Metadata and Adaptive Object-Models

Joseph W. Yoder [1], Reza Razavi [2]

[1] The Refactory, Inc. - 209 W. Iowa
Urbana, IL 61801 USA
yoder@refactory.com

[2] University of Paris 6, LIP6 - CNRS
Case 169, 4, place Jussieu,
75252 PARIS Cedex 05, FRANCE
Reza.Razavi@lip6.fr

Abstract. The unrelenting pace of change that confronts contemporary software developers compels them to make their applications more configurable, flexible, and adaptable. A way to meet such requirements is to use an Adaptive Object-Model (AOM). This paper describes common architectures for adaptive object-models and summarizes the results from our ECOOP 2000 workshop. Participants to this workshop focused on comparisons between the *Adaptive Object-Model's* approach and those of *Reflection* and *Metamodeling*. It emerged that there are common themes present in all three approaches and that these approaches can compliment one another for assisting developers in designing and building systems that can more quickly adapt to new and changing business requirements.

What is an Adaptive Object-Model?

The era when business rules were buried in code is coming to an end. Today, users themselves often seek to dynamically change their business rules without the writing of new code. Customers require that systems are built that can adapt more easily to changing business needs, that can meet their unique requirements, and can scale to large and small installations.

On the other hand, the same technique is adequate for the slightly different purpose of producing a whole line of software products: of course, a line of products may be obtained by variously instantiating a unique abstract model, but also by adapting a given initial system to various requirements that appear simultaneously instead of evolving in time. Moreover, the diversification of a successful product may also be seen as a form of reengineering.

Black-box frameworks provided early solutions for the design of flexible implementation of business rules [3]. Recent research in the different types of architectures to meet such requirements from an object-oriented perspective has been catalogued as *Adaptive Object-Models* [1, 2, 5, 13, 16]. An *Adaptive Object-Model* is where the object representation of the domain under study has itself an explicit object model (however partial) that is interpreted at run-time. Such an object model can be changed with immediate (but controlled) effect on the system interpreting and running it. Note that *Adaptive Object-Models* usually requires a thorough analysis of the domain at hand, which may very well include a black-box framework as an initial stage.

Objects have states and respond to events by changing state. The *Adaptive Object-Model* defines the object model, i.e. the objects, their states, the events, and the conditions under which an object changes state, in a way that allows for dynamic modification. If you change the object model, the system changes its behavior. For example, such a feature makes it easy to integrate a workflow mechanism, which proves useful in many systems [4, 11].

Adaptive Object-Models successfully confront the need for change by casting information like business rules as data rather than code. In this way, it is subject to change at runtime. Using objects to model such data and coupling an interpretation mechanism to that structure, we obtain a domain-specific language, which allows users themselves to change the system following the evolution of their business.

Metadata[1] is then often used in adaptive 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 the metadata to build and manipulate these runtime descriptions, it is easy to add new objects to the adaptive object-model, and make them immediately available to users. This approach has been validated by several successful industrial projects [6,7,9,11,15].

Metadata and Adaptive Object-Model workshops at the University of Illinois in 1998 and at OOPSLA '98 and '99 were devoted to "pattern mining" some of these ideas. Our workshop at ECOOP '00 (June 2000 at Cannes in France) was oriented toward a broader analysis of the possible uses of the *Adaptive Object-Model* technology, based on the examination of a range of examples. This paper describes the results of the concrete approaches for building dynamically adaptable systems.

Hot topics in Adaptive Object-Modeling

This workshop evolved from the previous workshops held at the University of Illinois and OOPSLA. Those workshops had many papers submitted from researchers and practitioners dealing with issues involving dynamically adaptable systems. This lead

[1] MetaData can be described by saying that 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. In other words, if something is going to change a lot, make it easy to change. The problem is that it can be hard to figure out what changes, and even if you know what changes then it can be hard to figure out how to describe the change in your database. Code is powerful, and it can be hard to make your data as powerful as your code without making it as complicated as your code. But when you are able to figure out how to do it right, metadata can be incredibly powerful, and can decrease your maintenance burden by an order of magnitude, or two. [R. Johnson]

to the following which is a list of the primary topics that we have been addressing in the field of *Adaptive Object-Models* and which was part of the call for participation to our workshop.

1. Experiences in developing systems, which have been developed with, or might have benefited from, an Adaptive Object-Model.
2. Building a common vocabulary (terminology, definitions, etc,); e.g. Typical AI terminology (ontologies) vs. software engineering terminology
3. Models and languages for Metadata and AOMs, including current standardization efforts and expressive (even ambiguous) type languages,....
4. Comparative works with similar approaches like Metamodeling and Reflection.
5. Specific implementation techniques for Adaptive Object-Models.
6. Design concepts and classification according to semantics and relevance.
7. Appropriate development process models for cultivating Adaptive Object-Models (analysis patterns, design patterns, ...), and the impact of AOMs on development process models (iterative, incremental, testing, ...).
8. How do requirements such as heterogeneous, distributed environments, smart software and agents, and user empowerment drive the evolution of Adaptive Object-Models? How can we overcome resistance to building Adaptive Object-Models from programmers and managers?
9. How do get Adaptive Object-Models to exhibit acceptable performance? Can we borrow from traditional disciplines, e.g. caching, Just-In-Time compilation, heuristics, statistics, stereotypical vs. customized interfaces, and tools?
10. How do measure such aspects of Adaptive Object-Models as stability, class library complexity and related metrics, learning curve, cost (development of the 'framework', building the applications, change, maintenance), and performance?
11. Dealing with views continues to be a challenge; can we use Adaptive Object-Models to solve some of these problems? For instance, some models replace a hard-wired view on classification of objects by a dynamic view based on observational equivalence of object behavior. This view takes into account dynamic properties of the parties involved in a contract. If Adaptive Object-Models are any help, which qualities must they possess?
12. Most Adaptive Object-Model implementations focus on one or two isolated aspects, e.g. object model, business rules, mapping objects onto a database. Other aspects are typically left to more traditional approaches. Hence, they often tend to interfere in ad-hoc and hard-wired ways. For instance, it is often hard to design good, pro-active user interface code that is cleanly separated from the business rules without introducing redundancy of one sort or another. Can we use weaver-like approaches in combination with Adaptive Object-Models to achieve more global-wide, yet dynamic reflection?

Comparative Summary of Contributions

This workshop brought together researchers and practitioners from a variety of areas in order to explore the themes that underlie such systems. Position papers for all workshop participants can be found at http://www-poleia.lip6.fr/~razavi/aom/papers.

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 AOMs. In the main, participants focused on comparisons between the Adaptive Object-Model's approach and those of Reflection and Metamodeling. It emerged that indeed all three approaches share the same four levels of abstraction.

The following is a more detailed view of the main positions that were presented for participation at the workshop.

Using AOMs for the description of types of phenomenon over a period of time

Analysis Patterns are solutions to recurrent problems in modeling a problem domain. However, analysis patterns are not necessarily easy to implement. We have developed several applications that capture medical observations about patients using the Observer pattern. This paper [9] describes how we dealt with problems while evolving from analysis to design, how we extended the *Observation* pattern to work in our environment, and the details of how we implemented this pattern. This is a very good example of using *Adaptive Object-Models* to allow end-users to dynamically customize their system without needing to write new code.

Using AOMs for building Flow-independent End-User Programming Systems

The goal of *end-user-driven application development* is to provide end users with greater computational power to create, customize and extend their software.

Adaptive Object-Model architectural style is particularly adequate for building end-user programming systems. By providing design patterns to create dynamically adaptable systems, they offer an appropriate foundation to the problem of facilitating programming for end users and domain experts.

However, creating flow-independent [4] end-user programming systems in the context of Adaptive Object-Models is still a problem.

Type Cube [14], as shown in Fig-1, is a model that addresses this problem by coupling micro-workflow [4] and the core of Adaptive Object-Models [6, 7]. It provides guidelines for designing object-oriented applications that can be extended and personalized by dynamic definition of new entity types and processes. This design work can be done by domain experts. *Type Cube* has been validated by two industrial projects.

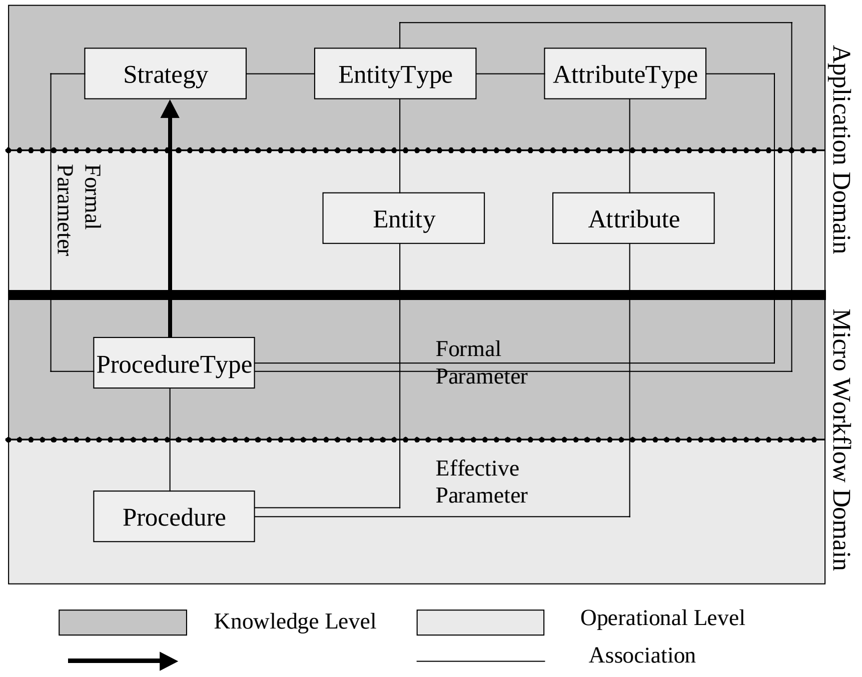

Fig-1. Type Cube Model building flow-independent end-user programming systems.

AOMs for building Data Warehouse and Business Intelligence Environments

The Common Warehouse Metamodel (CWM) is a recently adopted standard of the Object Management Group (OMG) for metadata interchange in the data warehouse and business intelligence environments. CWM extends the OMG's standard MOF/UML/XMI metamodeling architecture with data warehousing and business intelligence domain concepts. CWM supports a model-driven approach to metadata interchange, in which object models representing shared metadata are constructed according to the specifications of the CWM metamodel. Tools agree on fundamental domain concepts (as defined by the CWM metamodel) and, therefore, are capable of understanding a wide range of models representing particular metadata instances.

We believe that the OMG architecture (in particular, the MOF ontology, meta-layer stack, and semantics imposed on compliant metamodels) generally allows for the creation of metamodels whose instances readily align with (or reveal or expose) the fundamental patterns of *Adaptive Object-Models*. The CWM metamodel, by directly extending MOF/UML, drives support for Adaptive Object-Model patterns into the data warehousing and business intelligence domains, leading the way for a new generation of data warehousing and business intelligence tools that are dynamically configurable and highly adaptive to changing environments. Such tools would be driven by *Adaptive Object-Models*, with CWM serving as the foundational meta-model guiding the creation of those Adaptive Object-Models.

Support for Adaptive Workflow (CRISTAL Project at CERN, Geneva)

The current trend in software engineering is towards developing complex systems for use by growing and changing communities of users that require access to large-scale data repositories or data warehouses. Inherent in the approach to designing such systems is the ability for systems to cope with changing user requirements, to be inter-operable with existing (legacy) systems, to handle multiple versions of reusable code and to be configurable as needs evolve. One approach that is gaining favor is to develop systems that are Description-driven [12] i.e. that not only handle and allow manipulation of data objects but also objects that capture system description.

Using AOMs for Content Management Transactions

Another domain where *Adaptive Object-Models* have been identified is form-database interaction in a content management system. Handles for read/write data-base statements have properties, strategies and can be implemented as type objects [10].

An additional complication, which came up in this application, was the need for a transaction manager to orchestrate the interaction of individual objects, which raised the question of how to formalize different pathways through a tree of interdependencies.

Summary of discussions

A majority of participants observed that *Adaptive Object-Modeling*, Metamodeling and Reflection share the same dimensions of abstractions as shown in Fig-2. The top most level, called L3, represents *a language for describing languages*. The next level, L2, represents a domain specific language, derived by applying L3 to the application domain. The L1 level represents the specification for a particular software (system). Finally, the L0 level represents is an instance of that specification.

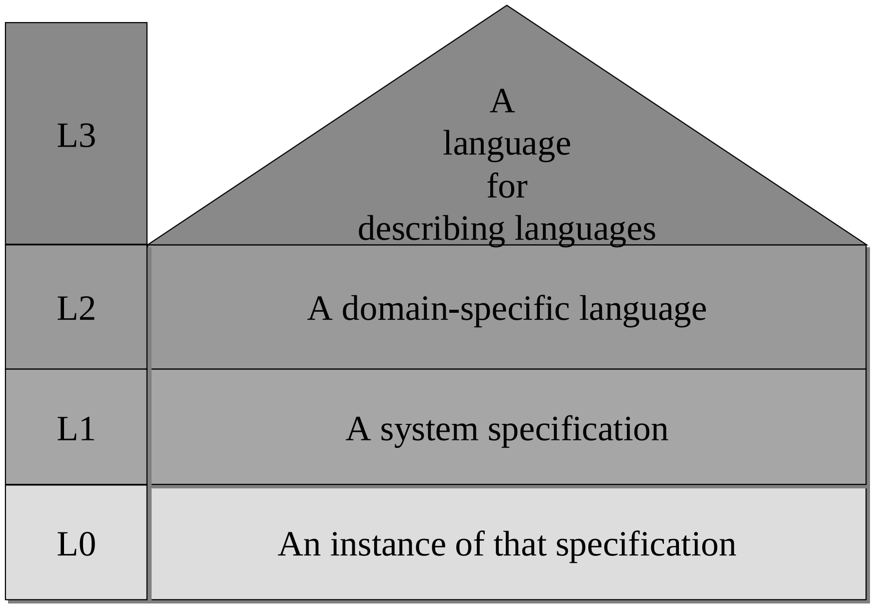

Fig-2. Dimensions of abstraction; AOMs, Reflection and OMG 's metamodeling Architecture.

Type Cube (Fig. 1) is an example of this conceptual layering from Adaptive Object-Modeling. Applying *Type Cube* to an application domain delivers an adaptive object-model-based domain specific language for that domain. This language allows experts to specify their applications, which are then instantly executable.

The Reflection community has also used similar approaches to provide adaptable programming languages. Many reflection techniques can be found in *Adaptive Object-Models*.

The Common Warehouse Metamodel (CWM) is another example from Metamodeling , that has already adopted this four level architecture. In fact, the MOF ontology, meta-layer stack, and semantics is imposed on compliant metamodels.

CWM extends the OMG's standard MOF/UML/XMI metamodeling architecture with data warehousing and business intelligence domain concepts. CWM supports a model-driven approach to metadata interchange, in which object models representing shared metadata are constructed according to the specifications of the CWM metamodel. Tools agree on fundamental domain concepts (as defined by the CWM metamodel) and, therefore, are capable of understanding a wide range of models representing particular metadata instances.

The OMG architecture generally allows for the creation of metamodels whose instances readily align with (or reveal or expose) the fundamental patterns of *Adaptive Object-Models*. The CWM metamodel, by directly extending MOF/UML, drives support for Adaptive Object-Model patterns into the data warehousing and business intelligence domains, leading the way for a new generation of data warehousing and business intelligence tools that are dynamically configurable and highly adaptive to changing environments. Such tools would be driven by *Adaptive Object-Models*, with CWM serving as the foundational metamodel guiding the creation of those Adaptive Object-Models.

Conclusions

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 these changes. A 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 reflected as 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. It was noted that principles of Reflection and MOF could be used to better support the goals of AOMs and vice-versa. We are looking forward to reconvening the members of this community to continue this work. The focus of the next meeting will be around on seeing where these communities meet and how they can support one another for achieving the goal of building dynamically adaptable systems.

References

1. 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.
2. 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, Sept 1998.
3. Don Roberts and Ralph Johnson. “Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks,” *Pattern Languages of Program Design 3,* Robert Martin, Dirk Riehle, and Frank Buschmann, eds., Addison-Wesley, Reading, MA., 1997.
4. Dragos A. Manolescu and Ralph E. Johnson , Dynamic Object-Model Workflow Framework for Developers. URL: http://micro-workflow.com/
5. Francis Anderson. “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, September 1998.
6. Francis Anderson and Ralph Johnson. The Objectiva telephone billing system. MetaData Pattern Mining Workshop, Urbana, IL, May 1998. Available on the Web at: http://www.joeyoder.com/Research/metadata/UoI98MetadataWkshop.html.
7. Jeff Oakes and Ralph Johnson. The Hartford insurance framework. MetaData Pattern Mining Workshop, Urbana, IL, May 1998. Available on the Web at: http://www.joeyoder.com/Research/metadata/UoI98MetadataWkshop.html.
8. John Poole - The Common Warehouse Metamodel as a Foundation for Active Object Models in the Data Warehouse Environment. Position paper to ECOOP‘2000 workshop on Metadata and Active Object-Model Pattern Mining - Cannes, France.
9. Joseph W. Yoder, Federico Balaguer, Ralph Johnson - From Analysis to Design of the Observation Pattern. Position paper to ECOOP‘2000 workshop on Metadata and Active Object-Model Pattern Mining - Cannes, France.
10. Holger Blasum, Thorsten Weigl - Content management transactions -applicability of a dynamic object model. Position paper to ECOOP‘2000 workshop on Metadata and Active Object-Model Pattern Mining - Cannes, France.
11. Martine Devos and Michel Tilman. A repository-based framework for evolutionary software development. MetaData Pattern Mining Workshop, Urbana, IL, May 1998.
12. P. Brooks, Z. Kovacs, J-M Le Goff, R. McClatchey, T. Solomonides & N. Toth. Ontologies, Patterns and Description-Driven Systems. Position paper to ECOOP‘2000 workshop on Metadata and Active Object-Model Pattern Mining - Cannes, France.
13. 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.
14. Reza Razavi - Coupling The Core of Active Object-Models and Micro Workflow --- Foundations of a Framework for Developing End User Programming Environments. Position paper to ECOOP‘2000 workshop on Metadata and Active Object-Model Pattern Mining - Cannes, France.
15. Reza Razavi. “Active Object-Models et Lignes de Produits”. OCM’2000. Nantes, France. May 2000. URL: http://www-poleia.lip6.fr/~razavi/ocm2000.
16. Joseph W. Yoder, Brian Foote, Dirk Riehle, and Michel Tilman Metadata and Active Object-Models Workshop Results Submission; OOPSLA Addendum, 1998.

Workshop participants

Our workshop gathered for the first time researchers from *Adaptive Object-Modeling* with those from MetaModeling and Reflection. Table 1 shows the list of all participants to our workshop:

Name	Contact
Y. Shaham-Gafni	gafni@il.ibm.com
J. Bézivin	jean.bezivin@sciences.univ-nantes.fr
H. Blasum	blasum@muc.de
V. Couturier	vincent.couturier-9333783@netserver.univ-lyon3.fr
N. Parlavantzas	arlavantzas@lancaster.ac.uk
J. Poole	John_Poole@hyperion.com
R. Razavi	Reza.Razavi@lip6.fr
N. Revault	Nicolas.Revault@lip6.fr
G. Rozsavolgyi	Gerard.Rozsavolgyi@lip6.fr
N. Toth	Norbert.Toth@cern.ch
J. Yoder	joeyoder@joeyoder.com

Table 1. Workshop participants