Chapter III

A Generic Model of Object-Oriented Patterns Specified in RSL

Andrés Flores, University of Comahue, Argentina

Alejandra Cechich, University of Comahue, Argentina

Gabriela Aranda, University of Comahue, Argentina

Abstract

Object-oriented patterns are a promising technique for achieving widespread reuse of software architectures. They capture the static and dynamic structures of components and frameworks in successful solutions to problems for a wide range of domains. However, patterns are invariably described informally in the literature, which makes it difficult to give any meaningful certification of pattern-based software. The design process could be enhanced by means of an automatic support for modeling and verification with a proper formal foundation. In this chapter, we show how formal specifications of GoF patterns, based on the RAISE language, have been helpful in developing that tool support, where we have adopted the well-known Java language upon its portability facet. Thus, the object-oriented design process is extended by the inclusion of pattern-based modeling and verification steps, the latter involving checking design correctness and appropriate pattern application through the use of the supporting tool, called DePMoVe (design and pattern modeling and verification).
Introduction

Design patterns are descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context (Gamma, Helm, Johnson, & Vlissides, 1995). By their very definition, Design patterns result in reusable object-oriented design because they name, abstract, and identify key aspects of a common design structure. Design patterns are a promising technique for achieving widespread reuse of software architectures. They capture the static and dynamic structures and collaborations of components in successful solutions to problems that arise when building software in domains like business data processing, telecommunications, graphical user interfaces, data-bases, and distributed communication software. Patterns aid the development of reusable components and frameworks by expressing the structure and collaboration of participants in a software architecture at a level higher than source code or object-oriented design models that focus on individual objects and classes. Thus, patterns facilitate reuse of software architecture, even when other forms of reuse are not feasible.

However, patterns are invariably described informally in the literature, generally using natural language narrative together with a sort of graphical notation, which makes it difficult to give any meaningful certification of pattern-based software. Particularly, patterns in the GoF catalogue (Gamma et al., 1995) are described by using a consistent format which is based on an extension of the object modeling technique (OMT) (Rumbaugh, Blaha, Premerlani, Eddy, & Lorensen, 1991). This form of presentation gives a very good intuitive picture of the patterns, but it is not sufficiently precise to allow a designer to conclusively demonstrate that a particular problem matches a specific pattern or that a proposed solution is consistent with a particular pattern. Moreover, it is difficult to be certain that patterns themselves are meaningful and contain no inconsistencies. In some cases, descriptions of patterns are intentionally left loose and incomplete to ensure that they are applicable in a range as wide as possible. This reduces understanding and interpretation upon appropriate patterns usage. Nevertheless, the availability of a more formal description could help alleviate these problems.

Formal methods are gaining prominence in software engineering as a way to insure that a specification is consistent with its intended meaning, and that two formally-rendered artifacts (e.g., a specification and an implementation) are consistent with each other in some precise way. Formal methods in the arena of software architecture tend to manifest themselves in representation technology. In sum, formal methods are useful to help a human organize thought patterns into a more disciplined form, thus heading off conceptual errors.

In a previous work (Cechich & Moore, 1999a, 1999b; Moore, Cechich, Reynoso, Flores, & Aranda, 2002), we have presented a formal model of a generic object-oriented design that was developed in RSL (the RAISE specification language) (RAISE Group, 1992, 1995), and based upon the extended OMT notation given on the GoF catalogue. We have shown how designs can be formally linked with patterns in this model, and how properties of individual patterns can be specified in the model, thus giving a basis for formally checking whether a given design and a given pattern are consistent with each other. Although we have mainly focused our attention to GoF patterns, the whole model is general enough to be applied in a similar way to formalize other patterns based on object-oriented notations,
Related Content

Leveraging Models of Human Reasoning to Identify EEG Electrodes in Images With Neural Networks
Optoelectronics in Machine Vision-Based Theories and Applications (pp. 106-133).

Edge Detection by Maximum Entropy: Application to Omnidirectional and Perspective Images
www.igi-global.com/article/edge-detection-maximum-entropy/59875?camid=4v1a
Radial Moments for Image Retrieval
www.igi-global.com/chapter/radial-moments-for-image-retrieval/197060?camid=4v1a

Moving Object Classification Under Illumination Changes Using Binary Descriptors
www.igi-global.com/chapter/moving-object-classification-under-illumination-changes-using-binary-descriptors/209833?camid=4v1a