A comparison of different types of declarative modelling

Maria Keet

Department of Computer Science, University of Cape Town, South Africa
mkeet@cs.uct.ac.za

HydroSciences Montpellier
Univsersité de Montpellier & Institut de Recherche pour le Développement
20 June 2024
Models and terminology

Conceptual data models

Ontologies

Conclusions
Outline

1. Models and terminology
2. Conceptual data models
3. Ontologies
4. Conclusions
Models galore

- Physical model; e.g., Lego brick house
- Mathematical model; e.g., climate change model, bacterial growth in cheese-making
- Machine learning & cs.; e.g., data-driven spellchecker, LLM
- Conceptual models; e.g., concept maps, UML diagrams, ontologies
Models galore

- Physical model; e.g., Lego brick house
- Mathematical model; e.g., climate change model, bacterial growth in cheese-making
- Machine learning & cs.; e.g., data-driven spellchecker, LLM
- Conceptual models; e.g., concept maps, UML diagrams, ontologies

⇒ Gentle, mostly non-technical introduction to conceptual models in computing, with a few half-baked props on ‘things with water’
An old TOY example – linking different perspectives
An old TOY example – linking different perspectives
An old TOY example – linking different perspectives
Matching things and terminology (1/2)

- **Stock** typically named with a noun (particular or universal)
- **Flow** named with a verb
- **Converter** as an attribute related to Flow or Stock
- **Action Connector** relates the former
- **Object** is candidate for an endurant
- **Event_or_activity** for a method or perdurant
- **Converter** maps to attribute_or_property
- **Action Connector** candidate for relationship between any two of Flow, Stock and Converter

More information and evaluation (with the microbial loop in ocean): Keet CM. Factors affecting ontology development in ecology. DILS’05.
Matching things and terminology (1/2)

- $\forall x (\text{Stock}(x) \rightarrow \text{EnDurant}(x))$
- $\forall x (\text{Flow}(x) \rightarrow \text{PerDurant}(x))$
- $\forall x (\text{Converter}(x) \rightarrow (\text{Quality}(x) \lor \text{State}(x)))$
- $\forall x (\text{ActionConnector}(x) \rightarrow \text{Relationship}(x))$

DOLCE: [Masolo et al.(2003)]
Some questions (part 1)

- Stella is rather outdated; which other (declarative) modelling languages could this be done with?
- What to convert it into?
  - Domain specific language
  - Conceptual data model, process model, ...
  - More generic terminology, classes, relations, and constraints (an ontology?)
Some questions (part 1)

- Stella is rather outdated; which other (declarative) modelling languages could this be done with?
- What to convert it into?
  - Domain specific language
  - Conceptual data model, process model, ...
  - More generic terminology, classes, relations, and constraints (an ontology?)
- Why even do this?
Some questions (part 1)

- Stella is rather outdated; which other (declarative) modelling languages could this be done with?
- What to convert it into?
  - Domain specific language
  - Conceptual data model, process model, ...
  - More generic terminology, classes, relations, and constraints (an ontology?)
- Why even do this?
  - integrating models, comparing models, checking for possible inconsistencies automatically
  - Management of the mathematical models: separate declarative from the imperative
Terminology, simplified processes, and some links

**Hydrological model development**
(conceptual, physical, and data-driven)

- Perceptual model
- Conceptual model (formalised perceptual model)
- Mathematical model
- Computational model (implemented version)

**Conceptual data model development**
(mainly conceptual, sometimes data-driven)

- Input (business rules, data, legacy software, ...)
- Conceptual data model (diagram, serialised, possibly formalised [in a logic])
- Object-oriented program code, SQL statements, etc.

**Ontology development**
(conceptual vs reality-based, may involve text/data-based approaches)

- Input (CDM, database schemas, taxonomies, vocabularies, text, design patterns, ontologies, ...)
- Sketch outline of ontology ('conceptualization stage')
- Ontology (formalised in a logic, typically serialised)
- Use in ontology-driven information systems

---

hydrological modelling information from [Solomatine and Wagener(2011)]
Terminology, simplified processes, and some links

**Hydrological model development** (conceptual, physical, and data-driven)

- Perceptual model
- Conceptual model (formalised perceptual model)
- Mathematical model
- Computational model (implemented version)

**Conceptual data model development** (mainly conceptual, sometimes data-driven)

- Input (business rules, data, legacy software, ...)
- Conceptual data model (diagram, serialised, possibly formalised [in a logic])
- Object-oriented program code, SQL statements, etc.

**Ontology development** (conceptual vs reality-based, may involve text/data-based approaches)

- Input (CDM, database schemas, taxonomies, vocabularies, text, design patterns, ontologies, ...)
- Sketch outline of ontology ('conceptualization stage')
- Ontology (formalised in a logic, typically serialised)
- Use in ontology-driven information systems

---

hydrological modelling information from [Solomatine and Wagener(2011)]
On the maths – scaffold generation
On the maths – scaffold generation

- **Object**
  \[
  \text{Object}(t) = \text{Object}(t - dt) + (\text{event}_{1} - \text{event}_{2}) \cdot dt
  \]
  
  INIT \( \text{Object} = \{ \text{Place initial value here... } \} \)

- **Infloows**
  \[
  \text{event}_{1} = \{ \text{Place right hand side of equation here... } \}
  \]

- **Outflow**
  \[
  \text{event}_{2} = \{ \text{Place right hand side of equation here... } \}
  \]

- **Stock**
  \[
  \text{Stock}(t) = \text{Stock}(t - dt) + (\text{flow}_{1} - \text{flow}_{2}) \cdot dt
  \]
  
  INIT \( \text{Stock} = \{ \text{Place initial value here... } \} \)

- **Infloows**
  \[
  \text{flow}_{1} = \{ \text{Place right hand side of equation here... } \}
  \]

- **Outflow**
  \[
  \text{flow}_{2} = \{ \text{Place right hand side of equation here... } \}
  \]

- **Attribute or property**
  \[
  \text{attribute}_{1} = \{ \text{Place right hand side of equation here... } \}
  \]

- **Converter**
  \[
  \text{converter} = \{ \text{Place right hand side of equation here... } \}
On the maths – scaffold generation

```
<table>
<thead>
<tr>
<th>Concentration_Pollutant (g/m³)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Has</td>
</tr>
<tr>
<td>Water_in_Pond</td>
</tr>
<tr>
<td>Volume (m³)</td>
</tr>
<tr>
<td>Inflow (m³/s)</td>
</tr>
<tr>
<td>Outflow (m³/s)</td>
</tr>
<tr>
<td>Forms</td>
</tr>
<tr>
<td>Pollutant</td>
</tr>
<tr>
<td>Amount (g)</td>
</tr>
<tr>
<td>Inflow (g/s)</td>
</tr>
<tr>
<td>Outflow (g/s)</td>
</tr>
</tbody>
</table>
```

Is in
On the maths – scaffold generation

![Diagram of Concentration Pollutant](image)

- **Concentration Pollutant** $(g/m^3)$
- **Water in Pond**
  - Volume $(m^3)$
  - Inflow $(m^3/s)$
  - Outflow $(m^3/s)$
- **Pollutant**
  - Amount $(g)$
  - Inflow $(g/s)$
  - Outflow $(g/s)$

The equation $C_{pol} \cdot W_{out} = P_{out}$ represents the relationship between concentration, flow, and pollutant amount.
On the maths – scaffold generation

- Concentration Pollutant ($g/m^3$)
  - Forms
  - Pollutant
    - Amount (g)
    - Inflow (g/s)
    - Outflow (g/s)
  - Water in Pond
    - Has
    - Volume ($m^3$)
    - Inflow ($m^3/s$)
    - Outflow ($m^3/s$)
  - $P_{acc} = P_{amount} + P_{in} - P_{out}$
On the maths – scaffold generation

\[ P_{acc} = C_{pol} \cdot W_{vol} - C_{pol} \cdot W_{out} + P_{in} \]
On the maths – a recent paper

Some questions (part 2)

- How to do something like this systematically?
- Will any of the current applied maths markup languages (e.g., SBML) suffice?
Some questions (part 2)

- How to do something like this systematically?
- Will any of the current applied maths markup languages (e.g., SBML) suffice?
- Why look into this?
Some questions (part 2)

- How to do something like this systematically?
- Will any of the current applied maths markup languages (e.g., SBML) suffice?
- Why look into this?
  - Annotation (declarative representation) of the formula separates the what (the formulae) from the how (the code how to compute it)
  - One declarative formula/model can be loaded ‘trivially’ across programming languages and executed
  - (Parts of) The declarative formula/model can more easily be reused, saving design time, facilitating interoperability
SBML: random example of an SBML model from the BioModels database

```xml
<listOfUnitDefinitions>
  <unitDefinition id="volume" metaid="_224f94f3-7d8c-4058-899c-ef158671d01e" name="volume">
    <listOfUnits>
      <unit exponent="1" kind="litre" metaid="b0b05891-9c17-4104-836c-e37a86b35f95" multiplier="1" scale="-3"/>
    </listOfUnits>
  </unitDefinition>
  <unitDefinition id="time" metaid="_45409b69-6247-4659-b564-97ab08e0e5cd" name="time">
    <listOfUnits>
      <unit exponent="1" kind="second" metaid="e38bd50b-5dda-4daa-a633-c6f6c51d2ed8" multiplier="3600" scale="0"/>
    </listOfUnits>
  </unitDefinition>
  <unitDefinition id="substance" metaid="_7abe9aca-312c-4b5d-baa2-c64840864da1" name="substance">
    <listOfUnits>
      <unit exponent="1" kind="mole" metaid="_5a1de13-c181-4205-a36f-9fbe53894ab3" multiplier="1" scale="-3"/>
    </listOfUnits>
  </unitDefinition>
</listOfUnitDefinitions>
```
SBML: random example of an SBML model from the BioModels database

```xml
BIOMD0000000531_url.xml

<listOfParameters>
  <parameter constant="true" id="Ka" metaid="COPASI2" name="Ka" value="1.44">
    <annotation>
      <COPASI xmlns="http://www.copasi.org/static/sbml">
          <rdf:Description rdf:about="#COPASI2">
            <dcterms:created>
            </dcterms:created>
          </rdf:Description>
        </rdf:RDF>
      </COPASI>
    </annotation>
  </parameter>
  <parameter constant="true" id="kb" metaid="COPASI3" name="kb" value="1.6E-10">
    <annotation>
    </annotation>
  </parameter>
</listOfParameters>
```
SBML: random example of an SBML model from the BioModels database

```
258     <listOfRules>
259     <assignmentRule metaid="_15d0c074-0652-4ed7-b795-9a1fd1a61655" variable="kat50">
260     <math xmlns="http://www.w3.org/1998/Math/MathML">
261     <apply>
262     <ln/>
263     <apply>
264     <plus/>
265     <apply>
266     <divide/>
267     <cn> 1 </cn>
268     <ci> kb </ci>
269     </apply>
270     <cn> 1 </cn>
271     </apply>
272     </apply>
273     </math>
274     </assignmentRule>
275     <assignmentRule metaid="d9883e24-e7cc-4373-b7f7-b79949435f44" variable="v5955">
```
Limitations of the domain models

- There are very many notations to learn
- Objects and arrows, but no way to specify how many outgoing arcs there may be
- Limited computational use among models
- Proliferation of incompatible modelling tools that are cumbersome to maintain
Rainfall questions – loose ends in the model

What are the names of the relations? the constraints? The relevant attributes? And their data types?
Solutions to limitations of the domain models

- Devise one notation for all
- More expressive (more features) than only objects and arrows
- Computational support
- (Try to) Standardise to make tooling development ‘economically’ viable
Solutions to limitations of the domain models

- Devise one notation for all
- More expressive (more features) than only objects and arrows
- Computational support
- (Try to) Standardise to make tooling development ‘economically’ viable

⇒ Conceptual data models
A class of models that capture the information about the data that are to be stored in the prospective software system (and possibly manipulated)

There are several conceptual data modelling language families and notations
A. Crow’s feet notation

<table>
<thead>
<tr>
<th>Entity type</th>
<th>Relationship (unspecified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>*Identifier attribute</td>
<td>zero or one</td>
</tr>
<tr>
<td>Attribute2</td>
<td>exactly one</td>
</tr>
<tr>
<td>Attribute3</td>
<td>one or more</td>
</tr>
<tr>
<td></td>
<td>zero or more</td>
</tr>
</tbody>
</table>

B. Chen’s notation

- **Entity type**
- **Identifier attribute**
- **Relationship**
  - **1**: zero or one
  - **1**: exactly one
  - **N**: one or more
  - **N**: zero or more

Note: no ‘stock’, ‘flow’ etc, but type of element for any subject domain
A. ER diagram using crow’s feet notation

- **Book**
  - *ISBN
  - Title
  - Format

- **Author**
  - *Name
  - Email
  - PhoneNo

- **Publisher**
  - *Name
  - Country

- **Book editor**
  - *Name
  - Email

B. Population examples

**Books**
- It
- Carrie
- The talisman
- Abcd

**Authors**
- Stephen King
- Peter Straub
- Efg Hijklm

**Editors**
- Editor1
- Editor2
- Editor3

**Publishers**
- Signet
- Pocket books
- Random house
- Abcd
Same example in UML and ORM

Convert UML class diagram (semi-)automatically into program code
Easier to communicate with other domain experts and programmers what’s in the code
Easier to reuse with other math formula that use same entities
Limitations of conceptual data model models in theory or practice

- For one specific application only – need to re-do it for each application, integration issues
- Solutions to recurring modelling issues re-invented time and again (and same mistakes made)
- Mostly informal diagrams that suffer from ambiguity (intentionally or not)
- Limited authoring guidelines
- Some quality control mechanisms

---

1 Mainly the CSDP for ORM [Halpin(2001)]; an example for EER: http://www.meteck.org/modellingbook/DanceSchoolExample.html; TDD proposal for UML [Tort and Olivé(2010)]
Outline

1. Models and terminology
2. Conceptual data models
3. Ontologies
4. Conclusions
Solving limitations of conceptual data model models with ontologies

- Model for a subject domain, of use across multiple applications for use, reuse, integration
- Provides solutions to recurring modelling issues, saves re-inventing
- Logic-based, as precise as permitted within the language
- Multiple quality control mechanisms (theories, methods, techniques, tools)
Simplified graphical rendering of a fragment of most popular one:

http://geneontology.org/docs/ontology-documentation/

(informally) an ontology is an engineering artefact in machine-processable format, which contains the entity types, their relationships, and constraints that hold over them of a particular (subject) domain.
In an ontology development environment (ODE)...
Models and terminology

Conceptual data models

Ontologies

Conclusions

... happenings behind the GUI ...

SubClassOf(awo:lion awo:animal)
SubClassOf(awo:lion ObjectSomeValuesFrom(awo:eats awo:Impala))
SubClassOf(awo:lion ObjectAllValuesFrom(awo:eats awo:herbivore))
... and underlying that serialisation
Why ontologies?

- For their own sake, possible future use
- Representing a scientific theory precisely
- Facilitating communication among humans
- Enabling communication between software applications or modules in a complex system
- Used for and in many different ontology-driven information systems: a.o., data integration, recommender systems, NLP, textbook enhancements, Q&A systems)
Why ontologies?

- For their own sake, possible future use
- Representing a scientific theory precisely
- Facilitating communication among humans
- Enabling communication between software applications or modules in a complex system
- Used for and in many different ontology-driven information systems: a.o., data integration, recommender systems, NLP, textbook enhancements, Q&A systems)
Scenarios for building Ontology Networks (NEON methodology)
Scenarios for building Ontology Networks (NEON methodology)

Knowledge Resources

Non Ontological Resources
- Glossaries
- Dictionaries
- Lexicons
- Conceptual Schemas
- Taxonomies
- Thesauri

Ontological Resources
- O. Design Patterns
- O. Repositories and Registries
- Flogic RDF(S)
- OWL

Ontological Resource Reuse
- O. Aligning
- O. Merging

Alignments
- RDF(S)
- Flogic
- OWL

Ontology Design Pattern Reuse

Ontological Resource Reengineering

Ontology Specification
- O. Conceptualization
- O. Formalization
- O. Implementation

Ontology Localization

Ontology Support Activities: Knowledge Acquisition (Elicitation); Documentation; Configuration Management; Evaluation (V&V); Assessment

1. Hydrological models
2. Non Ontological Resource Reengineering
3. Ontology Design Pattern Reuse
4. Ontological Resource Reengineering
5. O. Aligning
6. O. Merging
7. Ontological Resource Reuse
8. Ontology Restructuring (Pruning, Extension, Specialization, Modularization)
9. O. Localization

[Suarez-Figueroa et al. (2008)]
NeOn – some extensions
NeOn – some extensions
Ontology development at the ‘micro-level’ level (cf. macro)

- Figure out what to add: which vocabulary and axioms
- Need to get those axioms into the ontology somehow
Ontology development at the ‘micro-level’ level (cf. macro)

- Figure out what to add: which vocabulary and axioms
- Need to get those axioms into the ontology somehow
- The actual modelling, or *ontology authoring*, using *micro-level* guidelines, methods, and tools
  - Methods, such as reverse engineering and text mining to start, OntoClean and OntoPartS to improve an ontology’s quality
  - Tools to model, to reason, to debug, to integrate, to link to data
Align to it – manually or assisted with D3 or BFOClassifier
Test-driven development

A note on that automated reasoning – Illustration

\[
\begin{align*}
\text{B} & \subseteq \text{A} \\
\text{C} & \subseteq \text{A} \\
\text{D} & \subseteq \text{A} \\
\text{B} & \subseteq \exists R.\text{E} \\
\text{D} & \subseteq \exists R.\text{E} \cap \exists S.\text{F}
\end{align*}
\]
Rainfall \sqsubseteq \textit{Eventive} (‘rainfall is an event’),
\textit{Rainfall} \sqsubseteq \exists \textit{causes}.Runoff (‘each rainfall event causes some amount of runoff’),
\textit{PondingLoss} \sqsubseteq = 1 \textit{measure}.\textit{Volume} (‘each ponding loss has exactly one measure of volume’ (of the loss)), ...
Limitations of ontologies?
Limitations of ontologies?

- Yes, there are...
- ... but not part of this talk
# Feature-based comparison

<table>
<thead>
<tr>
<th>Table 7.1</th>
<th>Comparison of types of models along a set of properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model type</strong></td>
<td><strong>Feature</strong></td>
</tr>
<tr>
<td>Mind Maps</td>
<td>Basic structuring of a topic</td>
</tr>
<tr>
<td>Biology models</td>
<td>Visualise biology knowledge (structures and processes)</td>
</tr>
<tr>
<td>Conceptual data models</td>
<td>Capture characteristics of data to be stored and processed in an program</td>
</tr>
<tr>
<td>Ontologies</td>
<td>Represent knowledge of a subject domain precisely and in a computer processable way</td>
</tr>
<tr>
<td>Ontology</td>
<td>Characterise one small aspect of interest precisely and in much detail</td>
</tr>
</tbody>
</table>

(also from the 'what and how' book)
Recap and future work

- Representing knowledge of the domain, semantics of the equations, declarative-imperative separation
- Different types of declarative models: diagrams, conceptual data models, ontologies
- Which one suits best depends on the task
Recap and future work

- Representing knowledge of the domain, semantics of the equations, declarative-imperative separation
- Different types of declarative models: diagrams, conceptual data models, ontologies
- Which one suits best depends on the task
- Opportunities for declarative hydrological modelling
- Repurpose some of the design steps from computing to hydrological modelling?
- Use the resulting models for ontology and conceptual model development
Acknowledgments

- Collaborators on some of this work: Pablo Fillottrani (Universidad Nacional del Sur, Argentina), Zubeida Khan (CSIR, South Africa), Agnieszka Ławrynowicz (PUT, Poland), César Bernabé (LUMC, The Netherlands), Zola Mahlaza (UCT)

- Current and former students on some of this work: Kieren Davies

- Funding of this secondment to HSM:
Thank you!

Questions?

- My book on modelling,
- aimed at a broader audience, and
- available in hardcopy and eBook
Thank you!

Questions?

- My textbook on ontology engineering (aimed at computer scientists)
- Also available in paperback (College Publications):
T.A. Halpin.
*Information Modeling and Relational Databases.*

Joshua S. Madin, Shawn Bowers, Mark P. Schildhauer, and Matthew B. Jones.
Advancing ecological research with ontologies.

C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari.
Ontology library.

Hervé Pruvost and Olaf Enge-Rosenblatt.
doi: 10.1061/9780784483893.094.

J. Reese et al.
Generalisable long covid subtypes: findings from the nih n3c and recover programmes.
*eBioMedicine,* 87(104413), 2023.

DP Solomatine and T. Wagener.
Elsevier, 2011.
doi: 10.1016/B978-0-444-53199-5.00044-0.
NeOn methodology for building contextualized ontology networks.
NeOn Deliverable D5.4.1, NeOn Project, 2008.

Albert Tort and Antoni Olivé.
An approach to testing conceptual schemas.

K. Wolstencroft, R. Stevens, and V. Haarslev.
Applying OWL reasoning to genomic data.