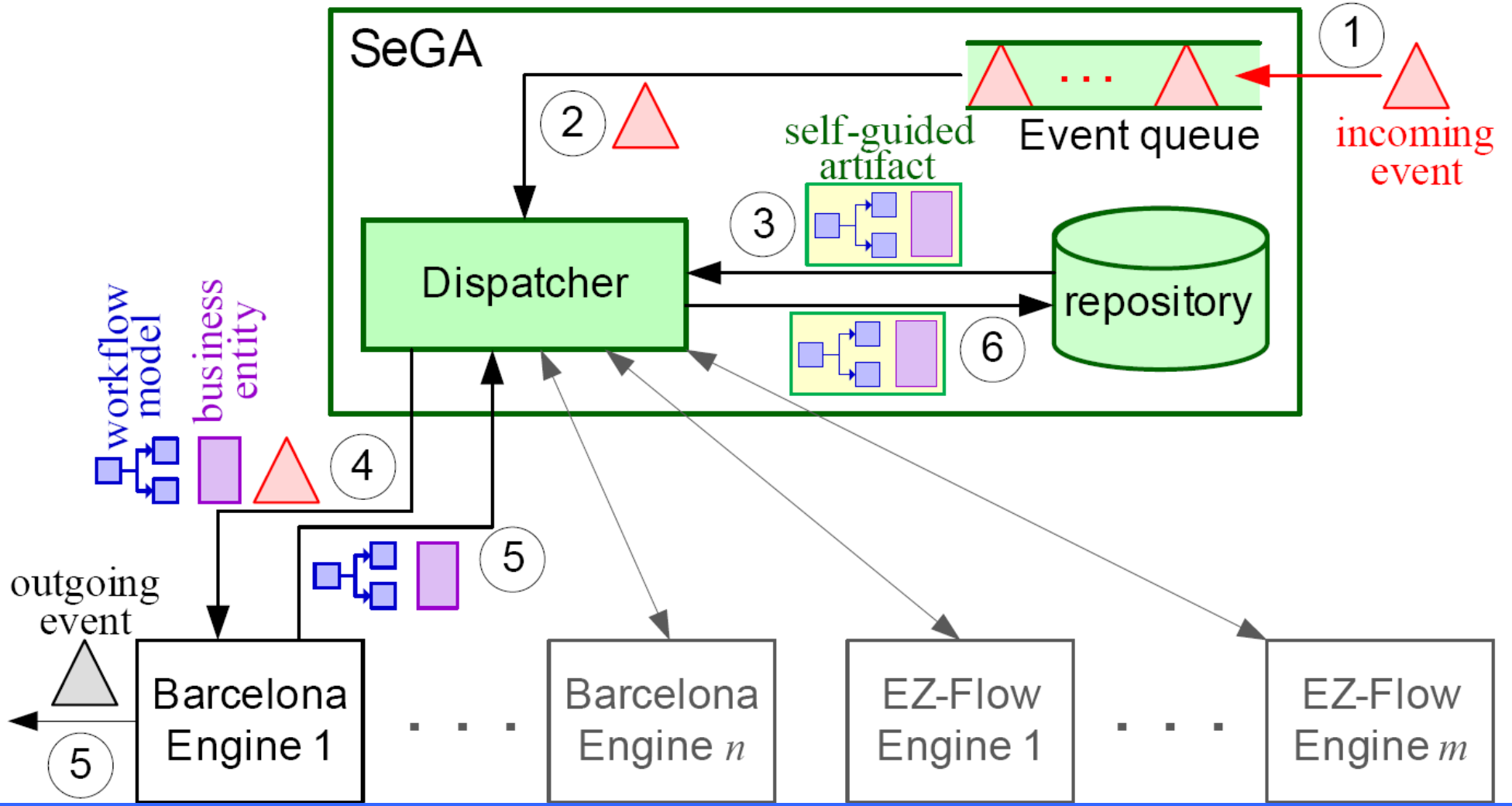
Outline

- Activity → data-centricity → artifact
- Lessons from practice
- BP as a Service
- Extending the artifact concept:
  Help from data integration? (or not)
- Cross reference paths
- The updatability requirement
- Isolation of process “footprints” or dataprints
- Many challenges ahead
- Conclusions

SeGA: A Service Wrapper/Mediator

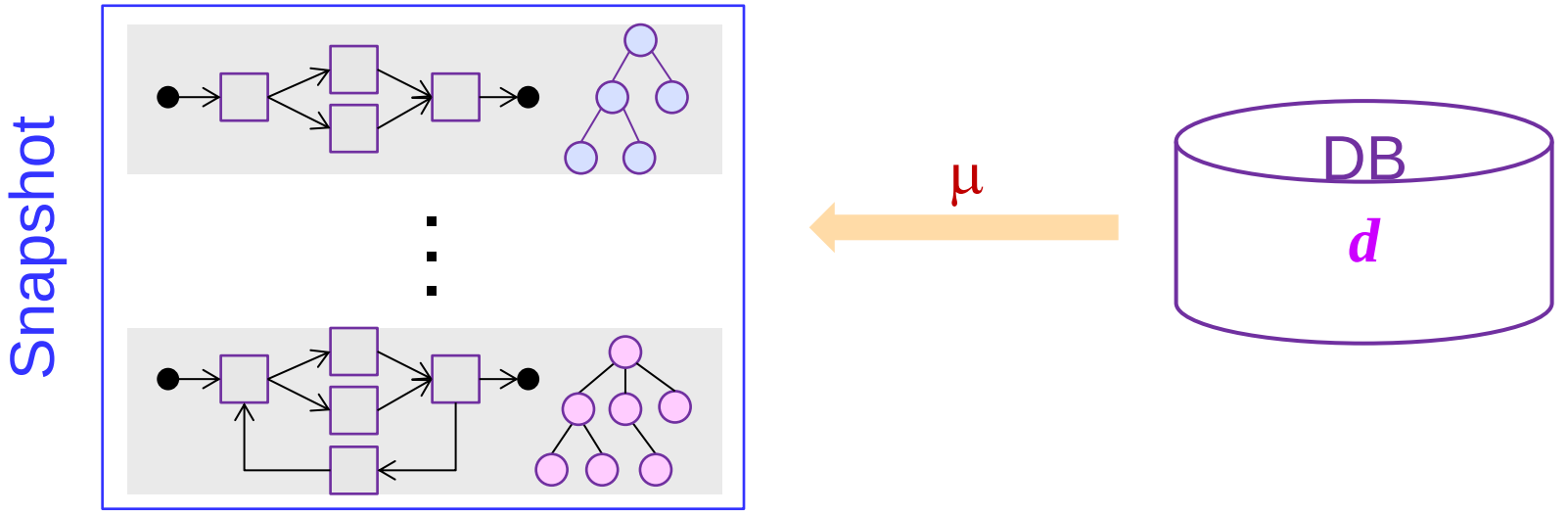
[Sun-Xu-S.-Yang CoopIS '12]

- SeGA separates data from execution engine
- Serves as a mediator



Possible only if "footprints" of BP instances disjoint

Isolation of BP Instances



- μ is isolating if each update on a single entity (instance) will not affect
  write (and/or read) attributes of other entity instances
- Theorem: Isolation can be tested
  - Testing "conflicting" updates
  - EXPTIME with conditional updates

[Sun-S.-Wu-Yang ICDE 14]

Outline

- Activity → data-centricity → artifact
- Lessons from practice
- BP as a Service
- Extending the artifact concept:
  Help from data integration? (or not)
- Cross reference paths
- The updatability requirement
- Isolation of process "footprints" or dataprints
- Many challenges ahead
- Conclusions

Connecting Biz Entities and Databases

- Fundamentals
  - What are these mappings?
    db queries phrased in 1960's, not understood until
    [Chandra-Harel JCSS 79, Bancilhon-Paredaens IPL 79]
  - Updatability, what else?
  - Mapping languages
- Design principles
  - Isolation, for lifecycles?, runtime mechanisms?
  - Data design completeness, needs ontology
  - Implementability: translating IOPEs on artifact to DB
- Transactions
  - Workflow vs databases

Outline

- Activity → data-centricity → artifact
- Lessons from practice
- BP as a Service
- Extending the artifact concept:
  Help from data integration? (or not)
- Cross reference paths
- The updatability requirement
- Isolation of process “footprints” or dataprints
- Many challenges ahead
- Conclusions

Conclusions

- Research on artifact BPs: need to look outside
- Data is the enabler/destroyer
- Holistic approaches including data and BPs can benefit practice, i.e., software design for enterprises
- BPaaS requires independence of service and data management [S. ICSOC'12]
- Need a new forum to explore holistic approaches

Data, Processes, and Sofewate Systems