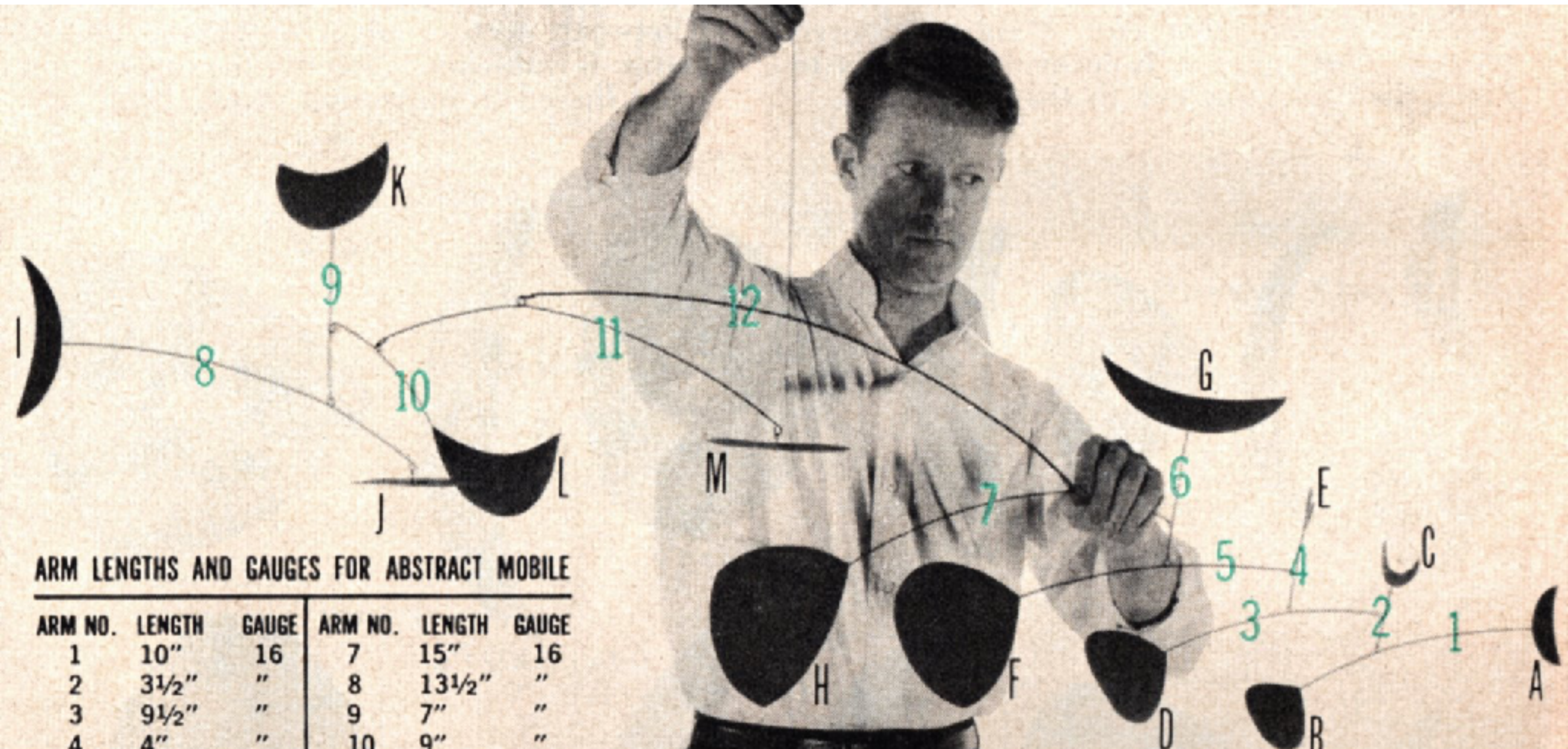
Verification of Data-Aware Processes

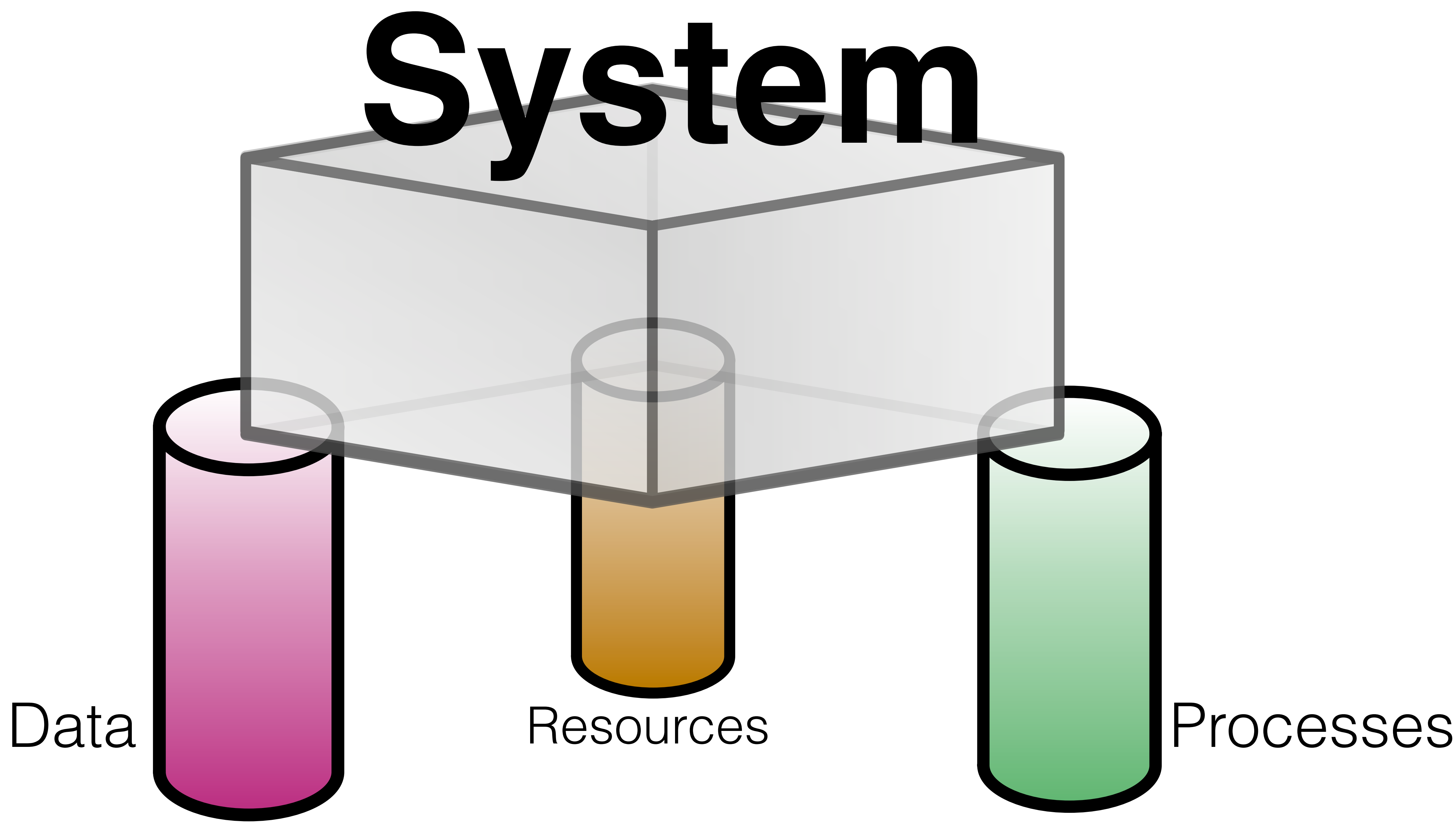
Diego Calvanese   Marco Montali

{calvanese,montali}@inf.unibz.it

Free University of Bozen-Bolzano



The Three Pillars of Complex Systems



In AI and CS, we know **a lot** about each pillar!

State of the Art

Traditional isolation between processes and data

- Why? To attack the complexity (*divide et impera*)

Logic and Computation have deeply contributed to the development of these two aspects

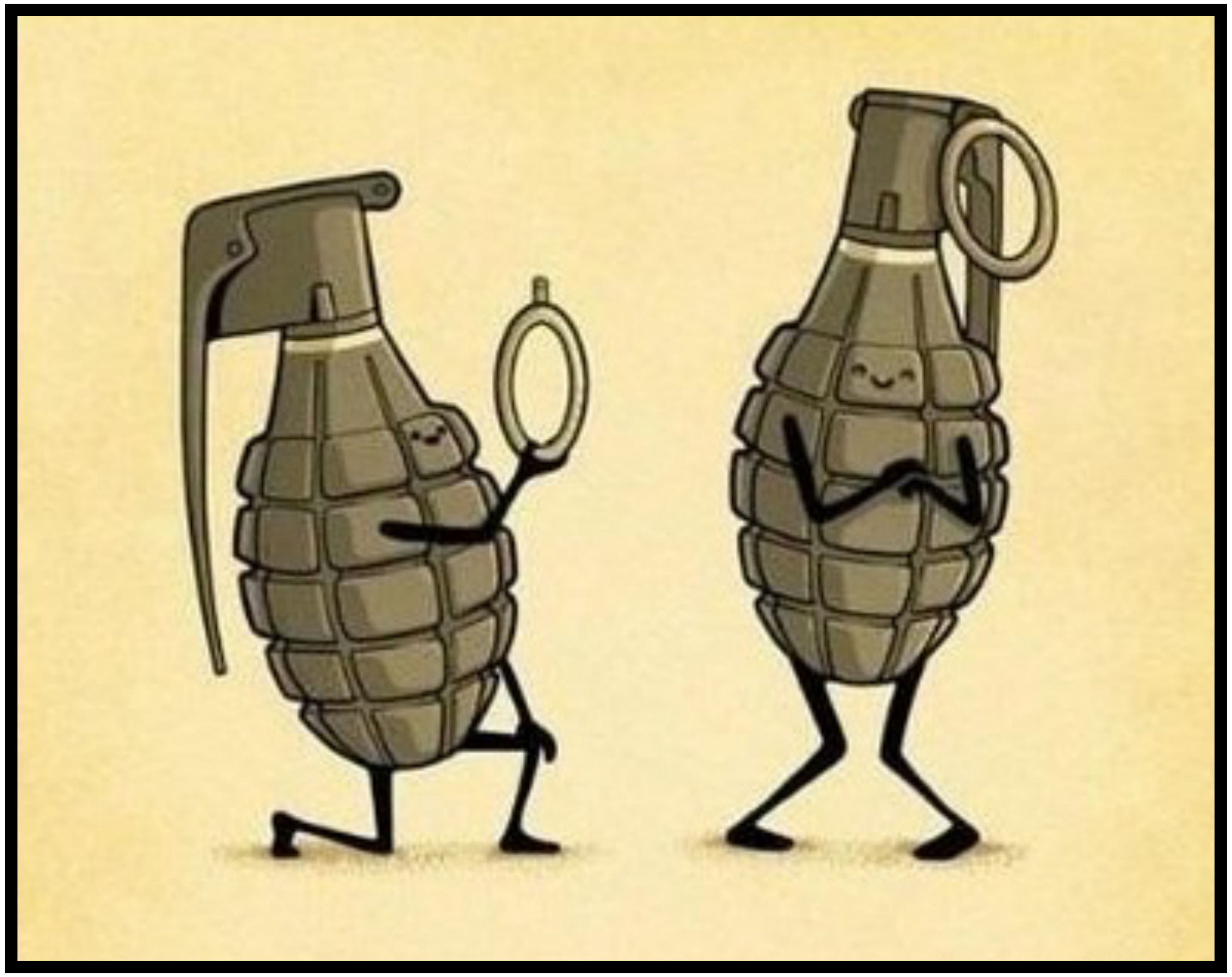
- *Data*: knowledge bases, conceptual models, ontologies, ontology-based data access and integration, inconsistency-tolerant semantics, …
- *Processes*: reasoning about actions, temporal/dynamic logics, situation/event calculus, temporal reasoning, planning, verification, synthesis, …

Information Assets

- **Data**: the main information source about the history of the domain of interest and the relevant aspects of the current state of affairs
- **Processes**: how work is orchestrated in the domain of interest, so as to create value
- **Resources**: humans and devices responsible for the execution of work units within a process

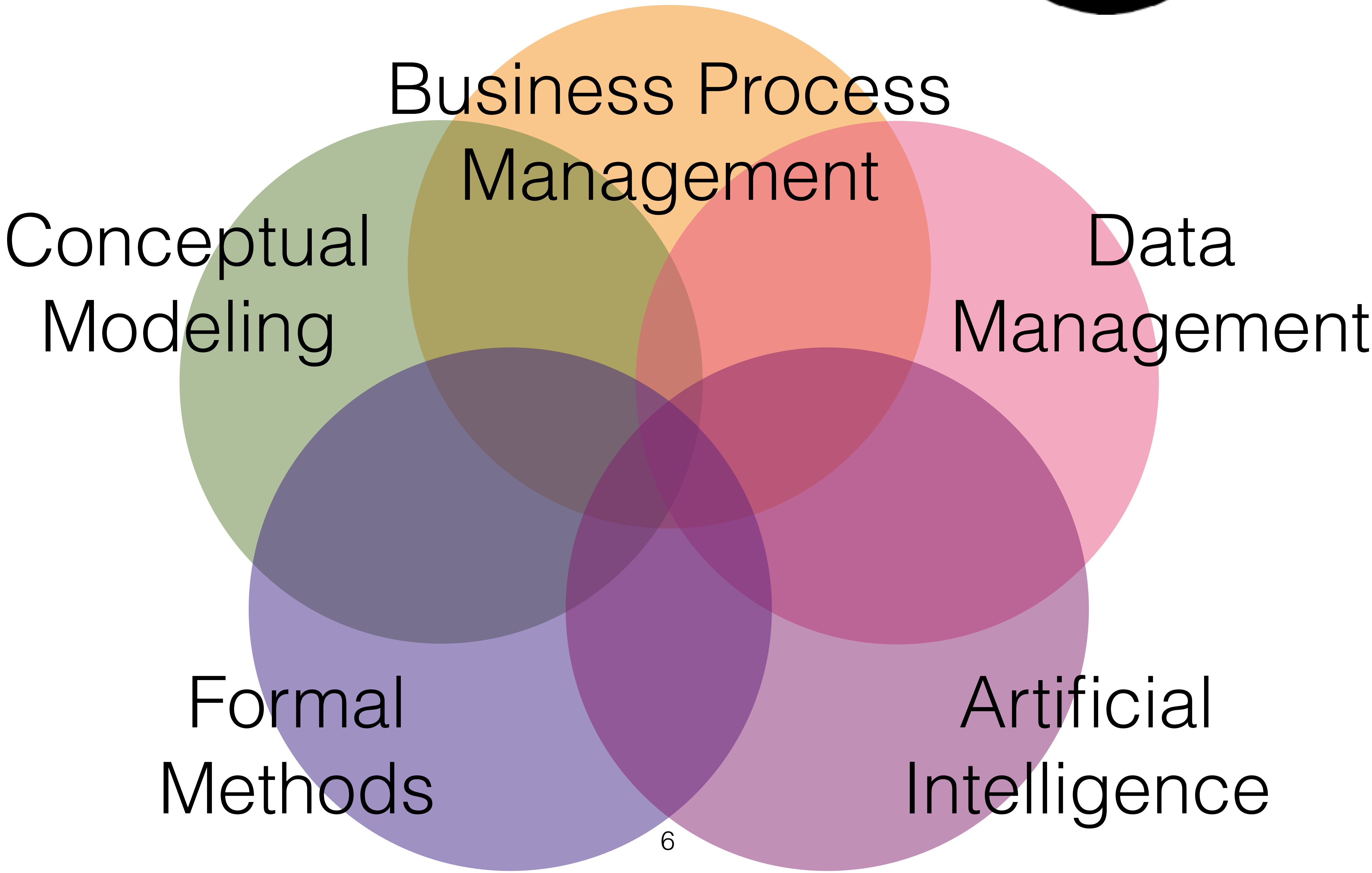
We focus on data and processes!

Marrying **processes** and **data** is extremely **challenging**….



… but is a **must** if we want to really **understand** how **complex dynamic systems** operate.

Our Research at KRDB



Our Research at KRDB

Practice

Theory

Our Research at KRDB

Theory

Practice

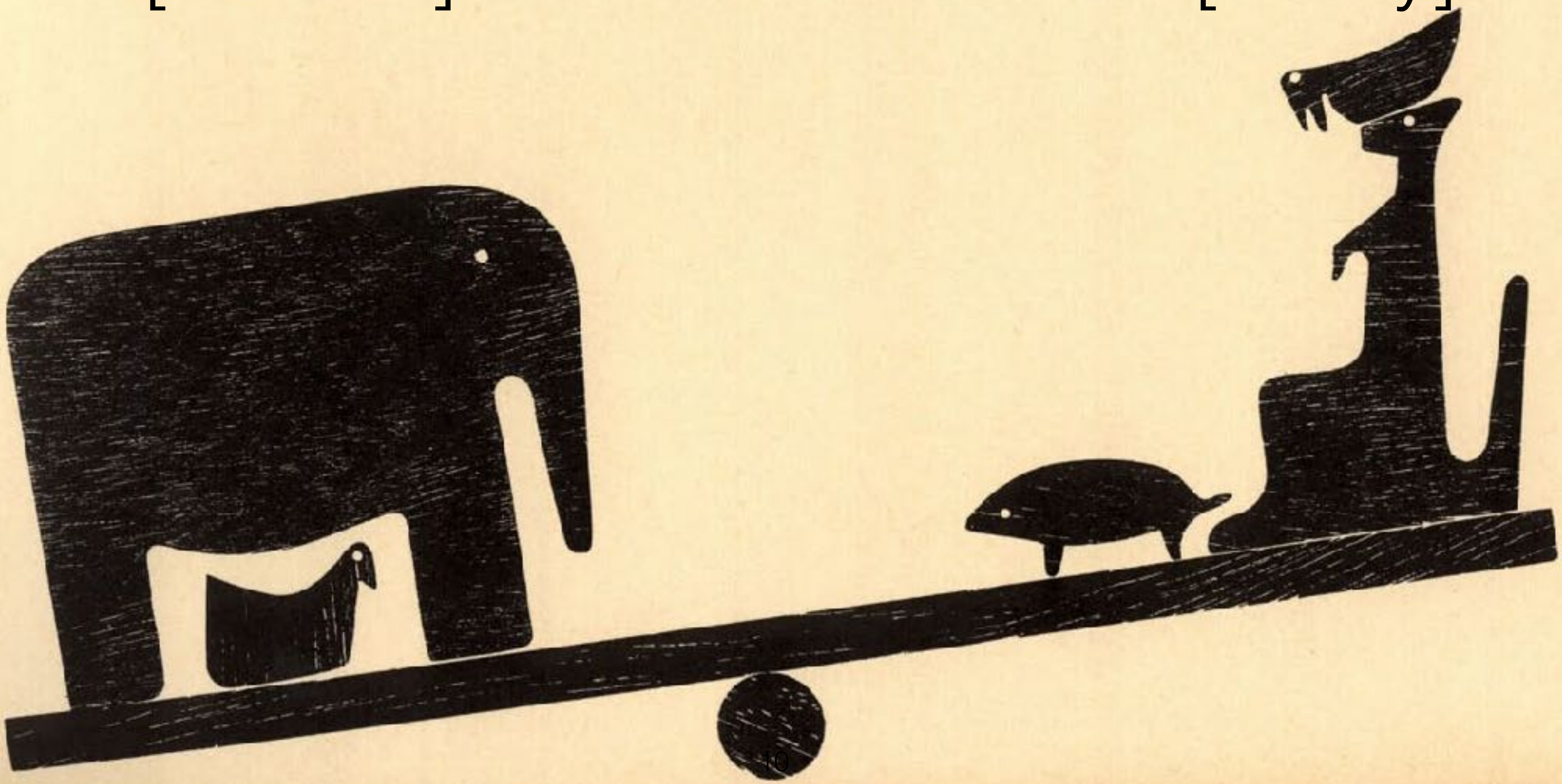
Outline

1. Introduction and motivation: **why processes + data**
2. The framework of **Data-Centric Dynamic Systems**
3. Verification **logics** and behavioural **indistinguishability**
4. Sources of **undecidability**
5. Control and conquer: **decidability** results
6. Connection to **concrete languages and systems**

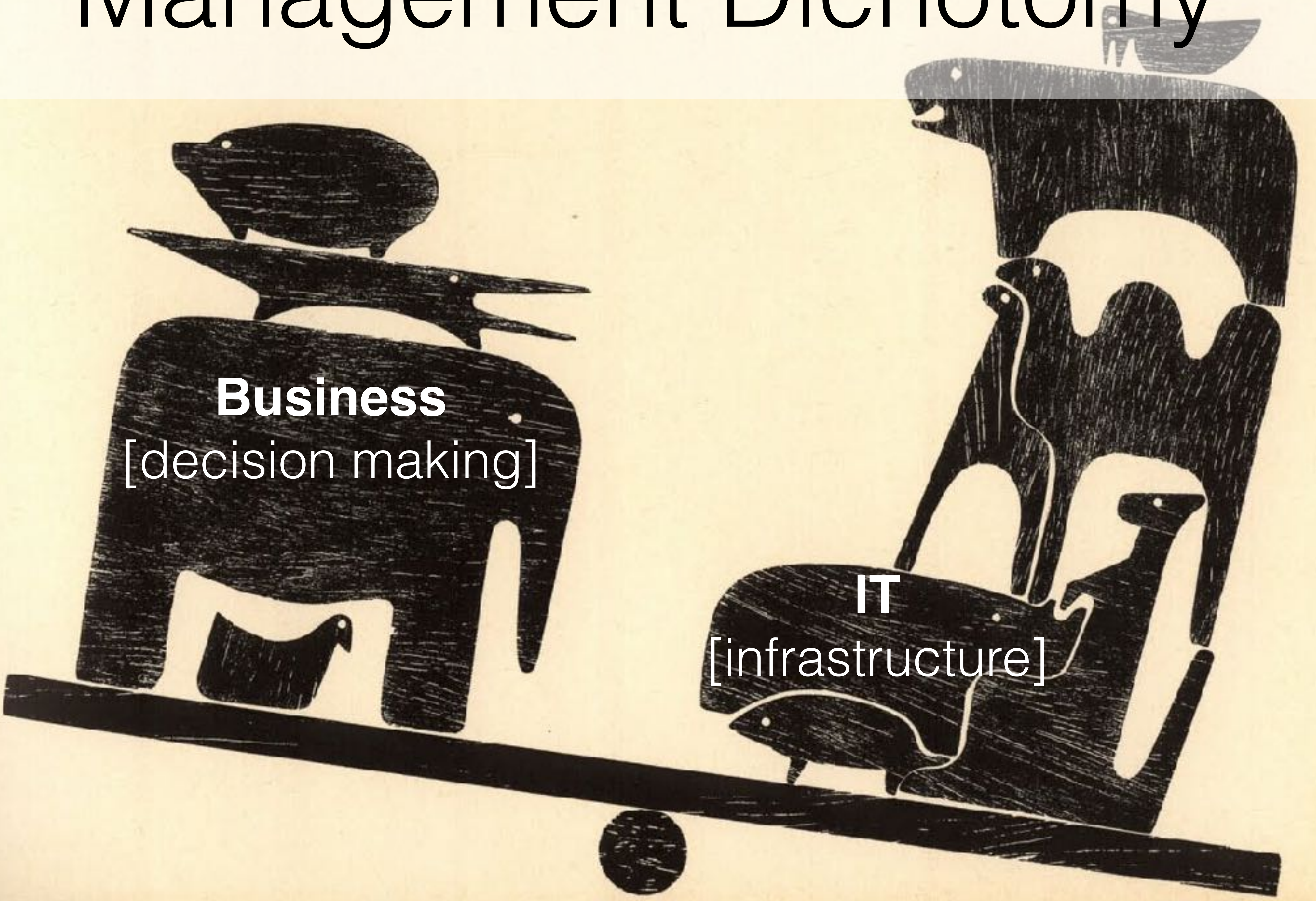
Experience Dichotomy

Management
[models]

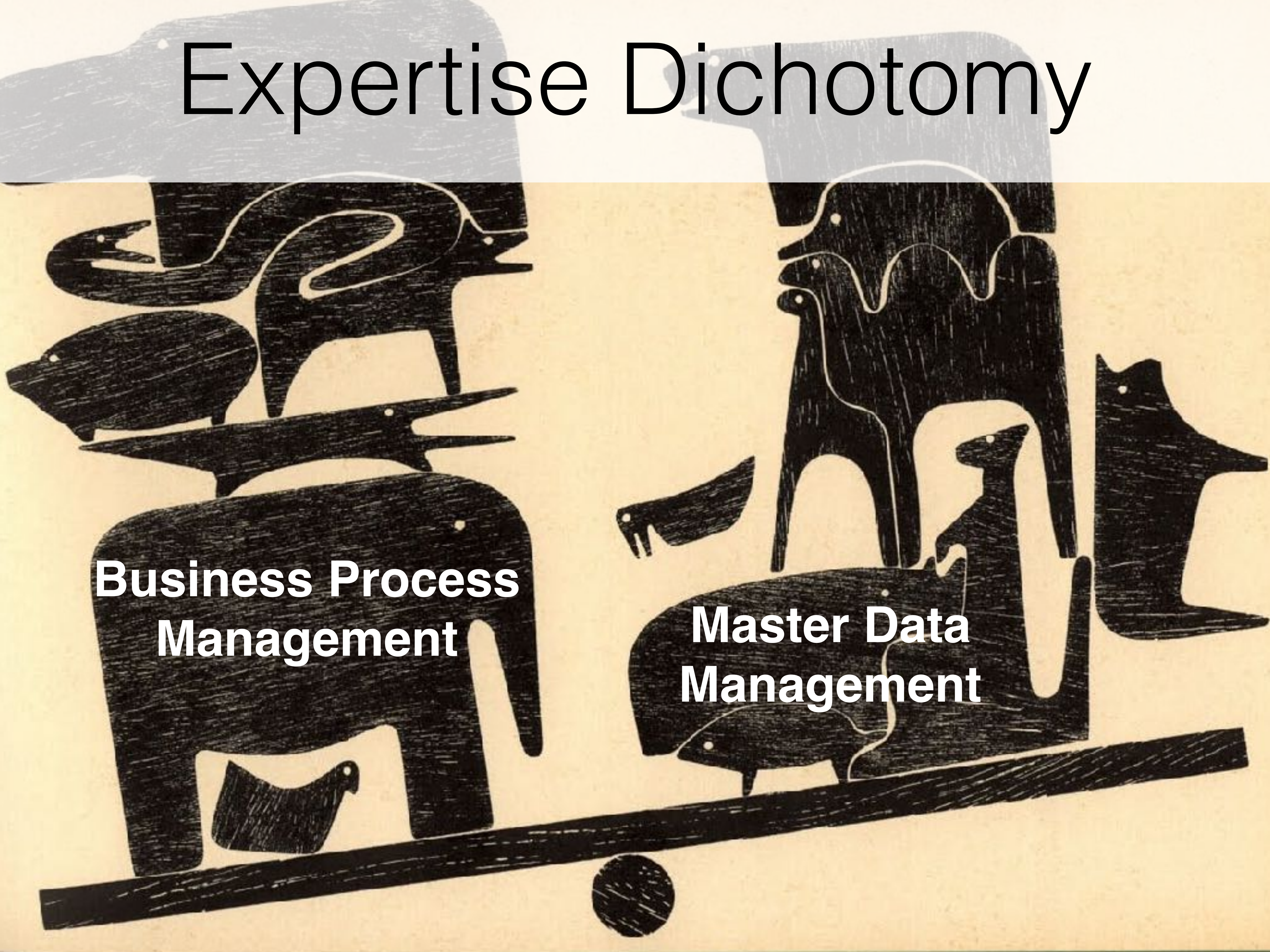
Workers
[reality]



Management Dichotomy



Expertise Dichotomy



A Successful Organization

Business Process

A set of **logically related tasks** performed to achieve a **defined business outcome** for a particular customer or market.

(Davenport, 1992)

A **collection of activities** that **take** one or more kinds of **input** and **create** an **output** that is **of value** to the customer.

(Hammer & Champy, 1993)

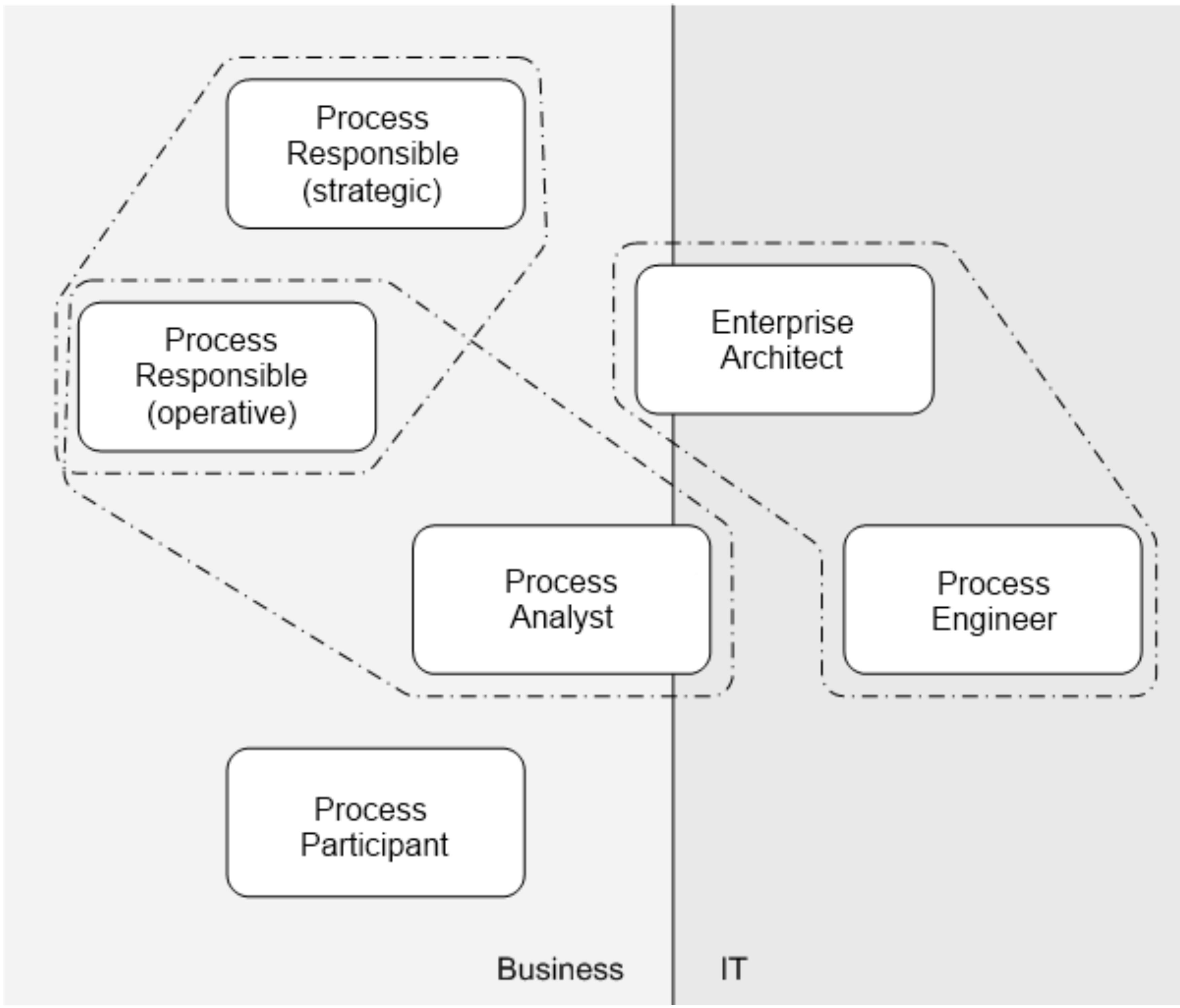
A **set of activities** performed in **coordination** in an **organizational** and **technical** environment. These activities **jointly realize a business goal**.

(Weske, 2011)

Business Process Management

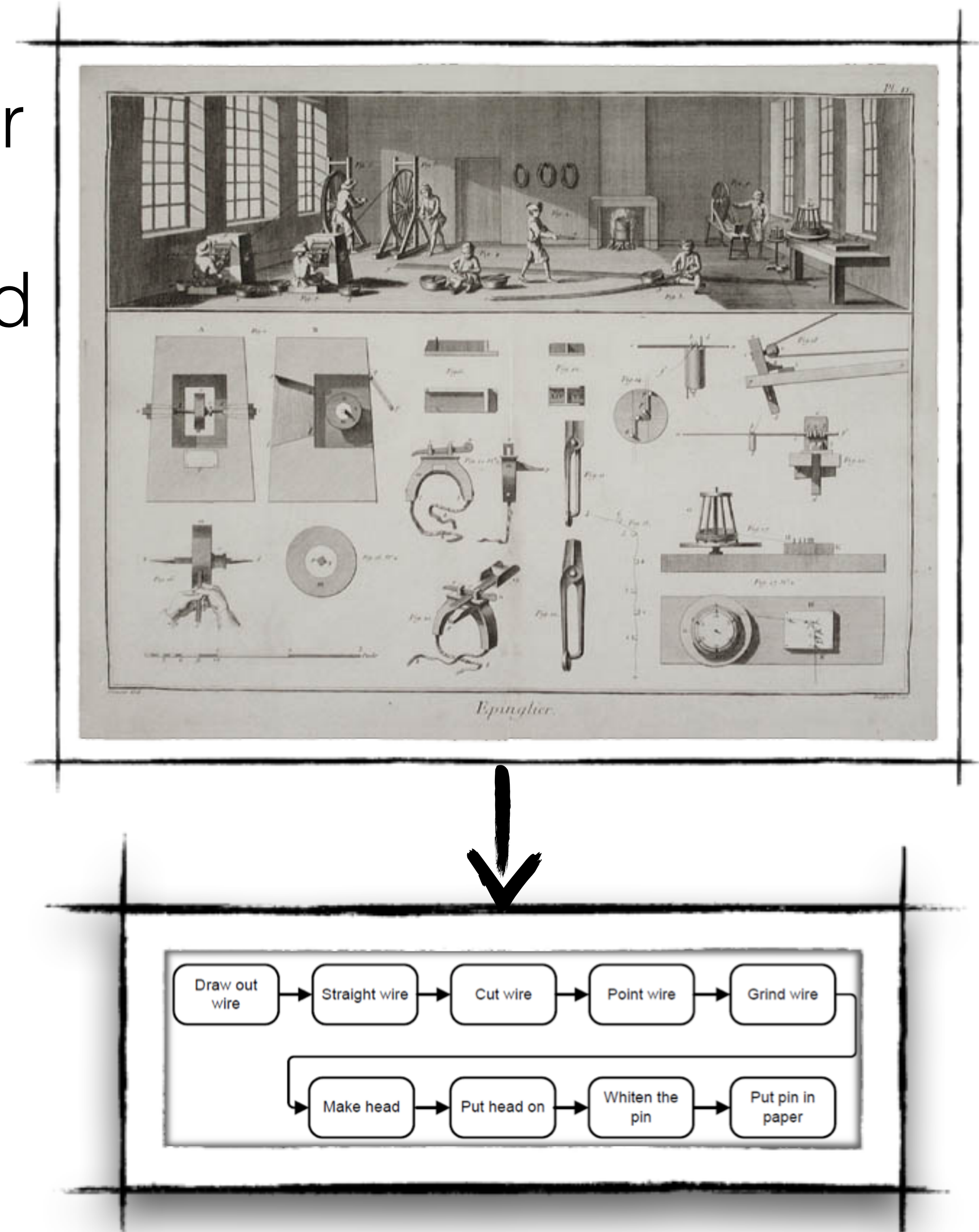
A collection of
concepts, **methods**, and **techniques**
to **support humans** in
modeling, **administration**,
configuration, **execution**,
analysis, and **continuous improvement**
of **business processes**

New Organisational Roles

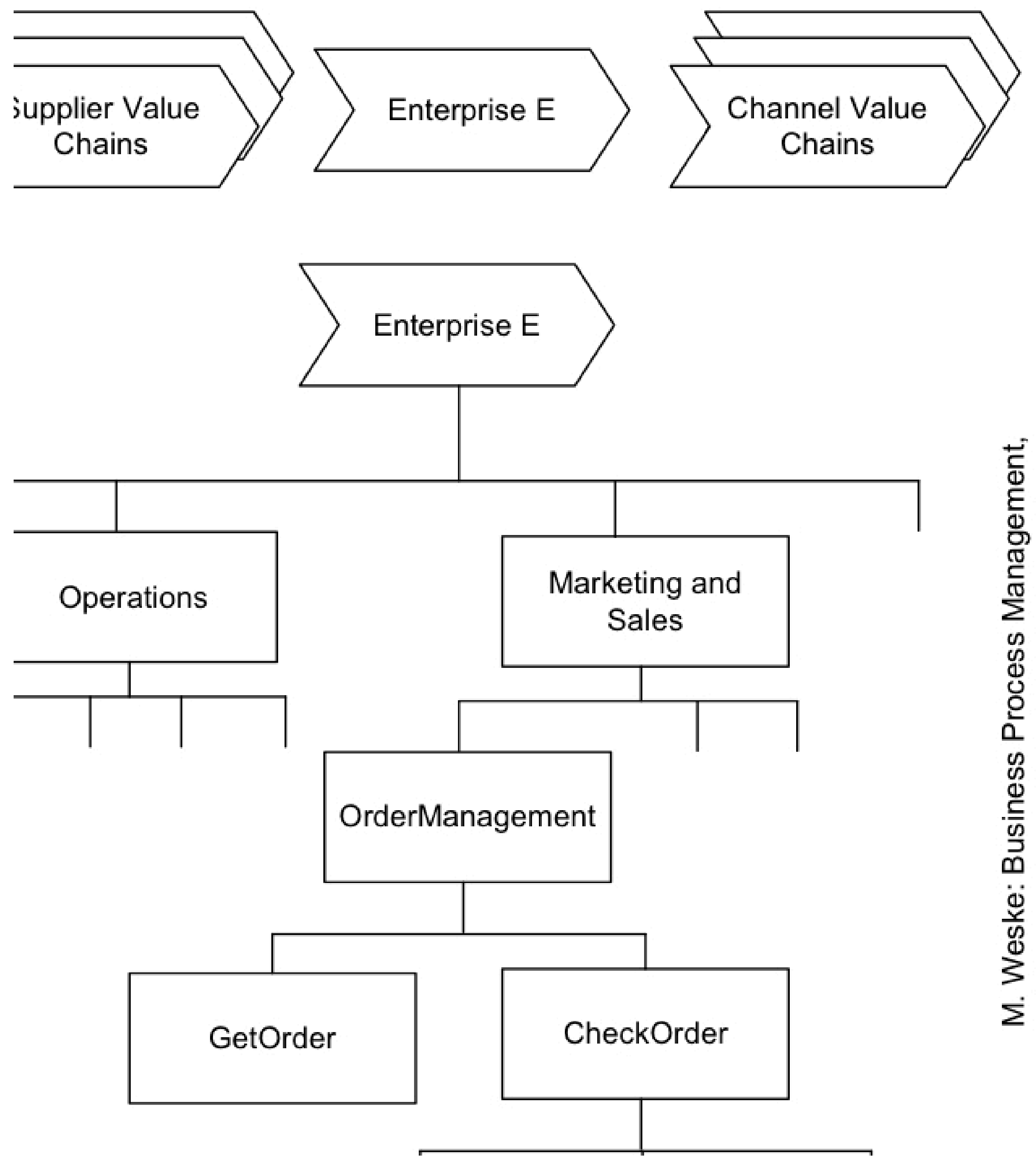


Short History

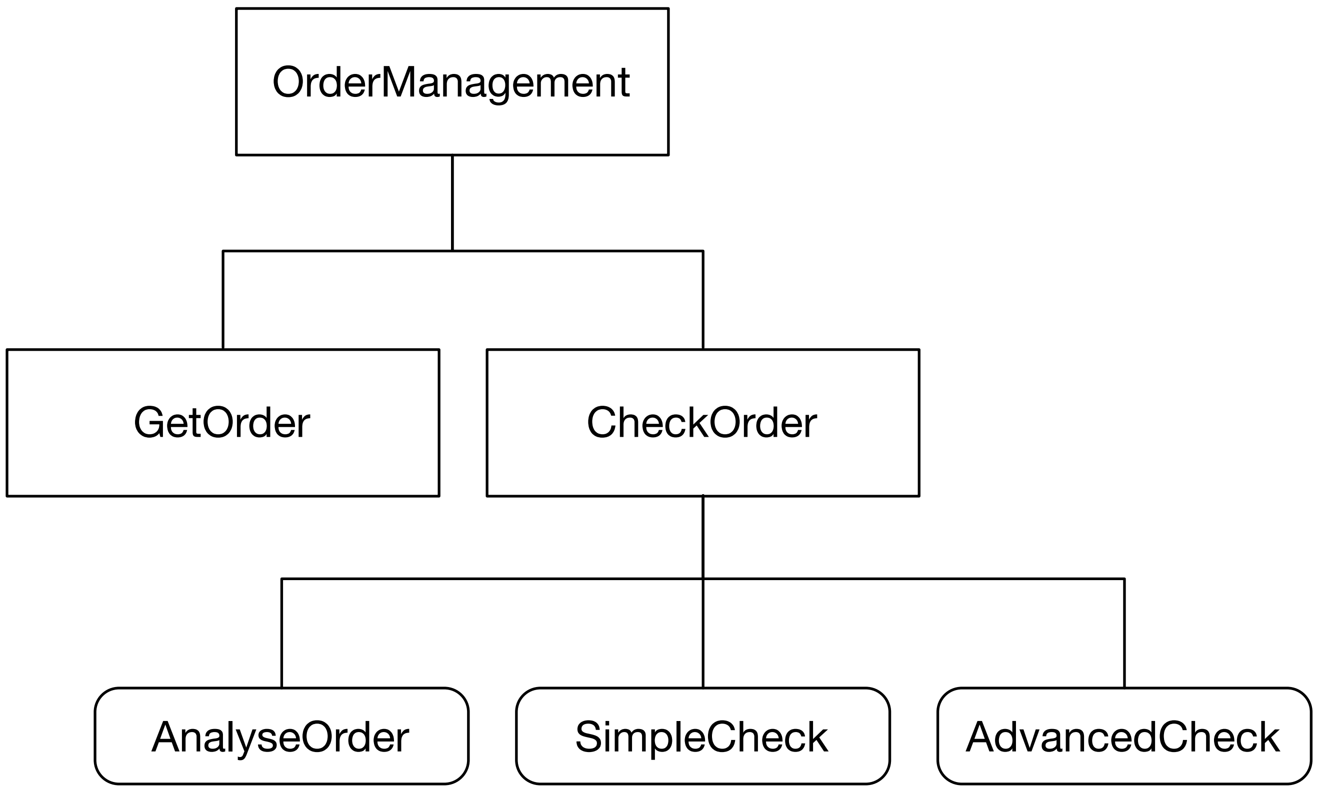
- Smith (~1750): division of labour
- Taylor (~1911): scientific method applied to organisations
- Hammer and Champy (~1990): processes as the basis for reengineering
- 2000s: business process lifecycle, process-orientation



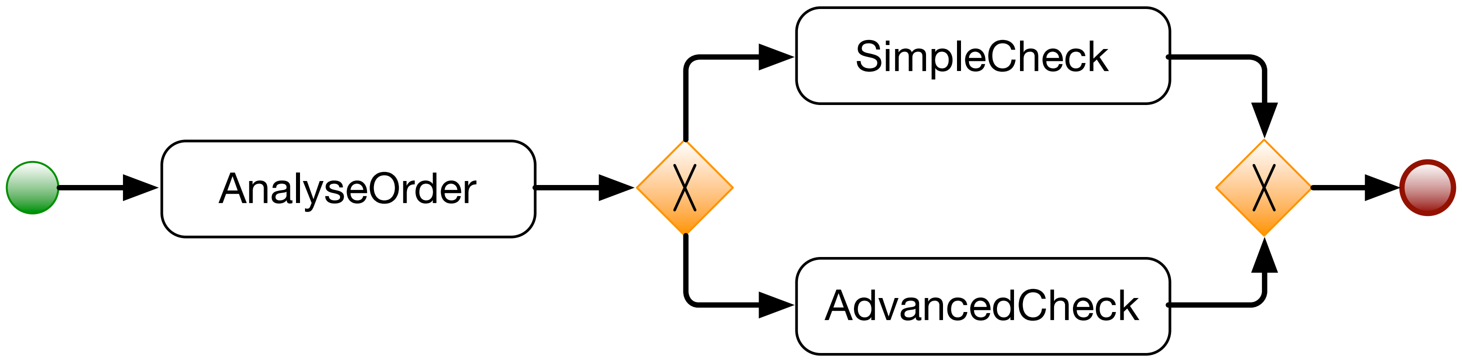
Value Chains, Business Functions, Tasks



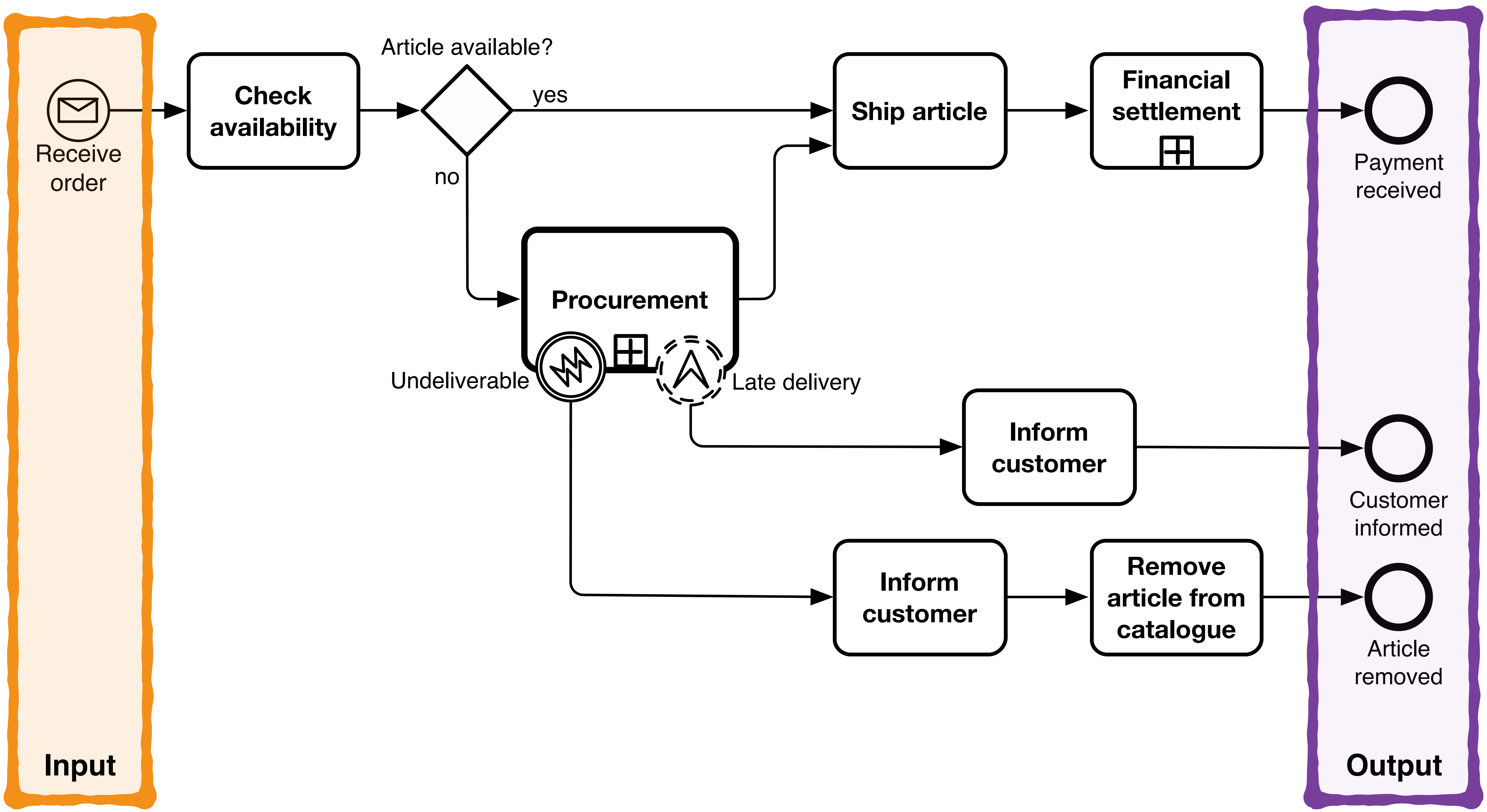
From tasks…



… to their coordination

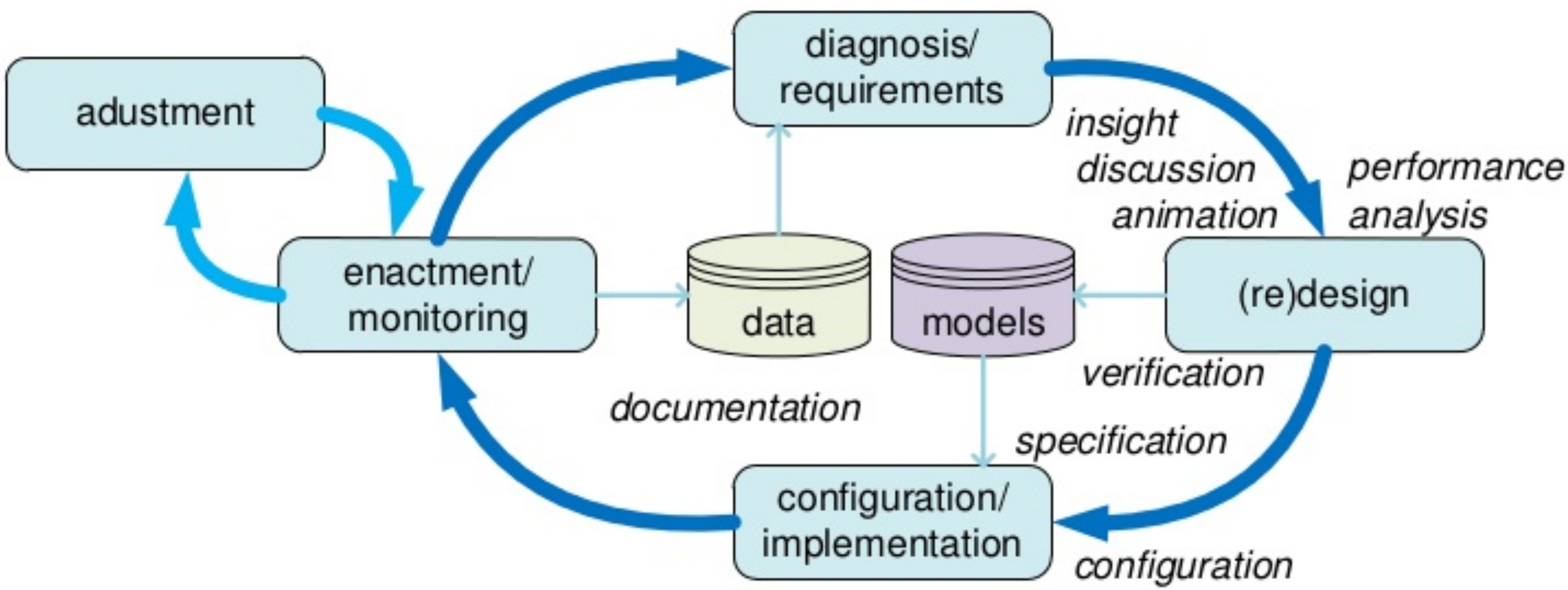


End-To-End, Reactive Behaviour



Order-to-cash, procure-to-pay, issue-to-resolution, …

Business Process Lifecycle



picture by Wil van der Aalst

Two Questions

How to **formally** and **conceptually** account for the **process+data** interplay?

How to verify such **BPMs**?

Two Questions

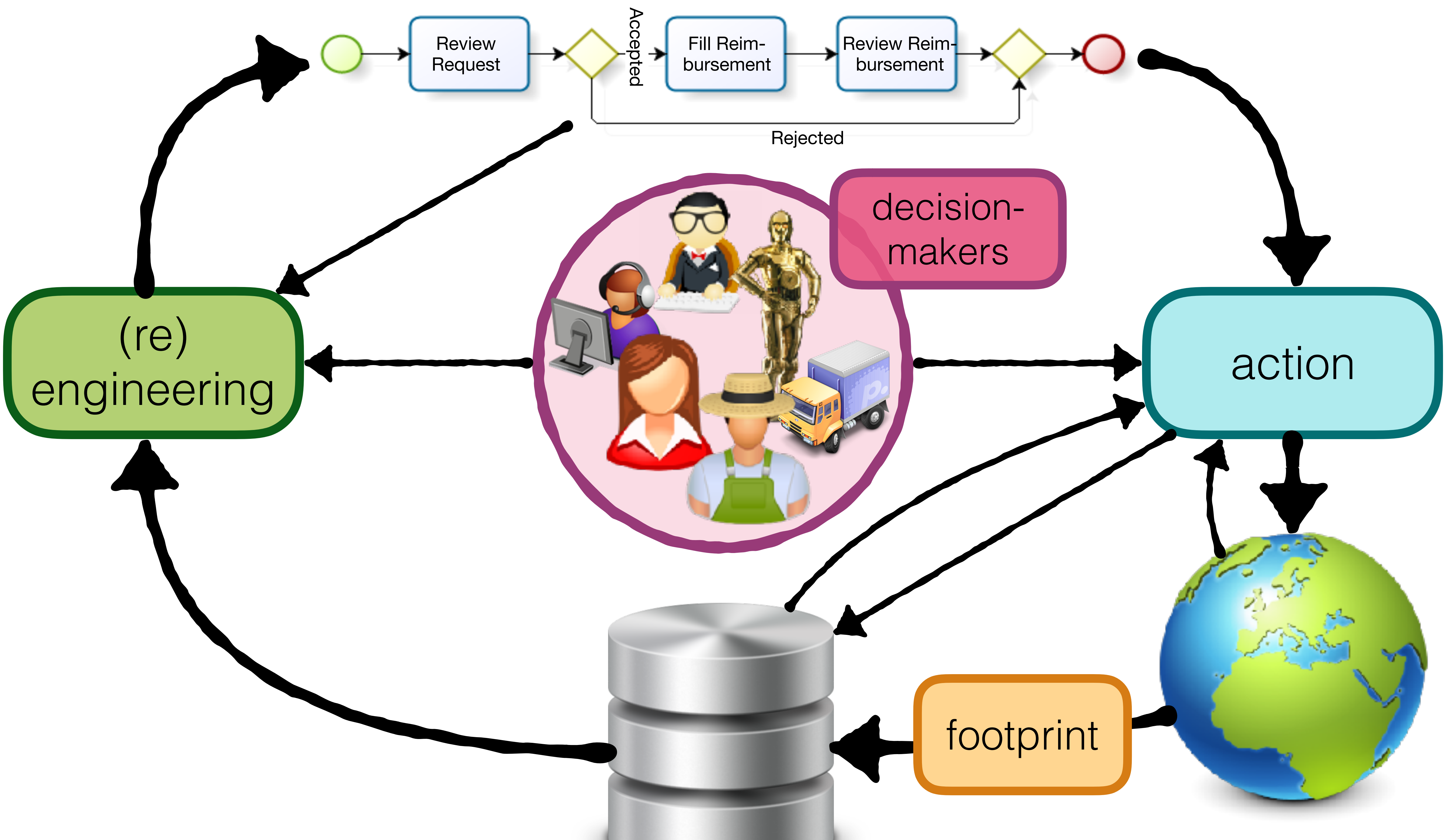
How to **formally** and **conceptually** account for the **process+data** interplay?

BTMs

How to verify such ~~**BPMs**~~?

Business Turing Machines

Data and Processes

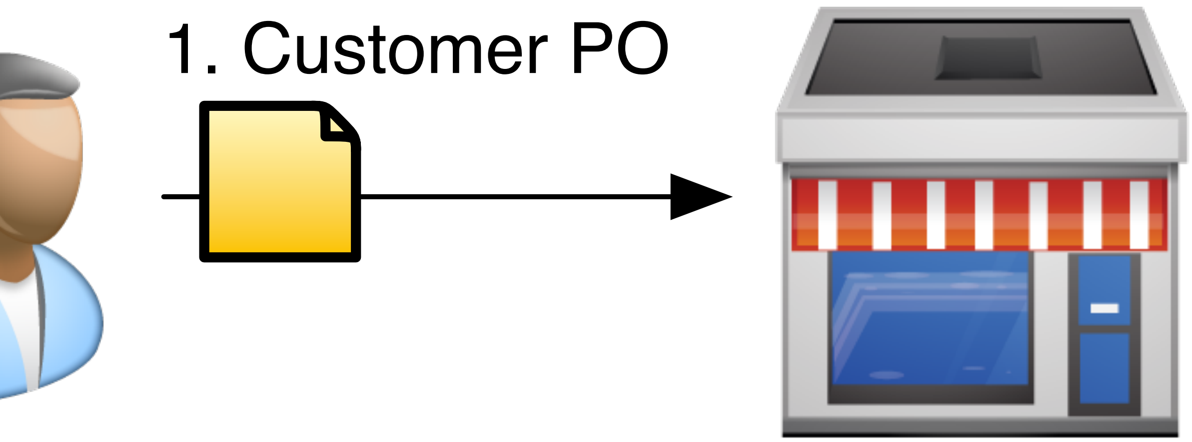


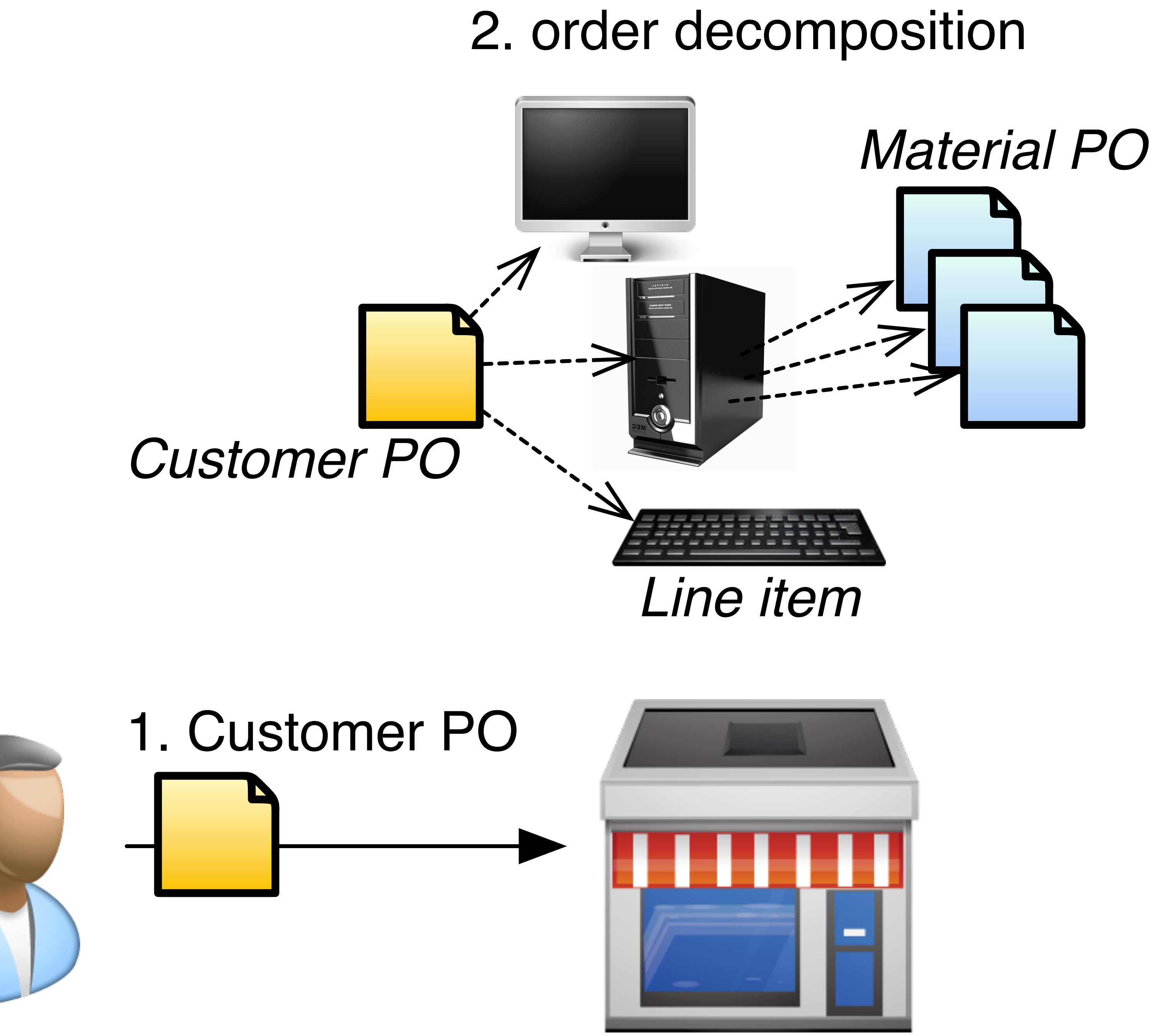
Is this Synergy Reflected by Models?

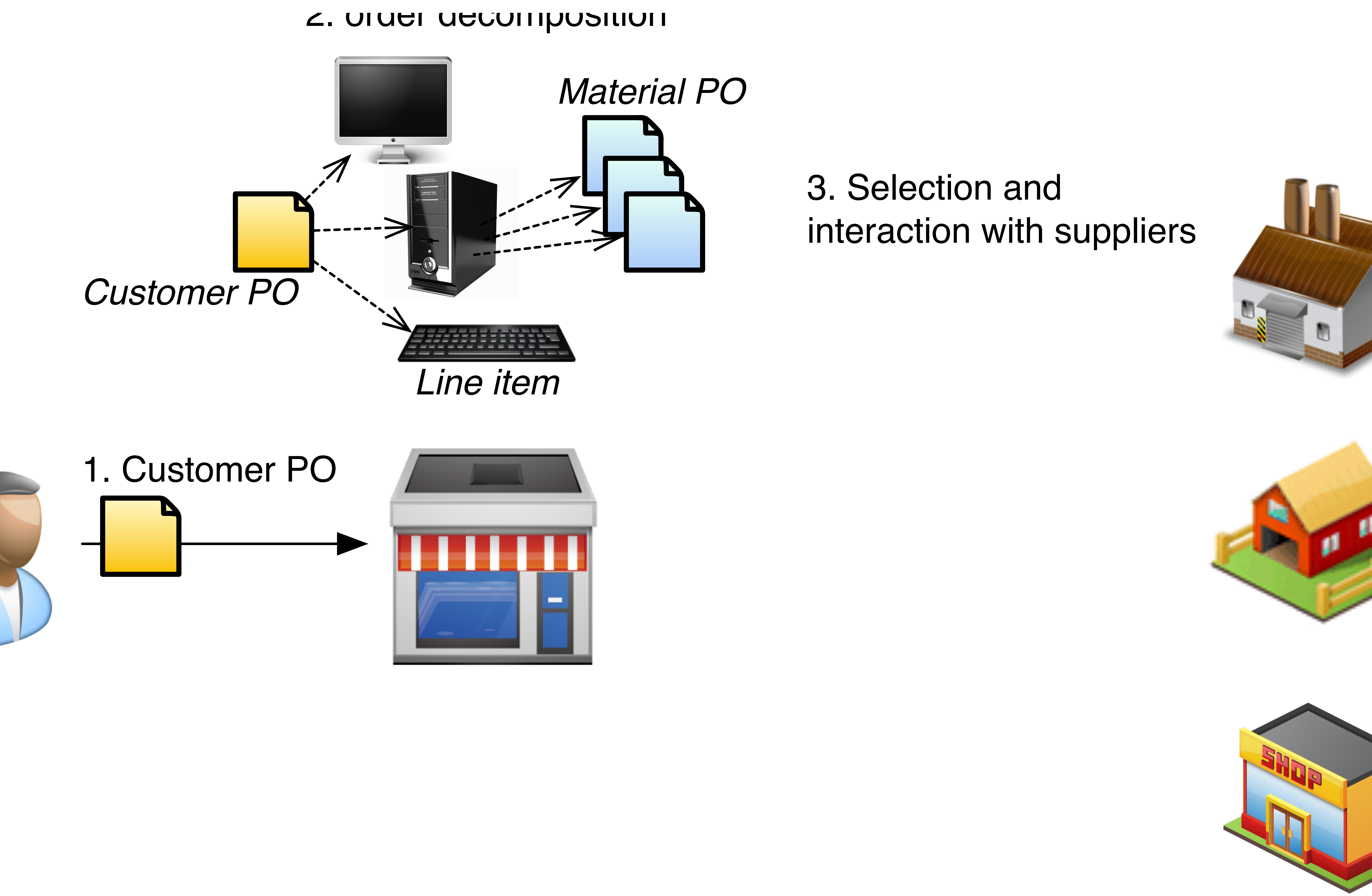
Survey by *Forrester* [Karel et al, 2009]: **lack of interaction between data and process experts**.

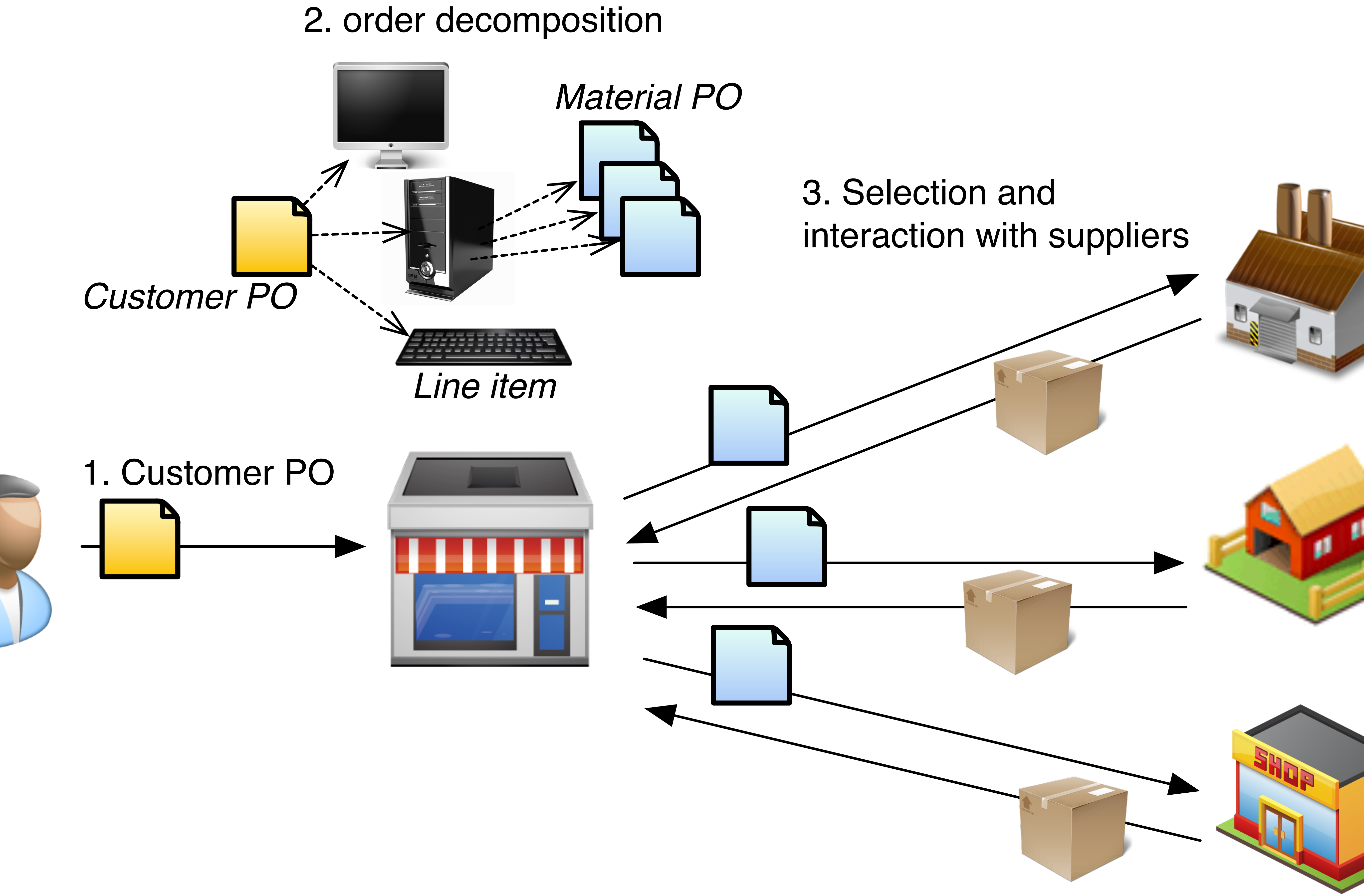
- *BPM professionals*: **data are subsidiary to processes**
- *Master data managers*: **data are the main driver** for the company's existence
- 83/100 companies: **no interaction at all** between these two groups
- This isolation propagates to models, languages and tools

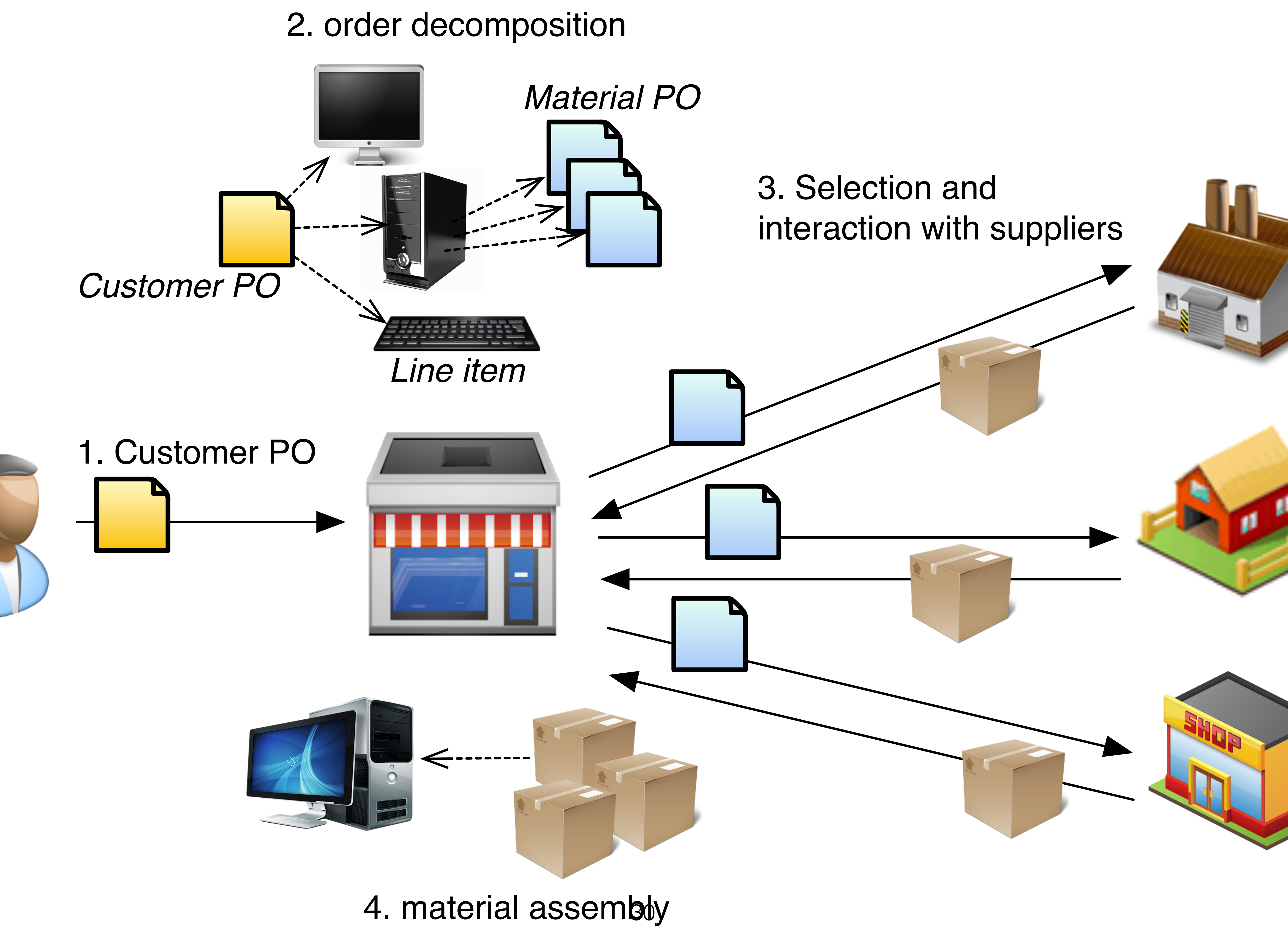
Example: Order-To-Delivery

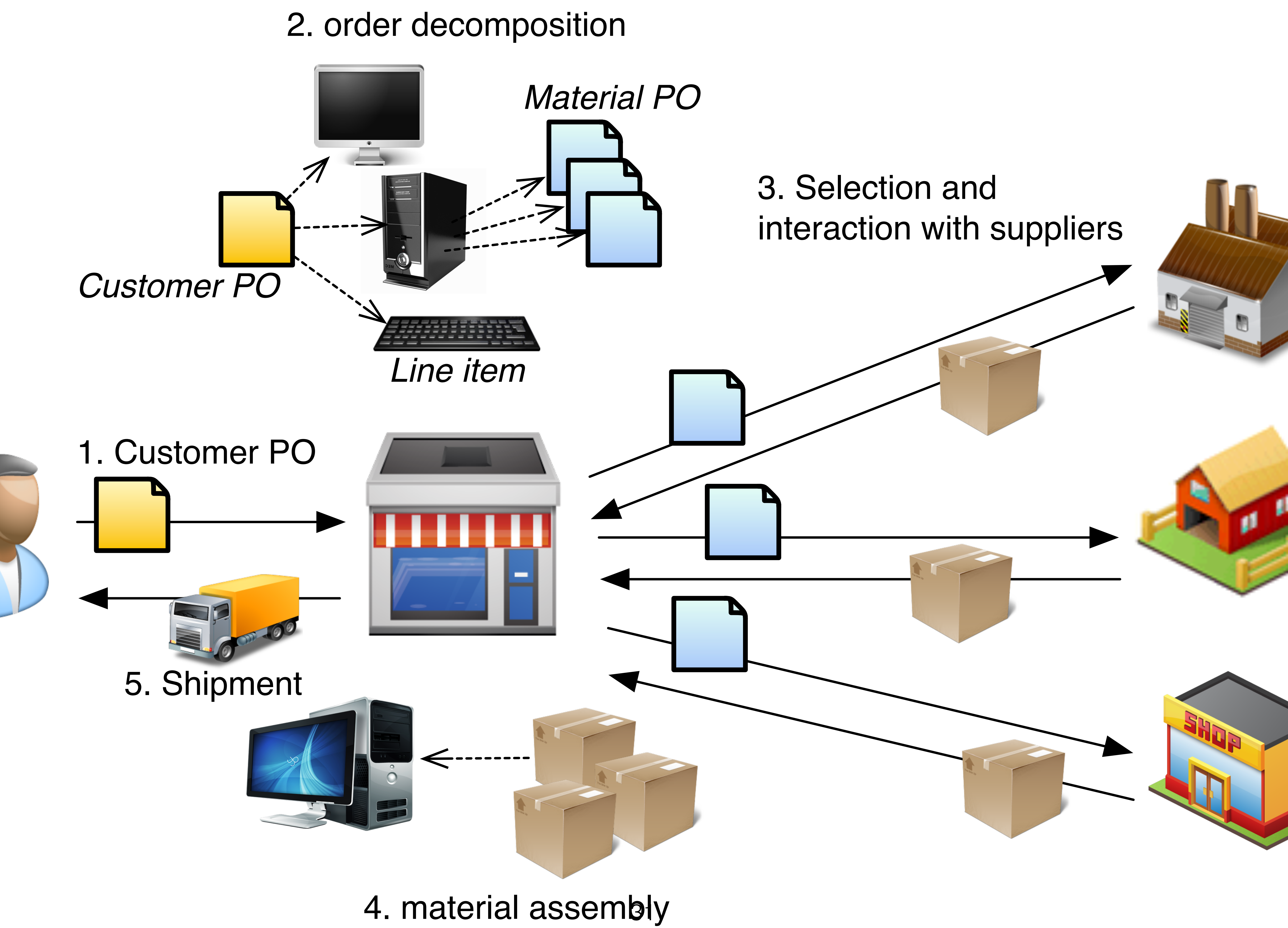



2. order decomposition
Material PO
Customer PO
Line item
1. Customer PO


2. order decomposition
Material PO
3. Selection and
interaction with suppliers
Customer PO
Line item
1. Customer PO
SHOP


2. order decomposition
Material PO
3. Selection and
interaction with suppliers
Customer PO
Line item
1. Customer PO
SHOP


2. order decomposition
Material PO
Customer PO
Line item
3. Selection and
interaction with suppliers
1. Customer PO
SHOP
4. material assembly


2. order decomposition
Material PO
Customer PO
Line item
3. Selection and
interaction with suppliers
1. Customer PO
5. Shipment
SHOP
4. material assembly

Observations

- A complex process, where the company acts as an intermediate hub between customers and suppliers

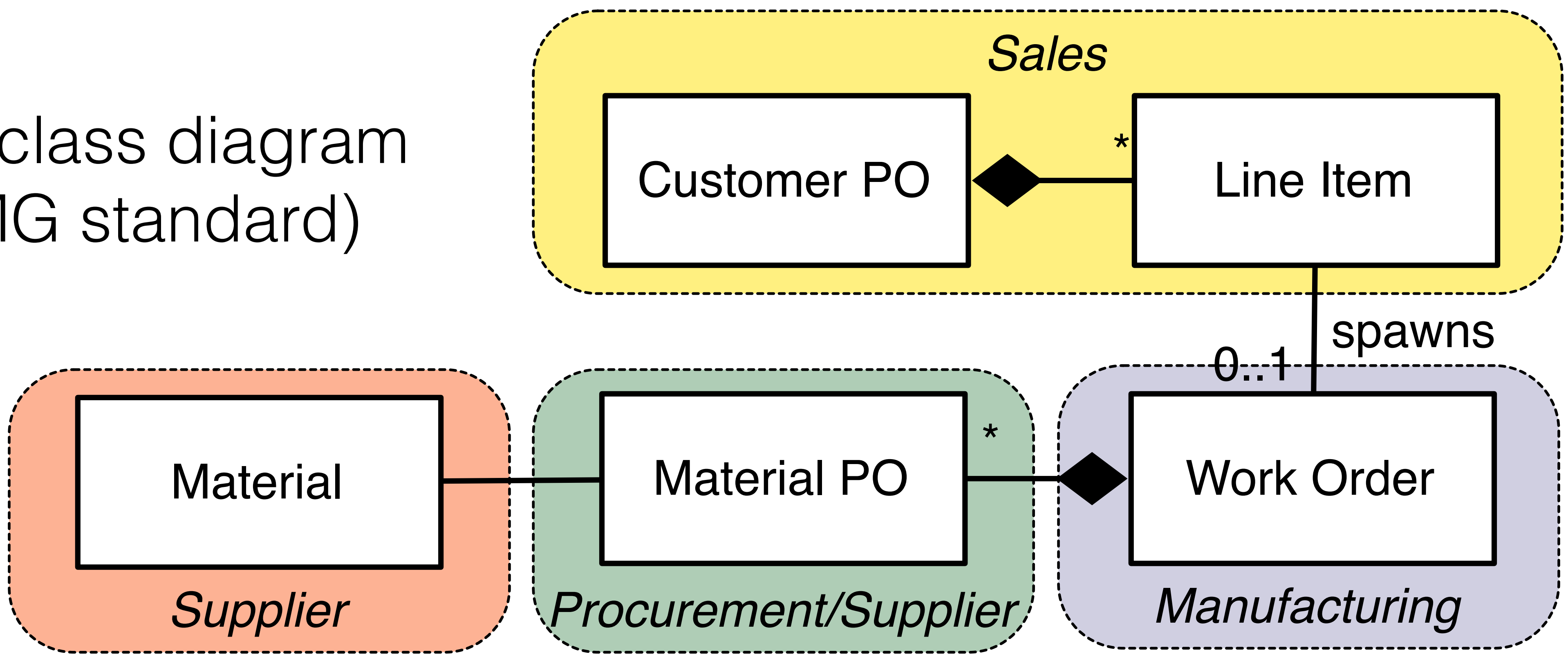
- **Happy path**
  1) The customer issues a purchase order
  2) The ordered material is obtained from suppliers
  3) The material is shipped, possibly using different packages

- One **exceptional path** (in general, there are many):
  1) The customer cancels the order
  2) A **cancelation policy** is applied to calculate a penalty

Conventional Data Modeling

Focus: revelant entities, relations, *static* constraints

UML class diagram
(OMG standard)

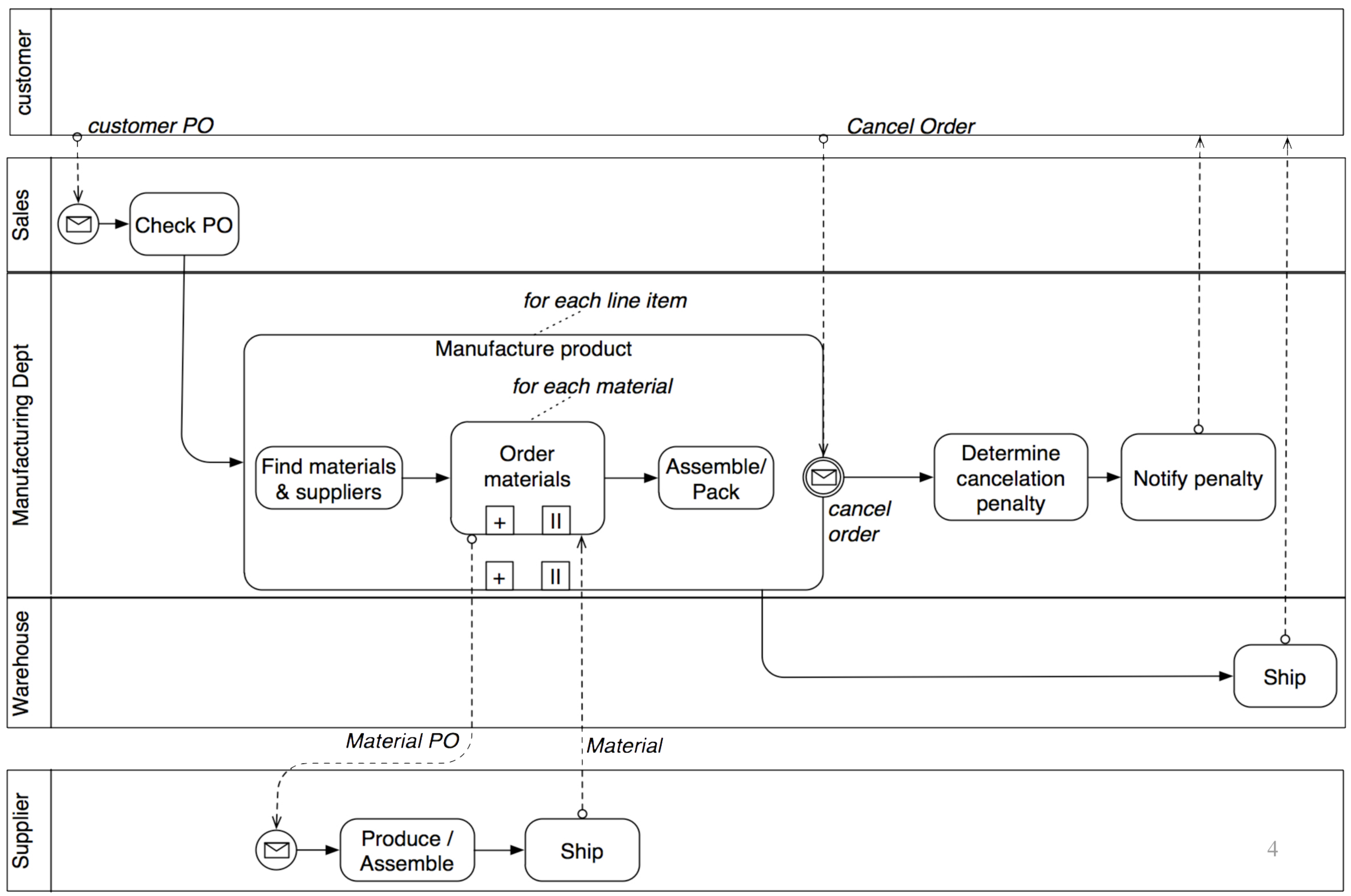


But… how do data evolve?
Where can we find the “state” of a purchase order?

Conventional Process Modeling

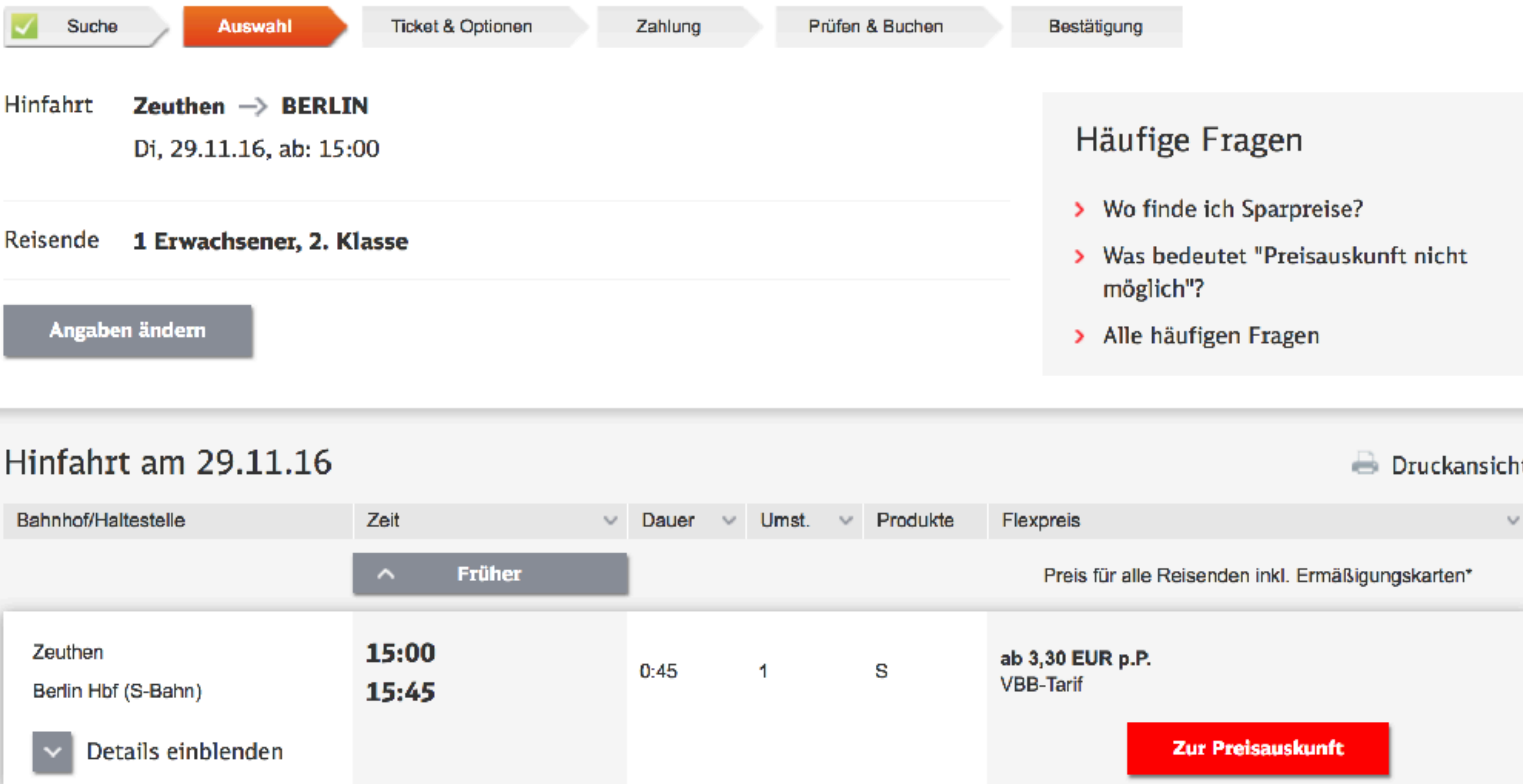
Focus: control-flow of activities in response to events

BPMN
collaborative
process
diagram
(OMG standard)

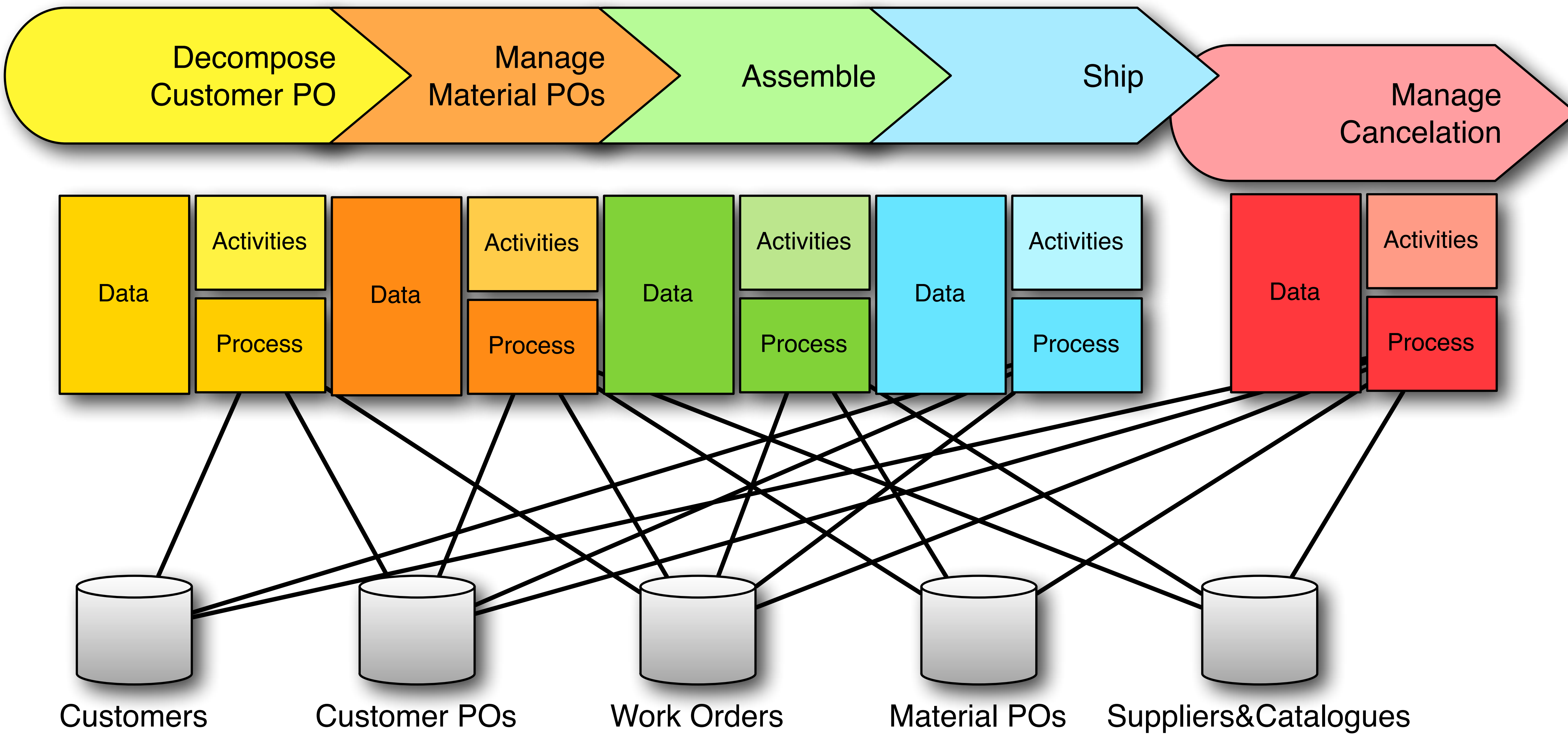


But… how do activities update data?
What is the impact of canceling an order?

A Deployed Process



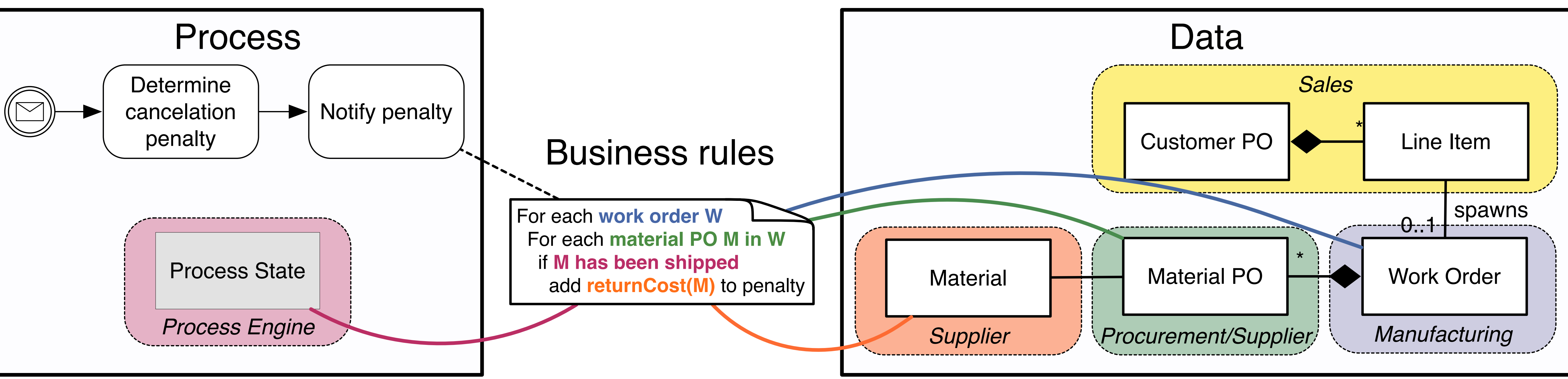
Do you like Spaghetti?



IT integration: difficult to manage, understand, maintain

Too Late!

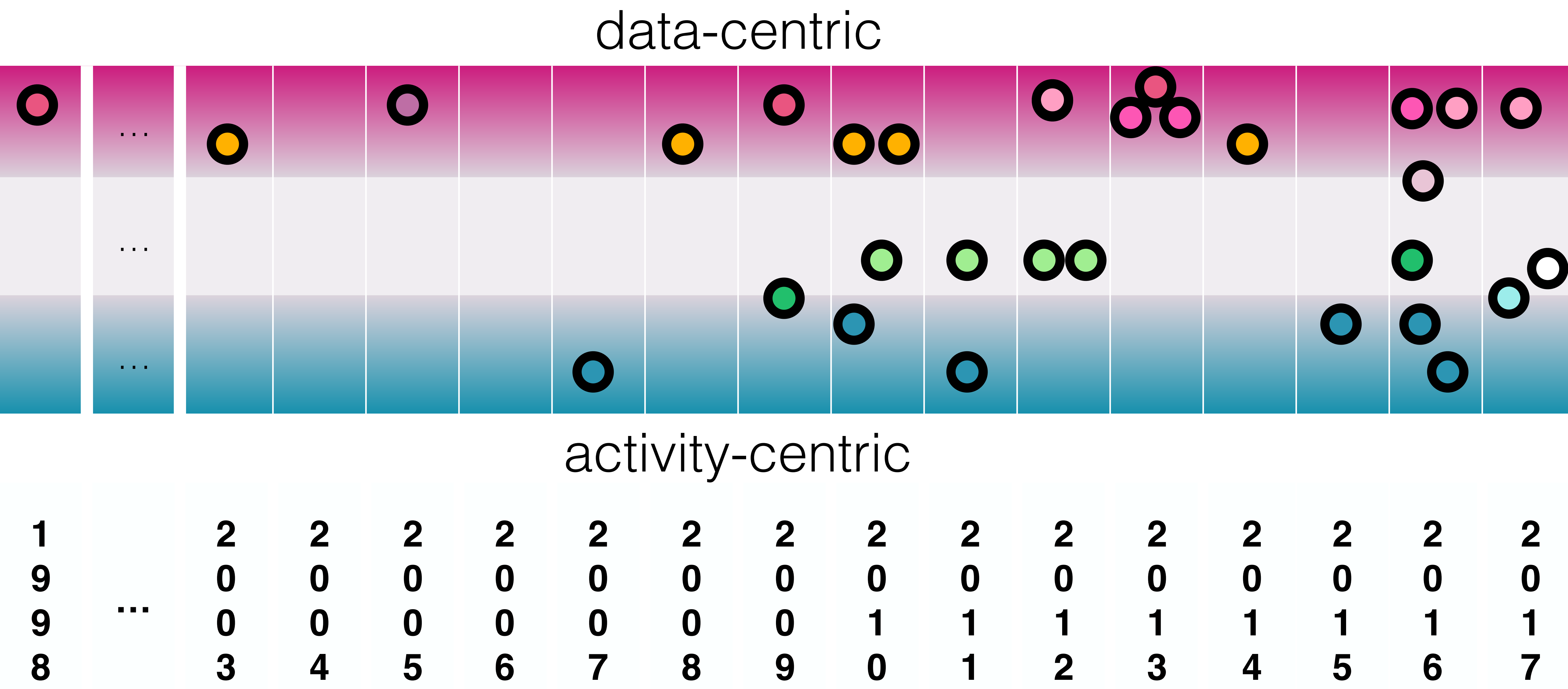
- Where are the data?
- Where shall we model relevant business rules?
- Consider an **order cancelation policy** that needs to check which material has been already shipped towards determining the customer penalty…

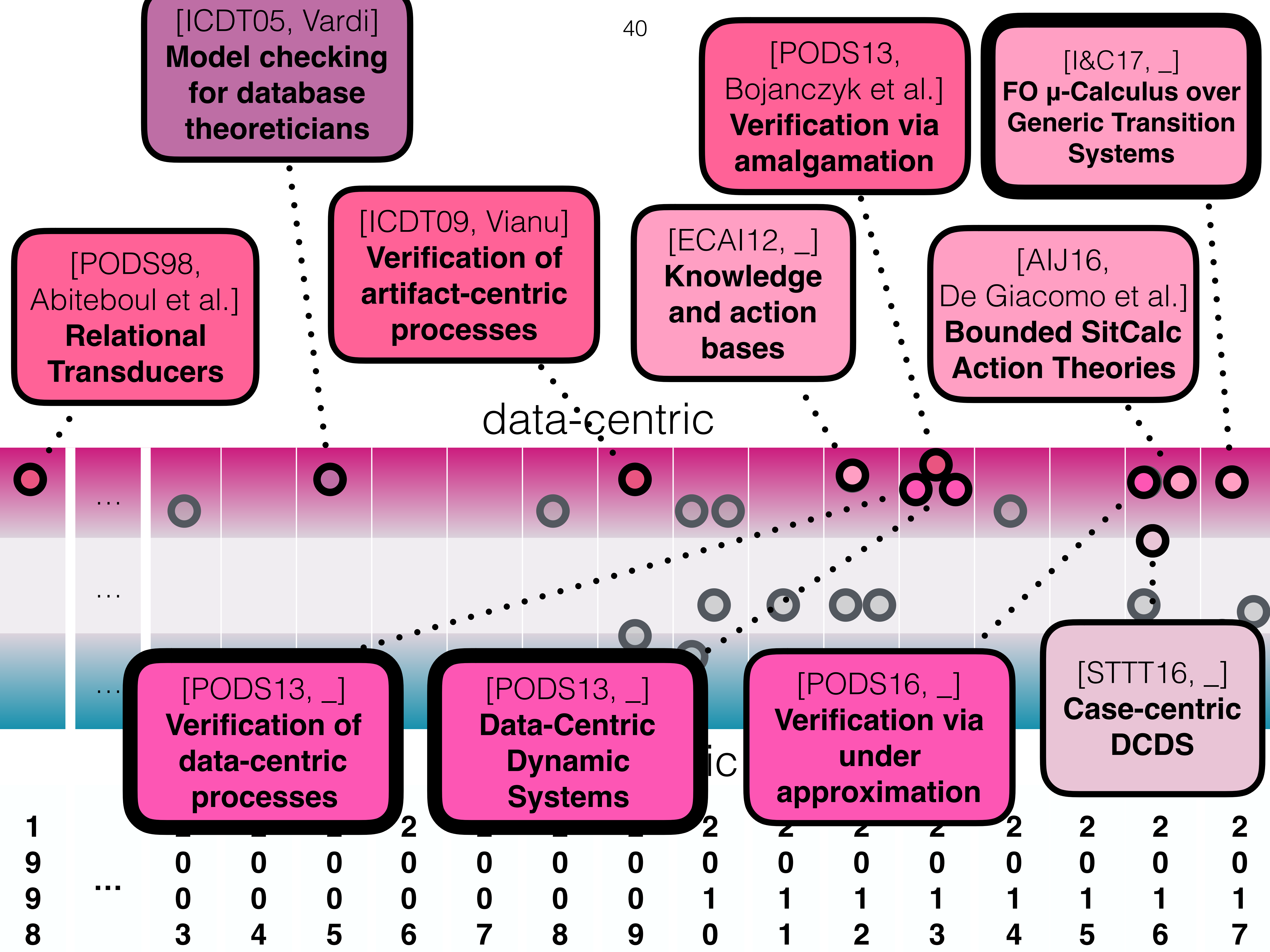


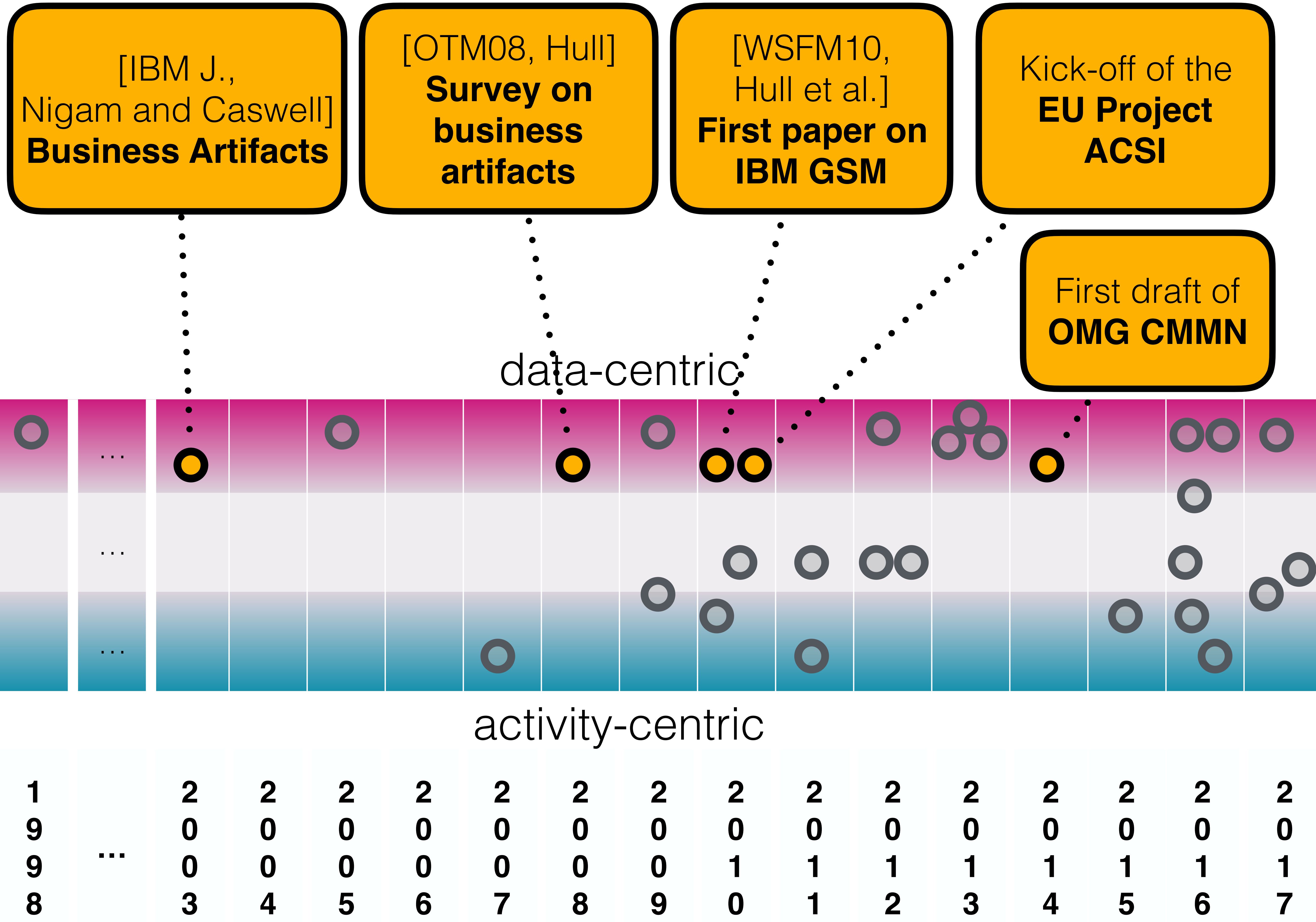

A LONG TIME AGO IN A GALAXY FAR FAR AWAY...
STAR WARS
A NEW HOPE
THX

N.B.: these are “sparse” dots!!!

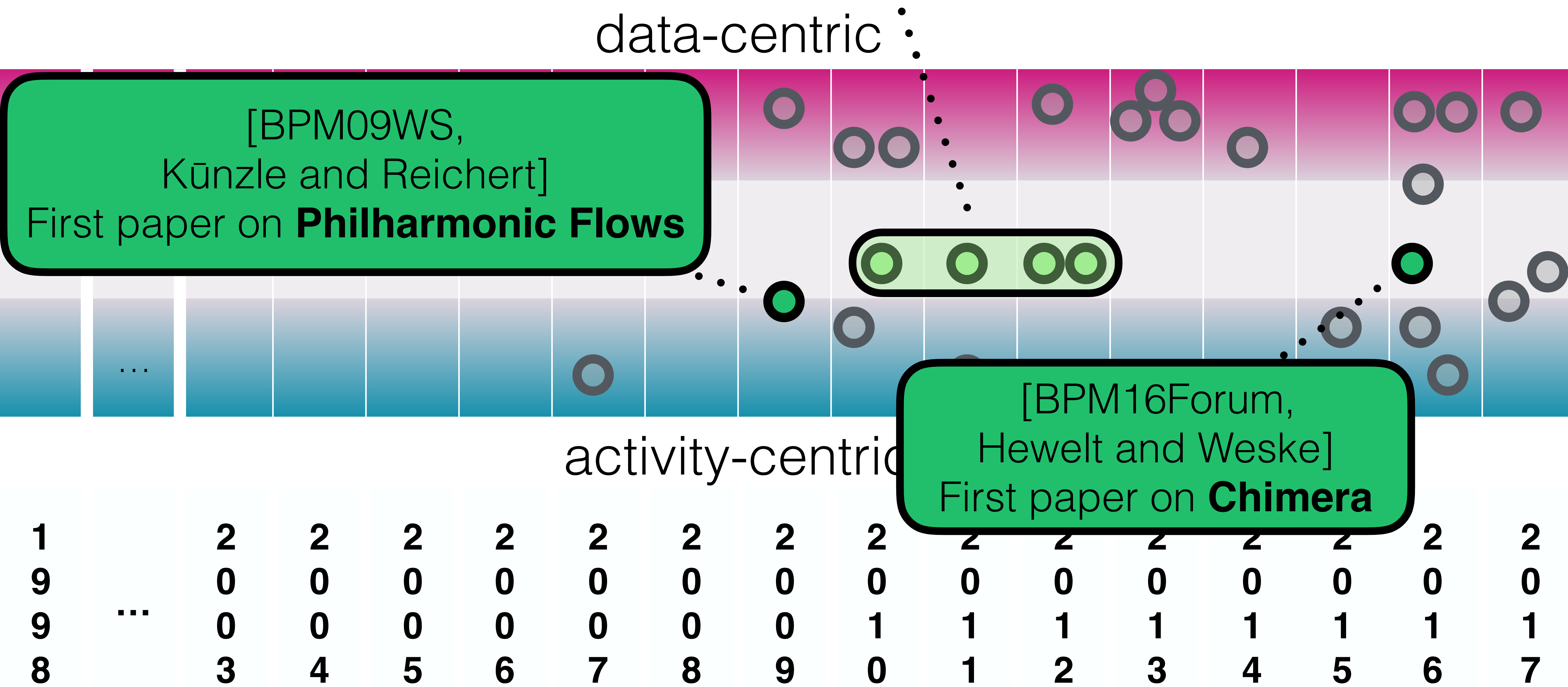
...There is Hope!

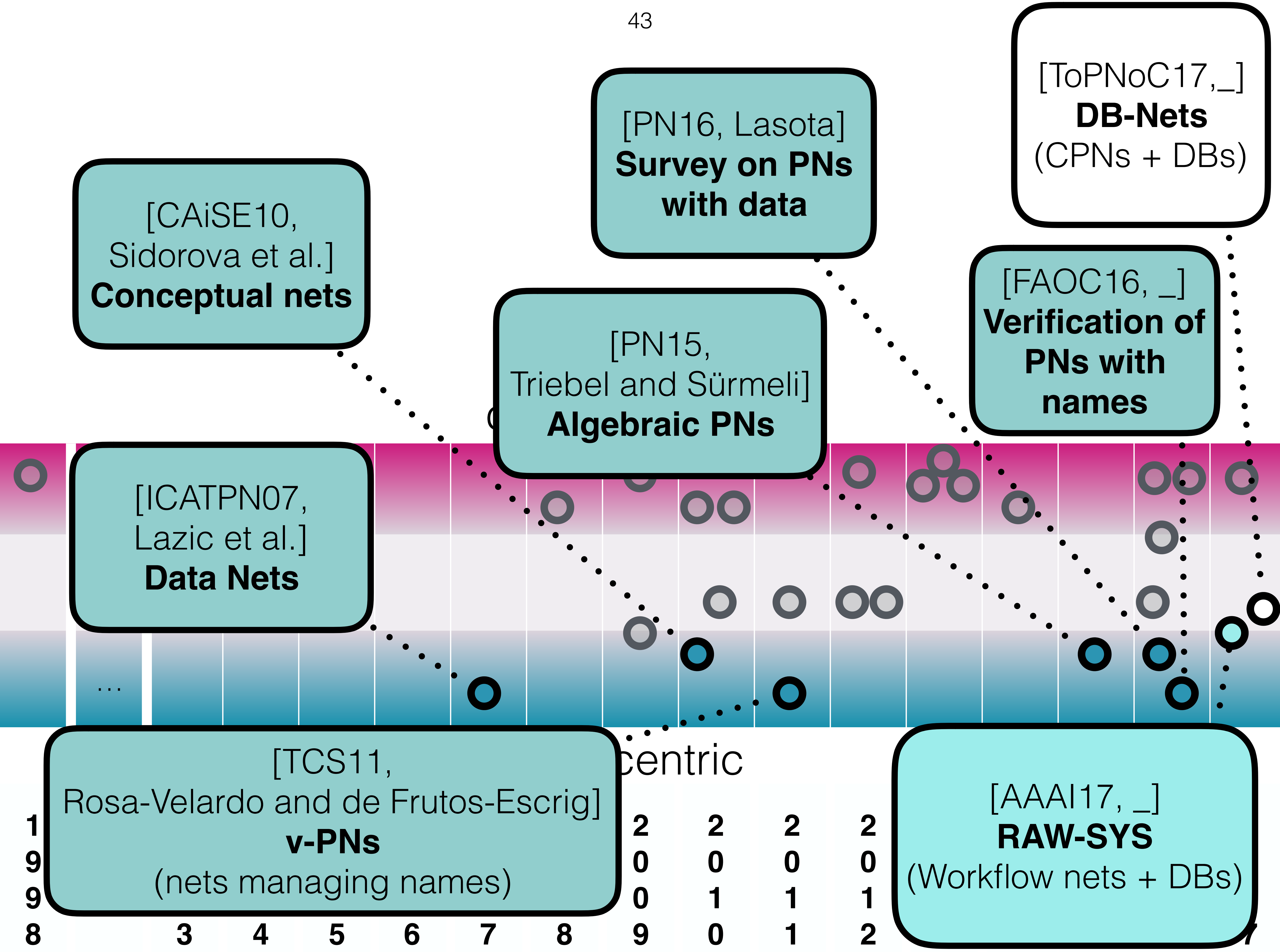



[ICDT05, Vardi]
Model checking for database theoreticians
[PODS13, Bojanczyk et al.]
Verification via amalgamation
[I&C17, _]
FO μ-Calculus over Generic Transition Systems
[PODS98, Abiteboul et al.]
Relational Transducers
[ICDT09, Vianu]
Verification of artifact-centric processes
[ECAI12, _]
Knowledge and action bases
[AIJ16, De Giacomo et al.]
Bounded SitCalc Action Theories
data-centric
[PODS13, _]
Verification of data-centric processes
[PODS13, _]
Data-Centric Dynamic Systems
[PODS16, _]
Verification via under approximation
[STTT16, _]
Case-centric DCDS
1998
...
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017


[IBM J., Nigam and Caswell] Business Artifacts
[OTM08, Hull] Survey on business artifacts
[WSFM10, Hull et al.] First paper on IBM GSM
Kick-off of the EU Project ACSI
First draft of OMG CMMN
data-centric
activity-centric
1998
...
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017

- [BPM2010, Richardson]: **BPM vs master data dichotomy**
- **Data+Process integration** key to:
  - assess **value of processes** and **evaluate KPIs** [Meyer et al, 2011]
  - **aggregate** relevant **info**, elicit **business rules** [ABDIS11, Dumas]
- [Reichert, 2012]: **“Process and data are just two sides of the same coin”**

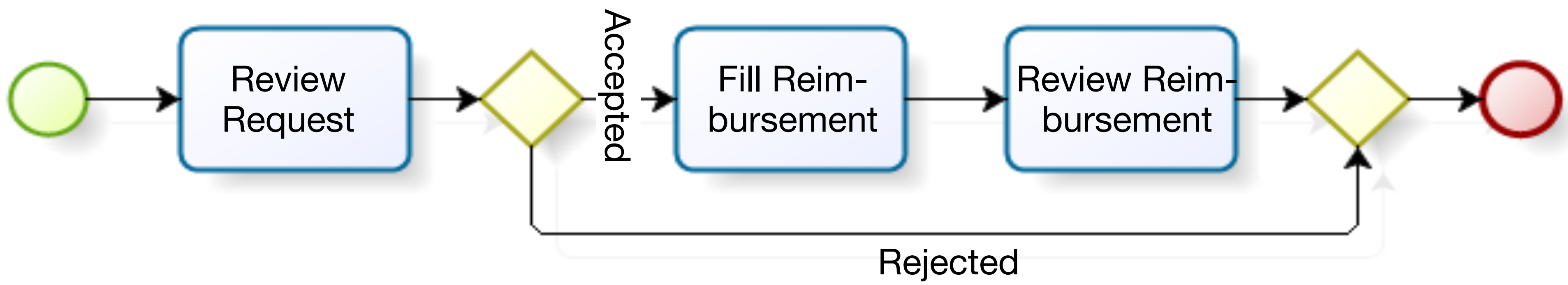



[ToPNoC17,_]
DB-Nets
(CPNs + DBs)
[PN16, Lasota]
Survey on PNs with data
[CAiSE10, Sidorova et al.]
Conceptual nets
[FAOC16, _]
Verification of PNs with names
[PN15, Triebel and Sürmeli]
Algebraic PNs
[ICATPN07, Lazic et al.]
Data Nets
...
[TCS11, Rosa-Velardo and de Frutos-Escrig]
v-PNs
(nets managing names)
centric
[AAAI17, _]
RAW-SYS
(Workflow nets + DBs)
1998
3
4
5
6
7
8
2009
2010
2011
2012
7

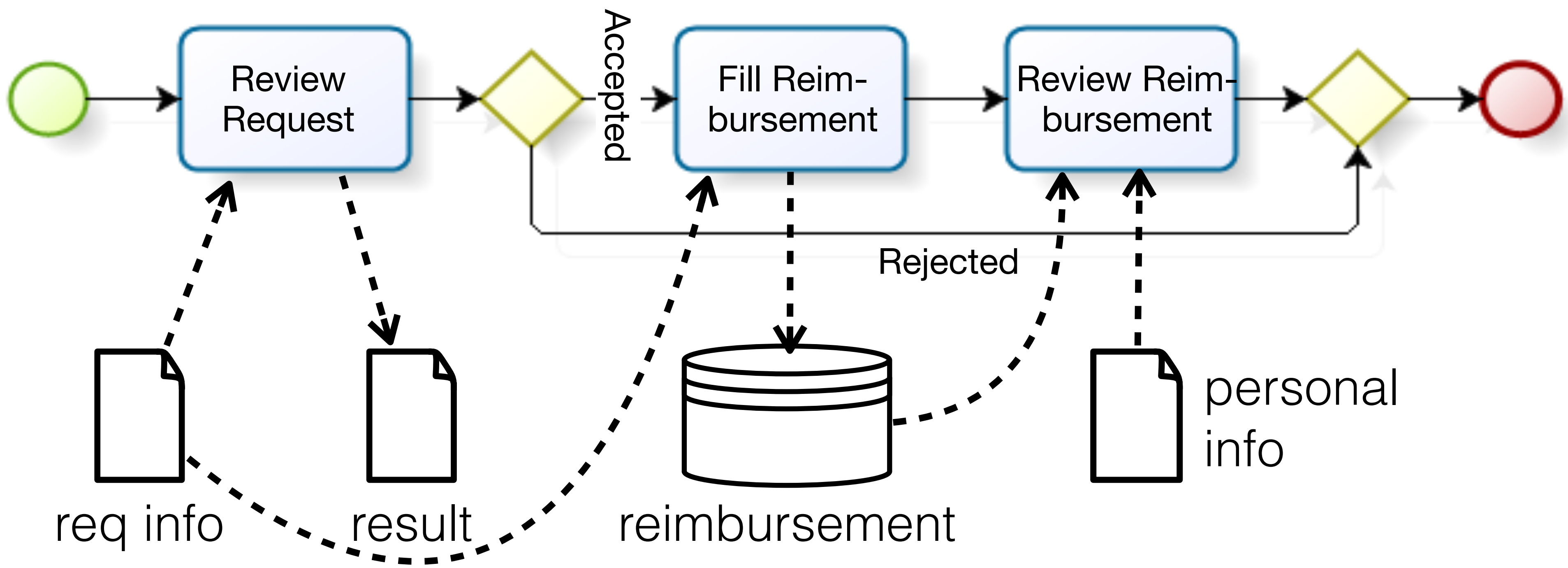
One Step Back…

How do
contemporary
activity-centric BPMSs
account for the
process-data interplay?

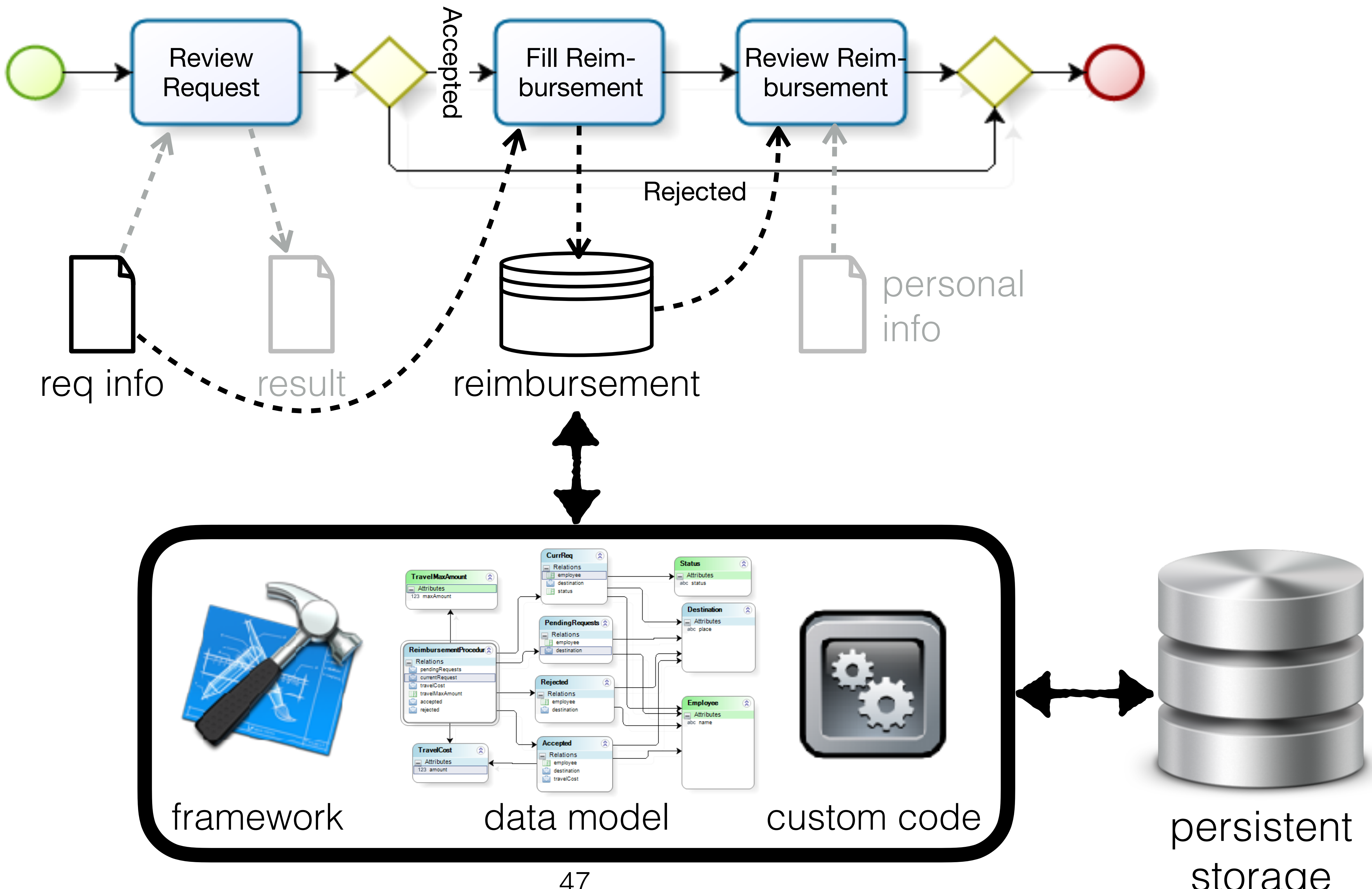
Example: BizAgi (~)



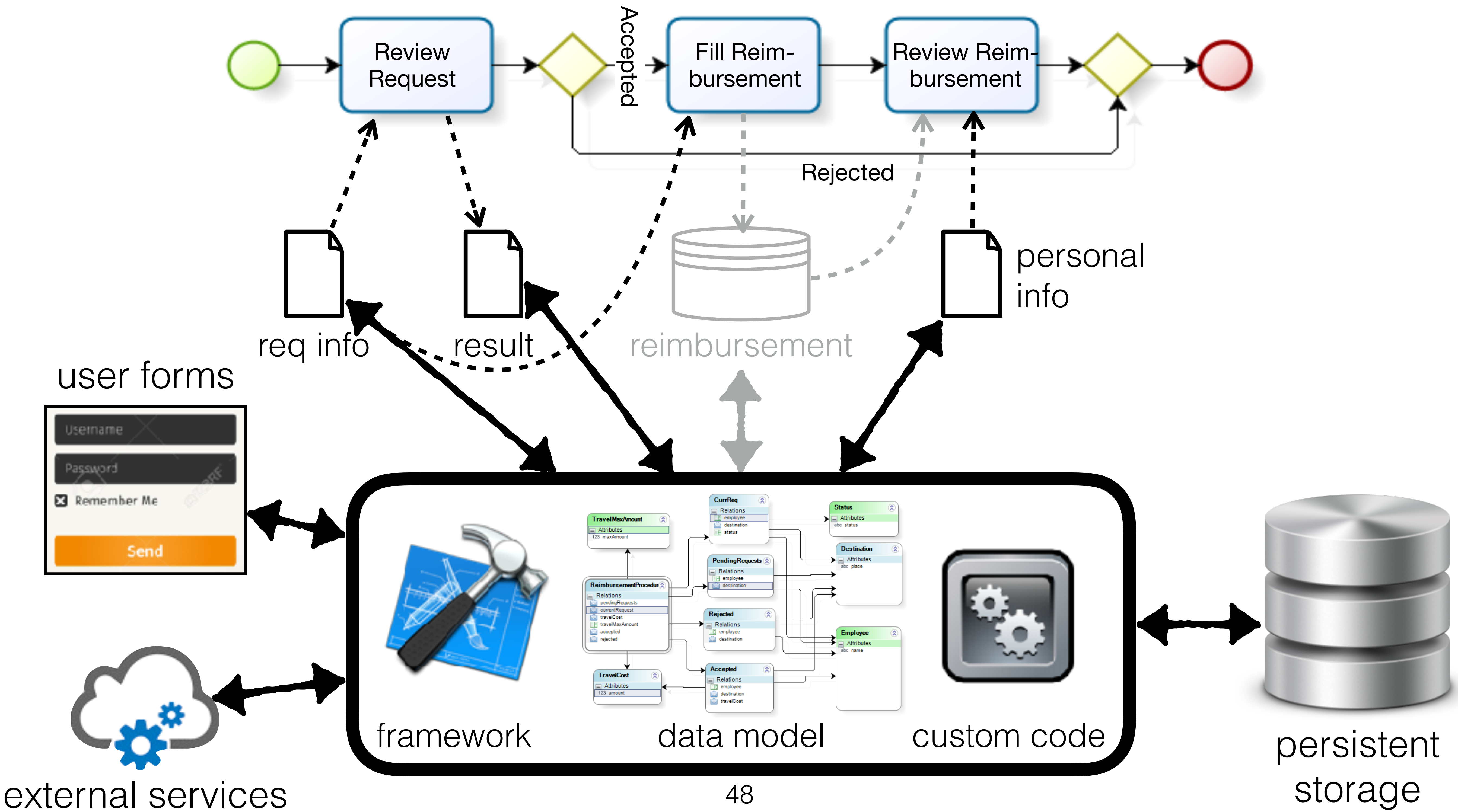
Case and Persistent Data



Persistent Data Engineering



Case Data Engineering



A General Recipe

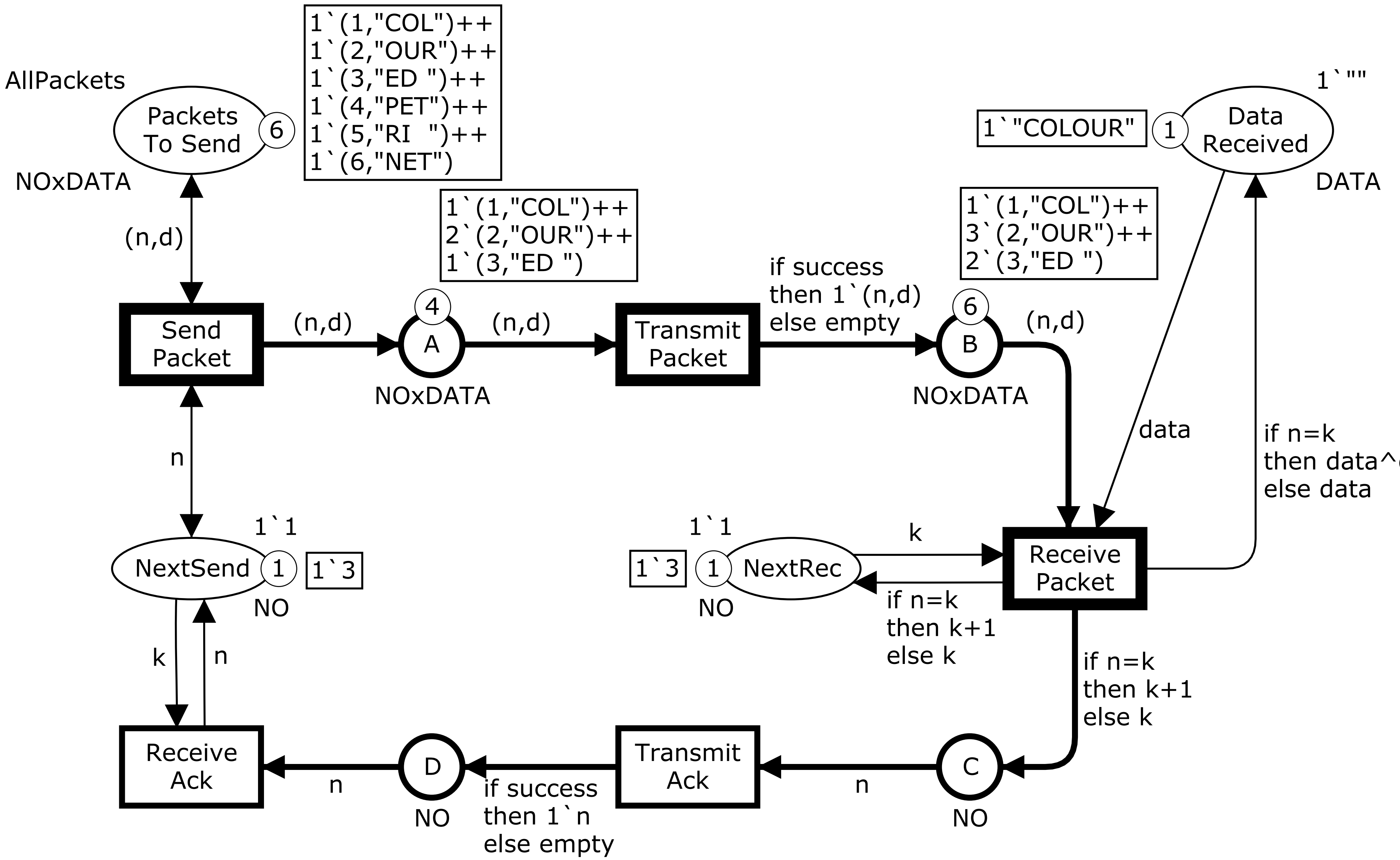
“REAL” PROCESS

- Explicit control-flow
- Local, case data
- Global, persistent data
- Queries/updates on the persistent data
- External inputs
- Internal generation of fresh IDs

Recipe?

BPMN
✓ Explicit control-flow
~ Local, case data
~ Global, persistent data
✗ Queries/updates on the persistent data
✗ External inputs
✗ Internal generation of fresh IDs

Colored Petri Nets



No conceptual representation of persistent storage

Recipe?

COLORED PETRI NETS

- ✔ Explicit control-flow
- ✔ Local, case data
- ✘ Global, persistent data
- ✘ Queries/updates on the persistent data
- ✔ External inputs
- ✔ Internal generation of fresh IDs

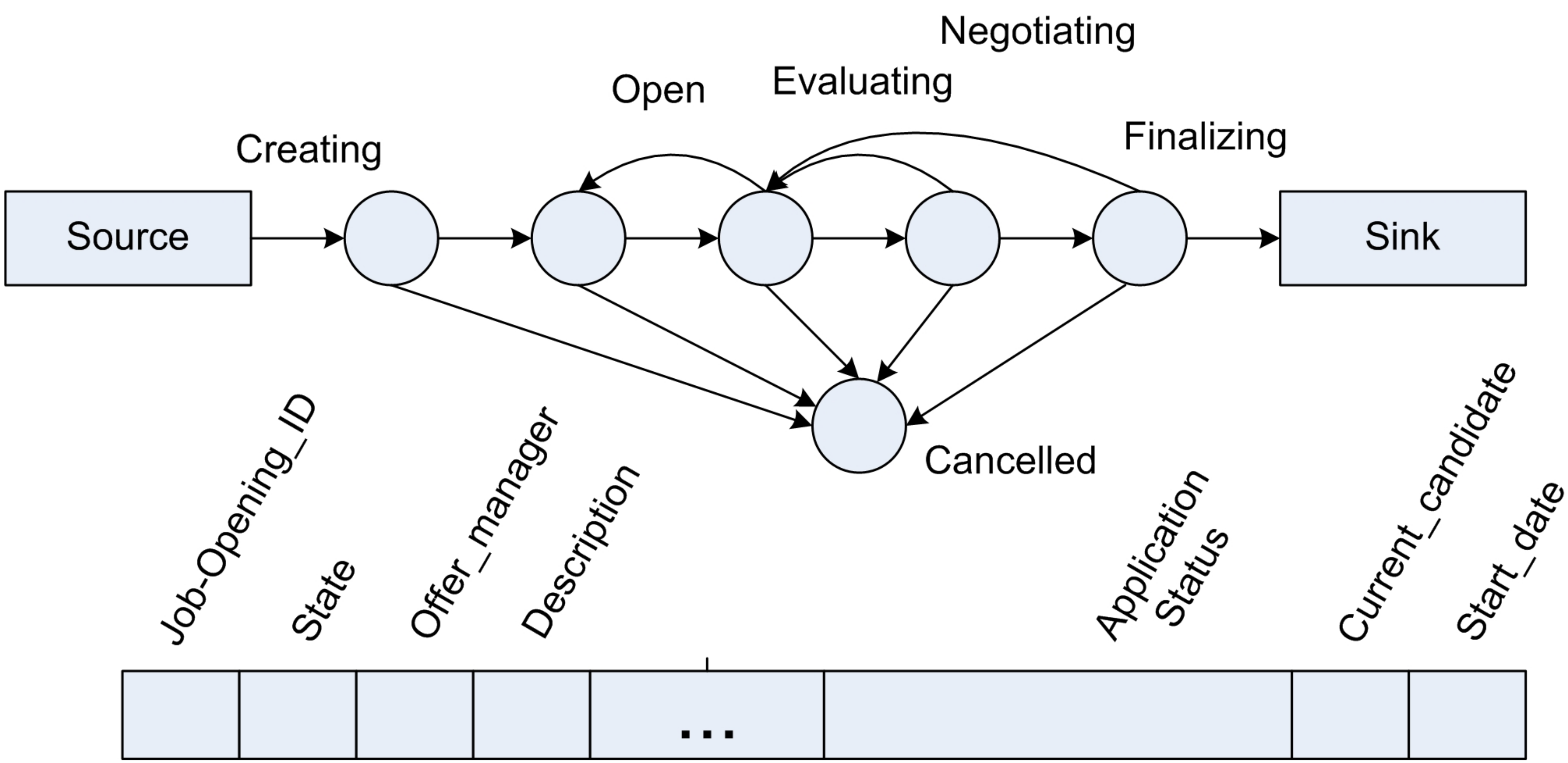
implicit, or using fresh variables

Business Entities/Artifacts

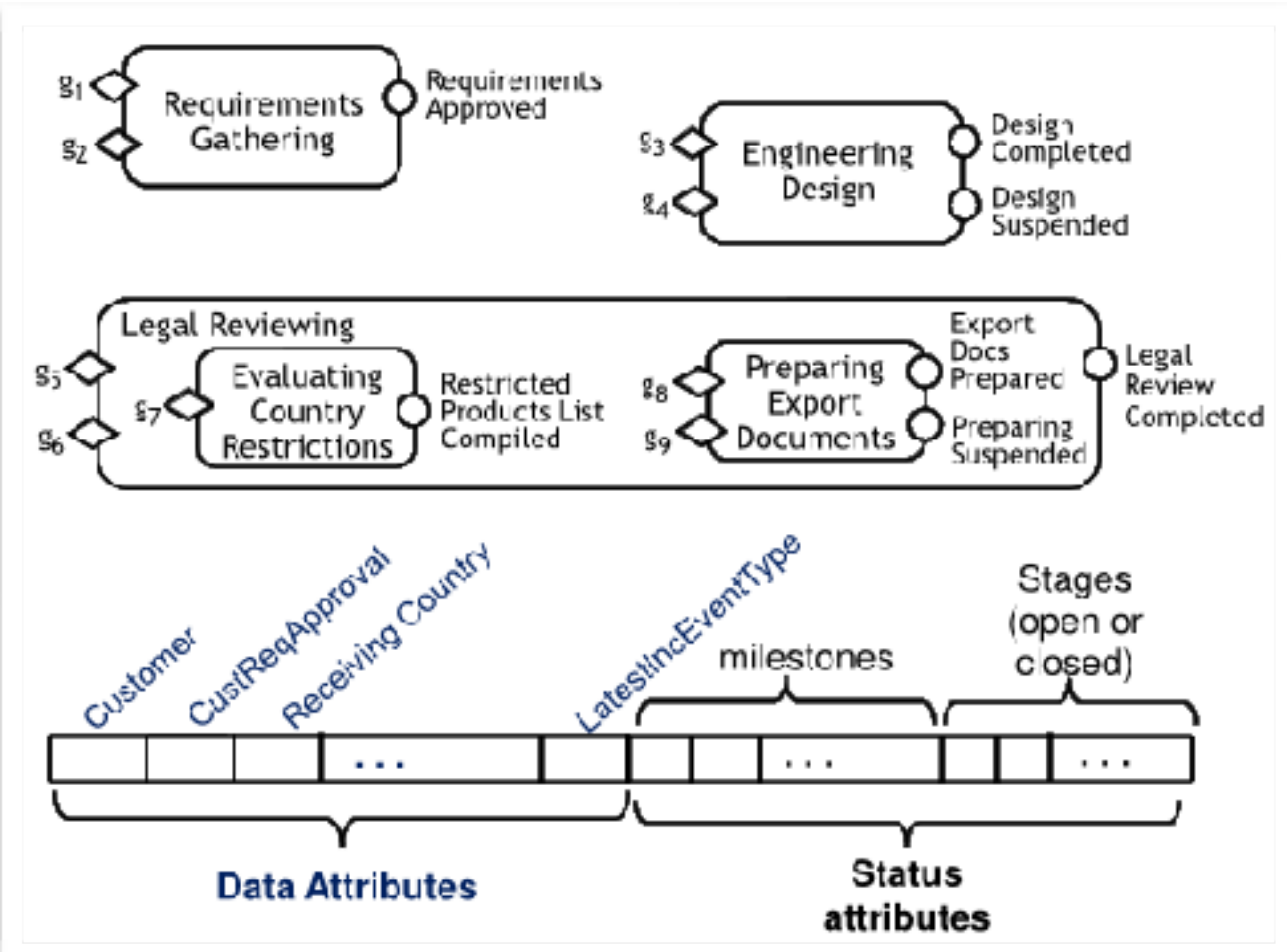
Data-centric paradigm for process modeling

- First: *elicitation of relevant business entities* that are evolved within given organizational boundaries
- Then: definition of the *lifecycle* of such entities, and how *tasks trigger the progression* within the lifecycle
- Active research area, with concrete languages (e.g., IBM GSM, OMG CMMN)
- Cf. **EU project ACSI** (completed)

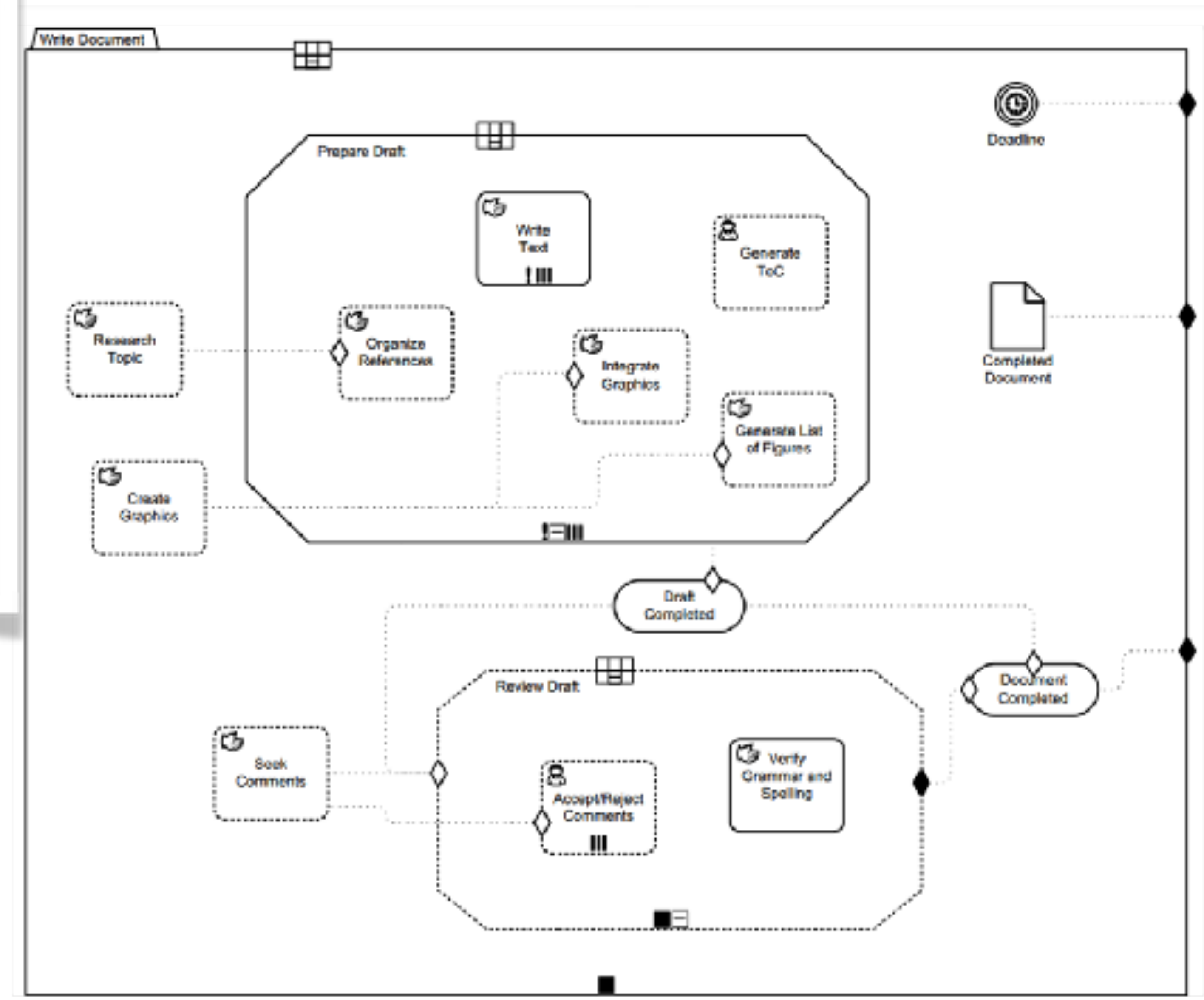
Finite-State Machines



GSM - CMMN

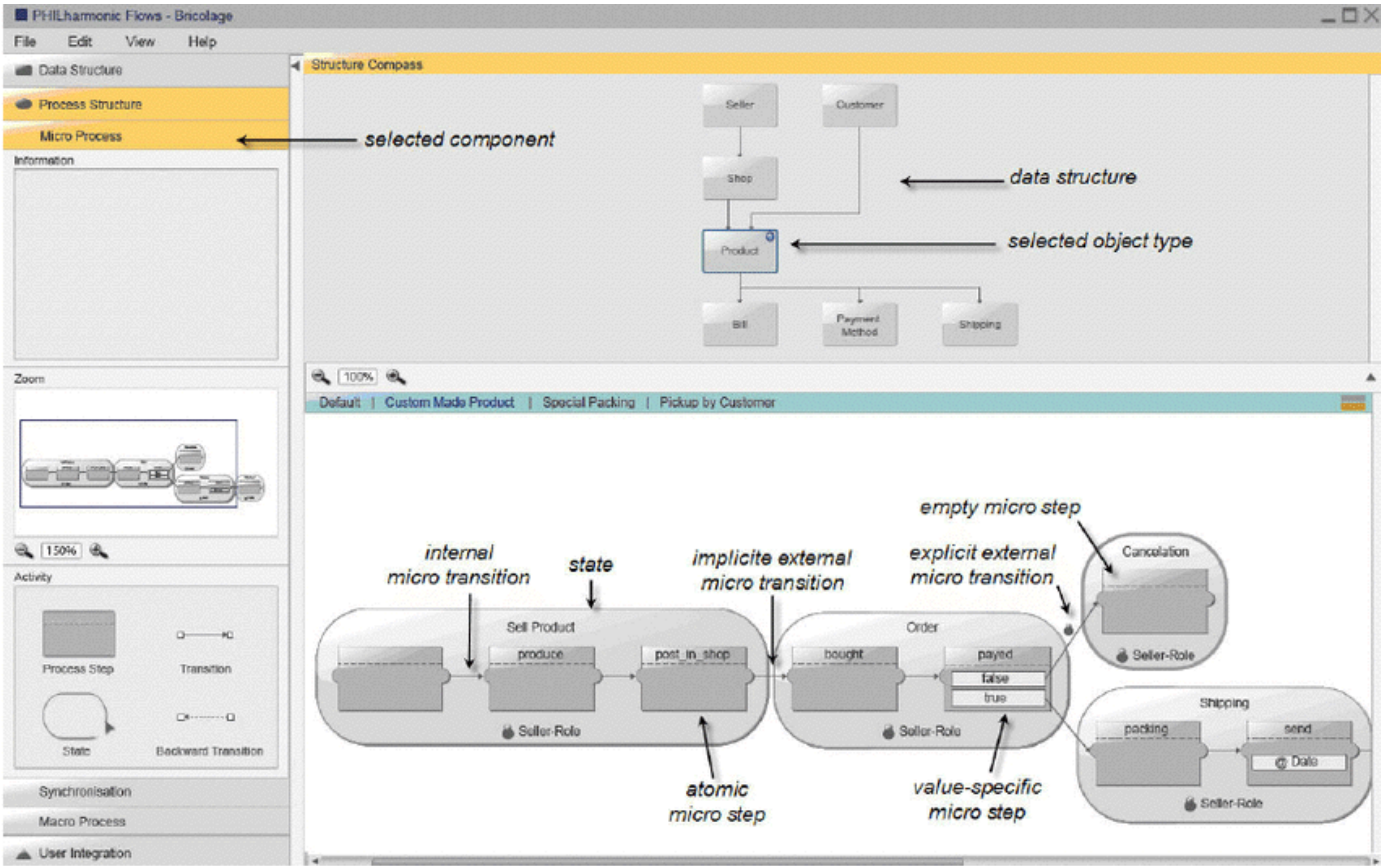


Requirements Gathering
Requirements Approved
Engineering Design
Design Completed
Design Suspended
Legal Reviewing
Evaluating Country Restrictions
Restricted Products List Compiled
Preparing Export Documents
Export Docs Prepared
Preparing Suspended
Legal Review Completed
Customer
CustReqApproval
Receiving Country
LatestIncEventType
milestones
Stages (open or closed)
Data Attributes
Status attributes



Write Document
Prepare Draft
Deadline
Completed Document
Draft Completed
Document Completed
Review Draft

Philharmonic Flows



Recipe?

ARTIFACT-/OBJECT-CENTRIC PROCESSES

- ~ Explicit control-flow
- ~ Local, case data
- ✓ Global, persistent data
- ✓ Queries/updates on the persistent data
- ~ External inputs
- ~ Internal generation of fresh IDs

Problem Dimensions

Dimension 1
Static Information Model

How are data structured?

- Propositional symbols —> Finite state system
- Fixed number of values from an unbounded domain
- Full-fledged database:
  - relational database
  - tree-structured data, XML
  - graph-structured data

Dimension 1
Static Information Model

Are constraints present? How are they interpreted?

- Complete data
- Data under incomplete information
  - ontology (with intensional part typically fixed)
  - full-fledged ontology-based data access system
- Hard vs soft-constraints (inconsistency-tolerance)

Dimension 2
Dynamic Component

- Implicit representation of time vs. implicit progression mechanism vs. explicit process
- When an explicit process is present:
  - how is the process dynamics represented?
  - procedural vs. declarative approaches (e.g., finite state machines vs. rule-based)
- Deterministic vs. non-deterministic behaviour
- Linear time vs. branching time model
- Finite vs. infinite traces

Dimension 3
Data-Process Interaction

How are data manipulated by the process?

- Data is only accessed, but not modified
- Data are updated, but no new values are inserted
- Full-fledged combination of the temporal and structural dimensions
- Hybrid approaches (e.g., read-only database + read-write registers)

Dimension 4
Interaction with the Environment

Is the system interacting with the external world?

- Closed systems vs. bounded input vs. unbounded input
- Synchronous vs. asynchronous communication
- Message passing, possibly with queues
- One-way or two-way service calls

Dimension 4
Interaction with the Environment

Which parts of the environment are fixed? Which change?

- Stateless vs stateful environment
- Fixed database vs. varying database vs. varying portion of data
- Multiple devices/agents interacting with each other
- Fixed vs changing topologies

Dimension 5
Formal Analysis

How are (un)desired properties formulated?

- Analysis of fundamental properties: reachability, absence of deadlock, boundedness, (weak) soundness

- Analysis of arbitrary formulae in some temporal logic

- Analysis of properties with queries across the temporal dimension (in the style of temporal DBs)

Dimension 5
Formal Analysis

Which forms of analysis?

- Verification
- Dominance, simulation, equivalence
- Synthesis from a given specification
- Composition of available components

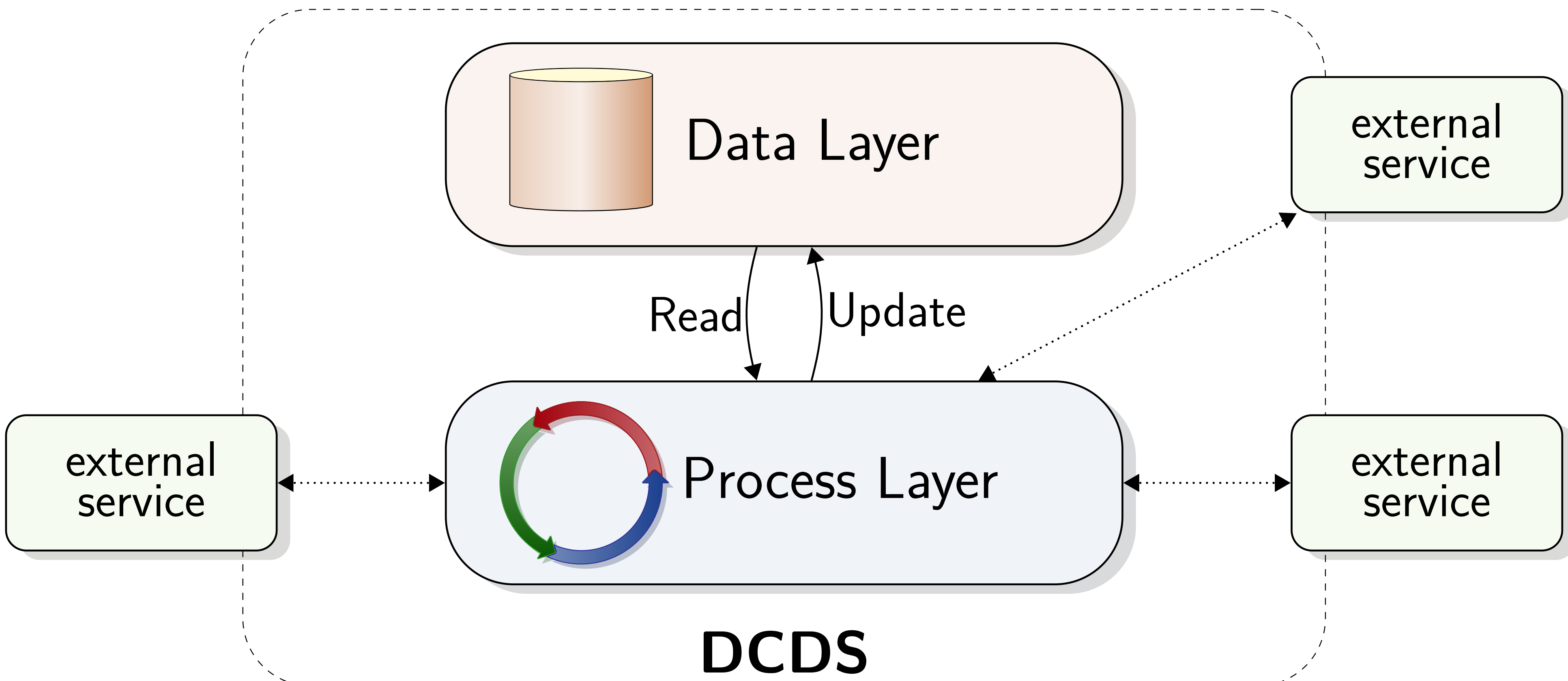
1) Go to the essential
2) Find boundaries of decidability in a general setting
3) Understand the connection with concrete languages
4) Implement

Fixing the main coordinates…

The Model

Data-Centric Dynamic Systems

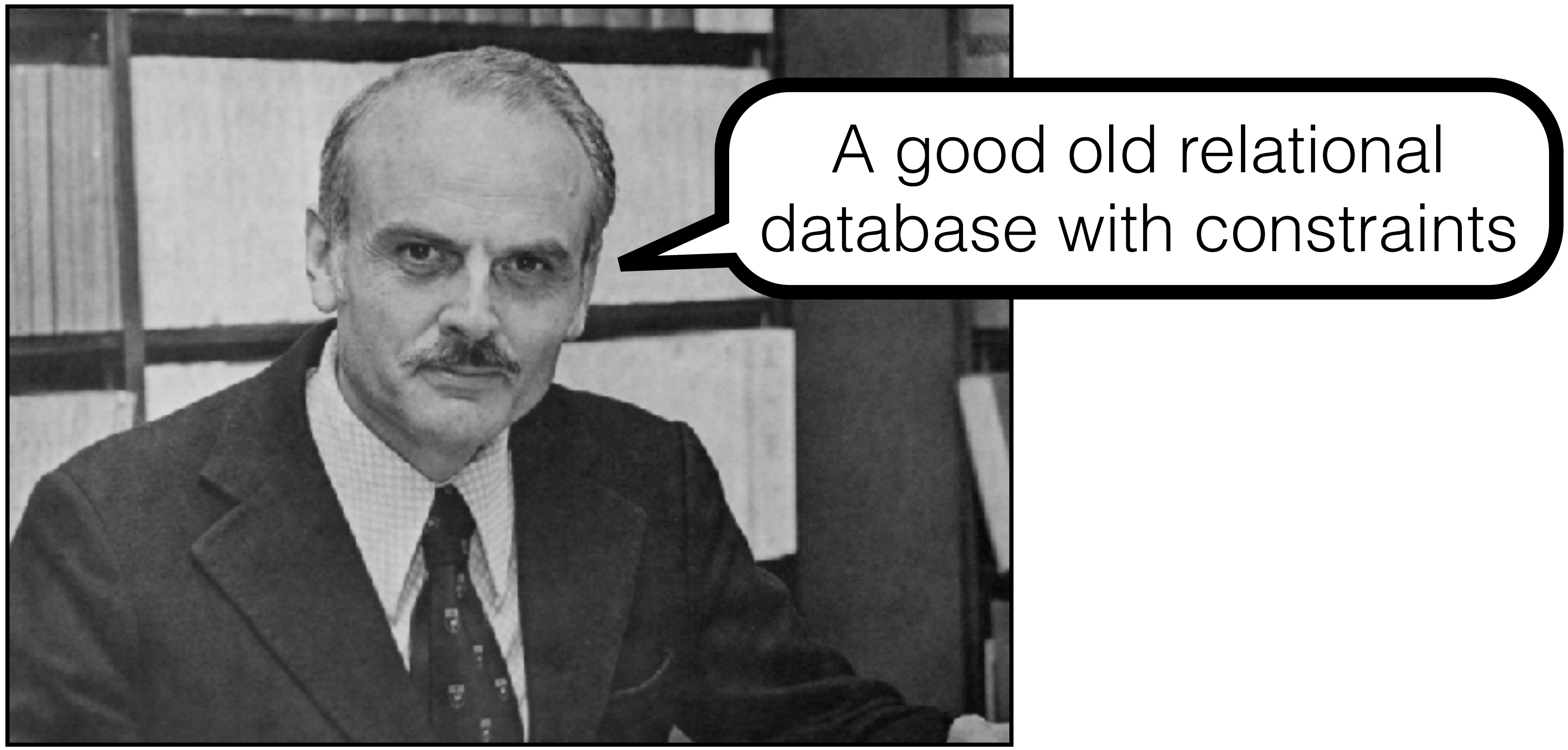
A pristine, yet very powerful framework for data-aware processes



Data layer: storage for persistent data

Process layer: declarative specification of system dynamics

Data Layer



Thus, a finite FO structure queried using domain-independent FO formulae

Data Layer

We fix an *infinite abstract data domain* Δ, and a finite subset Δ_0 of distinguished constants

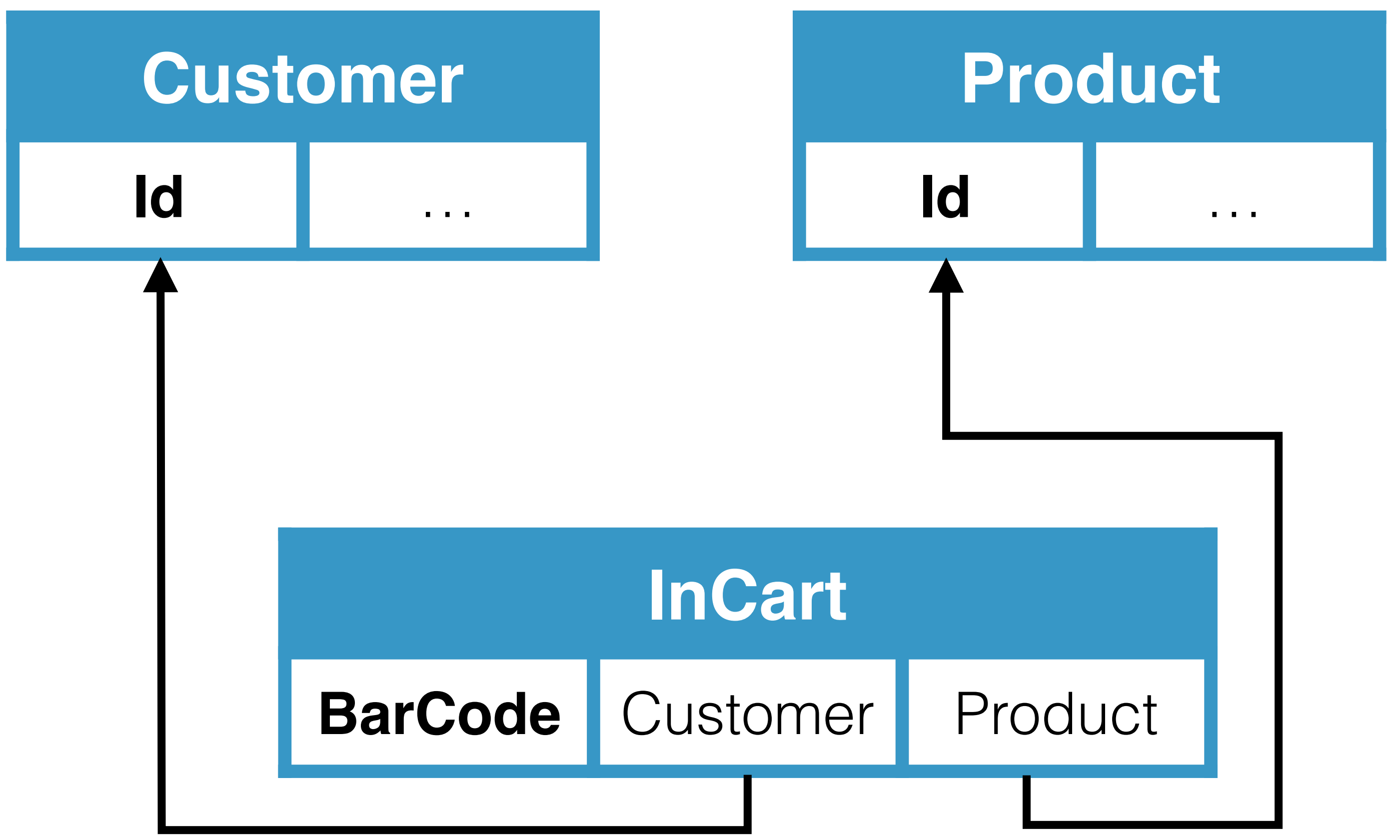
- **DB**: set of relation schemas
- **DB instance**: finite set of facts over DB using values from Δ
  - Active domain: (finite) set of values used in the instance

Data Layer

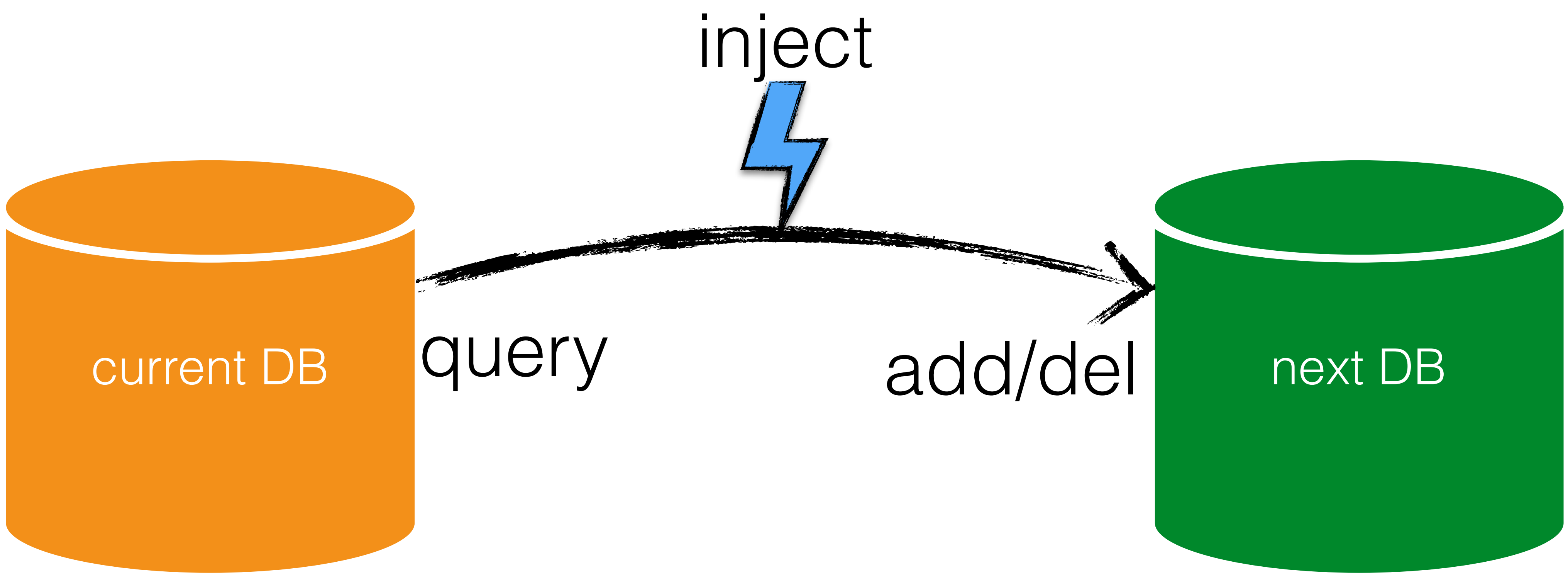
A DB instance is **queried** using possibly open first-order (FO) formulae with active domain semantics

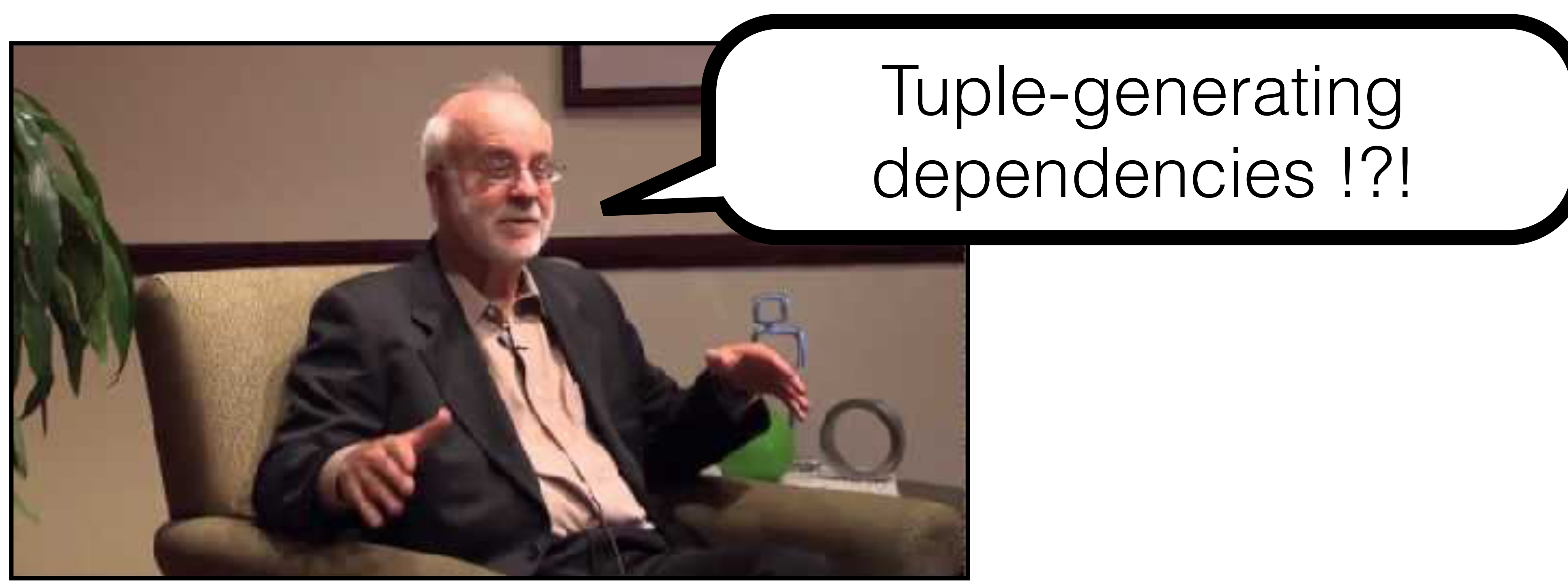
- **Constraints**: boolean queries, which *must be true in an instance*
  - *E.g.:* Keys, FKs, dependencies, multiplicities, …

Example: User Cart



Process Layer





Actions

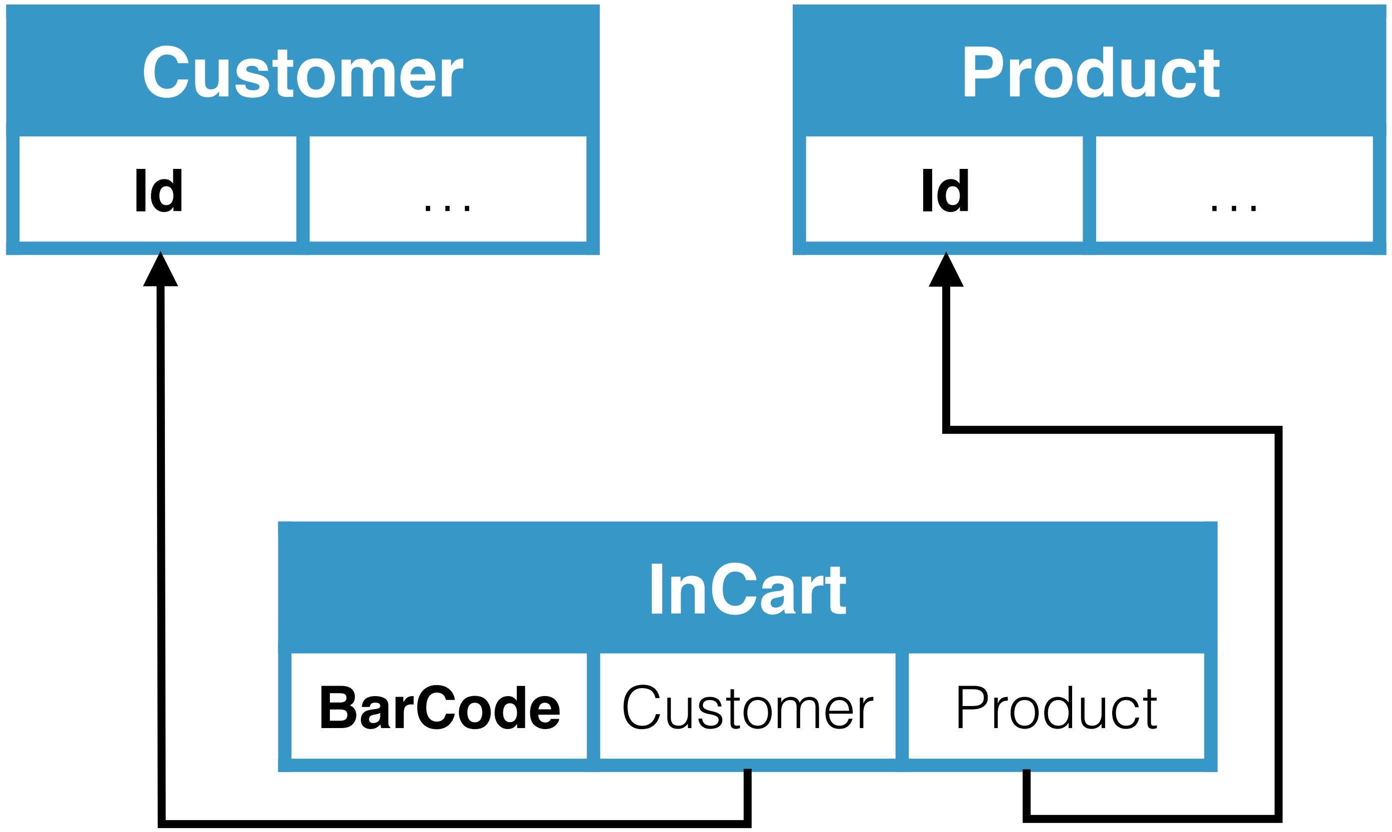
Each action encapsulates a **complex update** over the data layer

- **Action signature**: name + set of parameters
- **Action specification**: **conditional CRUD effects** (a là ADL in planning, or resembling SQL INSERT/UPDATE/DELETE prepared statements)

Action Effect

- Each effect is an IF-THEN rule
  - IF part: query over the current DB, possibly mentioning the action parameters
  - THEN part: ADD/DELETE facts, mentioning
    Action parameters
    Results to the IF query (bulk interpretation)
    Service calls to account for **new data**
- Cf.: ADL planning, tuple-generating dependencies, SQL insert/update/delete queries

Example: User Cart



User Cart Actions

- Any customer may decide to insert a new item of a given product into her cart

$$\exists \vec{y}, \vec{z}. Customer(c, \vec{y}) \land Product(p, \vec{z}) \mapsto \textsf{AddToCart}(c, p)$$

- Any customer may empty her own cart

$$\exists \vec{y}. Customer(c, \vec{y}) \mapsto \textsf{EmptyCart}(c)$$

User Cart Actions

- Adding to a cart…

$$\text{AddToCart}(c, p) :$$
$$\{\ true \rightsquigarrow \mathbf{add}\{InCart(\mathbf{getBarCode}(\mathbf{p}), c, p)\}\ \}$$

- Emptying a cart…

$$\text{EmptyCart}(c) :$$
$$\{\ InCart(b, c, p) \rightsquigarrow \mathbf{del}\{InCart(b, c, p)\}\ \}$$

Action Application

1. **Bind** the action parameters to actual values
(obtaining an instantiated action specification)

2. Issue the **condition queries**, retrieving ***all*** **answers**

3. **Instantiate** the add/delete facts using the parameters and all answers

4. **Evaluate** each **ground service call**, getting a corresponding value

5. Complete the **grounding** of add/delete facts

6. **Apply the update** on the current DB instance, first deleting, then adding

7. **If** the resulting DB instance **satisfies all constraints**: **commit**!
Otherwise: **roll-back**!

Sophisticated Inputs

Service calls are interpreted as being **purely nondeterministic** (e.g., user input).

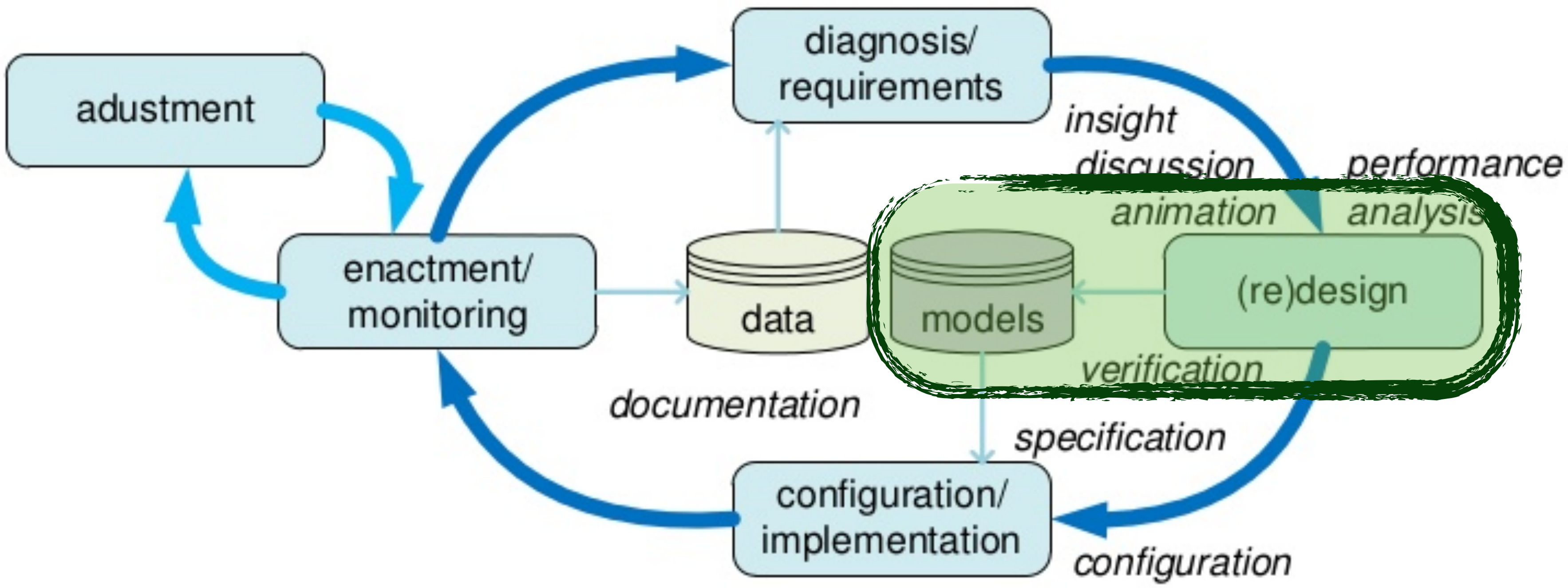
In many cases, it is useful to have:

- **constrained inputs** (e.g., comboboxes);
- **fresh value invention** (e.g., generation of a new primary key in a relation).

All this advanced features are syntactic sugar in DCDSs

Type of Analysis

Formal Verification



picture by Wil van der Aalst

Automated analysis
of a **formal model** of the system
against a property of interest,
considering **all** possible system behaviors

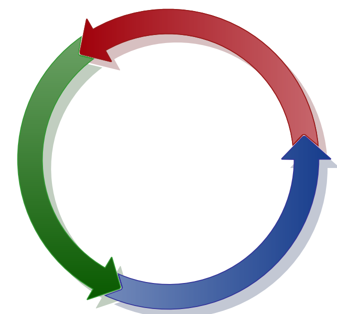
Guidelines

- System we verify = system we execute
- System compactly specified using a suitable modelling language: DCDS!
- A DCDS induces a transition system that provides the basis for verification
- Concurrency is interpreted as interleaving
- Various verification languages, with reachability as bottom line

Formal Verification

The Conventional, Propositional Case

Process control-flow

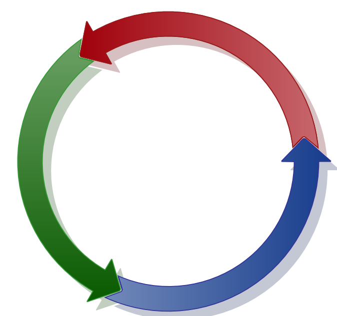


(Un)desired property

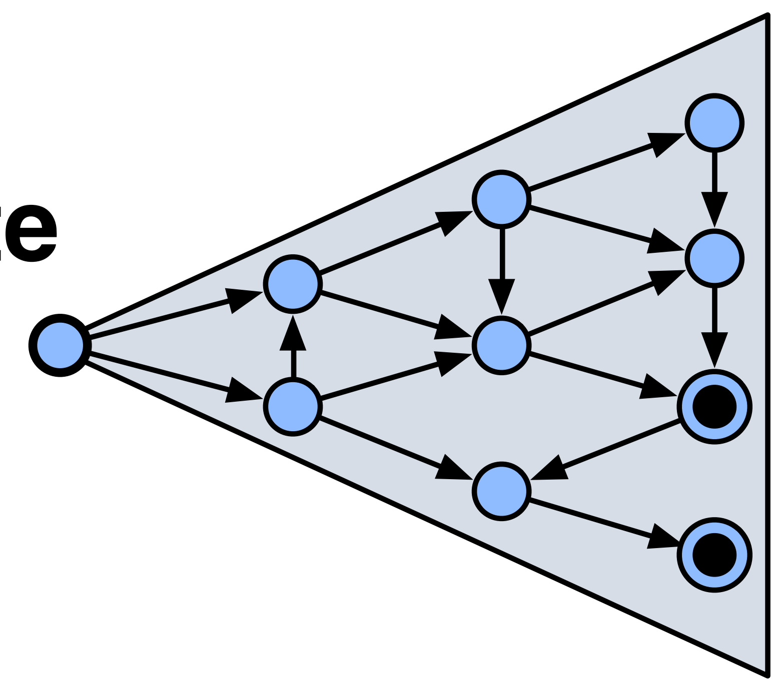
Formal Verification

The Conventional, Propositional Case

Process control-flow



Finite-state
transition
system

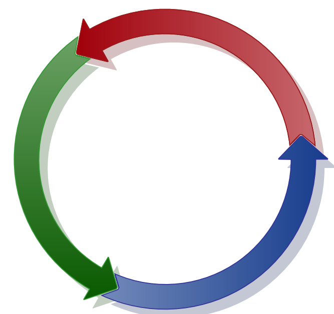


Φ **Propositional**
temporal formula

(Un)desired property

Formal Verification

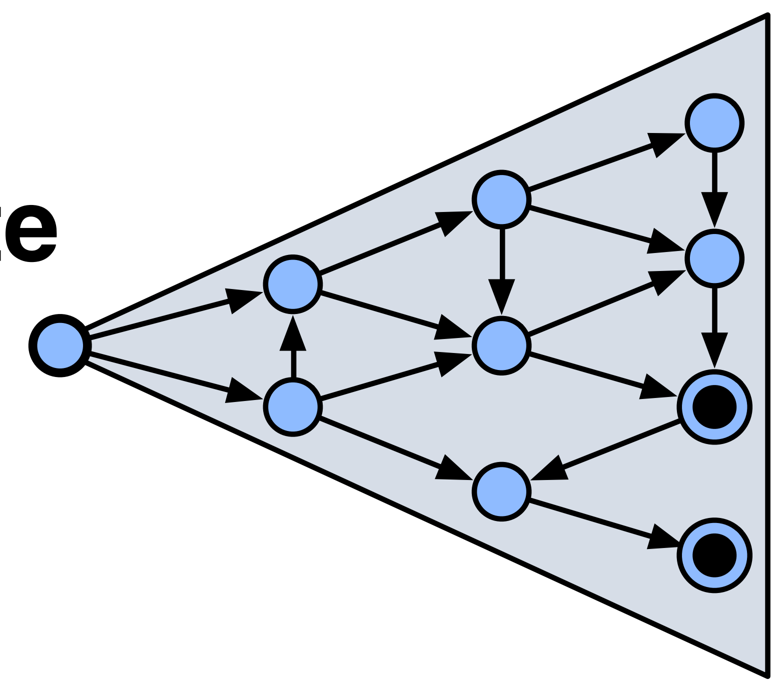
The Conventional, Propositional Case



Process control-flow

Verification via model checking
2007 Turing award:
Clarke, Emerson, Sifakis

Finite-state transition system



$\models \Phi$

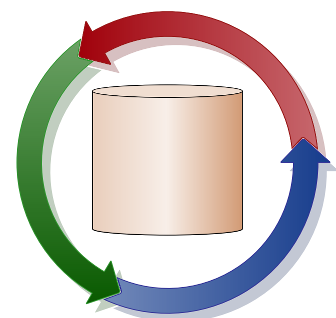
Propositional temporal formula

(Un)desired property

Formal Verification

The Data-Aware Case

DCDS (process+data)

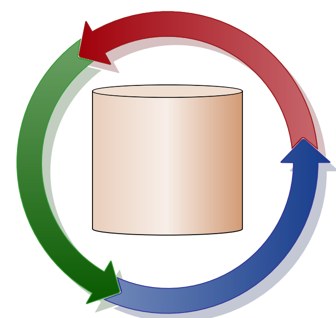


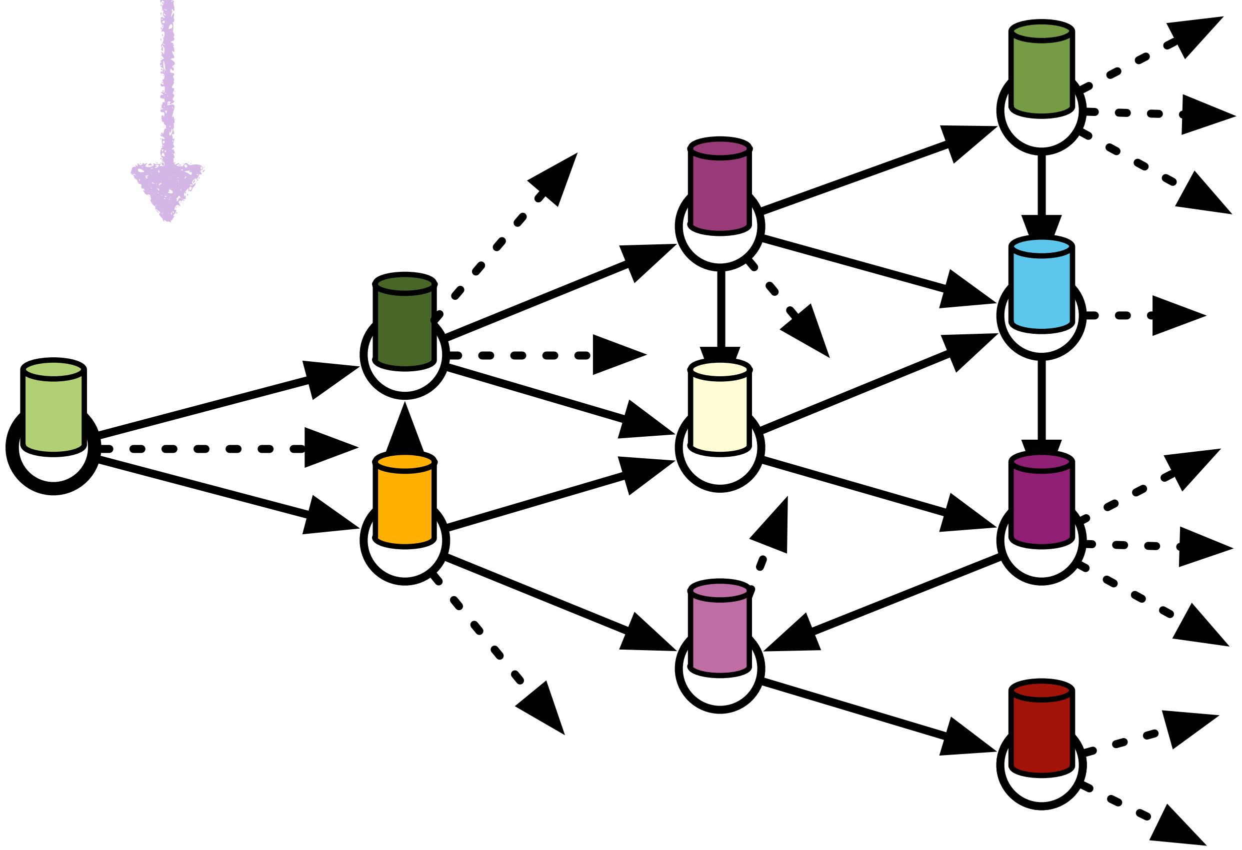
(Un)desired property

Formal Verification

The Data-Aware Case

DCDS (process+data)





Infinite-state, relational
transition system [Vardi 2005]

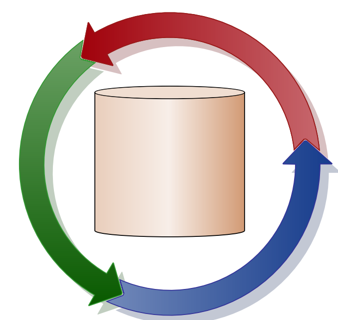
Φ **First-order**
temporal formula

(Un)desired property

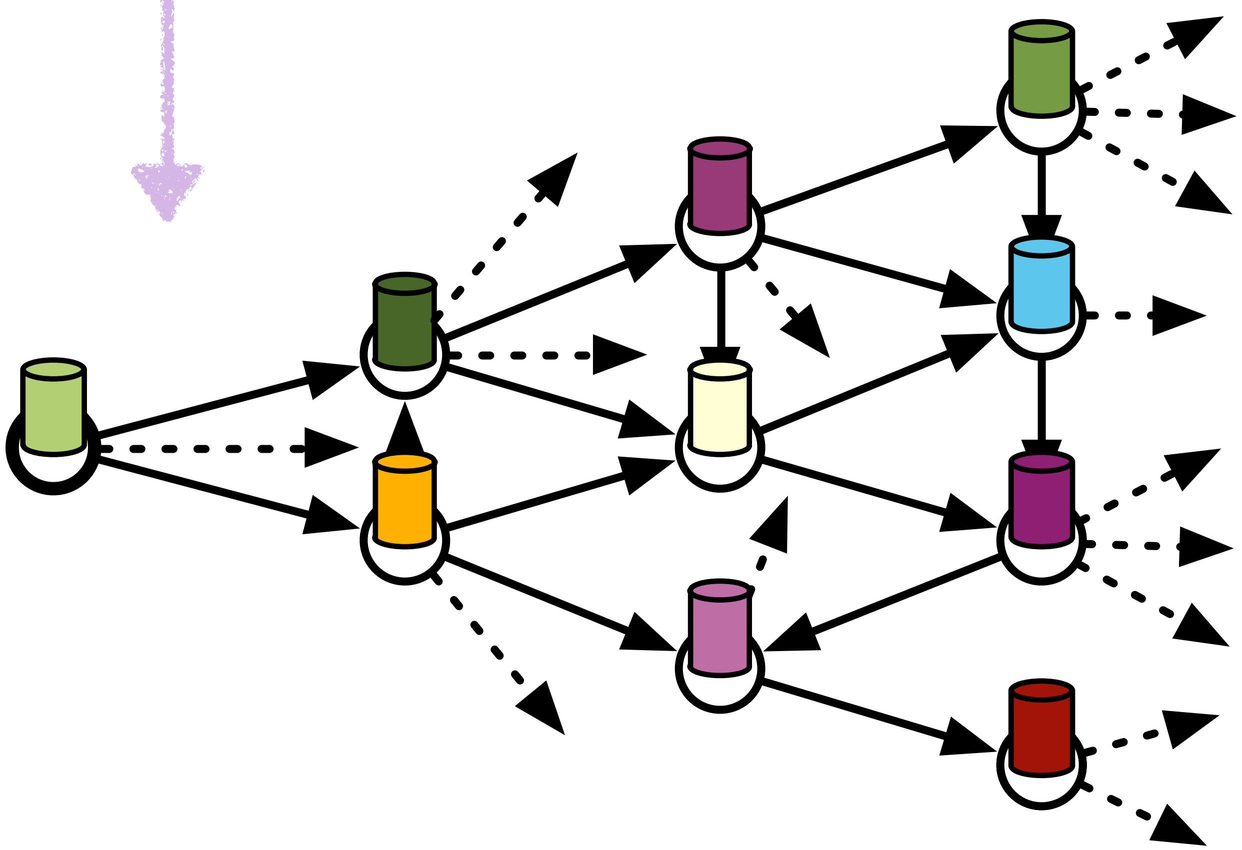
Formal Verification

The Data-Aware Case

DCDS (process+data)



?



$\models \Phi$

First-order
temporal formula

(Un)desired property

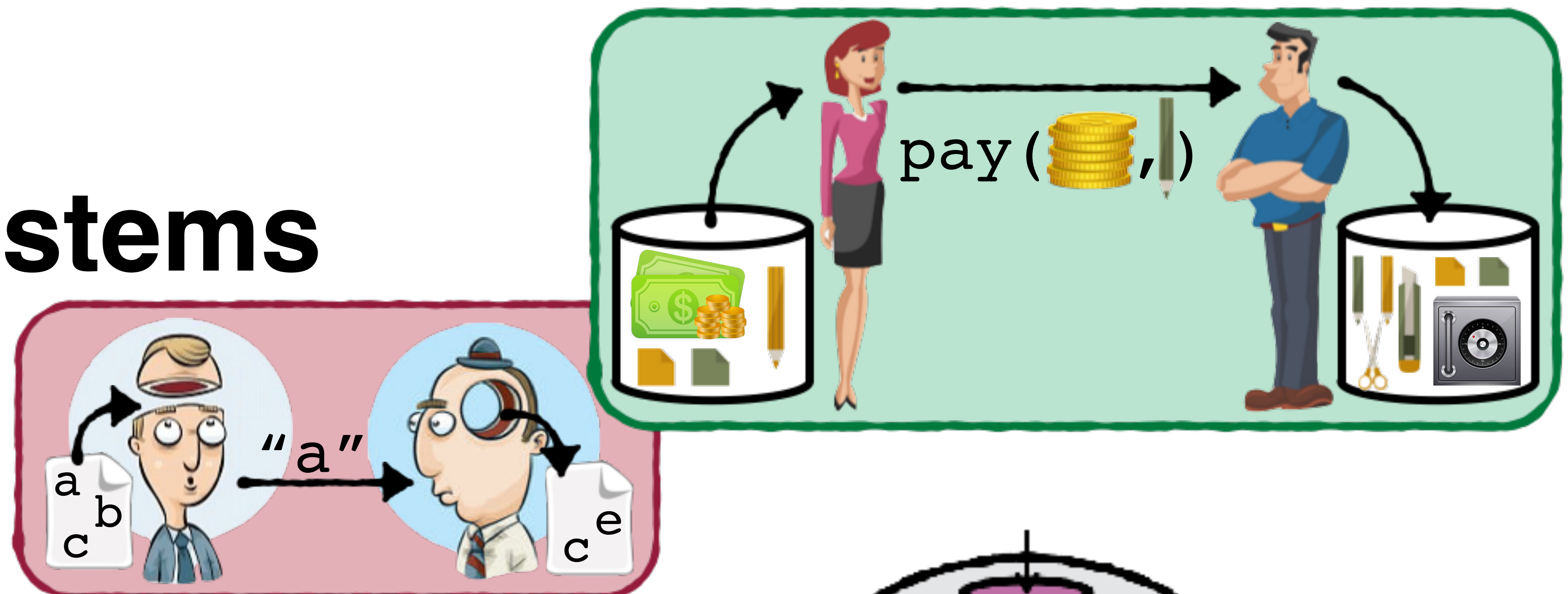
Infinite-state, relational
transition system [Vardi 2005]

Why FO Temporal Logics

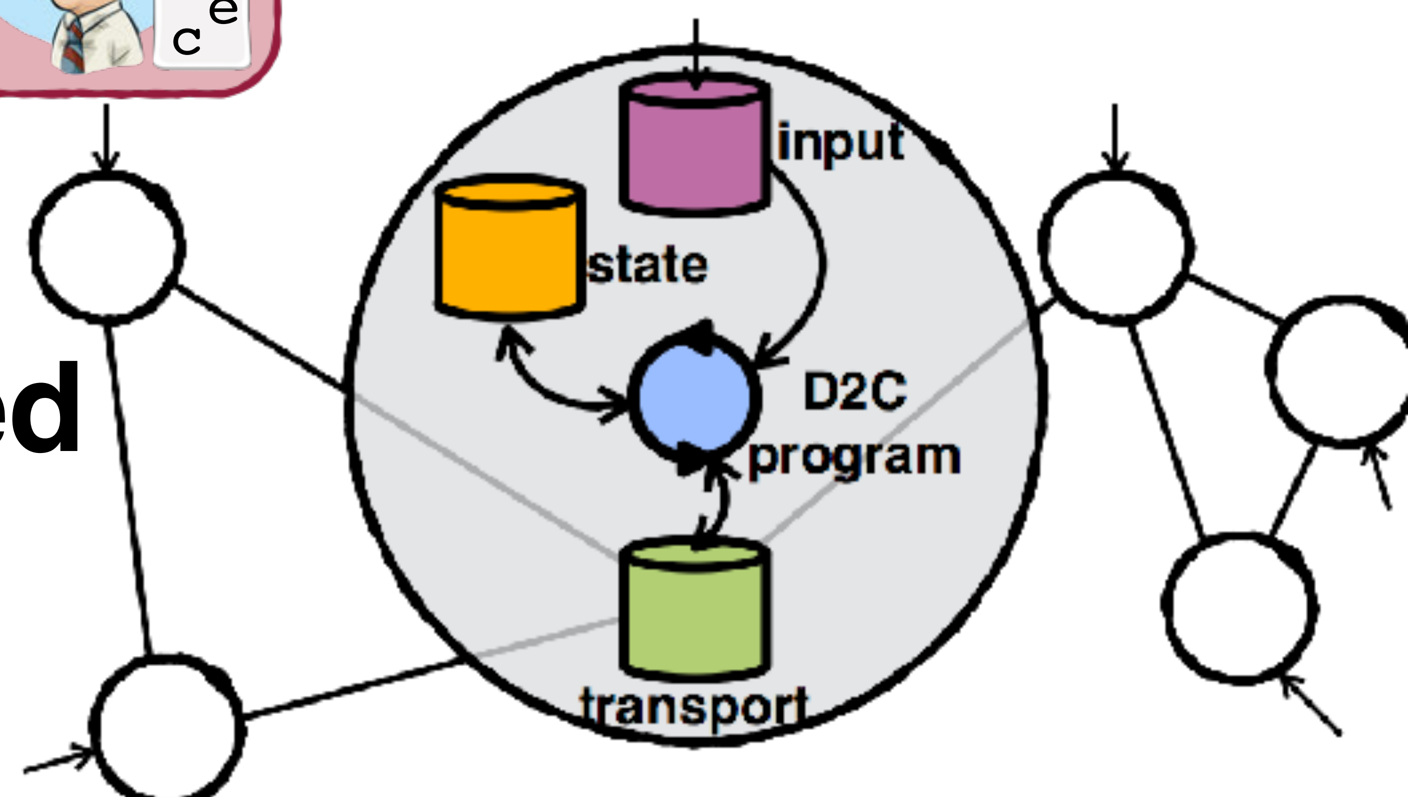
- To inspect **data**: **FO queries**
- To capture system **dynamics**: **temporal modalities**
- To track the **evolution of objects**: FO **quantification across** states
- Example: It is **always** the case that **every order** **is** **eventually** either **cancelled**, or **paid** and **then** **delivered**

Not Just Business Processes!

Relational Multiagent Systems



Declarative Distributed Computing



Software-Defined Networking

Let’s go!