Q&A
Liskov on Liskov

Barbara Liskov talks about her groundbreaking work in data abstraction and distributed computing.

Barbara Liskov, a professor at the Massachusetts Institute of Technology (MIT) and winner of the 2008 ACM A.M. Turing Award, has worked throughout her career to make software systems more accessible, reliable, and secure. We caught up with her recently to discuss a few of her most important accomplishments—and to find out what she’s working on now.

Let’s talk about CLU, the programming language you developed in the 1970s to handle abstract data types. Before I came to MIT, I was working on the VENUS system, and I got some ideas about a different way of modularizing programs around what I called multi-operation modules. When I came to MIT, I started to think of that in terms of data types. And then I decided the best way to continue the research was to develop a programming language.

How did your ideas differ from the research that was going on at the time?
When I started, the main way people thought about modularization was in terms of subroutines—of abstracting from how you wrote a procedure to calling that procedure, say, a sort routine, or a lookup routine. But they didn’t have any way of linking a bunch of procedures together.

And that’s what CLU’s clusters accomplish.
Yes. A cluster would have all the operations you needed to interact with a data object, and inside you could implement it and later re-implement it however you wanted.

Eventually, object-oriented programming evolved from your work on CLU.
Object-oriented programming evolved from two different strands. There was my work on data abstraction and some related work that was going on at [Carnegie Mellon University]. The other influence was Smalltalk. Both of these were sort of getting at the same idea in slightly different ways, but the big difference between my work and the Smalltalk work was that I focused on making a very strong distinction between what a module did...[Continued on p. 119]
You’ve since focused your attention on distributed computing. Can you tell me about your work on fault tolerance?

As you move to a distributed environment, where you have your storage on a different machine than the one you’re running on, you can end up with a system that is less reliable than before because now there are two machines, and either one of them might fail.

But there’s also an opportunity for enhanced reliability. By replicating the places where you store things, you can not only guarantee they won’t be lost with a much higher probability, you can also guarantee they will be available when you need them, because they’re in many different places.

Tell me about Viewstamped replication, the protocol you developed for replicating data in a benign environment.

The basic idea is that, at any moment, one of the nodes is acting as what we called the primary, which means it’s bossing everybody else around. If you have several different nodes, each replicating data, you need a way of coordinating them, or else you’re going to wind up with an inconsistent state. The idea of the primary was that it would decide the order in which the operations should be carried out.

What happens if the primary fails?

Well, you also need a protocol—we called it the view change protocol—that allows the other replicas to elect a new leader, and you have to do that carefully to make sure everything that happened before the primary failed makes it into the next view. The nodes are constantly communicating, and they’ve got timers, and they can decide that a replica has failed.

Did this work lead to the protocol you subsequently developed for coping with Byzantine failures?

It did, about 10 years later. It’s much harder to deal with Byzantine failures, because nodes lie, and you have to have a protocol that manages to do the right thing. My student, Miguel Castro, and I made a protocol that I can now see is sort of an extension of the original—of course, hindsight is very nice. But the primary is the boss, the other replicas are watching it, and if they feel there’s a problem, they go through a view change protocol.

Recently, you’ve worked on the confidentiality of online storage. If you put your data online, you want to be sure that it won’t be lost. Additionally, you want to know that it isn’t being leaked to third parties and that what’s there is actually what you put there.

How did you get interested in the subject?

In the nineties, I did some work with my student, Andrew Meyers, on information flow control, which is a method of controlling data not by having rules about who can access it, but by having rules about what you can do with the data after you’ve accessed it. That’s what I’ve been looking at recently, but the work with Andrew was programming language work, and then we just extended it.