Gurdip Singh: So yes, if

Gurdip Singh: You if you want to

Gurdip Singh: So,

Gurdip Singh: Good morning everyone.

Gurdip Singh: On behalf of NSF and sighs I welcome you to our Distinguished Lecture Series. Thank you all for joining. I know this is a busy time of the semester so appreciate you taking the time to to join.

Gurdip Singh: It is with great pleasure to introduce our speaker today, Dr. Barbara cough.

Gurdip Singh: Distinguished computer scientist and great inspiration to so many of us so

Gurdip Singh: Barbara received her BSc in mathematics from University of California, Berkeley and a PhD in computer science from Stanford University in 1968 to one of the very first women among the first women to receive a PhD in computer science.

Gurdip Singh: Subsequently, she worked at MITRE Corporation and then at MIT from 1972 onwards, where she is an institute professor and has held several leadership positions in the past.

Gurdip Singh: Her he has research interests are in distributed in parallel systems programming methodologies programming languages.

Gurdip Singh: She has a long list of accomplishments
Gurdip Singh: Going to go to all of them. So among a few among those say she's a member of the National Academy of Engineering, the National Academy of Sciences. The National Inventors Hall of Fame.

Gurdip Singh: And the messages. It's Academy of Sciences and very importantly for the science community. She received the ACM Turing Award in 2009 and the IEEE vendome and metal in 2004

Gurdip Singh: So today, she's going to talk to us about our reflections on programming methodologies. And so before I turn it over to her. I just want to let our audience know that if you have questions.

Gurdip Singh: Just feel free to type them into the chat session chat window as as the webinar progresses when once the presentation is completed, I will then go through those

Barbara Liskov: All right, thank you for that introduction. Um, I was asked as part of my talk to talk a bit about my career. This truck actually has a lot built into it anyway. But I wanted to start if I can figure out how to get to the next slide, which I can't

Barbara Liskov: Oh, here we go. Sorry. Okay, so this repeats, a bit of what you heard, I grew up in San Francisco. I went to Berkeley in those days. I mean, Berkeley is a great school and in those days it was actually free to go to Berkeley.

Barbara Liskov: I started off in physics, because I think I thought that was the hardest major but I realized pretty quickly that my physical intuition is not that great. So I switched to math. And I finished up with a BA. It was not a Bachelor of Science in math in 1961 and I applied, I should I should say, by the way, that there were almost no women in my classes I can remember.

Barbara Liskov: One other woman in some of my math classes, but that was about it. This was not really the thing you
were supposed to be doing. If you were a woman at that period of time.

Barbara Liskov: I applied for graduate school when I graduated. I got into a couple of places.

Barbara Liskov: But I decided that I really wasn't ready.

Barbara Liskov: To make that kind of commitment and so I decided instead to get a job.

Barbara Liskov: And so I moved to the Boston area that was where my father's family had come from. And I went there with a friend of mine who graduated from Stanford.

Barbara Liskov: And when I arrived there, I looked for a job and I couldn't find a good job as a mathematician, which really isn't surprising because you need a lot more math than what I knew to do something interesting.

Barbara Liskov: But I was offered a job as a programmer. I didn't even know computers existed at this point. And there were computers at Berkeley, but I was not in the engineering school and I really didn't know anything about them.

Barbara Liskov: And they were hiring people like me in those days, because there was no computer science major.

Barbara Liskov: And therefore, there wasn't a pool of trained people that they could rely on. And so they would hire people like me who didn't know anything, but they thought maybe we could do the work.

Barbara Liskov: And my first day on the job they handed me a Fortran manual and a little program to do. And they said, you know, go forth and write this program and so

Barbara Liskov: I was entirely self-taught in Fortran at that job I discovered something that I really, really liked. I was really, really good at it.

Barbara Liskov: I worked at MITRE for a year and then I switched to work at Harvard, and at Harvard, I was working on their machine translation project. So that was an early AI project.
Barbara Liskov: Where they not easily believe that just in a few years, they were able to be translated from English to some other language, although at that point.

Their program couldn't even successful in our sentences.

None of this had anything to do with me because I was just a programmer and my job was to maintain a very large program written in assembler.

So think about in those days it was all printouts think about a print out about two inches thick. That was the size of that program.

This was a great choice for me. I actually did it because I liked the commute better. But it turned out to be a great choice because now I got to learn how that machine work. It was a 7094.

And so I then could understand what the Fortran Compiler had been doing and what was really going on with programs executed. Plus, I got to see

A very big program that I had to maintain and this taught me a lot about good programming practice.

I early in that time. It's at Harvard, I decided to apply to graduate school, because I was learning very fast, but it was clear that I could learn a lot faster. If I had somebody teaching me and so

I went to graduate school, I decided to go to Stanford, because I wanted to go back to the Bay Area in started there in 1963 and I went there without any support.

I didn't even really know there was such a thing, and my recollection is that walking up the steps on the very first day there I met john McCarthy and asked for support.

In retrospect, this seems highly unlikely that I would do that. But at any rate I did end up working with McCarthy, he did support me with an RA.
Barbara Liskov: In retrospect, I suspect they thought I would be working in AI because of my work at Harvard, even though I wasn't a researcher in AI.

So Stanford was great. I think there were five people admitted the year I arrived. It wasn't even a computer science department, yet I realized partway through. And by the way, I was the only woman that your so Graham came following a year.

I realized partway through that I really would prefer to be in computer systems, but I decided to stick with AI to get my PhD and then I would be able to go on and make changes in what I was doing.

When I graduated in 1968 and I didn't have any offers and good faculty position, which I naturally thought made me think, well, I'm probably not good enough.

Which is what you think when something like that happens to you. But at any rate, I decided to go back to mitre. Now, I wanted to go back to the Boston area because the man whom I married was living there and

But I went to my daughter and instead of being a programmer. I was now a researcher and this turned out to be a great thing because it mitre

mitre works for the government, as I'm sure you know, and they were doing research in system programming systems and they needed people to do this work.

And so my first day on the job. I was handed this great project to, first of all, implemented machine architecture in micro programming. And then I implemented a time sharing system to run on top of that. And then in about 1978 when that project was finished.

I was asked to look into the software crisis. So what is the software crisis, many people today don't know what it was. But the problem was, at that point in time we simply did not know how to build programs that worked.

As a result, software development efforts failed, and it was very common in the 60s, the 70s, the 80s to pick up the newspaper and see a report.

About company x, who had spent millions of dollars and hundreds of man years. And in the end, they
just had to throw out the entire system.

Barbara Liskov: Because it didn't work. Part of the problem was that they didn't have well trained people part of the problem was that they had a tendency to skimp on the hardware.

Barbara Liskov: With the sort of naive idea that the software could make up the difference. There was very little understanding of what it meant to build software and how much work. It was

Barbara Liskov: But at any rate. This was clearly a crisis, the government was very concerned because they were building quite advanced software systems. And so I was asked to think about what could be done about that.

Barbara Liskov: I should say, by the way. The name of this talk is reflections. Because when I got the Turing Award in 2000. It was actually 2008 ago for some reason in 2010 although for some reason it's called 2009

Barbara Liskov: I started reflecting on my career and I went back and thought about all these early years in programming methodology.

Barbara Liskov: So here I am in programming methodology and so of course I did what any intelligent person does I start to read the literature.

Barbara Liskov: And I discovered that program methodology was about two main topics, one was about design and the other was about program structure.

Barbara Liskov: And I'm going to tell you about it, just a few of the papers I read some places are running

Barbara Liskov: Programs for their undergraduates, where they have them read these old papers. There are many interesting old papers that students ought to know about.

Barbara Liskov: So here's the first one. I'm sure you all know this and screw Dykstra, wrote, not a paper, but just a letter to the Communications of the ACM go true statement considered harmful.

Barbara Liskov: And another thing you may not be aware of is that in those days programming was held in very, it was
not thought to be an interesting intellectual challenge and programmers were kind of dismissed as doing interesting intellectual work.

Barbara Liskov: I actually saw this in fact even into the 80s when I would visit other departments where

Barbara Liskov: The departments were controlled by mathematicians and they sort of dismissed the intellectual content of programming.

Barbara Liskov: Director was partly making a point that programming is a difficult intellectual problem.

Barbara Liskov: He's talking about the fact that in order to understand whether programs are working correctly. We need to reason about them.

Barbara Liskov: And what we have in front of us is a piece of text, but what's actually happening is you have a program running. So you have to go from this

Barbara Liskov: Thing in front of your face to thinking about the thing that's happening under the covers and the reason he thought to go to State and was bad was because it disrupts the relationship between what you're looking on the text and what's happening in practice when the program is running.

Barbara Liskov: And the way I like to think about this is that you're debugging a program, you get to a place where there's an error in order to understand

Barbara Liskov: And if you have well structured programs with you know why how statements if statements and procedure calls it's relatively easy to do this.

Barbara Liskov: But if you have a program with go to send it. And these were sometimes referred to as bowls of
spaghetti, because it was like you had pointers everywhere. It can be very, very hard to figure it out.

Barbara Liskov: This paper was very controversial of partly because people thought that without go tues maybe they couldn't write the programs they needed to write

Barbara Liskov: Because programming languages were pretty primitive in those times compared to what we have today and their control structures were somewhat lacking, but the other one was that Dykstra tended to annoy people and people were very annoyed because they said

Barbara Liskov: I write very good programs and I use go tues and that is absolutely correct. You can write excellent programs in any lousy programming language and you can write terrible programs.

Barbara Liskov: And even a very good programming language, but nevertheless Dykstra was absolutely right to go to save it was not a good idea. And we were able to manage just fine without it. Okay, use me

Barbara Liskov: The second papers by Nicholas fear. This is about design top down design, which was also a topic of interest at the time.

Barbara Liskov: And Nicholas had a an idea that you're sort of wrote an abstract program. This is like what we think of today as computational thinking you know you bribed the program into tasks and you pick up one of those tasks and think about it some more.

Barbara Liskov: And his example was the Queen's problem. He said, let's think about this problem as adding the next queen to the next column. So you have a solution that works for him columns you tried to add the queen to the next one.

Barbara Liskov: If you can do it, you go forward. Otherwise you backtrack and and continue that way. That was his example.

Barbara Liskov: We know today that top down design really is the way to go because after all, if you don't think about the problem you're trying to solve. I don't see how you can possibly actually solve it, but of course it's not a simple thing where you can just sort of March, right down like that.

Barbara Liskov: Of the third paper. And the last one is one by Dave parness parness wrote a couple of papers about this and and now he's getting into this issue of modularity, which I think is really the key issue in programming
methodology.

Barbara Liskov: And he what he said in another paper was there are these things called modules, we know we want our programs to build out of them. But we don't know what they are.

Barbara Liskov: And that was the problem at the time in this paper. He talked about the connections between modules and the point he's making here.

Barbara Liskov: Is that when people thought about these modules, whatever they were they neglected all sorts of connections. So, for example, they might think about a procedure is a module. In fact that was basically the only module mechanism, they had.

Barbara Liskov: But they might neglect the fact that the procedure was communicating with the outside world via a whole bunch of global variables. And so he saying you have to capture everything.

Barbara Liskov: About what the module is doing in order to be able to make modularity work. Okay, so I read these papers I read a whole bunch of others. And I realized that I had invented. Oh, excuse me, I want to stop for a minute and just talk about modularity today so.

Barbara Liskov: Here's what we know today a program is a collection of modules, each module has an interface described by specifications. So here we're talking about day Parnassus thing.

Barbara Liskov: And we understand that that interface had better be a complete the complete way you interact with a module in the specifications and better describe all the behavior.

Barbara Liskov: And an example of a module sort routine where the interface is the arguments and results and the specification says when you return the array is in sorted order.

Barbara Liskov: So it's a difference between what the module does and how it doesn't. There's nothing in that description about the fact that I'm using whatever sort routine. I chose to use.
Okay.

Barbara Liskov: And then we know now that we can talk about correctness images implementation correct if it meets specifications and

Barbara Liskov: This gives us local reasoning, which is the powerful thing we're looking for, provided we can be certain that the rest of the code depends only on the specification and doesn't. In fact, somehow or other interact with the module in some other way.

Barbara Liskov: So actually, in 1970 when I was working at

Barbara Liskov: On this, we knew that we wanted modules. We didn't understand what they were we understood the benefits the local reasoning.

Barbara Liskov: And independent development was maybe the most important thing at that time because you had teams of programmers and you had to give them different things to work on.

Barbara Liskov: It, we may not have understood the Modify ability quite so well. But what that means is we understood that.

Barbara Liskov: You might have to change a piece of the program. And if he hadn't developed correctly, then you could rip out that piece, replace it with another

Barbara Liskov: And the system as a whole would continue to work the problem was we had absolutely no idea with the modules were

Barbara Liskov: procedures were all that we understood and these are nowhere near powerful enough to build systems and people didn't get the connections right as harness was saying.

Barbara Liskov: Okay, so I thought about all these papers and a bunch of others. And I realized that I had invented assignments for the I'm not a design methodology. That's what I call it, but more like a modularity idea.
Barbara Liskov: And this paper was actually about design. Also, but I'm going to talk about the modularity and this was when I was working on that time sharing system that I had developed in the previous project.

Barbara Liskov: Time sharing was a new idea at the time and

Barbara Liskov: I was very concerned, I had a small group of programmers about how we were going to be able to produce this fairly complicated system in a way that was effective. And the way programs were built at the time.

Barbara Liskov: There was an awful lot of communication through global variables. So imagine that there's some huge global state and then you had these modules.

Barbara Liskov: And whatever they are, and they're all interacting with one another to that global state. And this is really not a very good idea. So what I decided to do in the development of this venous system.

Barbara Liskov: Was to break up the global state into what I called partitions each partition owned a part of the global stage and the rest of the program.

Barbara Liskov: Did not access that stage so that state was local to the partition and in order to provide access the partition provided operations procedures.

Barbara Liskov: That could be called by the rest of the program and only those procedures access to that stage.

Barbara Liskov: So what you see here actually is a very modern way of looking at modularity and it was new at the time. I wasn't inventing a methodology. At the time, I was just trying to get that system to run

OK.

Barbara Liskov: OK, so now it's 1972 and I moved to MIT. So what happened was I wrote a paper on Venus, the

Barbara Liskov: Time sharing system that I had developed in my previous project. At MITRE, and by the way, I should stop for a minute and say, the fact that I went to mitre was really
Barbara Liskov: A gift because I was able to make that move from AI to systems without all the added pressure, I would have had in a university job of teaching courses and so forth.

Barbara Liskov: So, you know, it's like a door closed and another door opens and that actually has been my experience in my career mean that's what happened to me.

Barbara Liskov: When I didn't get a job in math and therefore I got into computer science instead

Barbara Liskov: And you know if I'd gone to university, I don't know what would have happened. But this way I felt like I had a candy box at

Barbara Liskov: mitre I had these wonderful projects and I was just doing them and learning a ton of stuff. Anyway, I had written a paper on Venus.

Barbara Liskov: In 1971 submitted it to SSP and which is, as you know, the top systems conference and it was a prize paper. So sp and meanwhile

Barbara Liskov: Title nine was about to pass and Title nine, as you know, had to do mostly with athletics access for women to athletics, but it was having an impact on access for women.

Barbara Liskov: To things in universities in general. And so some universities and MIT was one of them. I would say a handful in 1972

Barbara Liskov: Decided they were looking for women on the faculty and sitting in the audience at my talk at SSP was Corby Fernando Corbett show

Barbara Liskov: And Jerry Salter was professor at MIT. He was the head of my session and they were they were looking because Jerry Wiesner who was the president of MIT.

Barbara Liskov: had told them he was interested, this kind of stuff does come from the top. And so I was invited to apply to MIT. And I went there in the fall of 1972 and here's a kind of a funny story.
Barbara Liskov: God, I can't remember his name. The internet guy.

Barbara Liskov: I'll think of his name. Anyway, he tells a joke about how I got his job. He applied to MIT. They gave me the job instead.

Barbara Liskov: I mean, look how badly. He ended up in the end. And sometimes I'll think of his name later. But this is what happened. So should get old. Okay, so I moved to MIT in 1972.

Barbara Liskov: And honestly, this was a perfect time for me to move because at monitor the way it worked was we looked for calls for proposals from the government and

Barbara Liskov: So, you know, I would have worked on program methodology for a bit. It's not fair. I would have continued working on it because something else would have come up.

Barbara Liskov: But at this point, I was really hooked programming methodology was a very important topic and I was just really interested in it, and my what I was thinking about was this question about

Barbara Liskov: We had all these really good papers and they would describe a way of doing something and they would give you an example. And you read that paper and you think.

Barbara Liskov: Boy, that really is the right way to do it. But then when you came to the programming were working on and you tried to apply those ideas you just kind of fell apart.

Barbara Liskov: And I think maybe it fell apart, partly because it just wasn't very well defined. It was a lot of hand wavy. So I was thinking about

Barbara Liskov: You know, what can we do about this, and my way of working, by the way, is I work nine to five.

Barbara Liskov: More or less. Maybe it's, you know, five or six or whatever, very intensely, and then I stopped for the day.
Barbara Liskov: But that doesn't mean and I developed this in my year at mitre after my undergraduate program when

Barbara Liskov: I decided I discovered how effective you can be, if you really use those hours and how useful it is to
not work because this gives you a chance to rest. And then you come back to your work, rest of the next morning, plus

Barbara Liskov: Your subconscious is still working and I'm still thinking about stuff and so

Barbara Liskov: Every day I might go home with some problem I've been considering and overnight, maybe it got
solved in the morning when I was thinking about

Barbara Liskov: What am I going to do today. I often discovered I had a solution to a problem. I hadn't figured out how
to solve yesterday. So anyway, at some point. Somehow I got the idea of abstract data types and it seems kind of
obvious in I did it again. Okay.

Barbara Liskov: Seems kind of obvious in retrospect I sort of saw that you could think of partitions as data abstractions.
So here's the partition.

Barbara Liskov: It looks kind of like an object, but really it's not thinking about it as the the program, the class that
implements the data type.

Barbara Liskov: That provides you with a bunch of operations, the representation of the objects is hidden

Barbara Liskov: And some of these operations are constructors and others are the operations that you use to interact
with the objects.

Barbara Liskov: But nobody had made this connection yet. So I had this wonderful you know moment when all of a
sudden I saw that I could make this connection and I understood immediately that this was very important. And the
reason it was very important is because it links modularity to design

Barbara Liskov: Because the way we designed a spike in inventing abstractions. In fact, when I was teaching my course
and how to write big programs, I would talk about a methodology that said
Barbara Liskov: invent an abstract machine with all the operations. I eat the procedures and all the data types I eat the data abstractions that you would like to have

Barbara Liskov: And then you can just implement your program using that abstract machine and then you pick up one of those abstractions and continue going, but they're abstractions. They're not just pieces of code. Their abstractions and

Barbara Liskov: And that was really important. And furthermore, I didn't think programmers would have any trouble.

Barbara Liskov: With data types because they already understood how to invent procedures so they understood procedural abstractions. This is just another one.

Barbara Liskov: And starting to think about what kind of abstract data do I need that wouldn't be an issue for them. They understood data types already in fact of course there were data types.

Barbara Liskov: data abstractions in their higher level programming languages that were implemented by the compiler and

Barbara Liskov: But in order for this to work. We're going to need something in programming languages. So I started to think about programming languages.

Barbara Liskov: And I started to work on this with a with Steve zealous who at that point was a graduate student at MIT. He was also an IBM employee and he had had

Barbara Liskov: A similar idea. And so we decided to see what we could do with it. And what we're doing now is we're trying to think about what would it mean

Barbara Liskov: To have a programming language that supports of abstract data types. And of course, we read the literature, which is pretty well positioned because I

Barbara Liskov: Was very familiar with list since I had done my whole thesis and list.
Barbara Liskov: And Steve came from IBM and so he was up on all those IBM languages, which were the major languages at the time. And of course, we knew about other languages like the alcohol family and so forth.

Barbara Liskov: Okay. So Stephen I read the literature and I'm just going to mention a couple things. This is an early paper on simulate

Barbara Liskov: Clearly, a very important piece of work we looked at this, we felt that its main point had to do with hierarchy and

Barbara Liskov: We thought that was kind of not only were interested in. Plus, it had no encapsulation. So this idea that you limit access to internal state to just the module that wasn't in there. And also, we didn't go in that direction.

Barbara Liskov: I'm not going to tell you very about very many papers at that point. This one we found very interesting Jim Morris.

Barbara Liskov: Protection in programming languages and what Jim was doing there was, he was defining rules for modularity and what he said in that paper is that

Barbara Liskov: In order for modularity to work the first rule is that code outside of module must not manage the mass not access in it or must not modify the data managed by the module.

Barbara Liskov: And that's clearly necessary because if you want, what you want for modularity independent reasoning.

Barbara Liskov: You've got to have a barrier around the modules, this is going to work if some code on the outside, you can get in there and muck with your internal state.

Barbara Liskov: Then your reasoning isn't going to work at all. So that was the first point, Jim was making. He also pointed out that really the code on the outside shouldn't even look at your internal state.

Barbara Liskov: And this is important. It's really important for it is two ways of thinking about it if the if the internal state is observed this major specification ought to be covering it.
Barbara Liskov: But another way of looking at it is your specifications. Okay, just better protected because you want to be able to replace this module.

Barbara Liskov: This implementation with another one if for some reason discover it's not good. Like, it isn't implemented fast enough.

Barbara Liskov: And if code on the outside could have observed your internals. You are much more limited in what you can do.

Barbara Liskov: Okay, so we read them. However, Jim, then went on to say, well, how are we going to manage this and he suggested something akin to encryption.

Barbara Liskov: You certify encrypt all objects coming out of the module and when they come back in. You can tell that they've been mucked with.

Barbara Liskov: And so therefore, you're in control. But of course, this is obviously not a solution that's going to fly in practice. Okay. So Steve, and I thought about this and we worked on this for maybe only six or eight months.

Barbara Liskov: And our motive record of working was we met every day at lunchtime, or almost every day and my recollection of the summer of 1973 was that.

Barbara Liskov: It was sunny every day in Cambridge, Massachusetts, which obviously wasn't true, but we met outside the lovely area.

Barbara Liskov: And then we would think independently, and by the end of that summer we had written up our ideas and submitted it to a conference that I don't believe exists anymore, but was about programming language research at the time.

Barbara Liskov: And this program made a big splash because this was an idea whose time had come.
Barbara Liskov: Okay, so now it's the fall of 1973 and I took the obvious next step of, oh, you know, I forgot to say something about MIT. I'm going to go back for a minute.

Barbara Liskov: I meant to tell you that when I arrived at MIT. So I did tell you that Title IX came along and that its impact was that universities were looking for women. And in fact, when I had applied for faculty positions in 1968, there weren't very many women in faculty positions at universities and they really were not looking for women and maybe this was particularly true in STEM.

Barbara Liskov: But I think it was true across the board. And when I got to MIT in 1972 there were 10 women on the faculty of the faculty of almost 1000 people.

Barbara Liskov: I was the first woman in computer science, computer science wasn't a department, we didn't even it was in the double E department and later the W department became Electrical engineering and computer science which it is still today. But at that point it was just double E. And there was one other woman in my department Millie Dress, a very distinguished physicist.

Barbara Liskov: I was the first one in computer science. And one of the things I noticed in a couple of years after I joined the MIT was quite a number of women.

Barbara Liskov: were added to the faculty who had previously been in research position. So, I mean, this is what you know really confident women had been doing before things started to change. OK, so now continuing on my way here.

Barbara Liskov: Showed up in my office is the way I think about them in the fall of 1973 wanting to work with me. It wasn't quite that way. I mean, Russ Atkinson, I talked to him later.

Barbara Liskov: And he said, Well, what really happened was I was my you were my academic advisor that man I advise them about what courses to take and you suggested that I joined your research group. So, you know. Anyway,
these three students were the main ones.

Barbara Liskov: It was quite a large research group though me a lot of people came to our design meetings we had weekly design meetings.

Barbara Liskov: In this picture, that's me and around 1974 and that's jack Dennis, who was a professor at MIT. And he was one of the people that helped me

Barbara Liskov: In fact, I wrote my first NSF proposal at that time. And I think jack was a co author and he certainly helped me write the proposal and jack used to come to our weekly meetings.

Barbara Liskov: Of course, Steve was still involved. Although Steve was now working on his PhD thesis, which was about algebraic specifications for data types. And then there were another a number of other people who were

Barbara Liskov: As time went by more people join my research groups and but others were interested and they would come to the weekly meetings.

Barbara Liskov: Okay, so

Barbara Liskov: But the three graduate students. I mentioned they were the main designers and it was clear that the next step, had to be to design a programming language. But here's my rationale and I had a well thought out rationale

Barbara Liskov: The nice thing about a programming language is Scott rules, it has to be precise. It is a mathematical object.

Barbara Liskov: Even though in those days it was described by a programming language manual, which often happens stakes in it.

Barbara Liskov: Or ambiguities and so the first compiler would resolve the ambiguities in one way, and the second would do it differently. And so there was a mess.

Barbara Liskov: Because this was before we had formal specifications and so forth and so on. But nevertheless,
Barbara Liskov: That showed you an example and wages hands. This is much more precise. And the other thing is programming language is a tool.

Barbara Liskov: And you have to understand what the tool, whether it really works. So

Barbara Liskov: You know, it's important that it be convenient to use. It's important signal sufficiently simple, it does the things that needs to do needs to have the right expressive power.

Barbara Liskov: And has to have good performance. It doesn't have to have huge performance. Performance. This is overrated. But it has to be good enough that you don't mind using it so

Barbara Liskov: By designing and implementing a programming language. We're figuring out what programming what abstraction really was data abstraction. And we're also making sure that it would work in practice, and although it seems obvious today.

Barbara Liskov: It was not always the time that this was going to really work.

Barbara Liskov: And I'm not going to tell you much about clue. But I just want to point out some facts about it. So you have a sense of the world at that time.

Barbara Liskov: And first of all, we were absolutely determined to do compile time type checking

Barbara Liskov: And. And actually if you look at this slide, mostly what you see as a kind of a love hate relationship with list so

Barbara Liskov: I did my thesis. It was chess games in list and I was always so furious when I would have to discover an error in the program by debugging.

Barbara Liskov: That could have been caught by a compiler. If only we had compile time type checking. So I thought, really, it would be great to have something that enforces the rules and, furthermore,
Barbara Liskov: With this whole new idea of data abstraction, we need to get our head around what it meant to make sure that it worked. Okay so static type checking heat based

Barbara Liskov: That's list and this may surprise you, looking back today, but in those days, there was a lot of controversy about Pointers. Pointers are evil.

Barbara Liskov: You shouldn't have them. Don't want to heat ridiculous and Bill wolf and Mary Shaw. We're working on our farm at the same time, this was the other

Barbara Liskov: Programming Language research project that was investigating data abstraction and they were using a non heat based approach.

Barbara Liskov: And actually it caused him a big problem because the thing about data abstraction.

Barbara Liskov: Is it has each the data type has objects, you don't actually want to think about how they're implemented at the time that you're defining what the time means

Barbara Liskov: And that means you don't actually know how big they are, and with a heat based approach. No problem. You know, all you have in the stack is a pointer and you don't have to worry about how much space. It takes on the heat.

Barbara Liskov: But if it's in the stack. This is a big pain in the neck and it cost them a lot of trouble separate compilation. This is listen to

Barbara Liskov: Because that was just the way you did it. You wrote one procedure at a time. You just kept going. And that was also very free

Barbara Liskov: And then a bunch of stuff I didn't do, because when you're in gauge internet in a challenging research project you throw out as much as you can. I decided no concurrency jack Dennis would have liked to have seen concurrency, but I decided that
Barbara Liskov: We had enough on our plate without worrying about that too. No. Go to yes because I thought my sister was right no inheritance, because that seemed like a distraction. Okay. Now, just one more slide about clue.

00:38:07.200 --> 00:38:11.340
Barbara Liskov: It took us several years to design it and implement it because

00:38:12.360 --> 00:38:26.250
Barbara Liskov: We had to innovate in many more ways than you might have expected this again has to do with what programming languages were like if the time. First we invented a mechanism for implementing data types. Those are the clusters. And that's where the name comes from

00:38:28.410 --> 00:38:39.720
Barbara Liskov: But then we had to face up to polymorphous so at that point in time programming languages didn't have any notion that you could write a piece of code, and it would work for many different types

00:38:40.320 --> 00:39:01.410
Barbara Liskov: And it kind of worked okay when you didn't have data abstraction. Because what your programming language had in it was just a very small set of types. And so maybe this wasn't a big deal.

00:39:01.860 --> 00:39:09.840
Barbara Liskov: Although, honestly, it must have been annoyed if you had to rewrite your sort routine because it had to do now with arrays of rules, instead of a race of integers.

00:39:10.350 --> 00:39:16.290
Barbara Liskov: But anyway, as soon as you have data abstraction. And you can see new types marching down the road, you know, you're going to have to write code.

00:39:16.710 --> 00:39:25.290
Barbara Liskov: You know, not a set of integers, not a set of rules. You want to be able to define the set once and then have it worked for many different types, but it wasn't an officer, how to do this.

00:39:25.590 --> 00:39:37.230
Barbara Liskov: And it took us quite a while to figure it out. And we came up with a mechanism that was mostly but way ahead of its time, like what they have in Haskell now the the Haskell type classes.

00:39:38.340 --> 00:39:48.000
Barbara Liskov: And then we needed to deal with iteration, because many data types of collections. And when you have collections. You want to iterate over them without destroying them. And so we invented integrators
Barbara Liskov: Most like what are called generators or rather, they are called generators. I forget the modern terminology that was a very nice mechanism. It was not a data abstraction. It was more like a procedure.

Barbara Liskov: And we had to deal with exception handling at that point, people didn't know what exception handling should be like there were various proposals.

Barbara Liskov: John good enough wrote a paper in 1974 I think was talking about all the different things that were available.

Barbara Liskov: You know, there was a question about, you have the resumption model versus the termination model and we had to sort of way through all of this and we came out with a very nice mechanism.

Barbara Liskov: There were other issues that we had to worry about lockdown. You know, what is the normal case. And how do you write your code and so forth. Anyway, they deal with all this stuff. We had to innovate in many different ways. All right, so we finished clue.

Barbara Liskov: In the late 70s. I showed you the paper didn't show me the paper. I think I can.

Barbara Liskov: If I didn't, didn't which came out I think around 1977

Barbara Liskov: By then we had the language defined, we might have still been working on various implementations. We use the usual trick of bootstrapping through a Lisp variants. And then after that we both the compiler and flu and we had users. And so we had a

Barbara Liskov: You know, we were up there and going concern and it was time to to figure out a new research project and I seriously thought about, well first I thought about programming language research but I decided that I didn't have any great ideas. So I didn't want to go on in that direction.

Barbara Liskov: I even thought about commercialization, but it was different in those days, I would have had to, you know, form a company, it would have had to spend all this time maintaining code that was being used elsewhere. It wasn't a research path.

Barbara Liskov: And so I was looking around and I read a paper by Bob Kahn about the ARPANET, which was still a
fairly new thing in those days.

Barbara Liskov: And he talked about the dream of distributed programs or distributed computing where you could have a program that had components running at many different machines.

Barbara Liskov: And wouldn't that be great but nobody really knew how to do it. So I thought, oh, well, there's like great research project. And so I jumped into distributed computing.

Barbara Liskov: And what I have on the slide or some of the things I accomplished, but I'm not going to talk about those today. It's a fascinating field distributed computing.

Barbara Liskov: But I continue to work on Turkey methodology.

Barbara Liskov: Oops, let me go back just hang on just a minute. I'm going to go back a couple slides.

Barbara Liskov: And but I didn't think about what I was doing as research. I was doing it mainly in the context of a course that I was development in it since it's MIT doesn't have a name. It has a number six 170 and it was our second course in computing.

Barbara Liskov: So the idea was the kids already knew how to write small programs. And now we were going to tell them how to write big programs. So it was really appropriate methodology course.

Barbara Liskov: And I taught them about module program design I taught them how to reason about correctness. I taught them about specifications much of this was done jointly with john de tag.

Barbara Liskov: John had done his PhD thesis on abstract model specs know maybe he did I forget specifications for abstract data types and, you know, we had the students writing specifications. We had them implementing.

Barbara Liskov: The abstraction function and the rep invariant. I mean, we're teaching them all this stuff and so
Barbara Liskov: And so the only thing I'm going to tell you about is one thing that came along in the program methodology area and that was the stuff about type hierarchy. So in 1987 I believe

Barbara Liskov: I was asked to give the keynote at oops LA. So, whoops. There was a new conference there.

Barbara Liskov: And another thing about what was going on and research at that time, which seems kind of odd today, but was true, then, is the East Coast and the West Coast were very far apart and

Barbara Liskov: What was going on out there was different from what was going out in the east. So in the East Coast. We are working on data abstraction on the West Coast. They working on small talk and small talk.

Barbara Liskov: Ronnie's so they were over in the inheritance area. And when I was invited to give this keynote at oops la

Barbara Liskov: I thought, oh, well, this is a good opportunity for me to look into all that work that had been going on the west coast.

Barbara Liskov: And so I started reading lots of papers on small talk and other object oriented languages that were being designed based on the work that had been done on small talk. And of course, small talk is based on Simul 67

Barbara Liskov: And so what I discovered when I read these papers was that, as you know, small talk provided inheritance and of course it was the predecessor of the modern languages we use today, just like my work on clue was a predecessor and

Barbara Liskov: And people were interested in inheritance, but it was used for two different purposes, it was us first of all to implementation techniques. So if I have an implementation.

Barbara Liskov: Of something some data type, you know, like Windows, then I could have a subtype me a subclass that implements red windows by just sort of borrowing from the code and adding a little extra stuff.

Barbara Liskov: And I have never actually been all that interested in that stuff though I don't deny that seems to be
useful for various things. One of the things that struck me, though in reading the papers on

Barbara Liskov: These languages. These auditory languages, the way people didn't talk about specifications at all.

Barbara Liskov: They were talking about the meaning of a subclass by explaining its differences in code.

Barbara Liskov: What was going on and superclass. So that was kind of interesting and very different from what we were doing in six 170 where we were working on specifications.

Barbara Liskov: I also discovered, though, that they were interested in type hierarchy and that I thought was really interesting.

Barbara Liskov: And it was clear from reading the papers that they hadn't any idea what it ought to me because I read a paper that talked about stacks and hues.

Barbara Liskov: And this paper said that each was a subtype of the other and clearly what they were saying was that they had the same methods with the same arguments and they weren't at all concerned about their behavior.

Barbara Liskov: But it's obvious that if you have a program that expects a stack and you give it a cue isn't going to work because the data, the object that you pass to it doesn't do what it expects.

Barbara Liskov: So I gave a keynote at oops law in which I gave a simple common sense definition of sub typing. I said objects of some type, should behave like those of super types. If used to be a super type methods and I mean the behavior.

Barbara Liskov: Was the point I was making clearly you need the syntax to work out. Right. But the behavior is very important. And another thing that this sentence says is that you don't worry about the entire behavior. You only worry about what you can observe. If you think you're observing a super tight.

Barbara Liskov: Okay, so I said that in a newspaper on the newsletter keynote. I did. Subsequently, write a paper about it, which I have a reference for a little bit later. This became a big deal. I wasn't paying much attention. And then one day in
Barbara Liskov: The early 90s. I think I got an email from somebody saying, can you tell me if this is a correct interpretation of the list car substitution principle. So this name had appeared

Barbara Liskov: And I all of a sudden saw that there was this huge chatter going on on the internet about this principle. What did it mean what's the exact definition and so forth, and subsequently in the 90s Jeanette wing who, as you know, used to be a director of size.

Barbara Liskov: Suggested to me that really would be a good idea if we pin down what it really meant as opposed to just relying on this very loose.

Barbara Liskov: Common Sense interpretation. And so what I have here is, I have the original paper, which I don't think is a very good definition of what

Barbara Liskov: What behavioral subtype means. And then the paper with Jeanette, which really comes down to describing what it really is. Okay. So that's really what I wanted to talk to you about. I just wanted to tell you a funny story.

Barbara Liskov: When I got the Turing Award in 2010, nine, whatever it was.

Barbara Liskov: Whatever it was, it was not the year the word officially was formed anyway.

Barbara Liskov: Whatever it was, it was not the year the word officially was formed anyway.

Barbara Liskov: What is the things that was happening was my husband was on the internet every day.

Barbara Liskov: Looking at all the chatter. And as you know, the internet as we unfortunately know today is not necessarily a very nice place.

Barbara Liskov: And there were comments and some were nice and some are not so nice. This one that I'm going to tell you about was not intended as a nice comment the comment said basically, what did she get the award for we already know this anyway.

Barbara Liskov: But honestly, this was a huge compliment because we didn't know this anyway. In fact, I discovered to my amazement. When I got the award that many of my graduate students didn't know it.
Barbara Liskov: They did not realize that there was a time before data abstraction. They thought it was always there. So,

Barbara Liskov: It's lovely to think that something I invented coupled with the work on small talk and object oriented language led us to what we know today. And that's really the end of my talk. So thank you very much.

Gurdip Singh: Thank you, Barbara Thank you for the great presentation.

Gurdip Singh: For your perspective and your reflection on abstraction and methodology and taking us through this journey through time on to how modularity and abstraction sort of incorporated into the languages and time has become common practice today.

Gurdip Singh: So, so thank you once again for the great presentation. So now we have the floor, open for Q AMP. A. So if you have questions, please go through the Q AMP a tab and write down the equations there and and i will then read through through them. So while we were waiting for those

Gurdip Singh: Questions, of course, those who are panelists Margaret and all you could

Gurdip Singh: unmute yourself and ask questions as well.

Gurdip Singh: So let me ask a question by while others are putting the equation through. So could you sort of also maybe give your thoughts on how abstraction modularity is being incorporated in the undergraduate curriculum right now and how the

Gurdip Singh: current generation, the next generation of

Gurdip Singh: Students are being trained in this you know how it has evolved over. And what do you think your
Barbara Liskov: So I can't talk too much about this because I officially retired in 2014 I haven't been teaching since then. I've still been doing research with various colleagues, um,

Barbara Liskov: The modules are not necessarily what they ought to be people are violating the rules. Encapsulation is extremely important. If there isn't something that protects your modules from the outside the whole thing falls apart.

Barbara Liskov: Modern programming languages, unfortunately, don't. Many of them do not protect this the way that they ought to do. And so they don't have that helped.

Barbara Liskov: And it's always true in a big project with the weakest programmers are the ones that break the system.

Barbara Liskov: So it's unfortunate that we don't do a better job of enforcing encapsulation, because that would be a big help, but I also think it shows that

Barbara Liskov: The students are not being taught properly and I don't know. I can tell you at MIT, that the course that I developed six 170

Barbara Liskov: Is still there. It's sort of morphed a few times, but we are still teaching our students about program methodology and modularity and so forth. But I haven't looked at those courses lately, so I don't know. And of course, the world is changing with AI. And it's very interesting.

Barbara Liskov: I mean, the world is changing in many ways. One thing that's been great is to see how

Barbara Liskov: Program verification has come along.
Barbara Liskov: I used to think that it would never be possible to verify anything except informally, and I always taught my students about informal arguments.

But in some of the later work I did like the work on Byzantine fault tolerance boy would I like to have a formal verification of the core of that system because you can break the core of the system. The whole thing falls apart. And yet, you know, important code is based on that.

Anyway, I don't think that our educational system is in such great shape because somehow I think our students come out and they don't have a good enough understanding of modularity encapsulation and stuff like that but job.

You know, and I don't have a very broad knowledge and what happens at other universities anyway.

Barbara Liskov: Anyway, I don't think that our educational system is in such great shape because somehow I think our students come out and they don't have a good enough understanding of modularity encapsulation and stuff like that but job.

Gurdip Singh: Thank you. Thank you.

Gurdip Singh: So next question here from Edgar in Holland important do you think inheritance is now, you know, others may think that it's important has been overstated.

Barbara Liskov: Well, I've never been a fan of heritage. So I've never paid much attention to it. I think it has certainly muddied the water in some ways. I mean, if you think about what happened in Java.

Their solution to the inheritance was to try and use higher polymorphism so when Java first came out, I mean, I was delighted to see come out. It was the first mainstream language that really had data abstraction in it.
Barbara Liskov: And Andrew Myers was my student at the time. And we tried to get them to put polymorphous and into the language then

Barbara Liskov: And it didn't happen. And there was a mess, you know, and they were trying to do it through inheritance which really doesn't work. And even today, it's kind of a mess and

Barbara Liskov: I don't know that anybody has gotten this whole thing straightened out Andrew Myers is still interested in this question and I don't go to programming language conferences, nor pay much attention to what's going on at them. Um,

Barbara Liskov: So let me put it this way. I really haven't changed my mind. But that doesn't mean but I know Andrew thanks entire inheritances important so

Barbara Liskov: I don't know. You know, the level. I work at today is what one of the things I gave up was programming, I

Barbara Liskov: You know you have a limited amount of time that you can spend on stuff. And so I have chosen to focus on design and

Barbara Liskov: Higher level stuff abstraction, you know, the ideas that sort of put things together. I do tend to work right down to the system level where I think about how things really work.

Barbara Liskov: But the next step down of there's the code, you have to debug it and so forth. I stopped doing that it is extremely time consuming. It's also, of course, lots of fun, but I stopped doing that, um,

Barbara Liskov: I don't know where we started with this, but

Barbara Liskov: Anyway, I think, what I'm saying is, you know, I'm out there in the trenches and so it's a little hard for me to answer that question.

Gurdip Singh: Next question is my coke Bradley's which language today represents your best thoughts and wishes for software developer
Barbara Liskov: OK. So again, this is not something I'm paying attention to.

Barbara Liskov: So it's really hard and I really have not been looking at modern programming languages. I am a little familiar with Python but that's

Barbara Liskov: My knowledge. There's several years old now. I was sorry to see that it didn't have encapsulation. I think the idea of

Barbara Liskov: You know, dynamic static type checking is a good idea. I think, you know, or maybe just separating the idea of what you write. And what you read. So

Barbara Liskov: This is something I often say early in the talk I at the time that I started working on probably methodology. People were very focused on

Barbara Liskov: Making it faster to write code. And in fact, there was even was the name of that language. The one where you could write the one liners.

Barbara Liskov: So you could write these inscrutable little programs in a very few symbols and this was considered a wonderful achievement. But the truth is that

Barbara Liskov: Readability is much more important than right ability you know code is written.

Barbara Liskov: Then you have to read it. Other people have to read it, you know, five years down the line, somebody who isn't even connected to you at all is trying to read it. Um,

Barbara Liskov: So a way to sort of bridge that gap is to make it easier to write and have the support system fill in the details. So, for example, people don't like to write down data types.

Barbara Liskov: But it's easy for the, you know, for the runtime system to fill that kind of detail in. I think that's a great direction. I'm not following these conferences, so I don't really know what the current state of the art is
Barbara Liskov: But I think that is a promising direction and is helpful. And as I said, of course, cancellation.

Barbara Liskov: Yeah. And, and I get one more, one more thing I didn't do a lot of working concurrency recently and

Barbara Liskov: Of course encapsulation is even more important there. If you don't, if your module is not in charge of how the concurrency is working inside your highly concurrent data type, you know, all hope is over. I mean, it was already over without concurrency, but it's even worse now.

Barbara Liskov: And furthermore, you don't want a programming language deciding what that concurrency is because

Barbara Liskov: As you know about the time I did the work on clue. All we knew about was locks and then after while there was optimistic concurrency control and now there's all this RC you going stuff going on, you know, read copy update stuff where

Barbara Liskov: You don't even, you know, you don't have any locks and you know this stuff is all perfectly manageable. If you just have an encapsulated module in which the

Barbara Liskov: Complicated, you know, rules are being carried out, so it's encapsulation matters, having a programming language that prescribes your concurrency control mechanism, just not

Gurdip Singh: there yet. So the next question is,

Gurdip Singh: From

Gurdip Singh: malgorzata sketches. She thanks you for the great talk. And the question. She has is the use of global variables as bad as the go to State and

Gurdip Singh: It is the use of global variables as bad as the go to state.

Barbara Liskov: I know. Can you repeat that please because I didn't quite get
Barbara Liskov: Yes, it is as bad. Now notice that, I mean, in fact, it's interesting to look back and think about algos 60

Barbara Liskov: Okay so algos 60, you know, it had this great innovation. It had these inner blocks and the variables inside the inner block could not be accessed from the outer block. So that was good.

Barbara Liskov: But on the other hand, it was all set up so that you would communicate through the variables on the hour block, you don't want to do that at all. You really want

Barbara Liskov: You have your modules, they're independent, but notice that there is a kind of global variable and all of our programs and that is the file system.

Barbara Liskov: You know the database their global the global variables are still with us. I think they're using a much more disciplined fashion now than they once were. But they, we did not get rid of them. And I don't think you can

Gurdip Singh: Phillip show on past. So this is a question which you touched upon briefly earlier to the question I asked, it says, how do you approach teaching of programming modularity. What kind of examples or two. Would you use

Gurdip Singh: Could you give us some idea.

Barbara Liskov: Well, I can tell you what I used to do.

Barbara Liskov: It actually probably the person you who might have the best ideas. So I have a couple of colleagues who are basically teaching

Barbara Liskov: The kinds of stuff that was developed in six 170 so john doe tag.

Barbara Liskov: Has an introductory course and trainee given to us. Also, and they use examples, and I don't know what those examples are but they're very different what I used to use. I used to use quick index as an example. And it was, you know, a sequential program, but
And I think it was kind of boring for the students. You know, so you would use an example. I mean, first of all, I taught them about the methodology, the one I mentioned earlier about inventing abstractions and imagine you had an abstract machine.

And then I would illustrate it by doing a design, starting from a problem statement and going through now of course what I've been doing in the last 30 years 40 years.

Is thinking about systems which are not sequential systems. And so, you know, they're you're thinking more in terms of majors, you start with your major subsystems.

And I might work more from that level today if I was teaching that course I would think about. It's an interesting question. You know, you have to walk before you can run.

And I tend to think in terms of distributed systems and highly concurrent systems.

So, I think it's a great question. John has a book and I bet you he's got some interesting examples in that book that he uses for teaching. So that might be good place to look.

Quick inject quick index is a very good example, but of course I haven't actually taught that course since the 90s. So it's been a while.

Next question is from

Gurdip Singh: when when when from University of Georgia. So thank you for the wonderful insightful task talk and two questions. So the first one is, do you think abstractions also removes some potential optimization opportunities that can be exploited by components.
Barbara Liskov: Of abstraction is absolutely not in the way of what the compiler can do, in fact, a very powerful technique. Well, remember in line substitution. So, you know, that's a simple optimization technique where the compiler exposes to itself. The code of the procedure and then does all sorts of manipulations of the code so that avoid the cost of the calls and so forth and so on. And you can get rid of variables and all sorts of stuff and then

Barbara Liskov: If you change your idea you re implement procedure, no problem. You just re compile the deal. So in fact, I think it's quite the opposite. You know abstraction.

Barbara Liskov: Is something that you think about, but you don't actually reason in terms of the code that runs on the machine.

Barbara Liskov: Although of course if the compiler is doing stupid things you may run into a little problem. You know, like all those problems in the C plus positive either where it doesn't

Barbara Liskov: It gets a little bit over ambitious about what it's doing with your code. So the compiler had better be doing his job properly, but really the idea that you work in an abstract level and then the compiler comes in and does all this manipulation. That's a great use of abstraction.

Gurdip Singh: Yeah. Yeah, I agree. Yeah, I'm for the second question is big merger. Well, it says, given the recent advances in machine learning. Do you think computers will eventually be able to program themselves.

Barbara Liskov: Yeah, isn't that an interesting question. Yeah, let me put it this way. Well, first of all,

Barbara Liskov: You know, there's a lot of issues with what's going on in our world today and quite a few of them are due to machine learning.

Barbara Liskov: So I think we should be putting our energy into trying to, you know, do whatever technologically, we can do about fake news and

Barbara Liskov: I mean, just think of the problems we have to face of machine learning gives you
Barbara Liskov: 95% correctness. I mean it.

Barbara Liskov: Today, it doesn't give you 100% programming is very precise. So I don't think we're ready for prime time here, although I know I have colleagues who think this is cute and

Barbara Liskov: You know, somehow you get an almost correct solution. I mean, I suppose, if you're in an application we're almost correct is good enough, it might work.

Barbara Liskov: But that seems like the big deal to me. The fact that we don't quite get all the way there. And we want to get all the way there so that the code really works. And that's a big problem.

Barbara Liskov: But, you know, the issue about what's going to happen to software. What's going to happen to programmers.

Barbara Liskov: You know, our world is changing due to computer science and

Barbara Liskov: Who knows where we're going.

Gurdip Singh: So it's to have a show from

Gurdip Singh: lakin down live in June. So this is, this is not a question actually. But would be interested in your talked about this is the

Gurdip Singh: Modularity was developed about the same time in linguistics. In the 1970s, for the design of grammatical systems and Jerry photo famously promoted the idea for the design of mind in the modularity of mind in 1975

Gurdip Singh: My impression is that these ideas were not pursued with the same rigor, as they were in computer science.
Gurdip Singh: So,

Barbara Liskov: Yeah I you know I That's way beyond my level of expertise. It was just striking me

Barbara Liskov: You know, when we speak. We can be ambiguous.

Barbara Liskov: When we speak to a computer. We can't be ambiguous.

Barbara Liskov: I'm not sure that there is that much of a connection here, but I don't know its way out of my

Gurdip Singh: School into the equation. This another one on how we teach our students to design mode. So how do we teach our students to design more effectively. So I know you've already talked about it and you may have

Barbara Liskov: Well, I had the students work in teams. So I started them off working on very small projects where they still had to design, but it was small, they can do it as a single person.

Barbara Liskov: And then later in the course. I had them work on teams on a larger project where they really had to subdivide and come up with a

Barbara Liskov: System that worked. And that was, I would say only

Barbara Liskov: Hardly successful and the problems were not well first of all, I should say. I actually don't believe you can teach design.

Barbara Liskov: So, I believe, maybe you can teach small scale, you can teach the principles of design, but not everybody has the ability to design it it's a it's a skill. It's a, it's almost artistic, you know, because the design.
Barbara Liskov: A good design has kind of elegance to it. So all I could teach was the principal design. I could try to get across. You know, the modularity work. So if you came up with a good design, it would be better than if you didn't come up with a good design.

Barbara Liskov: The, the problem with that team project also had always had to do with people. It didn't have to do with, you know, and maybe these problems exist in industry to but teams didn't always work. And it was very hard to

Barbara Liskov: Monitor this and to help out the students, you know, you have teams where they'd be one really strong programmer and they would just take over everything, and nobody got to do much.

Barbara Liskov: Or you could have. There were lots of issues with the women you know they had teams where there was a woman on the team, and she was sort of sideline and

Barbara Liskov: So there were lots and lots of issues with running teams, but since large projects and modularity, have to do with teams and something you have to go there because

Barbara Liskov: Throttle you get something big, you know, a small program you write yourself. Well, you probably made more notice that it was a problem. So,

Gurdip Singh: Yeah, I think. Yeah. Modularity maps nicely our teams map nicely into modularity and they go ahead

Gurdip Singh: So car going on the coupon offers a question, what are your perspective on how to address the security caps that software code seem to have overall

Barbara Liskov: Yeah.

Gurdip Singh: Yeah you know this is hard. My people who aren't my friends who are
Barbara Liskov: Working in this area, you know, they think about it as a game, you know you appeal the security of your system.

Barbara Liskov: But the hackers are very very motivated to break it, it would be if we use better programming languages that might help you know the lot of the

Barbara Liskov: Ways, they did the vulnerabilities are often do to problems in the software that could have been avoided if we'd had a better programming language to begin with. It would be good. Maybe if we didn't have so much legacy code.

Barbara Liskov: The real problems are human interface problems, you know, how do we get people to not click on a link that they should not click on

Barbara Liskov: So in answer to your question. You can see I don't have any solutions.

Gurdip Singh: Yeah, and it's also goes back into how do you teach them to write more effectively and

Barbara Liskov: Yeah, but yeah. But I think what I'm saying is, even if you had a perfect system if there were any way that clicking on a bad link is going to be a problem. And you know, I mean, you can't help with that being a problem, you know, so people who have worked on.

Barbara Liskov: User Interfaces and trying to figure out ways to design better interfaces and and nobody has a good solution, you know, the

Barbara Liskov: You know, Pat. People don't like to use passwords that are complicated captures on they'll be useless because machine learning will be able to solve them anyway and and they're annoying so

Barbara Liskov: I think, you know, we, yes, we should get our act together in the in the

Barbara Liskov: In the systems and build a software that actually works. But there's even if we did that. There's a whole

file:///C/...ons%20on%20Programming%20Methodology%20by%20Barbara%20Liskov%201606700871/CISEDLLiskovtranscript.txt[12/18/20202:08:56 PM]
other area of problems, having to do with user interfaces.

444
01:13:58.260 --> 01:14:00.330
Barbara Liskov: And so it's a very hard problem.

445
01:14:02.820 --> 01:14:03.300
Gurdip Singh: Who

446
01:14:04.950 --> 01:14:17.370
Gurdip Singh: Knows, we should move from Johns and he asks, what would be the most disruptive phenomena in programming or computation in general. Since 2004

447
01:14:18.750 --> 01:14:20.520
Gurdip Singh: Your retirement, in your opinion.

448
01:14:20.730 --> 01:14:31.350
Barbara Liskov: Oh, well, what's going on the machine language because they really mean machine learning. Absolutely. So when I was at Stanford. I actually was interested in machine learning.

449
01:14:32.040 --> 01:14:47.910
Barbara Liskov: But machine learning in those days was you write a program that thinks like a person. And that really wasn't very effective. And one of the reasons I left AI was because I thought this is never going to work, or it's too hard for me or something like that.

450
01:14:49.680 --> 01:14:58.050
Barbara Liskov: But, you know, things are totally different now. And what you can do with machine learning is quite amazing. So I mean, things have really changed. There's no doubt about it.

451
01:15:02.760 --> 01:15:16.560
Gurdip Singh: So I see no more questions. So I have one question. So I know that if you talked about, you know, in terms of looking at program. We go to statements and all the abstractions made it difficult to

452
01:15:17.700 --> 01:15:33.630
Gurdip Singh: Orange or during CES 70s, 80s, the hardware itself a sequential where it was easy to sort of see how the program execute and debug. So I'm just curious about your thoughts on how the advances in hardware have

453
01:15:34.800 --> 01:15:35.250
Barbara Liskov: So I

454
01:15:37.380 --> 01:15:42.270
Gurdip Singh: The reasoning about them, or even how you write and all the

455
Gurdip Singh: abstractions that you may have. So

Barbara Liskov: So I'm I don't quite understand your what your are you asking me about how computers run today versus how they used to run

Gurdip Singh: Know, or how the advances in the hardware, the multi processing systems that we have and all impacted abstractions.

Barbara Liskov: I don't think they've impacted abstractions, or at least in my experience, I mean, I, I've spent a lot of time in the last five or six years working on multi core computers. So I'm talking about highly concurrent programs.

Barbara Liskov: abstractions are our salvation there. And you know what I said earlier about modularity and encapsulation being the, you know, the thing that really matters. Now I don't have experience.

Barbara Liskov: With a lot of the other stuff that's been going on. So I don't. I really can't say anything about that.

Barbara Liskov: But I can tell you that, as far as distributed computing is concerned, as far as highly concurrent programs are concerned abstraction. If anything is more important than that it isn't a sequential simple sequential world that I was working in. When I first started working in this area.

Gurdip Singh: So,

Gurdip Singh: See no more questions. So again,

Gurdip Singh: Thank you, Barbara once again for the great presentation.

Gurdip Singh: Really be thoroughly enjoyed it, and I'm sure all of our audience has also enjoyed the presentation. So thank you once again. So I just also want to remind everyone that at four o'clock today, at least for all of the NSF offers on the zoom call that we have will have office hours.

Gurdip Singh: With Barbara. So please do join us at four o'clock today, so thank you once again.

Bye bye.