Chapter X

A Catalog of Design Rules for OO Micro-Architecture

Javier Garzás, Oficina de Cooperación Universitaria (OCU) S.A., Spain
Mario Piattini, University of Castilla - La Mancha, Spain

Abstract

This chapter presents a catalog of different rules for help to design object-oriented micro-architectures. These rules form an important part of the object-oriented design knowledge. Rules, like patterns, or accumulated knowledge, are discovered, not invented.

Introduction

This catalog unifies, completes, and formats under the term “rule”: principles, bad smells, best practices, and so on. These rules are applicable to object-oriented (OO) micro-architectural design, according to the OO design knowledge ontology presented in Chapter II.
For the description of a rule, the catalog takes as a base the sections which Gamma, Helm, Johnson & Vlissides (1995) use to describe a pattern, generalizing and detailing these sections (see Chapter II). The relationships of rules with patterns and refactorings come under the following categories:

- **Implies the use of [patterns]:** Patterns that are necessary in a design resulting from the application of a rule. These patterns solve design problems of the new micro-architecture.
- **Is introduced by [refactorings]:** Refactorings or operations, which place the rule on the micro-architecture.

On the other hand, another important issue is that each rule is identified by a meaningful name. We have been careful in the choice of these names, in order to help the designer to identify speedily where and when a rule is violated. So, we have avoided names which may be attention-catching but which are not very meaningful (as happens, for instance with bad smells (Fowler, Beck, Brant, Opdyke, & Roberts, 2000) with names such as Lazy Class or Shotgun Surgery. Therefore, the rules are named according to their antecedent.

### If There are Dependencies on Concrete Classes

#### Intent

Depends on interfaces or abstract classes rather than on concrete elements.

#### Also Known As

Dependency inversion principle (DIP) (Martin, 1996)
Program to an interface, not an implementation (Gamma, Helm, Johnson, & Vlissides, 1995).
40 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the product's webpage:

www.igi-global.com/chapter/catalog-design-rules-micro-architecture/27365?camid=4v1


Recommend this product to your librarian:

www.igi-global.com/e-resources/library-recommendation/?id=1

Related Content

Distributed Object Business Engineering: Digital Legos for the Enterprise
www.igi-global.com/chapter/distributed-object-business-engineering/27353?camid=4v1a

A Framework Based on Design Patterns: Implementing UML Association, Aggregation and Composition Relationships in the Context of Model-Driven Code Generation
www.igi-global.com/chapter/framework-based-design-patterns/27359?camid=4v1a

Information Systems Analysis with FOOM
www.igi-global.com/chapter/information-systems-analysis-foom/18738?camid=4v1a

Agile Outsourcing to India: Structure and Management
www.igi-global.com/chapter/agile-outsourcing-india/25634?camid=4v1a