From Theory to Practice
Translating Schemas into Databases and Interfaces

Sarah Vela
MERIDIAN
Overview

• Introduction to the example case
• Why do schemas differ from tools? Some thoughts…
• Translating a schema
  • How it works
  • Towards a schema translation language
• Examples from the MERIDIAN Discovery Portal development
MERIDIAN Metadata Profile

- Visual version of profile: https://docs.meridian.cs.dal.ca/metadata/

- Tree structure indicates the hierarchy of classes, subclasses, and elements

- One of three requirement types:
  - Mandatory – must be included
  - Optional – can be included at user’s discretion
  - Conditional – may be required depending on a given condition

- Repeatable or non-repeatable
  - How many times an element can be included
The MERIDIAN Discovery Portal and Submission Form

- Online tool meant to support marine resource reuse by helping users find relevant data, software, etc.
- Not a data repository, but a niche search engine
- Relies primarily on manual metadata submission by resource holders
- Five types of resources:
  - Software
  - Model Output
  - Vessel-tracking Data
  - Acoustic Data
  - Bioacoustic Data

To contribute: https://discovery.meridian.cs.dal.ca/index.is
Philosophical Positions

• Schemas are very important
  • Speed up development
  • Prevent costly errors
  • Ensure tools are as useful as they can be

• Schemas are often not made by developers
  • Made by information specialists and subject-domain experts
  • Collaborative or heavily negotiated – not looking for further input

• Schemas want to be blueprints
  • Often difficult to translate directly
  • Arguably not desirable to translate directly
Design Philosophy

Balanced Design

Schema Set
Less Complexity
Less Rich Data
Minimum Data

Developer Set
More Complexity
Richer Data
Maximum Complexity
Translating Schema Language

- Relationships
- Repeatable classes and elements
- Requirement types
  - Mandatory
  - Optional
  - Conditional
Relationships

- Defines structure of database tables, form elements
- Gives direction for reusable components

- What works in theory doesn’t always work in practice
  - E.g. infinite hierarchies
Repeatable

- Defines database structure (one-to-one vs. one-to-many)
- JavaScript duplication of fields
  - Leads to overwhelming forms
- Multi-entry tools
  - Require instruction
  - More complicated validation and post-processing
• 2 hydrophones
  • Each calibrated once
  • 1 minor data corruption issue
• Attached to a rig deployed by 3 grad students
• Form validation
• Database schema requirement
• Gold for search design

• What if the value is not known?
  • Can placeholder values be used?
• Can any of the values be pre-filled?
  • Contextually set values
  • Presumed values (overwritable default)
• Usually indicated with an mark (*) or visual difference
• Limited search use - risks not returning relevant results
• Relational databases – empty fields increase size
• Code has to accommodate value maybe not existing
• Too many optional fields crowd forms and dissuade users
• The nature of the condition is paramount
• Simple conditions:
  • Different forms
  • Responsive forms
  • Responsive search
• Complicated conditions:
  • Provide clear instructions
  • Catch what you can (e.g. custom validation)
  • Hope

Whenever gmd:CI_Citation is used, there must be at least one gmd:CI_Citation with at least one gmd:contactInfo containing an electronic address.

You must provide either an email address or phone number for at least one person.
## A Translation Vocabulary

<table>
<thead>
<tr>
<th>Schema Term</th>
<th>Development Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mandatory</td>
<td>Mandatory - Default</td>
<td>Value is predictable (with or without override option)</td>
</tr>
<tr>
<td></td>
<td>Mandatory</td>
<td>If they don’t know, we don’t want the data</td>
</tr>
<tr>
<td>Mandatory/Conditional</td>
<td>Mandatory - Unknown</td>
<td>If they really don’t know, we still want the data</td>
</tr>
<tr>
<td>Conditional</td>
<td>Conditional – One Of</td>
<td>One field from a set is mandatory, but not each</td>
</tr>
<tr>
<td></td>
<td>Conditional – If/Then</td>
<td>If a field has a value, others are mandatory</td>
</tr>
<tr>
<td>Optional</td>
<td>Optional - Compromise</td>
<td>We would make this mandatory if we could</td>
</tr>
<tr>
<td></td>
<td>Optional - Other</td>
<td>Could be collapsed into one field with example values</td>
</tr>
<tr>
<td></td>
<td>Optional - Fishing</td>
<td>Very low likelihood of user completion, deprioritize</td>
</tr>
</tbody>
</table>
Thank you!
Any questions?