The Refactory, Inc.
7 Florida Drive. • Urbana IL 61801 • phone: (217) 344-4847 • fax: (217) 384-4458 • board@refactory.com

Title: Adaptive Object-Model Architecture: How to Build Systems That Can Dynamically Adapt to Changing Requirements

Speaker: Joseph W. Yoder

Abstract:

Architectures that can dynamically adapt to changing requirement are sometimes called "reflective" or "meta" architectures. We call a particular kind of reflective architecture an "Adaptive Object-Model (AOM)" architecture. An Adaptive Object-Model is a system that represents classes, attributes, relationships, and behavior as metadata. It is a model based on instances rather than classes. Users change the metadata (object model) to reflect changes to the domain model. These changes modify the system's behavior. In other word, it stores its Object-Model in XML files or in a database and interprets it. Consequently, the object model is adaptive; when the descriptive information for the object model is changed, the system immediately reflects those changes. We have noticed that the architects of a system with Adaptive Object-Models often claim this is the best system they have ever created, and they brag about its flexibility, power, and eloquence. At the same time, many developers find them confusing and hard to work with. This is due in part because the developers do not understand the architecture. This tutorial will give a description of the Adaptive Object-Model architectural style and will make it easier for developers to understand and build systems that need to adapt to changing requirements.

Outline:

- Introduction
- Motivation – a few examples
  - General Problem
  - General Solution
- Architectural Elements of AOM
- An Example in the Medical Domain – These are examples of the application we developed and the framework as it evolved
- Implementation Issues
- Advantages and Disadvantages
- Other Alternatives - Related Ideas and Architectures
- Summary and Questions

Duration: Half-day

Level: Advanced

Required Experience:

A good knowledge of object concepts is required. It would be useful if participants have a basic understanding of frameworks, though it is not necessary. A general understanding of the GOF patterns is required. A general understanding of Analysis Patterns (specifically knowledge levels) and Reflective Architectures can be helpful; though not required. Specifically we will be covering Composite, TypeObject, Properties, Strategy, Interpreter and the Builder Design Patterns along with the Party, Accountability and Observation Analysis Patterns.

Expected Audience:

The intended audience is for those that need to build, maintain, or understand flexible architectures that allow "power" users to change the object model at runtime. It is also intended for those that are working with meta-architectures to allow a system to adapt to changing requirements at runtime. People attending this tutorial will learn how to use Composite, TypeObject, Properties, Strategy, Interpreter and Builder Design Patterns to implement Adaptive Object-Models such as those described by dynamic systems modeled by Hay's and Fowler's Analysis Patterns.

References:

http://www.adaptiveobjectmodel.com

Presenter Profile:

Joseph W. Yoder has worked on the architecture, design, and implementation of various software projects dating back to 1985. These projects have incorporated many technologies and range from stand-alone to client-server applications, multi-tiered, databases, object-oriented, frameworks, human-computer interaction, collaborative environments, web-based, and domain-specific visual-languages. In addition these projects have spanned many domains, including Medical Information Systems, Manufacturing Systems, Medical Examination Systems, Statistical Analysis, Scenario Planning, Client-Server Relational Database System for keeping track of shared specifications in a multi-user environment, Telecommunications Billing System, and Business & Medical Decision Making. Recently his focus has been on how to build dynamic and adaptable systems. This has led to work on Adaptive Object-Models which are systems that have an architecture to allow for systems to adapt to changing requirements without programming.

Joseph W. Yoder has assisted many companies with the development of software applications, specifically object-oriented and web-based systems. Joe has mentored object-oriented developers and provided internal training on using patterns to assist with object-oriented development. Recently he has been teaching Java, Smalltalk, Patterns, Frameworks, Object-Oriented Analysis and Design and has mentored analysts and developers on many of their applications. He was also involved in the management and development of a reusable Enterprise Class Libraries.

Joe is the author of over two-dozen published patterns and has been working with patterns for a long time, writing his first pattern paper in 1995, and was the conference chair for the PLoP'97, conference on software patterns and was the programming chair of The Second Latin American Conference on Pattern Languages of Programming.

Joe enjoys building elegant and successful systems, helping people succeed, and learning new things. He wants to continue to provide analysis, design, and mentoring and to write papers that reflect these experiences.

Main topic(s):

- Adaptable Systems
- Architectural analysis/design/patterns
- Components
- Domain Specific Language
- Dynamic Object-Model
- Emerging technologies
- Frameworks
- Generative Programming
- Metadata
- Meta-Modeling
- Meta-Architectures
- Model-Driven Architecture
- Reflective and metalevel
- Reusable Black-Box Components
- Evolving Requirements

The following includes and example of the first few slides of the presentation.

Adaptive Object-Model Architecture

"How to Build Systems that can Dynamically Adapt to changing Requirements"

www.adaptiveobjectmodel.com
www.refactory.com

By: Joseph W. Yoder

Table of Contents

Table of Contents (cont.)

General Problem

- Requirements change within applications' domain.
- Requirements and 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

- An ADAPTIVE OBJECT-MODEL is an object model that provides “meta” information about itself 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
- Rules and Behavior are stored as descriptive (meta) information in ADAPTIVE

Adaptive Object-Model

- Represents classes, attributes, relationships, and behavior as *metadata*.
- Based on instances rather than classes.
- Users change the *metadata* (object model) to reflect changes in the domain.
- Stores its *Object-Model* in a database or in files and interprets it (can be XML/XMI).

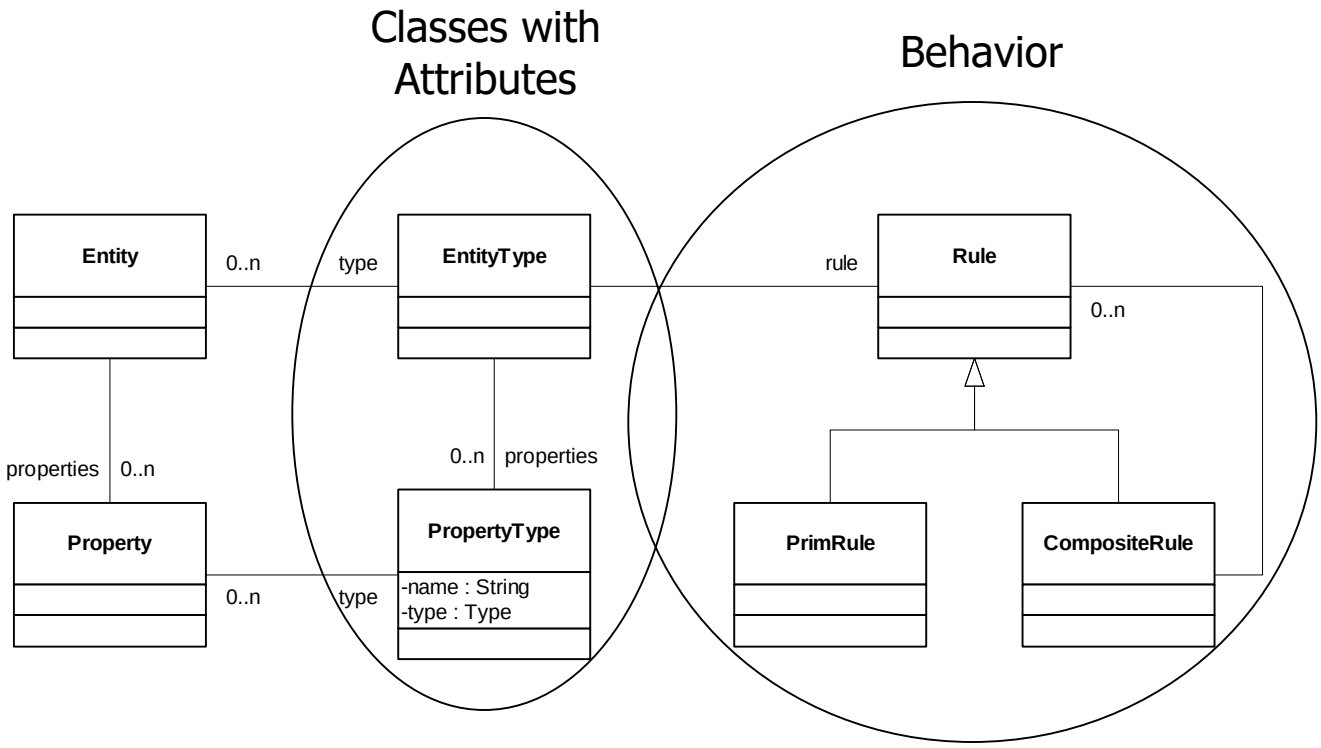
Consequently, the object model is adaptable, when you change it, the system reflects those changes immediately.

Architectural Elements of AOM

- Metadata
- TypeObject
- Properties
- Type Square
- Entity-Relationship
- Strategy/RuleObjects
- Interpreters/Builders
- Editors/GUIs

Sometimes called a "reflective architecture" or a "meta-architecture".

Adaptive Object-Model
(Very Common Structure)



- ECOOP & OOPSLA 2001 Yoder, Balaguer, Johnson