MIT CSAIL Alliances | Daniel Jackson Podcast

Hello, and welcome to the CSAIL Alliance Podcast series. My name is Steve Lewis. I am the assistant director for Global Strategic Alliances at MIT's Computer Science and Artificial Intelligence Lab, better known as CSAIL. In this podcast series, I will be interviewing principal researchers at CSAIL to discover what they are working on and how it will impact society. Today on our podcast, I will be speaking to Professor Daniel Jackson.

Daniel Jackson is a professor in the Department of Electrical Engineering and Computer Science as well as a MacVicar fellow. He leads a software design group and he received an MA from Oxford University in physics and an SM and PhD from MIT in computer science. He was a software engineer for Logica UK Limited and assistant professor of computer science at Carnegie Mellon University before joining the MIT faculty in 1997. His broad interest in software engineering, especially in development and design methodology, and safety critical systems, and through his research in these areas, he hopes to make software ultimately safer, more understandable, more reliable, and more robust overall.

Professor Jackson has won many awards, including the ACM SIGSOFT Outstanding Research Award for his pioneering work in software engineering, the MIT Martin Luther King Leadership Award. Daniel is an author of many books, including *Design by Concept, A New Way to Think About Software* as well as a book on photography called *Portraits of Resilience.* Daniel, thank you for your time today.

--So much, Steve.

Can you tell our listeners a little bit about the focus of your research and some of your bold aspirations?

So for the last few years, I've been working on a radical new approach to the design of software, which is predicated on the assumption that if you want to build great software, software that is usable and reliable, and secure, then not only do you need to have a deep and clear understanding of what your software does and why it does it, but you need to embed that understanding in the very structure of the software system. And this is actually not a new idea. Fred Brooks, in *Mythical Man-Month,* a very influential book, wrote about the idea of conceptual integrity.

And when he wrote an afterword in the 1995 edition of the book, 20 years after it had originally come out, and went back and talked about which ideas in the book were most important, he said, I still believe that conceptual integrity is the most important idea in the book. But he never told us what conceptual integrity is. And that's something that I've been trying to figure out. And I've developed a method of designing software around this notion of concepts that allows you to identify the concepts in a system, to analyze them systematically, and to put them together in a productive way to solve the kinds of problems that we want to solve. And this gives a radical and more flexible and simpler view of how to organize software and how to ensure that the software actually meets the needs of its users.

And what do you think about current design methods or strategies for developing software? Like agile for example, or is that a system that needs to be improved?

Yeah, I think that approaches like agile have certainly brought, in some respects, a revolution to software development, but they've been a process revolution. So what agile tells you is that the way you think about developing your software in phases should change, that it should be-- the phases should be shorter they should be more incremental, they should be more driven by immediate customer needs, and they should be responsive to feedback. And DevOps, of course, even takes that into the deployment phase. But agile really did nothing to rethink what software itself was really all about, how we would actually design software. At the end of the day, when you sit down to think about the needs of the users, and when you sit down to design the software and decode it, we're not really that far ahead of where we were in the 1970s.

Really? How so?

Of course, in the last 30, 40, 50 years since the advent of computing, we've seen some amazing advances in programming technology and in programming methods, and we understand programming much better now. And there are some remarkable ideas that came from research and from industry. You just have to look at ideas like the notion of abstract data types and objects, which completely revolutionize the way that we build software. Or ideas that are so fundamental, people don't even realize that they were actually invented. So now we have this assumption that you can talk about dependencies between different modules. This module depends or relies on this module. And the structure of the system has to be organized so that these dependencies are tamed.

Well, that idea was invented by David Parnas in the 1970s. Before that, nobody had heard of such an idea. So we have these great ideas and they've suddenly influenced the way we program. But we still, in many respects, missed the big picture. And so what I see is that when we think about usability in particular, you have visual designers and user experience designers, who tend to focus primarily on the skin of the user interface, and what the user interface looks like on its visual design, on the microscopic interactions you have with that interface. And then you have, of course, a mountain of software engineers and software developers who work on backend stuff. They build servers and databases, and elaborate architectures that lie behind the user interface.

But there's almost nobody bringing these two together. There's nobody thinking about what it actually means to design the fundamental structures of software to serve the needs of the users. And because we don't have that, we, so often, end up with systems that have remarkable technology, that powers them, but are still almost unusable. Or at least their usability is not in line with the power of the technology that lies behind it. I think you only have to look at something like Zoom, as an example of this. Zoom is one of the best pieces of software out there. And I think Zoom's popularity and success is well deserved.

And undoubtedly, that's in large part because Zoom's video technology is remarkable. Even over a fairly poor internet connection, you can get a pretty good picture of someone talking. And Zoom is remarkably reliable and effective. And its interface-- its user interface isn't bad either. But we've all been caught by these situations in which breakout rooms don't behave the way you expect them to, in which chat messages get lost. And Zoom itself, as you may recall, got into a lot of trouble at the beginning because of its security problems.

And so we're beginning to realize there's this disparity between the underlying technology, this remarkable networking video technology on the one hand, which so much research and effort is poured into. And on the other hand, this useability technology, which is often just neglected. Of course, companies know that it's completely fundamental, but researchers haven't paid so much attention to it.

I see. And what's your take on using machine learning or AI to help develop better software or more usable software? Is that some place we're at now or is that more so in the future?

I don't honestly think it's going to help. I think that software, to be more usable, has to be more simple. And generally, AI and machine learning are only going to make things more complicated. So I think that there are fantastic applications for AI and machine learning. Probably, we've seen, for example, the amazing explosion of applications in image recognition. And that has many domains in which it can be very useful, from a medical diagnosis to sorting your family photographs. But I don't see machine learning really being any help-- any major help in software design.

Interesting. Can you tell me a little bit about the Wild Card project and what's that all about? Wild Card is one in a series of projects that we've been conducting in my research group to really rethink the fundamental paradigms of programming. And Wild Card poses the question, what would it look like if you could take your favorite web apps and, as an end user, not as an expert programmer, customize those web apps yourself, change their behavior? And so what Wild Card allows you to do, for example, is to open the Airbnb page, to look at it and say, wait a minute, Airbnb has prices but it doesn't let me sort listings by price. I'll add that. And you then add that and you can sort them by price. Or maybe, I'd like to write some notes on these Airbnb listings so I remember which ones I like, share them with my partner who's going to go on a trip with me. You can add that feature.

And this can all be synchronized with the Airbnb app so it actually shows in the Airbnb web page. We've shown you can also connect to other web services. So maybe, for example, I'd like to know how walkable the neighborhoods are, that these Airbnb apartments are in. So I can add that as well, just through a little gesture in the user interface. And all of this builds up a different user experience using the web app that is completely invisible to the web service itself. So Airbnb doesn't even know you're doing it. They don't know you're annotating their records, they don't know how you're sorting them, they don't know that you're attaching walkability metrics to their listings.

All this is happening just in your browser. So that's what Wild Card allows you to do. And we see it as a kind of wedge in the door of a new kind of software development that's much more democratized, much more open and flexible, customizable, and in which the boundary between the professional developer who writes code and the end user who just uses the code, is eroded so that the end user can begin to really shape the applications that he or she uses.

So this requires no sort of expertise from the end user perspective and could tailor their experience just with simple queries in plain English.

It's not quite like that. I would say know we haven't got to that point yet. It does require some expertise but it requires the kind of expertise that many end users already have. So if you can create a basic spreadsheet, in fact, Wild Card uses a spreadsheet view. If you can create a basic spreadsheet and enter some simple formulas, then you can use Wild Card. So we think that there's a very-- there's a gold mine basically of, if you like programming opportunity for end users who all day long are really doing programming activities like recording macros, writing spreadsheet formulas, and so on, but they don't think of themselves as programmers.

And most of the programming environments would be completely inappropriate and unusable to them because they would require them learning all kinds of complex languages and all kinds of complex environments. So we get rid of all of that and we make this end user customisation experience something much more in line with what end users are already doing in the little programming tasks that they're doing every day.

Fascinating. How far away do you think this is from commercial viability?

Actually, I think Wild Card is pretty close. I think a bigger question with wild card is who would pay for it. I think that we're getting to the point where we will be able to customize a large variety of websites and I think it could be very powerful and useful to end users. But what we haven't really worked out yet is who would want to pay for such a thing, because end users are notoriously stingy. Understandably in wanting to pay for even the website as they do or many of the facilities that they use online.

So is this available as open source or is this more close to propitiatory.

It's available and we're continuing to develop it, where actually-- it's very exciting because until now each wild card deployment on a particular site depended on a little bit of programmer produced code to actually do the scraping on the website. So you had to download what we called an adapter. And this adapter will be written once for Airbnb, for example, and then it would allow the end user to customize the behavior of Airbnb in whatever way they chose. Now what we're doing is, we're making the scraping itself and use a programmable. And that's something that I expect that will have we already have it up and running. But I think it's going to be in really good shape by this summer. So I'm very excited about that.

Wow. That is exciting you're also leading another project called Alloy can you talk about that?

Sure Alloy is a design language that actually was originally developed in my group in the late 1990s. And we worked on it, I would say until the early 2000s. The key idea an alloy was that you could write down the essential aspects of a design in a very succinct and intuitive way and then you could run analysis on it, that would automatically find bugs in your design. And Alloy is a very radical idea because it expects you to think about design in a different way. It expects you to think about the very essence of your design and to record it in this simple logical language.

And so I Alloy has taken some time to grow. What's been very exciting for Alloy and I suspect the reason you asked about it, is that even though the Alloy project was officially a project that we started many years ago, Alloy is actually just growing and growing in its popularity and a bunch of other research groups in other universities have taken up the mantle of Alloy and have developed some very impressive extensions of alloy, most interestingly a tool called electrum which is so impressive that we're actually incorporating it into the Alloy code base.

An alloys being taught in many universities, it's taught for example in a course at Brown University and that is called logic for systems which teaches students who want to learn how to build computer systems to learn how to express their designs and analyze them and Alloy. And it's been used in a number of companies to find critical bugs in designs before those designs were even coded. So Alloy is a very exciting project and it now has a board of about 20 members who helped lead it from different universities and industrial labs and it's an ongoing project which I expect to continue for many years.

Excellent. How can our listeners find out a little bit more about Alloy?

They can Google it. They can go to alloytools.org, they can download it, if they just Google our language they should find a bunch of things. We're actually in the midst of revamping the website. So hopefully, soon it will be more accessible than it currently is.

Great. Let's switch gears a little bit and talk about self-driving cars. Can you tell listeners about your collaboration with Toyota research?

Sure. Well, we've had a very exciting collaboration with researchers at the Toyota Research Institute for the last couple of years. The question that we've been addressing is in the presence of all the automation and sophistication that is required to make a self-driving car. How can you ensure that it won't go wrong? How can you ensure that catastrophes won't happen because of flaws in this very elaborate and complicated system? And the basic idea that we've been pursuing is to build something that we call an interlock which essentially monitors the behavior of this very complicated system with a much smaller and simpler piece of code. And if there's a problem that it detects, it intervenes.

So does this get into the realm of having the artificial intelligence in the car make these moral decisions, when it comes to a potentially hazardous scenario?

Well, I wish I could say that the artificial intelligence systems are making moral decisions. I think that if you look at the kind of accidents that have happened with self-driving cars, they haven't actually been that a car face the well-known philosophical trolley problem, that it had to decide whether or not to plunge ahead and kill the driver or swerve and kill the child on the sidewalk. Instead the moral issues that have come up have normally been that the self-driving car is simply not safe enough and it's lack of reliability raises profound moral questions about whether the deployment of such cars is responsible.

We saw this with the Uba accident in Tempe, Arizona in which an Uber taxi killed a woman who was crossing the road, we've seen multiple accidents with Teslas for example, in which the car has plowed into a concrete barrier on a highway or a stationary fire truck in the middle of the highway and so on. And all of these accidents have happened, not because of some imperfect more decision making mechanism, but simply because autonomous cars are just not yet reliable enough to not crash.

It's just interlock get at the heart of this particular issue?

That's what it aims to do. I mean the idea of the interlock is to bring a much greater degree of confidence to our deployment of self-driving cars by essentially, you can think of it as kind of an envelope that it wraps around the behavior of the car and says, the car can't stray from certain behaviors. So if the car is about to do something crazy, that then the interlock will prevent it from doing it.

I see. And when do you believe that level five autonomy will be achieved by autonomous vehicles?

Well, I have to say I'm not very hopeful. I think it's going to be a very long time. I think that much more likely is that we will simplify the driving infrastructure, we will have not only better markings on roads, but we'll have probably electronic markings in the way of beacons and communication between cars and that if we build this infrastructure it will become much more plausible to have cars traveling autonomously, because they'll be able to rely on this much richer source of information and they won't fall prey to problems to do with, for example visual recognitions being misunderstood.

I happen to agree with you. I think it requires the whole ecosystem, the whole transportation system to be smarter in order for this particular level of autonomy to be achieved. Switching subjects again. Tell us about your photography book, *Portraits of Resilience.* What inspired you to create it?

Wow. Well, *Portraits of* Resilience came out of actually a very dark period at MIT. Back in about 2015, we had experienced a terrible, tragic spate of suicides at MIT. And the weight of depression and anxiety, particularly amongst our students was just debilitating. And I had noticed increasingly large numbers of students unable to do the work in my course. Who would come and talk to me about the problems the challenges they were facing with depression and anxiety.

And it seemed to me that one of the things that we needed to do at MIT was to move away from our traditional notion of anonymity and we needed to be much more open and honest and engage each other about these issues. So I had this idea that maybe I could make a gallery of people who'd experienced the problems of depression and anxiety and other mental health issues. And that these people would include not only students, but also faculty and staff members. And that by making this gallery, I would essentially be putting up images not of people who would be thought of as victims of these conditions, but people who could be thought of as inspiring heroes, people who had struggled with these difficult things and could teach the rest of us how to cope with them in our lives.

And this eventually led to the book*Portraits of Resilience,* that was published in 2018 by MIT Press and it included an array of photographs with verbatim interviews, with about two dozen members of the MIT community. Who told really remarkable stories about their lives and the challenges they faced and how they came to terms and in many cases overcame the challenges that had previously been holding them back.

That's wonderful work and our listeners find this book. So there's actually, we have a website portraits of resilience.com the website has excerpts from the book, it has pointers to where you can obtain the book if you'd like to. And it also has a subscription link for a postcard, a virtual postcard that I send out every month, which includes a photograph and some inspirational words.

Wonderful. And in closing Daniel, can you tell us your experience with Cecil alliances and some of our industry partners. We mentioned Toyota before is there anyone else that you've worked with in the program that you'd like to mention?

I've been just extraordinarily impressed by the amazing range and number of companies that are partnering with us through these alliances and the vibrancy of the programs and I've had the opportunity to meet our colleagues from these companies, who've come through MIT. Sadly not recently, all our meetings have been online but hopefully that will start again. And I've had the opportunity to participate in the Alliance's wonderful annual meetings. And so I think that the alliance plays just a crucial role in connecting us to industry, in exposing industry to the ideas that we're developing and exposing us to some of the important challenges and problems and ideas that come out of industry. So I think it's a hugely synergistic and mutually beneficial system and I'm thrilled to be part of it.

Great, we seem to think so too and we appreciate your time today Daniel. Thank you very much and we'll be talking to you take care man.

Thanks so much Steve, bye bye. If you're interested in learning more about the Cecil Alliance Program in the latest research at Cecil, please visit our website at cecil.mit.edu and listen to our podcast series on Spotify, Apple Music or whatever you listen to podcasts. Tune in next month for a brand new edition of the Cecil Alliance podcast series and stay ahead of the curve.