A Multiple Phases Approach for Design Patterns Recovery Based on Structural and Method Signature Features

Mohammed Ghazi Al-Obeidallah, Department of Computing, University of Brighton, Brighton, UK
Miltos Petridis, Department of Computing, Middlesex University, London, UK
Stelios Kapetanakis, Department of Computing, University of Brighton, Brighton, UK

ABSTRACT

Design patterns describe both structure, behavior of classes and their relationships. They can improve software documentation, speed up the development process and enable large-scale reuse of software architectures. This article presents a multiple levels detection approach (MLDA) to recover design pattern instances from Java source code. MLDA is able to recover design pattern instances based on a generated class level representation of a subject system. Specifically, MLDA presents what is the so-called Structural Search Model (SSM) which incrementally builds the structure of each design pattern based on the generated source code model. Moreover, MLDA uses a rule-based approach to match the method signatures of the candidate design instances to that of the subject system. As the experiment results illustrate, MLDA is able to recover 23 design patterns with reasonable detection accuracy.

KEYWORDS
Design Patterns, Detection, Gang of Four, Reverse Engineering, Rule-Based Systems, Static Analysis

1. INTRODUCTION

The detection of design patterns is a reverse engineering activity where design patterns are recovered depending on certain criteria. The idea of patterns was adopted by the so-called Gang of Four (Gamma, Helm, Johnson, & Vlissides, 1995) -henceforth GoF.

GoF have cataloged 23 design patterns. Each design pattern describes a problem that occurs over and over again, in an attempt to describe the core solution to that problem. This solution can be reused a million times over, without doing it the same way twice. In fact, design patterns vary in their levels of abstraction. Each design pattern solves a specific design problem by connecting a number of classes (participant classes) together using different relationships. According to the GoF’s catalog, each design pattern involves both structural and behavioral aspects. Structural aspects describe the static arrangement of classes and their relationships. On the other hand, behavioral aspects describe dynamic interactions between the participant classes.

DOI: 10.4018/IJSI.2018070103

Copyright © 2018, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
Design patterns at the source code level reflect the earliest set of design decisions that have been taken by the development team. In addition, the majority of current software systems involve instances of design patterns in their source codes. Consequently, the recovery of design patterns helps a number of stakeholders, such as system analysts, software engineers and architects to capture design and code information and enhance their understanding over an enterprise system. However, the recovery of design patterns is not an easy task since software documentation is not always available and the possible variants of pattern instances.

This paper presents a new detection approach named Multiple Levels Detection Approach (MLDA). MLDA provides a Structural Search Model (SSM) that is able to recover the instances of design patterns based on the generated class level representation of the Java source code.

Furthermore, MLDA builds the structure of each design pattern incrementally based on the generated source code model. To reduce the number of false positive instances, a rule-based system that is able to match the method signatures of the candidate design instances to that of the subject system has been developed. Since GoF design patterns have been attractive both to industry and academia, this paper focuses on the GoF design patterns.

This paper is organized as follows: Section two presents MLDA architecture, the structural search model and the rule-based system. Experiments and results, related work and threats to validity are discussed in Sections three, four, and five respectively. Finally, the conclusion is presented in Section six.

2. RECOVERING DESIGN PATTERN INSTANCES

The Multiple Levels Detection Approach (MLDA) is a research prototype, which has been developed to recover the instances of design patterns from Java source code. MLDA involves three main levels: a parsing level, a searching level and method signatures matching level. The architecture of MLDA appears in Figure 1.

The parsing level aims to recover the source code information and produce a source code model. Moreover, MLDA aims to recover the five key relationships, which may occur between classes and objects inside any object-oriented program. These relationships are Inheritance, Aggregation, Association, Dependency and Realization. On the other hand, the searching level of MLDA aims to examine the source code model that has been developed during the parsing level and tries to match it with the GoF’s catalog.

Specifically, MLDA introduces a Structural Search Model (SSM), which involves a searching algorithm for each design pattern. MLDA works on the principle of building the patterns incrementally based on the connecting relationships. The third level of MLDA is the method signatures matching level. The method signatures of the subject system are represented as a set of facts. On the other hand, the required method signatures of the candidate design instances are represented as a set of

---

**Figure 1. The architecture of MLDA**

[Diagram showing the architecture of MLDA]
Related Content

A Decision Making Method Based on Society of Mind Theory in Multi-Player Imperfect Information Games

ODACE SLA: Ontology Driven Approach for Automatic Establishment of Service Level Agreements
[www.igi-global.com/chapter/odace-sla-ontology-driven-approach/66798?camid=4v1a](www.igi-global.com/chapter/odace-sla-ontology-driven-approach/66798?camid=4v1a)
Software Metrics and Measurements
[www.igi-global.com/chapter/software-metrics-measurements/29388?camid=4v1a](www.igi-global.com/chapter/software-metrics-measurements/29388?camid=4v1a)

[www.igi-global.com/article/service-oriented-software-engineering/126635?camid=4v1a](www.igi-global.com/article/service-oriented-software-engineering/126635?camid=4v1a)