New dimensions

Over the next few issues of IEEE Software, you may notice changes that we hope will add new dimensions to the magazine’s scope.

Our highly regarded software reviews departments (Software Reviews and New Product Reviews) are being fine-tuned to bring you more technically oriented products. This means even greater emphasis on tools; leading-edge products for engineers, scientists, programmers, and their managers; and products designed for the specialist, developer, and knowledgeable computerist.

We are proud to increase our focus on two important dimensions in software engineering: quality assurance and human factors.

MCC’s Vincent Shen, a recognized leader in software-quality research, will edit the Software QualityTime department, which will report important results, techniques, and practical experiences about software quality, validation and verification, and project management. This department will regularly feature brief reports by widely recognized people in this field.

Koffler Group’s Kathleen Potosnak, one of those rare individuals who is interested and experienced in human factors and software design, will edit the Human Factors department. She will cover the latest developments in user interfaces, human factors, and human-computer interactions technologies — and will help you apply this technology to everyday designs.

Both departments start this issue.

Another new dimension I am personally working very hard to complete is the addition of Europeans, Asians, and other nationalities to IEEE Software’s editorial board to broaden the base of our technologists and give our readers insights into software developments around the globe.

Please let me know how we are doing: Your suggestions and concerns are valuable inputs to the process of adding new dimensions to IEEE Software.

Ted Lewis
Editor-in-Chief

SOFTWARE LETTERS

C++ impressions

To the editor:

The review of The C++ Programming Language (Book Reviews, July, p. 110) praises the book’s organization and readability but gives the reader almost no impression of the language itself. This is unfortunate because C++ is a relatively new programming language and the book is both its official definition and (for now, at least) its most readily available description.

The reviewers say, “The most useful feature of C++ is that it makes some of the things that are very complicated in C easier to accomplish,” but they forget to add “without any loss of efficiency.” This design goal (not feature) is achieved by several complementary language features.

For example, C++ has strong type checking to catch incorrectly formed function calls and incorrect use of pointers at compile time. This not only eliminates a class of runtime errors but also provides the basis for overloading functions with compile-time binding.

Also, C++ supports abstract data types with optional data hiding, member functions for controlled access, and automatic, programmer-defined instance initialization and destruction. This facilitates a more modular style of programming than C allows, with attendant improvements in reliability, maintainability, and reusability.

Binding function names to executable code can be postponed to runtime where necessary, allowing object-oriented programming to be combined with more conventional styles. Unlike most other object-oriented languages, C++ does not sacrifice compile-time type checking or impose a runtime performance penalty.

Also, the review had two incorrect or misleading impressions. First, C++ has no features specifically for windows. What it does have is a well-integrated set of features that let programmers cope with complexity, giving them the power, scope, and capability to build and interface to all kinds of complex software systems — including window managers. In this, C++ goes far beyond C.

Second, C++ is already more widely used outside AT&T than within it and is used more in traditional software development than in “hard-core computer science environments,” contrary to the reviewers’ assertions.

The C++ Programming Language is aimed at people who want to write better programs now, who are looking for more efficient ways to construct programs, or who are simply interested in programming languages. I think these people will find the book interesting and useful.

Jonathan E. Shopiro
AT&T Bell Laboratories
Murray Hill, N.J.

Distressed by reference limit

To the editor:

I recently learned, with some distress, of the IEEE Software policy that limits submitted articles to 10 references. As an editor myself, I feel strongly that editors should do everything possible to encourage computer scientists to cite each other’s work (when it is genuinely related).

Computer science is a young field that suffers severely from the “not invented here” syndrome. Unlike their colleagues in more established fields such as mathematics and physics, computer scientists do not have a generally acknowledged set of intellectual leaders or landmark contemporary advances.

Instead, we have many competing traditions and a great deal of commercial hype. As a result, our representation and effectiveness at the national level is greatly diminished (for example, in the National Academies of Science).

Correction

The label “Guest Editor’s Introduction” was inadvertently left off Will Tracz’s article “Reusability Comes of Age” on pp. 6-8 in the July issue. We regret the omission.