

Rearranging the Deck Chairs

To the Editor:

The problem with the software tools that Stephen J. Vaughan-Nichols refers to in “Building Better Software with Better Tools” (Technology News, Sept. 2003, pp. 12-14) is that they do not address the most vulnerable aspect of software development.

The programming “stack” consists of four layers. At the bottom is the programming language’s line-by-line syntax. On top of that are control structures for such features as iteration and decision making. The next layer up is the program structuring mechanisms, such as functions, subroutines, and methods. The top layer is the (software) system structuring mechanisms, including modules, packages, and classes.

Off to the side are the ancillary supporting mechanisms the article describes. The problem is that if the stack rests on a foundation of (quick)sand, all the ancillary tools in the world won’t allow the production of high-quality software.

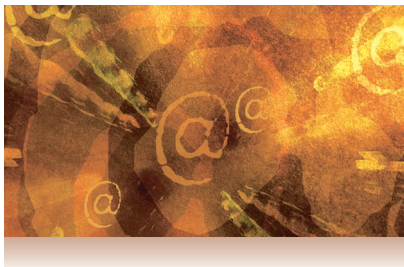
As long as programming languages continue to “inherit” their bottom layers from C—as is the case with C++, Java, and C#—tools such as those this article mentions will merely allow developers to rearrange the deck chairs on the Titanic.

*Mark Wallace
Irvine, Calif.
m.wallace@computer.org*

To the Editor:

The cost of fixing software has been known for two decades, but this hasn’t helped us make better software. How can tools build better software if tools are themselves software?

Factors other than quality determine



a product’s fate. To quote Bob Colwell (“Engineering, Marketing Teams, and Management,” *Computer*, Sept. 2003, pp. 5-7), “...some pretty poor products succeed. When they do, look closely and you probably will see some inspired marketer lurking just offstage and grinning maniacally.”

Most products are developed primarily for features and secondarily for quality. Some products actually rely on users to find bugs. Similar to optimization becoming irrelevant as a factor in choosing compilers, quality may become irrelevant in choosing products.

The need for quick revenue and the urge for accountability make quality less important. Consumers only notice the lack of quality when a disaster occurs.

According to James Bach (“What Software Reality Is Really About,” *Computer*, Dec. 1999, pp. 148-149), processes and tools aren’t the problem, people are. Until consumers buy a product because it effectively solves problems instead of being influenced by market hype or the product’s bells and whistles, quality will be a secondary goal—unless management proactively recognizes and rewards quality.

We ought to strive for quality proactively instead of reactively (when a disaster occurs). On the other hand,

quality may become irrelevant in the marketplace. In that case, we may still have debates about quality, but only for the sake of debating.

*Raghavendra Rao Loka
Palo Alto, Calif.
rao@acm.org*

The author responds:

Alas, I must agree. “Quality is job one” makes a nice motto, but it’s not one that’s often used with a straight face in software development. Indeed, all we need to do is look at the almost monthly reports of not mere errors, but error exploitation, with Microsoft’s desktop operating systems—which is approximately 90 percent of the desktop market—to see that quality is not at all the same thing as popularity.

SUCCESSFUL DESIGN REVIEWS

To the Editor:

I enjoyed reading Bob Colwell’s article on design reviews (At Random, “Design Reviews,” Oct. 2003, pp. 8-10). I would like to share some rules from successful projects at Bell Labs that made design reviews less stressful and more successful and ultimately led to the creation of reliable components for the telephone system:

Keep design reviews separate from design inspections. Design reviews were informal, high-level discussions about what was to be designed; inspections were formal meetings to find and reconcile defects in an existing design. The formality of the inspection reduced divisive defensiveness.

No managers allowed. Inspections ran better without having the bosses present to pander to. Only the author, peers, and the author’s “customers” (manufacturing) were invited to attend.

Metrics measure the process, not the people. The quality team used the statistics or metrics kept on an inspection, such as average reviewer prep time,

We welcome your letters. Send them to computer@computer.org. Letters are subject to editing for style, clarity, and length.

number of defects found, number of lines of code or pages reviewed, to evaluate the overall process. No one “cooked” metrics because we knew that it was impossible to isolate individual performance in the data.

Define the roles. Before announcing the meeting, the review team selected two members to act as moderator and recorder. The moderator ran the inspection and the recorder kept the error and issues lists. We believed—and our experience confirmed—that the author could not act as a recorder or moderator and maintain objectivity.

Criticize the design, not the designer. The participants must distance themselves emotionally from the design. For example, good criticism sounded like this: “Diode D23 is backwards.” We avoided questions like “What were you thinking when you placed D23 backwards?”

Don’t design it in the meeting. The design inspection was not the time or place for detailed design discussions. Authors were expected to work the design details with the appropriate teammates offline.

As Bob Colwell suggested, we produced some excellent stuff for the telephone network when everyone on the design team concentrated on making the design as successful as possible without bruising egos.

Jesse N. Alexander
Montclair, N.J.
jesse_alexander@ieee.org

BETTER SOFTWARE VERSUS MORE SOFTWARE

To the Editor:

I was amazed and dismayed to read Mark Halpern’s article, “VIM: Taming Software with Hardware” (Oct. 2003, pp. 21-25). His considerable verbiage amounts to the observation that more main memory will allow more loop unrolling, less garbage collection, and fewer pauses to access secondary storage. This is neither revelatory nor particularly interesting.

Halpern’s assault on fundamental programming constructs (does he really

believe software would be better without loops or hashes?) as not being “data-oriented” aside, the call for more memory overlooks something rather simple. Given the well-known tendency of software to fill all available space, there already is a demand for as much memory as we can provide. The economies of scale for main memory are currently at the local minimum between



fabrication cost and support cost, and they aren’t likely to advance any faster than Moore’s law already provides. Memory size and speed are unavoidably inverse: More memory is not an automatic performance guarantee.

Ultimately, the article is one more appeal for “more software” over “better software.” Even the author admits that petabytes of memory—or petahertz CPUs, or petabyte/second interconnect, or what have you—can’t solve every computing problem.

Rather than dreaming of hardware one day riding to the rescue, I recommend engineering better software design tools and languages to make better use of the hardware we already have.

Mark Neidengard
Hillsboro, Ore.
mneideng@ugcs.caltech.edu

The author responds:

Mark Neidengard is wildly off target in his comments on my article. He ignores my primary message—that effectively infinite memory would permit a drastic shrinking and simplification of all software.

According to him, the main advantage I promise is greater execution speed. I do claim a speed advantage, but only as a side benefit—halving our

software burden is the main prize. My article states this explicitly and repeatedly, starting with the title and the associated pull quote, and in passages such as, “But speed of execution, important as it is, is only an incidental VIM benefit. The main reward will be a radical reduction of the software burden...” and “...the primary benefit VIM would provide: eliminating half the software we now write.”

Saying that I want “more software” is such a perverse reading of my call for the elimination of half our software that I hope Neidengard mistakenly typed “software” for “hardware,” but I can’t be sure. He asks if I really believe that software would be better without loops or hashes. The answer is yes, of course; those constructs are simply expedients we resort to to conserve memory. Would he deny that programs that don’t need them would be simpler and faster?

Neidengard claims that Moore’s law blocks the economies of scale that I expect. But I don’t assume an increasing chip density, only that ramping up production of today’s chips could drastically reduce memory cost—an economic effect not subject to Moore’s law. He says that I admit that VIM wouldn’t solve every computing problem. True, and I accept the consequence of that admission: exclusion from the rogues’ gallery of charlatans.

GPU CLARIFICATION

To the Editor:

I enjoyed reading “The GPU Enters Computing’s Mainstream” by Michael Macedonia (Entertainment Computing, Oct. 2003, pp. 106-108), but I wanted to point out that a statement on page 107 might cause some confusion. The author says, “Precision, and therefore visual quality, increases when displayed with 128-bit floating-point color per pixel...” That isn’t quite true, and the URL cited presents marketing information that is a bit misleading about what higher precision means to graphics processing units (GPUs).

The new 128-bit floating-point for-

mat has no direct correlation with the actual display device. In fact, the GPU normally needs to convert 128-bit floating-point results to the usual 24-bit fixed RGB colors to display them.

The 128-bit floating-point format does a much better job of preserving precision than the 8-bit or 16-bit per channel fixed formats. That said, while 128-bit floating point is necessary in some cases, Nvidia and ATI recommend using the most compact format possible to preserve bandwidth.

In the area of increased visible color depth, various R&D groups recently have been creating displays with higher dynamic ranges. Last year, Matrox came out with its Parhelia system, which supports 10 bits per color channel. The image at www.realtimerendering.com/cubes.gif shows the problem of using only 8 bits per channel. The bottom of the center cube has discernible bands at 256 levels of gray scale, not visible on 10-bit displays. This year, Sunnybrook Technologies made a splash at Siggraph by demonstrating a display with a secondary, low-resolution LED backlight that provides intensity values that are brighter and with a much wider range (60,000:1).

*Eric Haines
Ithaca, N.Y.
erich@acm.org*

CHALLENGING A THEORETIC DISCUSSION

To the editor:

I am writing to criticize the decision to publish “Fighting Epidemics in the Information and Knowledge Age” by Hai Zhuge and Xiaoqing Shi (*The Profession*, Oct. 2003, pp. 116, 114-115).

Citing their own simulation studies, the authors advance, and defend, a theory in the realm of public health: that isolation control measures had no significant effect on containing the SARS epidemic. The rest of the article reinforces their claim that exchange of information, rather than isolation, was the important factor in the eventual containment.

Discussion of this type of theory does not belong in a publication whose readers, and reviewers, are not experts in public health and have no way of validating the authors’ claims or the correctness of their model. The authors do not cite any opposing view or theory, except for stating that their conclusions are “counterintuitive to the thinking of many people and some governments.” It seems appropriate that any theory of this sort be debated and peer-reviewed by public health researchers, not by computer professionals.



A particular contradiction in the article is noteworthy. The statement that “People will adopt intelligent behavior to avoid infection...” renders the analogy between e-viruses and biological viruses less than compelling. In fact, the best defense against e-viruses appears to be isolation control measures—firewalls, virus filters, discarding unsolicited attachments, and so on.

Information technology affects all aspects of modern society. The Web is a useful tool for everyone. This, however, does not provide a license for indiscriminate publication of a claim that is purportedly based on computer modeling. A balanced review of how computers are used in the public health arena would, of course, be quite useful to your readers.

*Behrooz Parhami
Santa Barbara, Calif.
parhami@ece.ucsb.edu*

The authors respond:

Our essay discusses the current and future role of the evolving Web in controlling epidemics as suggested by the course of the SARS epidemic. Our intention was to emphasize the implications

with regard to the relationship between knowledge sharing, the epidemic itself, the virtual environment, the natural environment, and society.

We do not agree that isolation control measures are the best defense against e-viruses. In effect, commonly used isolation measures only attack the symptoms. A study of the environment where the e-viruses start, and of the early stages during which they spread, would be the most effective means for developing professional policies and technical innovations to prevent e-virus epidemics.

Cross-disciplinary study is important to the progress of science and technology. Cooperation among professionals working in different areas helps generate significant innovation.

Simulation based on abstraction and simplification is an important way to find natural rules. One purpose of our essay was to show how what we observed in simulation impelled us to review the effectiveness of the preventive measures actually taken. We intend to publish the details, and we will welcome helpful discussion of them.

Fighting epidemics is not just a public health issue. It is an issue concerning the ecology of human society and its environment, and it is a major concern for our evolving global society.

Academic criticism, scientific attitudes, and academic freedom are inseparable aspects of the scientific spirit. We appreciate constructive criticism and productive discussion, but not when it is directed at the editors or the magazine, which offers a forum for exchanging different viewpoints.

CORRECTION

In *Computer's* July issue, the article by Ahmet M. Eskicioglu titled “Protecting Intellectual Property in Digital Multimedia” contained an error. In the table on p. 41, the column header should have said, “Estimated losses (in millions).” *Computer* regrets this error.