# Modelos Adaptativos de Objetos

Joseph W. Yoder
Ralph Johnson

Adriano Saturno Muniz



## Introdução

- Como construir sistemas que mudam suas regras de negócio freqüentemente?
- Como construir sistemas que podem ser alterados facilmente?
- Como construir sistemas que podem mudar suas funcionalidades em tempo de execução, sem a necessidade de recompilar código-fonte?



## Introdução (cont.)

- Arquiteturas reflexivas ou meta-arquiteturas.
- Adaptação a novos requisitos do sistema em tempo de execução.
- Uso de informações descritivas.
- Modelos Adaptativos de Objetos (MAO).
- Modelo Adaptativo de Objeto: classes, atributos, relacionamentos e comportamento como meta-dados.
- Sistema baseado em instâncias.



## Arquitetura MAO

- Difere da arquitetura usual de orientação a objetos.
- Arquitetura usual: classes representam entidades de negócio.
- Arquitetura MAO: meta-dados representam entidades de negócio.
- Meta-dados são interpretados em tempo de execução.



## Arquitetura MAO (cont.)

- Arcabouço construído a partir de padrões menores:
  - Objeto Tipo (Type Object);
  - Propriedade (Property);
  - Estratégia (Strategy);
  - Regra de Objeto (Object Rule);
  - Composição (Composite);
  - Intérprete (Interpreter);
  - Construtor (Builder).



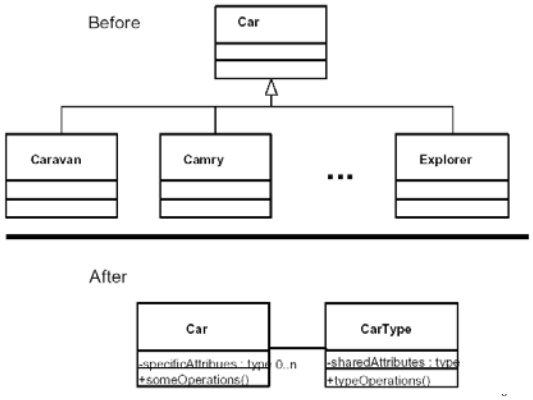
## Objeto Tipo

- Sistemas orientados a objetos convencionais:
  - Programa = conjunto de classes;
  - Uma classe define a estrutura e comportamento do objeto;
  - Uma classe para cada tipo de objeto;
  - Novos objetos implicam em novas classes;
  - Problemas com classes com número desconhecido de subclasses.

## Objeto Tipo (cont.)

- Aplicando o padrão Objeto Tipo:
  - Cada subclasse é uma abstração de um elemento do domínio;
  - Todas as subclasses tornam-se instâncias;
  - Novas classes criadas em tempo de execução;

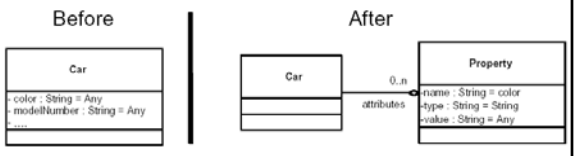
7




## Propriedade

- Sistemas orientados a objetos convencionais:
  - Atributos implementados pelas variáveis de instância;
  - Atributos definidos em cada subclasse;
  - Como variar atributos entre objetos de uma mesma classe?
- Usando o padrão Propriedade:
  - Crie uma variável de instância que recebe uma coleção de atributos.

9

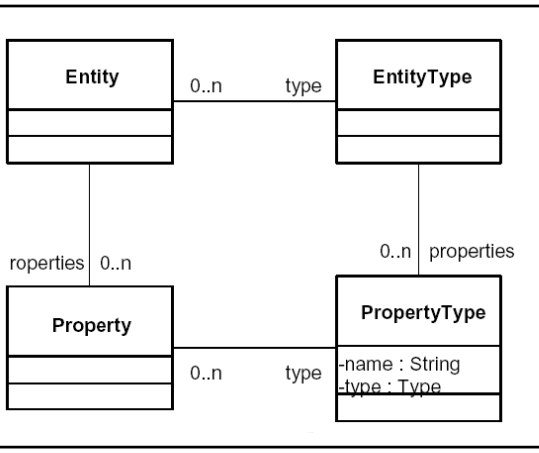



10

## Objeto Tipo + Propriedade

- Objeto Tipo é normalmente usado duas vezes: antes e depois do Propriedade;
- Objeto Tipo: Entidades x Tipos de Entidades;
- Propriedade aplicado às Entidades;
- Objeto Tipo nas Propriedades: Propriedades x Tipos de Propriedade;
- Cada Tipo de Entidade escolhe os Tipos de Propriedades para suas Entidades.

11



## Entidade - Relacionamento

- Como descrever os relacionamentos?
  - Atributos
    - propriedades referentes a tipos primitivos;
    - Associações de sentido único.
  - Relacionamentos
    - propriedades referentes a Entidades;
    - Associações de sentido duplo.
- Como separar atributos de relacionamentos?
  - Usar o padrão Propriedade duas vezes;
  - Criar duas subclasses de Propriedades;
  - Usar o valor da propriedade para separar atributos de relacionamentos.

13

## Estratégia / Regra de Objeto

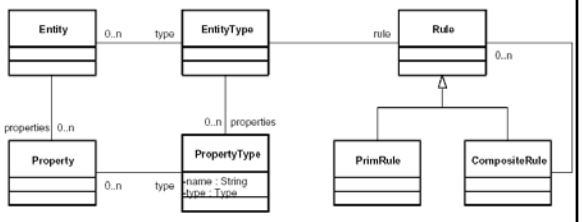
- Como representar regras de negócios?
- Algumas regras ainda não podem ser tratadas com o exposto até agora.
- São implementadas usando os padrões Estratégia e Regra de Objeto.
- Estratégia: objeto que representa um algoritmo.

14

## Estratégia / Regra de Objeto (cont.)

- Modelos Adaptativos de Objetos começam com estratégias simples.
- Estratégias são mapeadas para os Tipos de Entidades através de meta-dados.
- Informações descritivas interpretadas em tempo de execução.
- Objetos Regra: Estratégias mais complexas para regras de negócio mais complexas.

15




16

## Intérpretes de Meta-dados

- Meta-dados: Descrições das regras de negócio
- São interpretados duas vezes:
  - Instanciação dos objetos;
  - Interpretação das regras em tempo de execução.
- Podem ser armazenados em bancos de dados ou arquivos XML

17

## Intérpretes de Meta-dados (cont.)

- Meta-dados têm de ser inicialmente interpretados para a construção da estrutura do Modelo Adaptativo de Objeto.
- Podem ser usados os padrões Intérprete e Construtor.
- Construída a estrutura os Meta-dados são interpretados novamente: o comportamento do sistema é montado.

18

## Intérpretes de Meta-dados (cont.)

- Linguagens específicas a domínios podem ser criadas se necessário.
- Sistemas complexos requerem gramáticas, linguagens restritivas e intérpretes mais complexos.
- Regras gramaticais são difíceis de conceber, mas são bem aceitas pelo usuário final se forem intuitivas.

19

## Exemplo de MAO

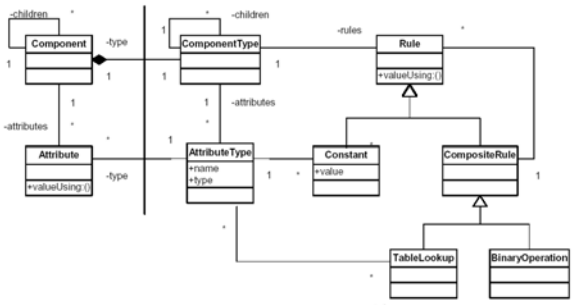
- Arcabouço Produto Definido por Usuário (User-Defined Product Framework).
  - Desenvolvido em Hartford, onde é usado para representar apólices de seguro;
  - Facilita a especificação, representação e manipulação de objetos complexos com atributos que são funções de seus componentes;
  - Permite criar novos objetos de negócio a partir de componentes conhecidos e deixa o usuário definir novos tipos de componentes sem a necessidades de programar;

20

## Exemplo de MAO (cont.)

- A arquitetura desse arcabouço é muito similar ao MAO descrito anteriormente.
  - Novos objetos de negócio são criados instanciando ComponentTypes, os quais definem seus AttributesTypes permitidos;
  - O padrão Estratégia foi aplicado em ComponentTypes e AttributeTypes para definir as regras a eles associadas

21

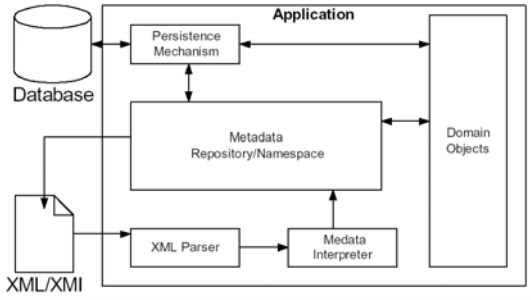



22

## Tópicos de Implementação

- Criando Modelos Persistentes
  - MAO representa meta-dados como objetos comuns;
  - Podem ser armazenados em bancos de dados através de regras conhecidas;
  - É possível usar XML para armazenas meta-dados;
  - Independente da forma de armazenamento o sistema deve ser capaz de interpretar os meta-dados e povoar um MAO corretamente;
  - Os padrões Construtor e Interprete podem ser usados nessa tarefa.

23




24

## Tópicos de Implementação

- Apresentando o modelo ao usuário
  - Interfaces gráficas são importantes no MAO;
  - A configuração do sistema é feita pelo usuário final através de alguma interface;
  - A construção de interfaces não se beneficia do reaproveitamento de código-fonte;
  - Interfaces gráficas tornam a apresentação / modificação do sistema mais amigável.

25

## Tópicos de Implementação

- Mantendo o modelo
  - Armazenar meta-dados num banco de dados não é tão simples quanto armazenar objetos comuns;
  - Uma solução é desenvolver ferramentas e editores para auxiliar os programadores;
  - Arcabouço estilo caixa-preta.

26

## Tópicos de Implementação

- Histórico de regras e dados
  - Manter uma trilha das mudanças nos valores, nas versões e nas regras ao longo do tempo;
  - O interpretador deve ser mudado para sempre pegar o valor atual, a versão e as regras corretas

27

## Conseqüências de MAO

- Vantagens:
  - Facilidade de efetuar mudanças;
  - Usuários "programam sem programar";
  - Converter um sistema num MAO reduz o número de classes e o tamanho do mesmo;
  - Fácil de manter com conhecimento do MAO.
- Desvantagens:
  - Esforço na construção;
  - Requer sistema para interpretar o modelo;
  - É mais difícil de entender;
  - Criação de linguagens específicas à domínios;
  - Difícil de manter sem conhecimento do MAO.

28

## Trabalhos relacionados

- Geradores de código (Code Generator)
- Programação Gerativa (Generative Programming)
- Meta-modelagem (Metamodeling)

29

## Conclusões

- MAO oferece uma alternativa à orientação a objetos usual;
- Projetos orientados a objetos atrelam entidades de negócio a classes e deixam pouca flexibilidade;
- MAO não atrela e por isso é mais flexível;
- MAO é útil em sistemas que enfatizam flexibilidade.

30

## Referências


- Joseph W. Yoder, Ralph Johnson, The Adaptative Object-Model Architetural Style, WICSA3, 2002.
- MetaData and Adaptive Object-Model Pages, http://www.adaptiveobjectmodel.com/


31