Design Patterns as Laws of Quality

Yann-Gaël Guéhéneuc, University of Montreal, Canada

Jean-Yves Guyomarc’h, University of Montreal, Canada

Khashayar Khosravi, University of Montreal, Canada

Hourari Sahraoui, University of Montreal, Canada

Abstract

Software quality models link internal attributes of programs with external quality characteristics. They help in understanding relationships among internal attributes and between internal attributes and quality characteristics. Object-oriented software quality models usually use metrics on classes (such as number of methods) or on relationships between classes (for example coupling) to measure internal attributes of programs. However, the quality of object-oriented programs does not depend on classes solely: it depends on the organisation of classes also. We propose an approach to build quality models using patterns to consider program architectures. We justify the use of patterns to build quality models, describe the advantages
and limitations of such an approach, and introduce a first case study in building and in applying a quality model using design patterns on the JHotDraw, JUnit, and Lexi programs. We conclude on the advantages of using patterns to build software quality models and on the difficulty of doing so.

Introduction

This chapter is a complete coverage of our current work on software quality models and on design pattern identification. In this chapter, we explore the idea of facts in science in relation with software quality models. We show how design patterns can be used as facts to devise a quality model and we describe the processes of building and of applying such a quality model.

In science, facts are the subject of observations by the scientists, who hypothesize laws to formalize recurring observations and theories to frame and to explain the laws. To the best of the authors’ knowledge, many facts have been recorded and published but only few laws and theories have been developed (Endres & Rombach, 2003). This lack of laws and theories impedes the successful development of software and reduces our trust in software and in software science.

The lack of laws and theories is particularly visible in software quality. There do not yet exist general software quality models that could be applied to any software. It is indeed difficult to build quality models without concrete laws on software and software quality. Thus, existing quality models attempt to link internal attributes of classes and external quality characteristics with little regard for actual facts on software quality and without taking into account some dimensions of the evaluated software, such as its architecture.

In the following, we use design patterns (Gamma, Helm, Johnson, & Vlissides, 1994) as general laws to build a software quality model. We choose design patterns because they are now well-known constructs and have been studied extensively. Design patterns provide “good” solutions to recurring design problems. They define a problem in the form of an intent and motivations and provide a solution in the form of a design motif, a prototypical micro-architecture that developers use in their design to solve the problem. Design patterns are said to promote software elegance through flexibility, reusability, and understandability (Gamma et al., 1994).

We assume that the design motifs provided by design patterns show flexibility, reusability, and understandability. Also, we assume that we can use design motifs as laws on software quality, as their authors intended, if not explicitly: