When I read the title of James Robertson’s article (“Why Analysts Should Invent Requirements”) in the July/August 2002 issue of IEEE Software, the first question I asked myself was “Does he really mean what I think he means?” If so, it’s significantly in conflict with what I teach and practice with regard to requirements engineering. A requirements analyst’s job is to find out what the users want in their new system, not tell them. Was he saying that we’re smarter and that users really don’t know what they want, so we have to guide them toward a goal of our own choosing?

Users do know what they want and can improve their systems on their own. They know their business. What they don’t know is how to turn that knowledge into a system specification. Unfortunately, the traditional approach to requirements analysis is to ask a lot of technology-based “how” and “what would you like?” questions as well as spend time distilling the original requirements from the current system. Analysts must invent requirements only when they don’t know how to ask the right questions. Robertson is correct in stating that sometimes we have to invent an answer to determine what the proper question should have been. But, again, that’s only because we don’t know how to properly create and structure the right questions. To change the results we get when doing requirements analysis, we must change how we go about doing the work. Essentially, that means changing the nature of the questions we ask. It means advancing requirements engineering from a craft to an engineering science.

Alan Goodbrand
Senior partner and principal
PowerPlus Systems
www.PowerplusSystems.com

James Robertson responds:

I agree with Alan when he says that users know what they want to do in their current jobs. However, when it comes to making significant improvements, users’ motives are suspect. Why? The big innovations might lessen or “de-skill” (perhaps even eliminate) their jobs; they will probably not be in those jobs when the software is finally delivered. In addition, users cannot be expected to know about new technology that is available.

Why is technology important? Because some new technologies create their own requirements. Before it existed, I doubt that a single user asked for the World Wide Web, but look at the businesses (several million requirements later) that have been built on the back of it. Users do not ask for requirements unless they think those requirements are technologically possible. Business analysts are in the technology business, and it is their responsibility to advise others about new technologies and invent requirements that are now possible because of those technologies.

We must remember that software is not an end unto itself, but a service that we provide for our clients’ businesses. The academic purity of software requirements might help those businesses, but innovation will help a lot more.

Distance learning not delivering—yet

In “Software Engineering Programs: Dispelling the Myths and Misconceptions” (September/October 2002), Hossein Saedian, Donald Bagert, and Nancy Mead highlight important points regarding software engineering programs. In their seventh myth, named “Software engineering programs will correspond to specific corporate requirements,” the authors emphasize industry’s current focus on specific languages and tools to the detriment of best practices. I agree completely.

Unfortunately, the IEEE Computer Society
LETTERS

has similarly erred in its distance learning courses. At the moment, these courses are dedicated to specific languages and specific tools. According to www.computer.org/distancelearning/benefits.htm, such courses “cover what you need to know.” The August issue of Computer (page 66) notes, “Next year’s distance learning catalog may also include prep courses for the IEEE Computer Society Certified Software Development Professional Program.”

But the proposed topics are only a small part of what software practitioners really need. SE practices such as the ones described in the Software Engineering Body of Knowledge project (www.swebok.org) are not covered.

I really hope the Computer Society offers members SE courses in the next several months. Courses on software practice are seldom widely accessible. When the distance learning program does offer such topics, the Computer Society will be supporting the wider dissemination of software engineering practices, which is one of its important roles.

André Sintzoff
Software Architect
Gemplus
sintzoff@computer.org

Not Really Pair Programming

In the November/December 2002 issue, Paul Freedman writes about pair programming and Edsger Dijkstra. Dijkstra’s experience has nothing to do with XP pair programming. He was just trying to catch typos, which I know from experience is hard to do when using punched cards or tapes.

Further, XP pair programming is for programmers who can’t or won’t express their designs so others can review them. With 20 years’ experience in development, I believe that coding effort is simply “compiling” or translating a design into some code that can in turn be compiled into machine language. There shouldn’t be much creativity in the coding process; that’s all done during design.

There are simpler, more effective ways to do peer review. One example is doing a low-level software design by writing all the comments for the code, but none of the code. This leaves no “unnecessary” documents lying around.

Mark Bullock
Quality assurance manager
The Cobalt Group
mbullock@cobaltgroup.com

Reverse Engineering Tools

Our tools help developers understand, document, and maintain impossibly large or complex amounts of source code.

They parse Ada 83, Ada 95, FORTRAN 77, FORTRAN 90, FORTRAN 95, K&R C, ANSI C and C++, and Java source code to reverse engineer, automatically document, calculate code metrics, and help your engineers understand, navigate and maintain source code that has grown too large for one person (or even a group) to know.

Big projects aren’t a problem. The tools can parse and later manipulate very large amounts of code. 1,000,000 SLOC projects (and larger) are common among our customers.

We also focus on exceptional customer support based on rapid response from a real engineer.

Download and try on your code:
http://www.scitools.com/

All downloads are fully functional, just time limited.