Chapter 7.1
A Survey of Object-Oriented Design Quality Improvement

Juan José Olmedilla
Almira Lab, Spain

ABSTRACT

The use of object-oriented (OO) architecture knowledge such as patterns, heuristics, principles, refactorings and bad smells improve the quality of designs, as Garzás and Piattini (2005) state in their study; according to it, the application of those elements impact on the quality of an OO design and can serve as basis to establish some kind of software design improvement (SDI) method. But how can we measure the level of improvement? Is there a set of accepted internal attributes to measure the quality of a design? Furthermore, if such a set exists will it be possible to use a measurement model to guide the SDI in the same way software processimprovement models (Humphrey, 1989; Paulk, Curtis, Chrissis, & Weber, 1993) are guided by process metrics (Fenton & Pfleeger, 1998)? Since (Chidamber & Kemerer, 1991) several OO metrics suites have been proposed to measure OO properties, such as encapsulation, cohesion, coupling and abstraction, both in designs and in code, in this chapter we review the literature to find out to which high level quality properties are mapped and if an OO design evaluation model has been formally proposed or even is possible.

INTRODUCTION

In the last two decades there has been a growing interest and effort put after the idea of improving the quality of the software processes (Humphrey, 1989). This increasing trend had it origin in the application of statistical process control techniques (Oakland, 1990) from the manufacturing industry to our sector, thus creating a new discipline that has been called software process improvement (SPI) (Humphrey, Snyder & Willis, 1991). This discipline aids organisations to improve their software producing processes by, firstly, identifying all the broad areas of the process, their goals

and the activities and sub-activities needed to achieve them and secondly by establishing a path through which the process can be incrementally improved, this path is a set of quality levels, each of them defined by the areas and their associated goals to be accomplished. Fundamental to the SPI are the associated metrics (Fenton & Pfleeger, 1998) that are the tool by which the organisation can tell at each moment where it is in the path, each of the aforementioned goals has an associated set of metrics that help to tell if it has been achieved and to what extent. Although there are alternative SPI models and methods, like CMMI (Paulk et al., 1993) or SPICE (ISO/IEC, 1999), an organisation can always adhere to a concrete definition of process quality and a way to measure it and improve it.

However the product arena does not seem to be so established in terms of quality improvement models. A fundamental question that managers and developers often face is when it is worth to improve a software product by reengineering it or on the contrary start it all over from scratch. Fowler (2000) states that

There are times when the existing code is such a mess that although you could refactor it, it would be easier to start from the beginning ... I admit that I don't really have good guidelines for it. (p. 66)

In this case Fowler was talking about the refactoring technique but something similar can be said about other OO design knowledge elements. There is plenty of knowledge, more or less formalised, about identifying situations in which to apply an specific design improvement (Brown, Malveau, Brown, McCormick & Mowbray, 1998; Fowler, 2000; Gamma, Helm, Johnson & Vlissides, 1995; Riel, 1996), but is there a formal method to know which design transformations should be applied first or are more important? Is it possible to establish which design transformations, pattern applications, refactorings, and so forth, are more important in a certain quality level?

An organisation or a project could be interested in attaining only a moderate quality level that is acceptable for the time being and it is foreseeable that will consume only a limited amount of resources. Such an organisation could be interested in a guide that tells what quality indicators are really crucial to that quality level, to what extent, in terms of a measurable quantity, how to measure them and which design transformation affect the properties object of the measurements.

First of all it would be necessary to define what is design quality, by identifying what general properties or high level indicators comprise it; second, to organise those indicators in sets that constitute an incremental ladder of quality, so that depending on the situations, the (non-functional requirements and the resources a designer can choose the target level for his or her design; thirdly, choose the metrics that help in the assessment of the goals accomplishment; and finally, define the OO knowledge elements that apply in each case. We are talking here about something that we could call an *OO design maturity model* that would help designers to assess and improve a design before having to implement it.

Product quality has been defined in ISO 9126 (ISO/IEC, 2001) by external and internal attributes that describe the quality of the final software product and its intermediate subproducts, such as design; according to that, design quality should be measured through internal attributes that will predict, somehow, the final outcome of the external attributes. Some authors (Bansiya & Davis, 2002; Basili, Briand & Melo, 1996) have proposed numerical relations between some internal quality attributes and general OO properties, such as coupling or cohesion, for which there are already defined metrics. Other authors (Miller, Hsia & Kung) have proposed directly to measure the levels of accomplishment of certain OO knowledge elements like design principles,