Chapter VIII

Formalising Design Patterns as Model Transformations

Kevin Lano, King’s College, UK

Abstract

This chapter describes techniques for the verification of refactorings or transformations of UML models which introduce Design patterns. The techniques use a semantics of object-oriented systems defined by the object calculus (Fiadeiro & Maibaum, 1991; Lano, 1998), and the pattern transformations are proved to be refinements using this semantics.
Introduction

Design patterns are characteristic structures of classes or objects which can be reused to achieve particular design goals in an elegant manner. An example is the “State” Design pattern, which replaces local attributes of a class that record its state (in the sense of a finite state machine) by an object which provides polymorphic functionality in place of conditionals depending upon the state.

An application of a pattern should be a functionally-correct refinement step, that is, all the required properties of the original model should still remain valid in the restructured model. We will consider Design patterns applied to models in UML (OMG, 2005), with detailed behaviour of classes defined using VDM++ syntax (Durr & Dusink, 1993).

The semantics of object-oriented systems has been given in a logical axiomatic framework termed the Object Calculus (Fiadeiro & Maibaum, 1994). The chapter (Lano, 1998) gives a semantics for the UML and VDM++ notation which we use here. The application of a Design pattern will be shown to yield a theory interpretation of the semantics of the original model into the semantics of the new model, so ensuring that the functional properties of the original model still remain true in the new model.

Model specialisation using such theory interpretation is distinct from instantiation of a metamodel to a model. Metamodel instantiation is more powerful and flexible than model specialisation, in general. For example, in the case of many Design patterns, such as Template Method, the pattern is based on the existence of certain methods without any restriction on the input and output parameters of these methods. In a generic model we must be specific about such parameters. This means that the pattern is described by an unbounded collection of generic models instead of a single metamodel.

However, the power of metamodelling also has a cost, which is the requirement for developers to have more advanced UML skills and detailed knowledge of the UML metamodel. Therefore we prefer an approach using configurable generic models to define model transformations, domain models, and patterns.

The object calculus is a more appropriate formalism for reasoning about patterns than OCL, because the object calculus deals directly with actions and operations as first-class elements, and (in our extension) to timing properties and properties of system states at general time points.

The introduction of a Design pattern is one example of model transformation. Other kinds of transformation are quality improvement transformations (e.g., factoring out common elements of a number of classes into a superclass) and refinement to a platform specific model (e.g., replacing many-one associations by foreign keys). The transformation verification approach described here can also be applied to these other forms of transformations.

Section 1 introduces the object calculus, and the syntax of VDM++, which we will use to illustrate Design patterns. Section 4 addresses creational patterns, Section 5 addresses structural patterns, and Section 6 addresses Behavioral patterns. In each section we consider some typical patterns from Gamma, Helm, Johnson, and Vlissides (1994) and other sources.
Microscopic Biopsy Image Segmentation Using Hybrid Color K-Means Approach
www.igi-global.com/article/microscopic-biopsy-image-segmentation-using-hybrid-color-k-means-approach/177202?camid=4v1a

3D Face Modeling for Multi-Feature Extraction for Intelligent Systems
Zahid Riaz, Suat Gedikli, Michael Beetz and Bernd Radig (2011). *Computer Vision for Multimedia Applications: Methods and Solutions* (pp. 73-89).
www.igi-global.com/chapter/face-modeling-multi-feature-extraction/48314?camid=4v1a