Chapter II

A Visual Language for Design Pattern Modeling and Instantiation

David Maplesden, Orion Systems Ltd., New Zealand
John Hosking, University of Auckland, New Zealand
John Grundy, University of Auckland, New Zealand

Abstract

In this chapter we describe the Design pattern modeling language, a notation supporting the specification of Design pattern solutions and their instantiation into UML design models. DPML uses a simple set of visual abstractions and readily lends itself to tool support. DPML Design pattern solution specifications are used to construct visual, formal specifications of Design patterns. DPML instantiation diagrams are used to link a Design pattern solution specification to instances of a UML model, indicating the roles played by different UML elements in the generic Design pattern solution. A prototype tool is described, together with an evaluation of the language and tool.
Introduction

Design patterns are a method of encapsulating the knowledge of experienced software designers in a human-readable and understandable form. They provide an effective means for describing key aspects of a successful solution to a design problem and the benefits and tradeoffs related to using that solution. Using Design patterns help produce good design, which helps produce good software (Gamma, Helm, Johnston, & Vlissides, 1994).

Design patterns to date have mostly been described using a combination of natural language and UML-style diagrams or complex mathematical or logic based formalisms, which the average programmer finds difficult to understand. This leads to complications in incorporating Design patterns effectively into the design of new software. To encourage the use of Design patterns, we have been developing tool support for incorporating Design patterns into program design. We describe the Design pattern modeling language (DPML), a visual language for modeling Design pattern solutions and their instantiations in object-oriented designs of software systems. We have developed two prototype tools, DPTool and MarmaDPTool, realising DPML and integrating it within the Eclipse environment. Significant contributions of this work include the introduction of dimensions as a proxy for collections of like Design pattern participants and the instantiation of patterns into designs rather than directly into code. These both fit naturally with model driven design approaches.

We begin by describing previous work in Design pattern tool support. We then overview DPML and describe its use in modeling Design pattern solutions and pattern instantiation. We discuss two prototype tools we have developed to support the use of DPML, together with an evaluation of their usability. We then discuss in more detail the rationale and implications of the design choices we have made in designing DPML and the potential for more general applicability of some of those design features, before summarising our contributions.

Previous Work

Design patterns, which describe a common design solution to a programming problem, were popularised by the seminal “Gang of Four” book (Gamma et al., 1994) and Coplien’s software patterns (Coplien, 1996). Design patterns have become very widely used in object-oriented software development, and their influence has spread to areas of software development other than design, such as the development of analysis patterns (patterns in the analysis phase) and idioms (language specific programming patterns). Design patterns are typically described using a combination of natural language, UML diagrams, and program code (Gamma et al., 1994; Grand, 1998). However, such descriptions lack Design pattern-specific visual formalisms, leading to pattern descriptions that are hard to understand and hard to incorporate into tool support. The UML standard for modeling Design patterns relies upon UML profiles and the UML metamodel (Object Management Group, 2006). This presents difficulties for modeling Design patterns, particularly because they are constructed using similar concepts to object models, and hence are simply prototypical examples of that object model. This does not allow enough freedom to model patterns effectively. Design pattern representations look like existing UML models and linking pattern elements to standard UML elements.
Moving Vehicles Detection in Traffic Video Using Modified SXCS-LBP Texture Descriptor
www.igi-global.com/article/moving-vehicles-detection-in-traffic-video-using-modified-sxcs-lbp-texture-descriptor/159805?camid=4v1a

Categorization of Plant and Insect Species via Shape Analysis
www.igi-global.com/chapter/categorization-of-plant-and-insect-species-via-shape-analysis/139593?camid=4v1a