Chapter 4
Synthesis of Object-Oriented Software Structural Models Using Quality Metrics and Co-Evolutionary Genetic Algorithms

André Vargas Abs da Cruz
Pontifícia Universidade Católica do Rio de Janeiro, Brazil

Dilza Szwarcman
Pontifícia Universidade Católica do Rio de Janeiro, Brazil

Thiago S. M. Guimarães
Pontifícia Universidade Católica do Rio de Janeiro, Brazil

Marco Aurélio C. Pacheco
Pontifícia Universidade Católica do Rio de Janeiro, Brazil

ABSTRACT

One of the biggest challenges for the developer of object-oriented software is the modeling and developing of the objects themselves, so that they are easily reusable in complex systems. The final quality of the software depends mostly on the quality of the modeling developed for it. Modeling and specification of software are fundamental steps for making the software development an activity of engineering. Design is the activity in which software behavior and structure are elaborated. During this phase many models are developed anticipating several views of the final product and making software evaluation possible even before the software is implemented. Consequently, the synthesis of a software model can be seen as a problem of optimization, where the attempt to find a better configuration among the elements chosen through the object-oriented paradigm, such as classes, methods and attributes that meet quality design criteria. This work studies a possibility to synthesize higher quality modelings through the evolution of Genetic Algorithms, a technique that has proved to be efficient in dealing with problems involving large

DOI: 10.4018/978-1-61520-809-8.ch004
search spaces. The work was divided in three main stages: a study of object-oriented software engineering; the definition of a model using genetic algorithms; and co-evolution of species for the synthesis of object-oriented software modeling, aiming at quality improvement; and at the implementation of a model for case study. The study of object-oriented software engineering involved the establishment of software development phases and the characterization of the representation used in modeling phase and, in particular, the characterization of class diagrams based on UML. The study also investigated software quality metrics such as Reutilization, Flexibility, Understandability, Functionality, Extensibility and Effectiveness. The specification of genetic algorithm consisted in the definition of the structure of the chromosome that could provide a good representation of modeling diagram and a function of evaluation of the design that could take the software quality metrics in to consideration. As a result, the chromosomes represent metadata of a simplified UML diagram of classes, which may later be used as an entry of a CASE (Computer Aided Software Engineering) Tool that can create the implementation code in the chosen pattern. The evaluation function was defined focusing at the synthesis of a higher quality object-oriented software modeling. In order to observe the use of more than one objective at the same time the Pareto technique for multi objective problems was used. The evolution is directed towards the improvement of quality metrics by searching for a qualitatively better modeling, based on Bansiya’s (Bansiya and Davis, 2002) study. The construction of a co-evolutionary model consisted in defining distinct species so that each one would represent part of the problem to be evolved, thus enabling a more efficient representation of the model. The co-evolutionary model allowed the evolution of more complex structures, which would not be possible in a simple Genetic Algorithm. The chromosomes related to each species codify metadata and that is why the solution assembly (design) makes use of a decoder. A case study was done to synthesize the modeling of an elevator controller. The results achieved in this section were compared to the modelings produced by specialists, and the characteristics of these results were analyzed. The GA performance in the optimization process was compared to that of a random search and, in every case, the results achieved by the model were always better.

1. INTRODUCTION

1.1. Motivation

The concern and the efforts to improve the practices of software development, looking for productivity, quality growth and lower costs, bring to evidence new perspectives for software development.

The difficulty and delay in implementing a complex system can be reduced if the previously developed and tested components can be used. However, a big initial effort to accomplish this goal is necessary for a software development company to create a repository of reusable software components.

Nowadays, the object-oriented programming paradigm is widely used and, as a consequence, huge effort has been made to reuse software. A new technique has been developed by Dandashi (Dandashi, 1998) to infer the re-usability of a software component by means of measurements taken directly from the implementation.

The object-orientation paradigm has changed the elements that are used to infer the quality of the software. Traditional metrics of software products such as size, complexity, performance and quality had to be changed to include new concepts such as encapsulation, inheritance, and polymorphism, which are innate to object orientation. This way, new metrics have been defined (Chidamber & Kemerer, 1994; Hitz & Montazeri, 1996; Li &