Design Patterns Formal Composition and Analysis

Halima Douibi, University Abdel Hamid Mehri, Constantine, Algeria
Faiza Belala, University Abdel Hamid Mehri, Constantine, Algeria

ABSTRACT
Informal description of design patterns was adopted to facilitate their understanding by software developers. However, these descriptions lead to ambiguities limiting their successful use in real applications. Recently, several formal methods have been employed to guarantee precise specification of design pattern elements without insisting on their behavior. The main goal of this article is to propose a rewriting logic-based framework for enhancing formal description and reasoning on design patterns composition. The proposed model is integrated and executed under Maude using K-Maude tool. It allows first to associate a meta model notation, according to MDA approach, to all design patterns elements and then, a Maude semantic basis for specifying both their structural and dynamic aspects, is given. In particular, it will be shown how the authors’ combined modeling approach is exploited for defining a set of operations on patterns, and how it will be used to formal analyze patterns composition.

KEYWORDS
Design Patterns, Formal Methods, K-Framework, Maude, Meta-Model, Pattern Composition

1. INTRODUCTION
In Gamma catalog (Gamma et al., 1995) Design Patterns are described in a consistent and structured way. A pattern is presented by several properties whose name, purpose, an example of a design problem that it solves and a structure of the proposed solution in the form of class diagrams. Although these Design Patterns are well documented, the decision to apply them is manual and remains largely dependent on the user. In fact, Design Patterns are elements of Reusable Object-Oriented Software, described using a combination of natural language, UML diagrams, and program code. Formal specification of Design Patterns can help to use them more effectively and correctly in software development. It can eliminate ambiguity by clarifying the different pattern’s concepts, at the same time this formal specification provides a solid foundation for reasoning about their composition and inter-relationships.

Several attempts have been made to deal with patterns formalization problem; they try to specify patterns in terms of mathematical structures and then, to check formally their accuracy (Blewitt et al., 2001; Gasparis, 2007; Herranz et al., 2002; Taibi & Ngo, 2003). Recently various formal methods (e.g.

DOI: 10.4018/IJITSA.2019070101

Copyright © 2019, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
event sets, process algebras, First Order Logic, Temporal Logic of Actions), each with a particular semantics, have been used in this context. However, existing works do not pay more attention to model Design Patterns elements in order to ensure specific properties of the produced software or to increase the correctness of patterns composition. The adopted formalisms lack appropriate constructs and natural semantics to deal with specification and analysis of dynamic operations on Design Patterns.

Rewriting logic, introduced initially by Jose Meseguer (Meseguer, 1992), has been shown as one of the most appropriate formalisms to describe concurrent and non-deterministic systems. This logic unifies several formal models (Process Algebras, Petri Nets Labeled Transition Systems, ECATNets, Event Structures) (Bettaz & Maouche, 1993; Bouanaka, 1998; Marti-Oliet & Meseguer, 1996). Besides, it has several tools and operational environments; this encourages its use for systems prototyping and analysis.

K-Maude tool (Serbanuta & Rusu., 2010) is one of the most recent interface implementation, which represents a semantic framework allowing the definition of programming languages, calculi, as well as type systems or formal analysis tools (Rusu et al., 2016). In addition, it may use all Maude system surrounded verification and checking tools, as the LTL model-checker (Clavel et al., 2007b).

Another interesting feature of K-Maude tool is that thanks to its programming language description and execution capabilities, one may define its own language, enough expressive and well understood by designers. Our contribution in this work is to show how this language is extended to improve specification and analysis of Design Patterns relationships and particularly their composition. Thus, an incremental approach to formalize Design Patterns is given. This approach consists of four main phases:

1. Providing a more abstract and generic notation of any pattern: DP-MM (Design Pattern Meta Model), starting from the most known patterns of GOF (Gamma et al., 1995), according to MDA technique;
2. Defining a pattern as an instance of the proposed (DP-MM) Meta model;
3. Integrating this pattern description model in Maude (Clavel et al., 2007a) in a transparent manner, allowing a non-familiar user to manipulate our Maude based executable specifications;
4. Extending both previous models to deal with dynamic operations on Design Patterns (as composition, restriction, etc.).

This paper is organized as follows. Section 2 describes briefly the fundamental concepts of K framework and its Maude interface. Our incremental approach for formalizing structural and behavioral aspects of Design Patterns is detailed in section 3, while a more attention is given to formalize some operations on Design Patterns. Section 4 shows how our formalization approach is applied to compose Design Patterns and to analyze them through an illustrative case study. Section 5 outlines the relevant related work and discusses our contribution. Finally, in Section 6 we conclude and present our future work.

2. K-MAUDE TOOL PRESENTATION

K is an executable semantic framework (K semantic framework, 2015), based on rewriting logic used to specify and analyze programming languages. It consists of three components: a concurrent rewrite abstract machine, a language definitional technique and a specialized notation (Serbanuta & Rusu, 2010; Rusu et al., 2016). K-Maude is a tool implementing K on top of Maude language (Clavel et al, 2007a). It provides an interface accepting K modules and a collection of tools for transforming K language definitions into Maude rewrite theories for execution or analysis or into LATEX for documentation purposes. The current K-Maude prototype is successfully used to define the semantics of several languages. It translates K-definitions (K modules), using several complex algorithms,
Related Content

Globalization of Instruction: Developing Intellectual Capital
www.igi-global.com/chapter/globalization-instruction-developing-intellectual-capital/44316?camid=4v1a

From the Garden of Eden to the Knowledge Society
www.igi-global.com/chapter/garden-eden-knowledge-society/24879?camid=4v1a

E-Business Knowledge
www.igi-global.com/chapter/business-knowledge/25045?camid=4v1a
Towards a Conceptual Framework for Open Systems Developments
www.igi-global.com/article/towards-a-conceptual-framework-for-open-systems-developments/109089?camid=4v1a