Revealing the Structure of JAVA Classes with Concept Lattices

M.Sc. Research by Uri Dekel, Department of Computer Science, the Technion

Why do programmers "reinvent the wheel" instead of reusing existing classes?

One reason for avoiding reuse is the effort involved in learning to use a third-party class. The interface of such a class can be very large, often spanning dozens or even hundreds of methods. This makes it difficult to intuitively locate a method that provides a specific functionality, especially if the original developer used a different vocabulary. This complexity also introduces the risk of errors and inconsistencies in the interface and the implementation of the class.

Most tools display methods and fields in a simple alphabetical list.

The premise of our work is that the usage patterns of methods by field are fundamental to understanding the behavior of the class, and that methods that use the same fields are related semantically. Hence, our work suggests using the mathematical classification technique of formal concept analysis (FCA) to semantically partition the methods of the class into groups of methods with related functionality, and to display them hierarchically in a concept lattice. On the right side of the figure below, each concept (a node of the lattice) represents a set of methods which use the same combination of fields. This combination is the union of the sets of fields of that concept and of every concept below it.

In order to examine the class at different levels of abstraction, we suggest several techniques to abstract the class information ("zoom-out") and to explore it in more detail ("zoom-in"). Both techniques focus on adding/removing fields (attributes) and straightforward. Some of the more interesting techniques are demonstrated in the figure below.

Our research heuristically groups related methods based on their use of fields, and displays them in a concept lattice.

Lists of Methods and Fields (Sorted alphabetically as in JavaDoc)

Concept Lattice
(Methods are clustered in concepts according to their use of fields)

Zoom-In and Zoom-Out tools demonstrated on class Pnt3D

The concept lattice and the zooming techniques are the basis for a three-stage methodology for analyzing classes with a purpose of understanding, quality assurance, or reverse engineering. In the first stage, the interface of the class is studied using a lattice that includes only interface methods. The user repeatedly zooms out to obtain an understanding of the general structure of the class, and then zooms in to explore specific functionalities. In the second stage, implementation is examined and the lattice is used to understand its implementation and discover flaws. In the third stage, the lattice is used to select an effective order for a class-wide code inspection.

Our methodology helped discover errors in actual third-party classes!

Our current directions of research include applications for version comparison, class design, and class metrics.