Editor: Diomidis Spinellis ■ Athens University of Economics and Business ■ dds@aueb.gr

Dear Editor

Diomidis Spinellis

Machines should work. People should think. —Richard Hamming

Dear Editor,

I know that you're nowadays often taken for granted and that some programmers consider you a relic. Yet many programmers continue to spend an inordinate amount of time with you and often listen to your advice. As you've no doubt observed, we programs are often mistreated; in this letter, I've written my most common grievances hoping you can convince programmers to treat us better.

In the past, I know that you had a more important position. Developers would fight to determine which of your two cousins, vi or Emacs, was most versatile. Creating an editor was no mean feat. The (now famous) programmers who brought your cousins to life, Bill Joy and Richard Stallman, had to overcome limitations such as slow terminal lines and CPUs, a small address space, and the idiosyncrasies of the numerous mutually incompatible terminals. Today, the pervasiveness of GUI libraries, fast processors, and abundant memory space make developing an editor a weekend project.

I believe you're not less important than you were 20 years ago. One of the best things you can do for programmers (and, incidentally, us programs) is to convince them to use their heads—taking advantage of your advanced facilities—instead of their fingers. First of all, this switch will reduce their risk of suffering from repetitive stress injury: If they can accomplish the effect of 100 keystrokes by giving you a 20-character search and replace command, they've saved their fingers from the impact force of 80 keystrokes.

Furthermore, by devising complex commands instead of repetitively typing they remain attentive rather than bored, and frankly, I trust you a lot more than programmers in performing repetitive actions. Most importantly, each time programmers think of a way to automate a complex editing task by giving you an appropriate command, they sharpen their mental skills. In contrast to their finite typing capacity, their mental skills appear to be infinitely expandable; over the years, I've encountered programmers who could almost perform magic with their editor.

My dear editor, let me give you some examples of how expert programmers can let their brains work instead of their fingers. If one of my classes contains fields named `x1`, `y1`, `x2`, and `y2`, a deft programmer will look for any of them in the code using a simple regular expression. Changing the variable names by adding an underscore between the letter and the digit with a single command is more complex, but again doable using a regular expression search-and-replace command. In cases where such a command becomes too complex, I've seen programmers locate the code element using a regular expression search but perform the replace-

 0740-7459/05/$20.00 © 2005 IEEE

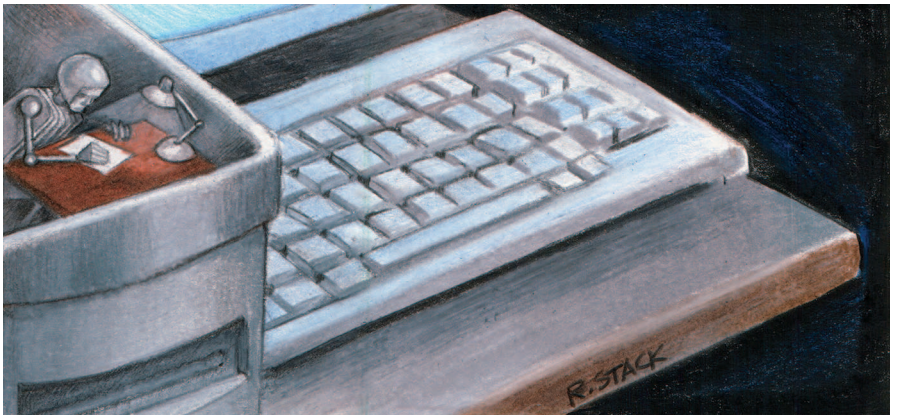
ment using the editor's "repeat last command" feature.

Of course, we both know that an editor isn't always what you want for modifying programs. External tools also come in handy, and for this, I really appreciate editors that can pipe a range of my body through an external filter. For example, if the order of two elements is reversed in a structure, the initialization data can also be reversed by piping it through the awk `{print $2, $1}` one-liner.

I don't consider all automation beneficial. Many programmers use your autoformatting facilities to beat us programs into shape. However, this is highly inconvenient for us. First of all, autoformatting isn't a substitute for good taste. Often, by judiciously adding some space in one of my code blocks, I can become easier to understand and maintain. Blindly applying autoformatting can destroy carefully laid out code elements. Additionally, a version-control system will store a new version every time I pass through with a new format, recording thousands of unimportant changes and confusing the programmers who will maintain me in the future.

Autoformatting introduces another problem into the development process; a risk-analyst would call it *overcompensation*. Programmers, confident that you, the editor, will handle all formatting tasks for them, completely neglect formatting us, leaving us worse than how we'd be without your help. Scientists have observed this phenomenon in the real world: After the introduction of safety caps in medicine bottles, parents neglected locking medicine cabinets and accidental child poisonings actually increased. About a week ago, I overhead with horror a programmer commenting that he didn't know Java's formatting guidelines because his editor handled formatting for him.

I hear you say, "You're asking me to stop doing my work. I simply can't sit idle watching the programmer hack you programs to death!" My dear editor, don't worry. You'll have plenty of useful work, if only the programmer asks. For instance, take s*yntax coloring*. Through syntax coloring, programmers can easily identify keywords, variables, constants, and comments. Often, they can also spot silly syntax errors (such as a missing quote), avoiding the distraction of an unproductive compile-edit cycle.



This brings me to another useful service you should be providing: *error highlighting*. Identifying a missing operator or semicolon also helps in the same way, as long as you do it correctly and unobtrusively. Don't distract programmers with false alarms while they write a statement, and never highlight errors that aren't. False error reports will make programmers simply switch this useful feature off.

A deep understanding of the language we're written in will also help you better serve programmers. Most modern languages follow a block structure identified by indentation; in some languages (such as Python, Occam, and Haskell) indentation is even semantically significant. An editor that won't allow programmers to easily *increase and decrease indentation levels* of our code blocks simply isn't suitable for programming.

Another language-specific service you can offer is *marking matching delimiters* (brackets, braces, square and angle brackets, and, dare I suggest it, XML tags). Of course, I know that some of your kind go even further and provide complete *refactoring support*: changes of variable names and method signatures, field encapsulation, extraction of local variables and constants, and movement of my code elements. I'm all for that; things that you can do automatically and reliably let programmers spend more quality time with me.

As an editor, you should also help programmers navigate within their increasingly complicated environment. By providing *online help* for API elements and convenient facilities for *browsing* my code's structure, I'll be less fearful of growing fat and ugly by programmers who reinvent the wheel instead of using an API feature or one of my existing classes or functions.

I've been babbling for far too long, so I'll close this letter with a few words on a feature I'm sure you're really proud of: editor macros. I'm sorry to tell you, but from my experience, these macros often indicate a hidden software design deficiency. If a programmer has to use a macro (or a wizard, in some integrated development environments) to create a new GUI element or code template, the program or its development environment probably has a design bug. In most cases, programmers using macros and wizards to save repetitive typing are programming at the wrong abstraction level.

So, dear editor, please pass our programmer friends the following advice: Don't type what you can automate in the editor environment, and don't use the editor features for what you can code.

Sincerely,

A Program

Diomidis Spinellis is an associate professor in the Department of Management Science and Technology at the Athens University of Economics and Business and the author of *Code Reading: The Open Source Perspective* (Addison-Wesley, 2003). Contact him at dds@aueb.gr.