Thanks to Andrew P. Black, Candy Yiu, and the National Science Foundation for partially funding this research under grant number CCF-0520346.

Hayashi and Colleagues
This tool underlines copy and paste duplication to indicate that it should be removed.

Bisanz
This tool displays smells by underlining smell occurrences in an Eclipse editor.

Fokaefs and Colleagues
This tool highlights the feature envy smell in source code with editor annotations.

Simon and Colleagues
This tool shows program entities as glyphs in a 3D visualization; distances between glyphs represent coupling between entities.

van Emden and Moonen
This tool shows a visualization of smells over the entire codebase as a colored graph, where each node represents a program entity.

Parnin and Görg
This tool provides a family of visualizations to indicate the presence and extent of code smells using a variety of small visualizations.

UnScalability
Small detectors should scale well as the number and kinds of smells increase. With more than 20 smells described by Martin Fowler and others, including comments, feature envy, and large methods, nearly every piece of code in your editor could smell. Thus, underlining everywhere a smell existed could quickly overwhelm the programmer.

InExpressivity
It’s not enough to say where a smell comes from. For many smells, such as feature envy, an explanation of why the smell exists may help programmers understand and correct it.

Higher Relevance
This tool achieves relevance by displaying smells only related to the method the programmer is currently viewing or editing.

Better Scalability
This tool achieves scalability by summarizing smells in a single ambient visualization behind the editor.

More Expressivity
This tool achieves expressivity by showing smell details only when the programmer wants more information.

IrRelevance
Smells may exist anywhere in code. A smell detector that draws the programmer away from where she is currently working is a distraction. While visualizations may be useful for high-level code inspections, low-level coding may make best use of a smell detector if the most relevant smells are presented first.

Petal diameters change as the programmer scrolls or moves the cursor in the editor. Mousing over a member emphasizes that member in the source code with a black outline. Clicking on the [+] next to a smell name activates the detail view. In the feature envy detail view, each referenced class is assigned a distinct color. Mousing over a member emphasis that member in the source code with a black outline.

The visualization is visible at all times. Petal placement and color roughly indicates obviousness. Each petal represents an individual smell. Petal radius represents “how bad” a smell is. The visualization is visible at all times. Petal placement and color roughly indicates obviousness. Each petal represents an individual smell. Petal radius represents “how bad” a smell is.