ROWE84
Panel Discussion at SIGPLAN '83: Symposium on Programming Languages Issues in Software Systems
Moderator/Editor: Lawrence A. Rowe (U.C. Berkeley)
Panel Members: Peter Deutsch (Xerox PARC), Stu Feldman (A.T.&T. Bell Laboratories), Butler Lampson (Xerox PARC), Barbara Liskov (MIT), Terry Winograd (Stanford University); SIGPLAN Notices, V19 #8, August 1984 "At the SIGPLAN '83 conference, a panel discussion was held to discuss what the important programming language issues might be in the 1980's. This paper is an edited transcript of the discussion." One major theme throughout the discussion can be broadly labelled "programming language design is not an island." Peter Deutsch introduces this theme when Lawrence Rowe asks "if data abstraction is the great idea of the 70's, what will be the great idea of the 80's?" Without guessing what the great ideas of the 80's will be Deutsch predicts that they are likely to come from outside the mainstream of programming language research. Some are likely to come from applying programming languages to real problems. Deutsch justifies this predication by citing three more "great ideas of the 70's". The class/instance inheritance mechanism was made into a real programming tool in Smalltalk. But the idea originally came from simulation. Logic programming came from Logic. The idea of a portable machine-oriented systems programming language was embodied in C. But this idea came from the engineering and application of programming languages. This theme comes up again in response to a question from Alan Newell. His claim is that all of the important programming language ideas have come from AI, including goal-based programming, logic programming and symbolic manipulation systems. He asks why all the new ideas seem to come from outside the programming language community. Barbara Liskov says that the purpose of programming languages is to express solutions to problems, so it's no surprise that ideas come from actually solving problems. What programming language researchers do is take the ideas that seem to have wide applicability and flesh them out. The ideas that more specific are incorporated into problem-oriented languages. Deutsch adds that "there is a big difference between a programming technique expressed by convention, or expressed as an algorithm, and that same technique expressed as a matter of the programming language". Incorporating a technique into a language can be a major creative act in itself. A second theme might be called "programming languages aren't just for programming anymore". This theme, like the first, arises in response to the question about the great ideas of the 80's. Terry Winograd's answer is that he sees lots of different kinds of languages developing: specification languages, protocols, text-formatting languages. None of these is a standard points to this as an example of a problem that must be dealt with in a wide class of languages, not just programming languages. A related theme is that even within 'programming' languages, there is still a wide range. Different special purposes will require different languages. The interface to a standard statistical package is a limited language. Visicalo can be thought of as a language. "Professional languages" are being developed, to be used within a limited domain--by auditors for example. These languages lie somewhere between general purpose and problem-oriented languages. In addition to the topics already mentioned, the panel discusses garbage collection, programming environments, extensible languages, optimisation, strong-typing, formal verification, large memories, fast processing, cheap chips and lazy programmers. This paper is easy reading. It is also fun reading. "(LAUGHTER)" appears more than a dozen times in the transcript. This review has avoided spoiling any of the jokes.