Barbara Liskov
# Table of contents

1. Introduction ........................................... 2  
2. Early life and career ................................ 3  
   2.1. Roles of Society .................................. 3  
   2.2. Venus Computer .................................. 4  
3. Work in programming methodology ................. 5  
   3.1. CLU & ARGUS .................................. 5  
   3.2 Liskov Substitution Principle ..................... 5  
4. Work in Distributed Systems ......................... 7  
   4.1. Byzantine Fault Tolerance ....................... 8  
5. Awards ................................................ 8  
6. Conclusion ............................................ 9  
Sources .................................................. 10
1. Introduction

Barbara Liskov is an institute professor at the Massachusetts Institute of Technology (MIT). She is most famous for winning the Turing award in 2008. She also won the 2004 IEEE John von Neumann medal [CSA04].

She has done research on many different computer science subfields, including artificial intelligence, programming methodology and distributed computing. Currently she leads the Programming Methodology Group at MIT. Her and the group’s current research is focused on distributed computing and Byzantine fault tolerance.

Being a contemporary computer scientist, there is as of yet no biography book on Liskov. Therefore almost all the references used in this seminar paper are by necessity from the Internet - mainly, MIT’s resources and Liskov’s interviews and her own writings.

Image 1.1. Liskov in 2010
By Kenneth C. Zirkel
2. Early life and career

Barbara Liskov (née Barbara Jane Huberman) was born in 1939 in California [WiLis].

Liskov first studied mathematics in the University of California, Berkeley. She earned her BA in 1961. After graduating she worked for a few years. She could not find an interesting job in mathematics, so she took a job as programmer. This was what lead her to the field of computer science and got her interested in the intricacies of programming.

She wanted to learn more about programming, so she decided to get back to university and went to graduate school in Stanford University. As this was in the sixties, and computer science as a field was only emerging, Stanford did not quite have a computer science department yet. But one was rapidly forming and Liskov took exams to qualify her in one of the first actual computers science groups [EnG].

2.1. Roles of Society

Later on Liskov reflected back on her career and contrasted her life experiences with the societal roles of women in her youth and young adulthood:

“When I look back on my early career, one thing that strikes me is the randomness of the process that led to where I am today. I didn't have a plan for where I was going; instead I reacted to obstacles and opportunities. I believe that some of this was due to being a woman. When I was young, it was uncommon for women to think about having a career. The effect on me was that I just focused on doing work that was interesting but expected to stop working when I had a family. I thought about things in this way even after I had my Ph.D. However, as I got into my research in software systems I realized that I was really committed to my work and would not give it up. Later when my husband and I had a family, I continued to work full time. It's possible that my lack of focus on a career early on slowed me down, but it also freed me up to take advantage of opportunities that I otherwise might have missed.”

Women of that time were often expected to give up their career in favor of becoming full-time homemakers, but Liskov wanted to keep working [EnG].
2.2. Venus Computer

Liskov got her PhD in 1968 in the field of Artificial Intelligence. For her thesis she did a computer program to play chess endgames. [WiLis] She then went back to working in industry, because she had not found a good academic position. She worked for not-for-profit organization Mitre and switched from AI to systems research. In Mitre, she worked on operating systems and created the “Venus Computer” [AMC].

The Venus computer was a machine architecture /multiprogramming system that experimented on multi-user time sharing. From Liskov’s thesis paper abstract:

*The Venus Operating System is an experimental multiprogramming system which supports five or six concurrent users on a small computer. The system was produced to test the effect of machine architecture on complexity of software. The system is defined by a combination of microprograms and software. The microprogram defines a machine with some unusual architectural features; the software exploits these features to define the operating system as simply as possible. In this paper the development of the system is described, with particular emphasis on the principles which guided the design [Lis72].*

In hindsight it seems that fault-tolerance was interesting to Liskov already at this point. The Venus was an experimental system, built so to that several users could use the system at the same time. Users were connected to a virtual machine “*so that major errors would not compromise the entire system, only the virtual machine for that user [AMC]*”.

In 1971 Liskov quit Mitre and joined the MIT as a professor. From that time on, she concentrated on programming methodology.
3. Work in programming methodology

3.1. CLU & ARGUS

CLU and ARGUS are two programming languages that she and her students developed in the seventies and the eighties. Developed between 1974 and 1975, CLU had to do with solving the software crisis at the time: as computers got more powerful, programs were getting larger, more complex and more difficult to manage, as the programs often did not have an easily manageable structure [WiCLU].

Some years before, Edsger Dijkstra had published his famous pamphlet, “Goto considered harmful”. CLU was, among other things, a goto-less language - and one that brought programming paradigms of the time closer to ones we have now. CLU was not widely-used, but remarkable because it had many features that were later adopted into commonplace usage. CLU was not object-oriented, but rather “object-based”; it featured classes and constructors. Liskov later reflected on the importance of CLU:

“The (programming) languages people use today are heavily based on CLU, even though the people who are writing the programs may not realize it,” she says. “The world gets more applications that are better designed than they would be in the absence of this work. Mind you, though, this work was waiting to happen. I just happened to be in the right place at the right time. [SKa]”

Argus was a language for distributed system applications, devised in the 1980s. Liskov has devoted much of her energy to distributed systems research later in her career, but after CLU, she also had a very strong presence in the field of programming methodology.

3.2 Liskov Substitution Principle

Programming methodology deals with how programs and programming projects should best be organised, the very structures and paradigms of programming itself. It is research and analysis on which practices work best and make the most sense in programming projects. For example, the decision between a top-down versus a bottom-up design approach can be said to me a methodological decision. Liskov is a pioneer of programming methodology research; this and object-oriented programming in general are what mainly gave her the Turing award.
On the field of object-oriented programming, Liskov is also famous for her “Liskov Substitution Principle” (LSP) - one that she did not name herself:

"Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it."

The formal definition is

"Subtype Requirement: Let \( \phi(x) \) be a property provable about objects \( x \) of type \( T \). Then \( \phi(y) \) should be true for objects \( y \) of type \( S \) where \( S \) is a subtype of \( T \)." [WiLSS]

This principle has to do with the correct assignment of subtypes to their respective superior types: one should not derive a subtype from a too specific supertype. In Liskov’s own words:

“So it’s a very simple rule, it’s really very intuitive and what it means is that if you have a type that is a subtype of another type and you use an object of that subtype in a context where you expect an object of the supertype, then the object of the subtype ought to behave like you expect. In other words you’re depending upon the specification of the supertype and the object should meet that specification even though it might belong to a subtype. [InfoQ]”

In layperson’s terms, if you have a supertype “Cat”, and the subtype “Kitten” derived from it - if you use the object of Kitten() in a context where you expect the supertype Cat, then a Kitten() object ought to behave like you would expect a Cat object would behave. Kitten should meet the specification of Cat, otherwise strange and unpredictable behavior might occur.
4. Work in Distributed Systems

Another computer science field that deals with unpredictable behavior is fault tolerance. Fault tolerance means the ability of a (computer) system to keep on working even in the presence of failures. The system is tolerant to faults and does not break down from every failure or unexpected turn. Early computers were very much not fault-tolerant, and even in contemporary computers this property is not nearly perfect, nor has even the current level of fault-tolerance been easily come by. Liskov is one of the pioneers of creating more fault-tolerant systems. In the 90s, she focused on this research in general, and byzantine fault-tolerance in particular.
4.1. Byzantine Fault Tolerance

If a complex system fails in an arbitrary, unforeseen way, we talk about a byzantine failure. Byzantine fault tolerance is a hard thing to achieve, since the more complex a system gets, the more things can go wrong.

The word “byzantine” in this context means complex and intricate, like the political struggles and intrigues of the ancient Byzantine empire. The term refers to a Leslie Lamport 1982 paper, "The Byzantine Generals Problem" in which reliable communication with unreliable parties is discussed [LSP82]. This was considered a hard problem.

In Liskov’s and Miguel Castro’s 1999 paper, “Practical Byzantine Fault Tolerance”, the problem is dealt with one more way: with a replication algorithm that is able to tolerate Byzantine faults. This paper was very influential in rekindling interest in byzantine fault-tolerance research [CaL99].

5. Awards

In 2004 Liskov was awarded the John von Neumann Medal “For fundamental contributions to programming languages, programming methodology, and distributed systems” [AMC]. [http://amturing.acm.org/award_winners/liskov_1108679.cfm]

In 2009, she was awarded the Turing award. Liskov is the second woman ever to win the Turing award. The first one was Fran Allen in 2006. The Turing award was for her work in the design of programming languages and software methodology that lead to the development of object-oriented programming [WiLis]

This pioneering with programming languages work is what got her the 2009 Turing award.

“Liskov’s achievements underpin virtually every modern computing-related convenience in people’s daily lives [MIT09].”
6. Conclusion

Liskov got her Turing award for lifetime achievement: much of her pioneering work in programming methodology so important, that these days it has become de facto norm. Many object-oriented programming staples derive from her work - and the whole face of the field might look very different if it was not for her work.

Liskov is also an influential and inspirational figure as a woman in computer science - male-dominated as the field has been. Throughout her career, Liskov has also promoted diversity in the field by encouraging and helping female students, and making computer science a more welcoming field for women [Nao09].

These days Liskov is interested in data security and cloud computing security - or rather, the lack of it.
Sources

[CSA04]
http://www.csail.mit.edu/node/384
[9.5.2016]

[EnG] Engineer Girl: Barbara Liskov
http://www.engineergirl.org/engineers/directory/2914.aspx
No author nor date given
[9.5.2016]

[WiLis]
Wikipedia: Barbara Liskov
https://en.wikipedia.org/wiki/Barbara_Liskov
[9.5.2016]

[AMC] AMC A. M. Turing Award: Barbara Liskov. Van Vleck, T.
http://amturing.acm.org/award_winners/liskov_1108679.cfm
No date given.
[9.5.2016]

[Lis72] ACM Digital Library: The design of the Venus operating system (abstract). Liskov, B.
http://dl.acm.org/citation.cfm?id=361272
Original publication date of abstract 03/1972
[9.5.2016]

[WiCLU]
Wikipedia: CLU (Programming language)
https://en.wikipedia.org/wiki/CLU_%28programming_language%29
[9.5.2016]

[SKa] Spectrum: Massachusetts Institute of Technology. Top Prize, Institute Professor honored for innovations in computing. Karagianis, L.
http://spectrum.mit.edu/articles/top-prize/
[9.5.2016]

[WiLSS] Liskov Substitution Principle
https://en.wikipedia.org/wiki/Liskov_substitution_principle
[9.5.2016]

[InfoQ] InfoQ: Interview with Barbara Liskov: Blewitt, A.
2.4.2013
http://www.infoq.com/interviews/barbara-liskov
[9.5.2016]

[9.5.2016]

[MIT09] MIT News: Barbara Liskov wins Turing Award
10.3.2009
No author given
[9.5.2016]

[Nao09] MIT Technology Revies: Driven to Abstraction. Naone, E.
https://www.technologyreview.com/s/416783/driven-to-abstraction/
[9.5.2016]

[9.5.2016]

IMAGE SOURCES

Image 1.1. Liskov in 2010
By Kenneth C. Zirkel - Own work, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=32799771

Image 2.1.
Photo by Thomas Gearty
http://spectrum.mit.edu/articles/top-prize/