Chapter 8.16
Model-Driven Software Refactoring

Tom Mens
University of Mons-Hainaut, Belgium

Gabriele Taentzer
Philipps-Universität Marburg, Germany

Dirk Müller
Chemnitz University of Technology, Germany

ABSTRACT

In this chapter, we explore the emerging research domain of model-driven software refactoring. Program refactoring is a proven technique that aims at improving the quality of source code. Applying refactoring in a model-driven software engineering context raises many new challenges such as how to define, detect and improve model quality, how to preserve model behavior, and so on. Based on a concrete case study with a state-of-the-art model-driven software development tool, AndroMDA, we explore some of these challenges in more detail. We propose to resolve some of the encountered problems by relying on well-understood techniques of meta-modeling, model transformation and graph transformation.

INTRODUCTION

In the current research and practice on software engineering, there are two very important lines of research for which tool support is becoming widely available. The first line of research is *program refactoring*, the second one is *model-driven software engineering*. To this date, however, the links and potential synergies between these two lines of research have not been sufficiently explored. This will be the main contribution of this chapter.

Model-Driven Software Engineering

In the realm of software engineering, we are witnessing an increasing momentum towards the

Copyright © 2009, IGI Global, distributing in print or electronic forms without written permission of IGI Global is prohibited.

use of models for developing software systems. This trend commonly referred to as model-driven software engineering, emphasizes on models as the primary artifacts in all phases of software development, from requirements analysis over system design to implementation, deployment, verification and validation. This uniform use of models promises to cope with the intrinsic complexity of software-intensive systems by raising the level of abstraction, and by hiding the *accidental complexity* of the underlying technology as much as possible (Brooks, 1995). The use of models thus opens up new possibilities for creating, analyzing, manipulating and formally reasoning about systems at a high level of abstraction.

To reap all the benefits of model-driven engineering, it is essential to install a sophisticated mechanism of *model transformation*, that enables a wide range of different automated activities such as translation of models (expressed in different modeling languages), generating code from models, model refinement, model synthesis or model extraction, model restructuring etc. To achieve this, languages, formalisms, techniques and tools that support model transformation are needed. More importantly, their impact on the quality and semantics of models needs to be better understood.

Program Refactoring

Refactoring is a well-known technique to improve the quality of software. Martin Fowler (1999) defines it as "A change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior".

The research topic of refactoring has been studied extensively at the level of programs (i.e., source code). As a result, all major integrated software development environments provide some kind of automated support for program refactoring.

As a simple example of a program refactoring, consider the refactoring *Extract Method*, one of the more than 60 refactorings proposed by Fowler. Essentially, it is applied to a method in which part of the method body needs to be extracted into a new method that will be called by the original one. The situation before this program refactoring on a piece of Java source code is shown in Figure

Figure 1. Java source code example before applying the Extract Method program refactoring (©2007 Tom Mens, UMH. Used with permission)

```
    protected LectureVO[] handleFindLecture
      (java.lang.String title, domain.Weekday day, domain.Time time)
      throws java.lang.Exception
  *  { SearchCriteria c = new SearchCriteria();
  *    c.setDay(day);
  *    c.setTitle(title);
  *    c.setTime(time);
      Collection coll =
            getLectureDao().findLecture(LectureDao.TRANSFORM_
LECTUREVO,c);
      LectureVO[] lectures = new LectureVO[coll.size()];
```

32 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/model-driven-software-refactoring/29571

Related Content

A Case Study on the Selection and Evaluation of Software for an Internet Organisation
Pieter van Staaden (2009). *Software Applications: Concepts, Methodologies, Tools, and Applications (pp. 2137-2152).*
www.irma-international.org/chapter/case-study-selection-evaluation-software/29499

Knowledge Management Software
Rodrigo Baroni de Carvalhoand Marta Araújo Tavares Ferreira (2009). *Software Applications: Concepts, Methodologies, Tools, and Applications (pp. 438-449).*
www.irma-international.org/chapter/knowledge-management-software/29401

Information Needs of Vocational Training From Training Providers' Perspectives
Agnes Wai Yan Chan, Dickson K.W. Chiu, Kevin K.W. Hoand Minhong Wang (2018). *International Journal of Systems and Service-Oriented Engineering (pp. 26-42).*
www.irma-international.org/article/information-needs-of-vocational-training-from-training-providers-perspectives/231506

Developing a Blockchain Solution for West Virginia Medicinal Cannabis
Ludwig Christian Schaupp (2019). *International Journal of Systems and Service-Oriented Engineering (pp. 1-11).*
www.irma-international.org/article/developing-a-blockchain-solution-for-west-virginia-medicinal-cannabis/256133

Estimating Interval of the Number of Errors for Embedded Software Development Projects
Kazunori Iwata, Toyoshiro Nakasima, Yoshiyuki Ananand Naohiro Ishii (2014). *International Journal of Software Innovation (pp. 40-50).*
www.irma-international.org/article/estimating-interval-of-the-number-of-errors-for-embedded-software-development-projects/120089