A Conceptual Data Model is a Process Model too

Monique Snoeck
Separate worlds?

**Business Process Modelling**
- Petri Nets
- BPMN
- Activity Diagrams
  - Declare
  - CMMN
  - IDEF
  - ...

**Conceptual Data Modelling**
- (E)ER
- Niam
- Class diagrams
- Ontologies
- OntoUML
  - ...

Separate worlds?

Business Process Modelling
- Petri Nets
- BPMN
- Activity Diagrams
- Declare
- CMMN
- IDEF
- ...

Sebastian Steinau et al.
Software & Systems Modeling
https://doi.org/10.1007/s10270-018-0695-0
Separate worlds?

Conceptual Data Modelling

- (E)ER
- Niam
- Class diagrams
- Ontologies
- OntoUML
Separate worlds?

• Asymmetry:
  • Process Modelling seems NOT (totally) agnostic of data
  • Data Modelling seems agnostic of Process Modelling

• For good reason ...
  • In software architecture:
    • the process layer will use the data layer
    • the data layer will NOT use the process layer
Separate worlds?

• The data layer does not invoke the process layer, but ... Nevertheless: The data model does imply a number of sequence constraints on tasks.

• A process model defines a set of scenarios
  • e.g.

  ![Diagram](attachment:image.png)

  defines the scenarios \{X.Y, X.Z\}

• A data model defines a set of allowed scenarios too
Separate worlds

• A data model defines a set of scenarios.
• The Business Processes also define scenarios.
• The Business Processes are constrained by the Data Model.
Process model defined by the CDM

• CDM & BPs of A web shop ...
CDM and BPs of a Web Shop

Basket

Create

Update/Modify

Terminate/End
CDM and BPs of a Web Shop
CDM and BPs of a Web Shop

Customer \( \rightarrow \) Basket

\[(\text{frozen})\]

1..1 \( \rightarrow \) 0..*
"0..*" = unbounded interleaving of Baskets as a subprocess of Customer
CDM and BPs of a Web shop

(frozen)

1..1  0..*

Customer  Basket

Note: the semantics of the MI parallel marker is not a correct match with the semantics of "Parallel interleaving (0..*)". BPMN has no correct symbol for this. This can be expressed with PNs.
CDM and BPs of a Library

"0..1" = Iteration (with gaps) of Loans as a subprocess of Book
CDM and BPs of a Library

Diagram showing the relationship between Loan and Book with a 0..1 relationship from Loan to Book and a 1..1 relationship from Book to Loan. The diagram also includes other processes such as Book Registration, Book Management, and Book Disposal, with various transitions and states.
CDM and BPs of a Library

Loan Processes
CDM and BPs of a Web Shop

Basket

Product
CDM and BPs of a Web Shop

Basket

Product

0..*

0..*
CDM and BPs of a Web Shop

Basket

0..* 0..*

Product

remove from basket

add to basket

add to basket

add to basket
The reification move

- Our **domain of discourse** is typically much smaller than our **cognitive domain**
- When a property or a relation holds, it presupposes some **hidden entities** we can’t talk of, unless we put them in the domain of discourse.
- The act of putting in the domain of discourse an entity otherwise hidden, although presupposed by the language, is called **reification**
- Reification allows us to **talk** of these hidden entities, adding details about them

Reifying relationships

- **Reifying** relationships allows us to talk of:
  - their nature
  - the way they change in time
  - their interaction with the world
  - how they compare with similar relationships

---

Sometimes, we must talk of a relationship…

- Guarino & Guizzardi, “We need to discuss the relationship”: revisiting relationships as modeling constructs. CAiSE 2015
- Guarino, Sales, Guizzardi. Reification and truthmaking patterns. ER 2018. 21
Let's talk about the item in your basket? Was it delivered? Cancelled? Paid?
Basket

BasketItem

0..* 0..*

Basket

Product

add to basket

add to basket

add to basket

remove from basket
CDM and BPs of a Web Shop

Process Nesting (M. Jackson: "Marsupial Entity")
CDM and BPs of a Web Shop

• In MERODE, the reification is required, until an "Existence Dependency Graph" (EDG) is obtained

• EDG is a "DAG"
• "Top Down Level" of the class can be visualised.
• The TDL number coincides with the process nesting level.

• "Top-down" arrangement of the classes makes it easy to "see" the implied "business process" playground.
Example: Order Administration
Seeing the BP in the Class Diagram

• Event Management

CDM as submitted
Quite impossible to quickly evaluate ...

Are Ticket Types & their associated permissions defined before or after invitations are sent to the guests?
Seeing the BP in the Class Diagram

- Event Management
- Re-arranged CD
+ adding of TDL
0: People
1: Per Client: Event
2: Per event:
   - Website, mgr, Area's, groups of ??, invitations
3-4: In parallel
   - Invitees, then Guests
   - Permission Classes, then Assigned Areas
5-7: ticketing, then identifiers, then permissions & area presence
Different class diagrams support different business processes. When defining the class diagram, one should be aware of the implications a modelling choice has on the BP layer.

1. Define Event
2. Define the Ticket Categories per Event
3. Issue invitations per category
4. Issue tickets per invitation

1. Define the event
2. In parallel, per event:
   - Define Ticket Categories
   - Issue invitations
3. Issue Tickets per invitation in category of choice
BP Modelling versus Class Diagram

• A Class diagram is a "behavioural model"

• Class on mandatory side of an association exists first
  • ➔ Tasks that triggers creation event of mandatory objects come first
  • ➔ Tasks that triggers creation of dependent come next
  • ➔ “Top-down” organisation of class diagram gives sequences of tasks

• Mandatory Class is deleted last
  • ➔ "ending" happens in reverse order.

A domain model (UML Class diagram) is NOT "just structure": it also implies some behaviour.
BP Modelling versus Class Diagram

• Some business objects are a direct representation of a business process
Introducing Object Orientation: object lifecycles

Create → Update/Modify:
- Confirm
- Ship
- Pay

Terminate/End → Basket

Basket lifecycle:
- ME_cr_Basket
- ME_CancelOrder
- ME_end_Basket
- ME_Confirm
- ME_Pay
- ME_Ship
- ME_cr_BasketItem
- ME_end_BasketItem
Introducing object lifecycles

- Basket

Create → Update/Modify: Confirm, Ship, Pay → Terminate/End

These actions do not occur in a random order
Introducing object lifecycles

Create

Update/Modify:
- Confirm
- Ship
- Pay

Terminate/End
BP versus OLC: example
BP versus OLC: example

"Customer pays invoice"

➔ BP Perspective
   Customer performs payment
   ➔ TASK pay in BP of customer (subject of verb)

➔ OLC perspective
   invoice is paid
   ➔ event "Pay" triggers transition in OLC of invoice (object of verb) to state "Paid".
BP versus Object Life Cycles

- BP: group behaviour according to aspects of work organisation
- FSM: group behaviour per business object type
Positioning in the Zachman Framework

<table>
<thead>
<tr>
<th>Contextual/Scope</th>
<th>Conceptual/Enterprise</th>
<th>Logical/IS Functionality</th>
<th>What</th>
<th>How</th>
<th>Where</th>
<th>Who</th>
<th>When</th>
<th>Why</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain Model:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- information content of the business components</td>
<td>- rules managing the behavioural aspects of business components</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Enterprise Model

Business Process Model
BP versus OLC: example

• Basket Management
Compatibility checks required

• Global behaviour of Domain Layer
  • Behavioural aspects of existence dependency
  • Parallel composition of FSMS of single object types
  • Synchronisation by means of joint participation to events
  • Additional constraints such as EDG Cardinality, Referential Integrity, Preconditions

• Behaviour induced by Business Process Layer

Compatibility between Business Process Models and Object Life Cycles needs to be managed as well
CDM scenarios versus BPM scenarios

- Scenarios accepted by the domain model
- Scenarios not used by business processes
- Business process scenarios accepted by the domain model
- Business process scenarios rejected by the domain model
- Scenarios modelled by the business process models
The CDM sets the boundaries of what is possible
Discussion

• Good Conceptual Modelling requires
  • managing cross-model quality
  • managing the different perspective simultaneously

• M. Jackson "The World & The Machine"
  • Modelling is also about engineering OF the world

• Process/behavioural view of the CM cannot be underestimated
  • See again Jackson: it should be about the shared events
  • ==> that kind of behaviour is NOT about work organisation/Business processes
Implications for training good modelers
Good Conceptual Modelling requires

• managing cross-model quality
  • Formal Foundations

• managing the different perspective simultaneously
  • Tool Support!
  • Educational Support!
Managing cross-view Consistency

• "Existence Dependency" allows arranging Object Types in a DAG
  • Existence Dependency builds on the notions of
    • "Weak Entity" from ER modelling
    • "Marsupial" from Jackson Systems Development
    • Set inclusion "$\subseteq\" of Formal Languages (a JSD diagram is a regular expression)
    • Notion of "more deterministic than" ($\leq\) for processes

• This maps to a mathematical "Lattice" structure
  • Combine Relational Algebra
  • With Process Algebra
    • MERODE relies on "CSP" from C.A.R. HOARE to define the behavioural aspects of object types
Tool support

• Understanding & managing the different perspective simultaneously is really HARD!

• Using Existence Dependency simplifies the set of modelling concepts
  • The modelling language is smaller
  • Code generation becomes easier

• Educational support:
  • Simulation
  • Built-in Feedback: explaining application behaviour based on the model
  • Experiments demonstrate that fast prototyping and adding feedback to the prototype improves models understanding

• See http://merode.econ.kuleuven.be/ for resources
MERODE & MDE

• Create Model

• Generate Prototype to validate model

• E.g. Library Model
MERODE Prototyping
MERODE Prototyping
MERODE Prototyping
World & Machine

Four facets of the relationship

- Models
- Interface
- Engineering
- Problem
Relationship between the World & the Machine: Models
Four Facets of the relationship

- Models to simulate and have information about the World & the Machine
  - The machine embodies a *model* or a *simulation* of some part of the world.
  - There are *data models*, *object models*, *process models*.
  - Provides efficient and convenient access to information about the world.
What's in the World? What's in the Machine?

- Where's the copy of my old book?
  - KU Leuven's library system:
  - At Home?
Relationship between the World & the Machine: Interface

- What is mirrored?
- What is captured and what not?
  - Student registers for a course
  - Student attends a class?
  - Student coughing in class
Relationship between the World & the Machine: Engineering

- What to control?
  - Student registers for a course?
  - Student borrows a book?
  - Student steals a book?
Four Facets of the relationship

- Models to simulate / have info about the World

- Interfaces define what phenomena are shared between the world and the machine
  
  "He died last week". "He doesn't exist any more"

- The engineer must understand the properties of the world and manipulate and exploit those properties to achieve the purposes of the system

The Interface facet & the Engineering facet require us to deal with behavioural aspects as well
Wrap up

• A data model is a business process model too
  • Because objects have behaviour
    • Default
    • Defined via OLC
  • Because objects are related
    • Relationships put constraints on behaviour
  • Because objects represent (parts of) business processes

• Awareness of the CDM behavioural part is important because
  • It defines the "playground" for BPs
    • Could be too restrictive or not restrictive enough
  • Consistency with Business Processes needs to be managed
  • Reflecting about the "shared phenomena" is needed to establish a proper relationship between the World and the Machine

• MERODE offers a concrete methodology & tool support for "behaviour-aware" conceptual data modelling
Useful references

• The World & the Machine:

• MERODE - BPModelling
  • M. Snoeck, Enterprise Information Systems Engineering: The MERODE Approach, Springer
  • M. Snoeck, G. Dedene, Existence dependency: The key to semantic integrity between structural and behavioral aspects of object types, IEEE TSE, 24, 233-251, DOI: 10.1109/32.677182
Useful references

• MERODE:
  • Teaching
  • Tooling - Impact of Simulation on Learning
  • Consistency checking

• see also full publication list on KULeuven's who-is-who
Questions?