

# Metadata and Active Object-Models

Joseph W. Yoder  
University of Illinois  
j-yoder@uiuc.edu

Brian Foote  
University of Illinois  
foote@cs.uiuc.edu

Dirk Riehle  
UBS AG, Ubilab  
Dirk.Riehle@ubs.com

Michel Tilman  
Unisys  
mtilman@acm.org

<http://www-cat.ncsa.uiuc.edu/~yoder/Research/metadata>

## Abstract

An object model is an abstract representation of a particular domain, using objects as the description. An active object-model is an object model whose object representation is interpreted or generated at run-time and can be changed with immediate (but controlled) effect on the system interpreting and running it.

Metadata are data that describe other data. When data are accompanied by such descriptions, they can be integrated into new applications on-the-fly. Indeed, when these descriptions are sufficiently rich and powerful, they can constitute a second-level language. This is distinct from the domain level in that it helps to specify the domain in such a way that it is dynamic and flexible. It makes it so that your domain objects can be reified.

Neither metadata nor active object-models are new. Indeed, "meta is beta" was a mantra for database people during the '70s, and the designers of contemporary object-oriented systems build active object-models every day, without knowing they are doing anything special.

This workshop brought together practitioners, language theorists, pattern mavens, architects, and academics to look for the patterns that underlie their disparate interests. By identifying the patterns and establishing a common vocabulary, we hope to relieve those who follow of the burden of reinventing these approaches yet again.

## Introduction

The volatile nature of contemporary business requirements forces developers to make their applications more configurable, flexible, and adaptable. The era where business rules are buried in Cobol code is coming to an end. Today, users need to dynamically change their business rules. Customers require systems that more easily adapt to changing business needs, meet their unique requirements, and scale to large and small installations. Multi-tiered systems often demand that the data that move through

them carry with them their own descriptions. There have been a number of successful frameworks and applications implemented and delivered in different areas of industry that use domain specific languages, meta-data, good object-oriented design, and flexible implementation of business rules to address these sorts of needs.

At OOPSLA'97 more than half of the demos had an explicit representation of their object model that they would interpret. Each system used a different name for this idea, often including "meta" as part of the name. But the name that seemed the clearest was Active Object-Model.

A system with an active object-model has an explicit object model that it interprets at run-time. When you change the object model, the system changes its behavior. For example, a lot of workflow systems have an active object-model. Objects have states and respond to events by changing state. The active object-model defines the objects, their states, the events, and the conditions under which an object changes state. Suitably privileged people can change this object model "without programming". Or are they programming after all? Changing the active object-model changes the way a company does its business, and seems to be programming in a very restricted domain.

Metadata is often used in active object-models to describe the object model itself. When runtime descriptions of these objects are available, users can directly manipulate these objects. Since the system can interpret and manipulate these runtime descriptions, it is easy to add new objects to the active object-model, and make them immediately available to users.

This workshop built on one held in May 1998 at the University of Illinois. Both workshops focused on how real world systems used metadata and active object models to enhance their flexibility, maintainability, ease of use, and power.

Position papers for all workshop participants and the UoI papers can be found at <http://www-cat.ncsa.uiuc.edu/~yoder/Research/metadata>.

## Business Case

People are building systems that use metadata and active object-models from the ground up for a variety of reasons. Indeed, a sound, sober business case can be made. Workshop participants identified the following issues relating to when you want to build these types of systems and reasons that cause active object-models to fail.

- I) *Business Case*
  - a) *Higher overall ROI*
  - b) *Higher flexibility to change*
  - c) *Cheaper to build applications*
  - d) *Foster business innovation*
  - e) *Incremental development and prototyping*
- II) *When to Build Active Object-Models*
  - a) *Need for flexibility*
  - b) *High pace of business change*
  - c) *Need for experimentation*
  - d) *Need to empower users*
- III) *Reasons for Failure*
  - a) *Inadequate bridge between business and technology level*
  - b) *Communication: different universe of discourse*
  - c) *Unclear operations and deployment structure*
  - d) *High availability/runtime evolvability*
  - e) *Security*

*Anything you can do I can do Meta*

## Forces

There are a number of positive and negative forces to consider when developing these types of systems:

- ✓ more or less static type safety
- ✓ flexibility, commonality, generality
- ✓ combinatorial explosion
- ✓ rate of change
- ✓ fuzzy and changing requirements
- ✓ skill level, cleverness
- ✓ users may not break the system
- ✓ understandability; comprehensibility
- ✓ unexpected things happen
- ✓ learning organization
  
- ✗ Performance
- ✗ Complexity meta vs. framework
- ✗ Cost to architect
- ✗ Capability vs. Constraints
- ✗ Responsibility on Users

## Patterns or Techniques

The following is a list of candidate patterns or techniques that were mined at the workshop.

- 1) Parameterization
- 2) Configuration
- 3) Expressions
- 4) Scripts
- 5) Table Driven
- 6) Properties
- 7) Validation
- 8) Type Object
- 9) Variable State
- 10) Strategies
- 11) Rules
- 12) Named Values Namespace
- 13) Context, Naming Services
- 14) Schema
- 15) Specs
- 16) Message Routing
- 17) Editor
- 18) Visual Builder
- 19) History
- 20) Composite
- 21) Builder
- 22) Dynamic Relationship
- 23) Observers
- 24) Value Holders
- 25) DAGs – connection between structure and behavior
- 26) Dynamic Accessors
- 27) Boot Strapping – Initial Population
- 28) Scripting Tools
- 29) Pluggable Behavior
- 30) Prototypes
- 31) Factories
- 32) Views

The participants began to categorize these into:

Ways to Describe Data and Behavior  
4, 5, 12, 13 14, 15, ...

How to Implement Active Object-Models  
1, 2, 3, 5, 6, 7, 8, 9, 10, ...

Tools to Define and Implement such models  
17, 18, 25, 28, ...

A number of papers have been written describing the qualities and patterns that make up these types of dynamic systems [Anderson 98, Foote and Yoder 96, Foote and Yoder 98, Johnson and Oaks 98, Roberts and Johnson 97, Johnson and Woolf 97, .].

## Terminology

A perhaps unfortunate legacy of our reflection heritage is its arcane vocabulary. What follows is a list of terms that were identified by the workshop participants as central to discussion.

- Meta
- Reflection
- Business Rules
- Self Describing Data Structures
- Runtime Modifiable Systems
- Self Contained Structures
- Self Aware Software
- Introspection
- Dynamic System
- Black-Box Framework
- Domain Specific Language
- Causal Connection

## How to Teach People Meta

When dealing with meta systems, training and knowledge transfer can be much more difficult. We identified different areas of knowledge transfer, ways to document systems, and process models for communicating this knowledge.

- Dimensions of Knowledge Transfer
  - Thought Limited Developer
  - Manager (Business Case)
  - Users of the System (End Users / Core Developers)
- Documentation
  - Patterns, Cookbooks, ...
  - Tools (Browsers, Auto Generated, ...)
  - Bring Closer to Mentoring
- Process Model
  - End-to-End (Vision of Overall System)
  - Continuity
  - Mentoring
  - Cross Breeding
  - Communication Culture

## Participants

The participants for the workshop for the workshop included Ali Arsanjani, Jens Coldewey, David Delano, Randy Dong, Brian Foote, James Long, Donald Malcolm MacQueen, Lourdes Tajés Martínez, Richard McClatchey, Eliot Miranda, Vince Nibler, Joseph Pelrine, Claudia Pons, Dirk Riehle, Arnon S. Rosenthal, Wolf Siberski, Peter Sommerlad, Michel Tilman, Layda, Torsten, Weerasak Witthawaskul, Bobby Woolf, and Joseph Yoder.

## Conclusions

A good sign that a technical area is ripening is that a number of people independently discover it. This would seem to be the case with metadata and active object-models. This workshop brought together researchers and practitioners from a variety of areas in order to explore the patterns and themes that underlie such systems.

A dominant theme was that the need to confront change is forcing system architects to find ways to allow their systems and users to more effectively keep pace with it. One way to do this is to cast information like business rules as data rather than code, so that it is subject to change at runtime. When such data are objects, these objects can, in time, come to constitute a domain-specific language, which allows users themselves to change the system as business needs dictate.

A major accomplishment of this workshop was to finally get this disparate group together, and to establish this dialog. However, we've only begun the task of fleshing out these architectural issues, uncovering the patterns, and of better defining our vocabulary. We are looking forward to reconvening the members of this community to continue this work.

## References

1. Anderson, Francis. "A Collection of History Patterns," *Collected papers from the PLoP '98 and EuroPLoP '98 Conference*, Technical Report #wucs-98-25, Dept. of Computer Science, Washington University Department of Computer Science, September 1998.
2. Brian Foote and Joseph Yoder. "Architecture, Evolution, and Metamorphosis," *Pattern Languages of Program Design 2*, John M. Vlissides, James O. Coplien, and Norman L. Kerth, eds., Addison-Wesley, Reading, MA., 1996.
3. Brian Foote and Joseph Yoder. "Metadata and Active Object-Models," *Collected papers from the PLoP '98 and EuroPLoP '98 Conference*, Technical Report #wucs-98-25, Dept. of Computer Science, Washington University Department of Computer Science, September 1998.
4. Don Roberts and Ralph Johnson. "Type Object," *Pattern Languages of Program Design 3*, Robert Martin, Dirk Riehle, and Frank Buschmann, eds., Addison-Wesley, Reading, MA., 1997.
5. Ralph Johnson and Jeff Oaks. "The User-Defined Product Framework," URL: <http://st-www.cs.uiuc.edu/users/johnson/papers/udp/>
6. Ralph Johnson and Bobby Woolf. "Type Object," *Pattern Languages of Program Design 3*, Robert Martin, Dirk Riehle, and Frank Buschmann, eds., Addison-Wesley, Reading, MA., 1997.