Inventing language

November 10 2009, by Larry Hardesty

Institute Professor and Associate Provost for Faculty Equity Barbara Liskov. Photo: Donna Coveney

(PhysOrg.com) -- Last Thursday, the day after the New York Yankees won their first World Series of the 21st century, MIT Institute Professor Barbara Liskov, the 2008 recipient of the Turing Award — frequently called the Nobel Prize for computer science — delivered the first lecture of the 2009 Dertouzos Lecture Series.

John Guttag, the Dugald C. Jackson Professor in the Department for Electrical Engineering and Computer Science, introduced Liskov, and he greeted the crowd of more than 150 people wearing a Yankees cap. The cap, he claimed, had a purpose other than pouring salt in the wounds of disappointed Boston Red Sox fans: His PowerPoint presentation was titled “Barbara Liskov: the Derek Jeter of Computer Science” — a

reference to the Yankees' star shortstop that drew hisses from the audience. Guttag's presentation also featured photos of Liskov dressed in renaissance clothing and putting a tray of cookies in the oven — "moonlighting as a baker to put her son through Harvard," Guttag said, "which was all the more poignant, since he could have gone to MIT for free."

Guttag waxed serious long enough to describe Liskov as one of his own most valued mentors, and then Liskov took the podium. She opened with a joke of her own: In large part, her Turing Award honored work she had done in the 1970s, laying down principles for the organization of programming languages that today are almost universally followed. After the award was announced, she said, her husband spent a lot of time on the computer Googling the reaction, and at some point, "he came upon a quote from someone who said, 'What did she get this award for? Everyone knows this, anyway.'"

After that, Liskov was all business. The thrust of her talk was that, in the 1970s, it was emphatically not the case that "everyone knew this," and she described in great detail the intellectual environment in which she did her pioneering work. The talk was not for the uninitiated: she began by describing several papers from the early 1970s from which she had drawn inspiration — with titles like "Go To Statement Considered Harmful" and "Information Distribution Aspects of Design Methodology" — but first conceded to the audience that while "many of you have read them, I'm not sure all of you have."

Liskov explained that, in the fall of 1972, after reviewing the literature in the field, she came up with the idea for what today are called abstract data types. Traditionally, a computer program would be a long list of exhaustively detailed instructions, and anyone reading the code — including the original programmer — could easily get lost. Abstract data types are, effectively, repositories for the computational details of the

program, which let the programmer concentrate on the big picture. A complicated program turns into some rather simple interactions between the abstract data types. And indeed, the programmer can later change the details of the data types' instantiation — how they do their low-level computations — without changing the overall structure of the program.

Liskov explained how, after coming up with the idea of abstract data types, she and some collaborators created a programming language, CLU, which put some of her ideas into practice. The rest of her talk was largely a demonstration that CLU prefigured most of the ideas that are commonplace in today's programming languages — ideas with names like polymorphism, type hierarchy, and exception handling.

During the question-and-answer session that followed the talk, Liskov was asked the secret of her success. Part of her answer — which must have chagrined some members of the audience — was that "I don't work that many hours a day." "I always went home at night, and didn't work in the evening," she said. "I always found that downtime to be really useful." She also, however, emphasized the importance of pursuing research that's interesting — rather than, say, the research that will generate the most publications. That way, she said, "at the end, if you fail, at least you did something interesting, rather than doing something boring and also failing." After the laughter died down, she added, "Or doing something boring and then forgetting how to do something interesting."

Provided by Massachusetts Institute of Technology (news : web)