Separating Semantics and Implementation

From a Single Ontologically Sound Conceptual Model to Multiple Physical Schema Languages

Bruce T. Bauman, U.S. DoD
Agenda

- Motivation
- Challenge
- OntoUML
- Compiling
  - RDFS
  - XSD
  - SQL
- Summary
- Further Work
- Conclusion
Model Types

Perspective

Enterprise
- Conceptual

Project
- Logical

System
- Physical

Representation
- Neutral

Technology
- Specific

Product
- Specific

Implementation
- Optimized Physical

Greater Reuse
Fewer Models
Greater Detail
Reflects Semantics
Greater Complexity
Model Types - Languages

Perspective

Enterprise

Conceptual

OntoUML

OntoUML with limited encoding annotations

Logical

Full use of encoding annotations

Physical

Technology specific tooling

Optimized Physical

System

Project

Neutral

Technology

Product

Implementation
Why existing languages don’t cut it

- Languages pitched at analysis and design (e.g., UML, ERD)
  - Optimized for design targeting specific technologies
  - Don’t have a well defined semantic mapping
- Implementation languages (e.g., XML Schema, DDL, OWL, RDFS)
  - Clearly they have made implementation trades.
- Natural Language
  - Not precise enough.
Unified Foundational Ontology (UFO)

- Created by Giancarlo Guizzardi
- For Conceptual / Analysis modeling
- Ontologically Based
- UML class diagram notation
Our subset of OntoUML

- **Classes**
  - <<kind>> <<category>>
  - <<role>> <<roleCategory>>
  - <<dependent>> <<associative>>
  - <<event>>
- **Associations**
  - <<non-dependency>> <<dependency>>
  - <<composition>> <<aggregation>> (merelogical relations)
- **Attributes**
- **Datatypes**
  - <<primitive>> <<domain>>3 <<enumeration>>
  - <<structure>> <<union>>3
## Classes

<table>
<thead>
<tr>
<th>Concept</th>
<th>Rigidity</th>
<th>Identity</th>
<th>Existential Dependence</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;kind&gt;&gt;</td>
<td>Rigid</td>
<td>Unified</td>
<td>Independent</td>
<td>Person</td>
</tr>
<tr>
<td>&lt;&lt;role&gt;&gt;</td>
<td>Anti-Rigid</td>
<td>Unified</td>
<td>Independent</td>
<td>Spouse</td>
</tr>
<tr>
<td>&lt;&lt;dependent&gt;&gt;</td>
<td>Rigid</td>
<td>Unified</td>
<td>Dependent</td>
<td>Mental State</td>
</tr>
<tr>
<td>&lt;&lt;associative&gt;&gt;</td>
<td>Rigid</td>
<td>Unified</td>
<td>Dependent</td>
<td>Marriage</td>
</tr>
<tr>
<td>&lt;&lt;category&gt;&gt;</td>
<td>Rigid</td>
<td>Dispersive</td>
<td>Independent</td>
<td>Party</td>
</tr>
<tr>
<td>&lt;&lt;roleCategory&gt;&gt;</td>
<td>Anti-Rigid</td>
<td>Dispersive</td>
<td>Independent</td>
<td>Customer</td>
</tr>
</tbody>
</table>
The finished product
XSD – key assignment

```
<<kind>> Document
| created by |
| 1..1 |
| 1..* |

<<associative>> Document Creator
| created |
| 1..* |

<<role category>> Creator
| identifier : string |
| type = "auto" |

<<category>> Agent
| {abstract} |

<<category>> Party
| {abstract} |
| name : string |
| type = "auto" |

<<category>> Person
| {abstract} |

<<category>> Organization
| {abstract} |

<<role>> Person Creator

<<role>> Organization Creator
```
## Association Encoding - embed

### Source
- **id**: string
- **attrib**: integer

### Target
- **id**: string
- **attrib**: integer

**Association**: A → B

**Include Association**: true

**Navigable**: false

**Include Endpoint**: true

**Encoding**: embed

### Code Example

```xml
<Source>
  <Id>Value</Id>
  <Attrib>Value</Attrib>
</Source>
```

```xml
<Target>
  <Id>Value</Id>
  <Attrib>Value</Attrib>
</Target>
```
Association Encoding - reference

Include Association: true ☑ false ☒ Global ☑

<table>
<thead>
<tr>
<th>A</th>
<th>N/A</th>
</tr>
</thead>
</table>

Include Endpoint:
- true ☒ false ☑
- Encoding: reference

<Source>
  <Target>
  <Id>FK Value</Id>
  </Target>
</Source>
Association Encoding - link

Include Association: true ☐ false ❌ Global ☐

A
N/A

B
Navigable ❌
Include Endpoint
true ❌ false ☐
Encoding: link

<Source>
<Target href=""/>
</Source>
Association Encoding - parts

Source
- id : string
- attrib : integer

Target
- id : string
- attrib : integer

Association:
- A
- 1..*
- association
- B
- 1..*

Include Association:
- true □ false ❌
- Global □

A
- N/A

B
- Navigable ❌
- Include Endpoint
- true □ false ❌
- Encoding : reference

<Source>
<Target>FK Value</Target>
</Source>
Association Encoding - parts

The diagram shows an association between two entities, Source and Target, with attributes and constraints.

### Source
- **id**: string
- **attrib**: integer

### Target
- **id**: string
- **attrib**: integer

The association is defined as follows:

- **Include Association**: true (checked box)
- **Navigable**: false (unchecked box)

The encoding for the association is:

```xml
<Source>
  <Association>
    <Target>
      <Id>FK Value</Id>
    </Target>
  </Association>
</Source>
```
Association Encoding - parts

Source
- id : string
- attrib : integer

Target
- id : string
- attrib : integer

Association A
association
B

Include Association: true ✗ false □

Global ✗

A
N/A

B
Navigable ✗
Include Endpoint true □ false ✗
Encoding : reference

<Association>
<Source>FK Value</Source>
<Target>FK Value</Target>
</Association>
Successes / Observations

- It actually does work
- Used on many projects
- Model reuse is occurring
- Encoding rules are sufficient however new encoding patterns are still being discovered
- Projects select very different encodings
Challenges

- Hard but not too hard
  - Physical Model Duality (encoding can seem like magic)
  - Tools
- Really Hard
  - Talent
  - Culture
- Things to consider
  - Return on Investment
Future Work

- Better visualization
- Change management
- Continue to improve documentation
- Prototype other implementations
- Explore semi-automatic translation
Questions

Bruce Bauman
Senior System Architect
U.S. Department of Defense

btbauma@gmail.com