Editors' Corner

John Pugh

Paul White

CERTAINLY IT HAS been demonstrated over the past few years the merits of using Smalltalk for application development in a wide variety of disciplines. It has successfully been deployed in areas as diverse as banking, insurance, telecom, and utilities, and the list goes on. With that given, it is amazing the number of times we still end up having to justify Smalltalk as a valid choice for software development projects. The misconceptions with respect to speed, difficulty of development, the proprietary nature of the software and so on still persist. It still seems impossible to avoid the "why should I even consider Smalltalk for my application development?" type of questions.

It's never quite clear why these misconceptions cannot be eradicated. Perhaps it's just the nature of the software industry, and engineering in general. Many of the often stated shortcomings of Smalltalk of course have long been addressed. Dynamic compilation, for example, provided a big boost in execution speed, and puts Smalltalk very close in execution speed to C++, when it is used as a fully object-oriented system. Garbage collection is another criticized feature even though today's implementations are so well tuned that these criticisms are for the most part unfounded.

The arguments (for using Smalltalk) must be raised to issues of business value, not technical merit.

The reason for writing is not to complain about the unfair criticisms, but instead to highlight the work we all have left to do if we're to take Smalltalk to the next level. If we really believe Smalltalk is worth sharing with the rest of the software world, then we need to do a much better sales job. Of course, many companies are working very hard to do just that. GemStone, for example, has taken a big step forward by positioning their OO database product as an application data server that uses Smalltalk as the primary development environment. Given the visibility they've been receiving of late, this is a positive step for all of us. IBM has also clearly demonstrated that Smalltalk is a major part of their future in the software arena. Their moving VisualAge and Smalltalk to their various platforms makes it a viable solution for software development shops that otherwise never would have considered it. And investors in ParcPlace/Digitalk are certainly expecting growth in the industry.

We have to realize that having Smalltalk progress on a pilot project by pilot project basis is not the scale of growth necessary. We need to see it being adopted by organizations as their leading choice for mainstream application development. And those of us who make a living from this technology cannot afford to stand idly by. We have to take the lead in educating "the masses" in why Smalltalk is a sound business choice. We're now to the point that making arguments strictly on a technology basis will never enable us to achieve this growth. The arguments must be raised to issues of business value, not technical merit. As an example, we believe that for software development to qualitatively improve as an industry, we need to move toward the adoption of notions such as software components. These components need to be self-contained, testable, and well specified on the basis of their behavior, constraints, input types, and output types. If we make this argument convincingly, then it becomes much simpler to convince people that Smalltalk is probably the best choice for achieving this type of software organization. We urge you to try arguing the case for Smalltalk in this type of context, rather than "but it has garbage collection which saves you a lot of work"—we think you'll be more successful.

The other issue with respect to "selling" Smalltalk is that we cannot afford to wait much longer in succeeding. We probably need to take a type of "storm the beaches" mentality to this, since the time is ripe now for market penetration. The simple fact is that virtually every software development shop is facing the same problem, namely that the systems they are being asked to build are significantly more difficult to construct than systems of even two or three years ago. The new systems being created today must capture business rules and business logic to be considered successful. Well, let's argue for Smalltalk's use because it is the best tool for capturing this knowledge. We think it is possible to demonstrate that Smalltalk is, purely from a language point of view, better positioned for application development than languages such as PowerBuilder, VisualBasic, and even C++. Let's use this to our advantage, rather than arguing whether or not Smalltalk's window creation facilities are as slick as other products—in the longer run, business decisions will always be based on business value! We hope you'll enjoy the issue.