Metadata and Adaptive Object-Models

"Anything you can do, I can do Meta"

Metadata: If something is going to vary in a predictable way, store the *description* of the variation in a database so that it is easy to change. —Ralph Johnson

"Meta is Beta"
**General Problem**

- Requirements change within applications’ domain.
- Business Rules are changing rapidly.
- Applications have to quickly adapt to new business requirements.
- Changing the application is costly, it generally includes code and data-storage.
- There are cycles of: build-compile-release.

**General Solution**

- Create an object design (meta-model) that describes the domain objects which includes attributes, relationships, and business rules as instances rather than classes.
- The domain objects are instantiated through a description given by the user or domain expert.
- Each new requirement is satisfied by creating a new description and a new instantiation.
Adaptive Object-Model (Dynamic Object-Model)

- An ADAPTIVE OBJECT-MODEL is an object model that provides “meta” information about the domain so that it can be changed at runtime
  - explicit object model that it interprets at run-time
  - change the object model, system changes its behavior
- ADAPTIVE OBJECT-MODELS usually arise as domain-specific frameworks
- Business rules can be stored in ADAPTIVE OBJECT-MODELS

Architectural Elements of AOM

- Metadata
- Type Object
- Properties
- Type Square
- Entity-Relationship
- Strategy/Rule Objects
- Interpreters/Builders
- Editors/GUIs
**Type-Object**

Before

SomeClass

SubClass1  SubClass2  ...  SubClassN

After

Entity

- EntityType
  - typeOperations()
  - specificAttributes : type

PLoPD3 - Johnson and Woolf

**Properties**

Before

Entity

- firstAttribute : String = Any

After

Entity

- Property
  - name : String = firstAttribute
  - type : String = String
  - value : String = Any

PLoP98 - Foote and Yoder
Strategies

Adaptive Object-Model
(Very Common Structure)

Type Square
Adaptive Object-Model
(Advantages)

- Can more easily adapt to new business requirements
- Smaller in terms of classes so possibly easier to maintain by experts
- Time to market can be reduced
- More closely

Adaptive Object-Model
(Disadvantages)

- Can be hard to understand and maintain
- Can have higher start up costs
- Can have poor performance
- Requires support tools
Interpreters / Builders: Solution

Medical Observations

Observation Example

Metamodel and GUI

- The metadata can simplify building user interfaces. Special GUI components can be developed for using the metadata.
- Example: The Observation model includes widgets that display list of values from the DiscreteValidators and also EntryBoxes that use RangeValidator.
- A Mediator and Adaptor layer was developed for managing the interactions between the domain objects and the GUIs.
PartyType: Metadata-Editors

Accountability: Metadata-Editors
Observation: Metadata-Editors
### Observation: Metadata-Editors

![Metadata-Editors](image)

### MOF

<table>
<thead>
<tr>
<th>L3</th>
<th>language for defining languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2</td>
<td>A language for defining domain specific software</td>
</tr>
<tr>
<td>L1</td>
<td>A specific software</td>
</tr>
<tr>
<td>L0</td>
<td>An execution of that software</td>
</tr>
</tbody>
</table>

Dimensions of abstraction in Adaptive Object-Models, Reflection and OMG's metamodeling Architecture

---

Adaptive Object-Model

When to Build AOMs

- Need for flexibility
- High pace of business change
- Need for experimentation
- Need to empower user

Advantages of AOM

- Systems can more easily be adapted to domain changes.
- Changes do not require recompiling the system.
- Power Users can change the business rules.
- Shorter time-to-market.
- Smaller in terms of classes so can be easier to maintain by experts.
Disadvantages of AOM

- Developing AOM is expensive.
  (higher startup costs)
- Can be hard to understand and maintain.
  (user-model and meta-model)
- It requires skilled human resources.
- Can have poor performance.
- It demands having infrastructure for storing, building, interpreting metadata (special support tools, editors, etc).

Other Approaches and Technologies

- Black-box Frameworks
- Code Generators
- Metamodeling Techniques
- Table-driven Systems
- Generative Techniques