Menlo Park CA
41:17
m4v

Brian Gerkey

An interview conducted by
Selma Šabanović
with
Peter Asaro

June 9 2011

Q: Tell us your name and where you were born and when.

Brian Gerkey: My name is Brian Gerkey and I was born in December 1976 at the Hahn Air Force Base in West Germany.

Q: And did you go to school there or stay there for –

Brian Gerkey: We only stayed there for a couple of years before moving on. My dad was in the Air Force, so we moved around a fair bit. Lived in North Carolina, Southern California, lived in Madrid for a couple of years. Ended up in the South. Went to high school in rural Georgia.

Q: How did you get interested in science and engineering?

Brian Gerkey: I'd always been interested in computers. I knew from pretty early on that I wanted to do something with computers. When I was in third grade my dad brought home an Apple 2GS, I think it was. It was the kind that it didn't have a screen. It would fit it, but it could just barely fit inside a briefcase, so he would carry it to and from work and when he brought it home, he'd hook it up to our TV and then we could play games on it and write programs in BASIC. So when it came time to go to college, I said I wanted to go into computer engineering. I remember in college not being able to connect at all with people who couldn't pick a major. When I applied I wrote I'm going to do computer engineering and then four years later graduated with a degree in computer engineering. The whole idea of wandering around and not knowing what you wanted to do was totally foreign to me.

Q: And so where did you go to college?

Brian Gerkey: Tulane University in New Orleans. I got my degree from a department that no longer exists because in the wake of Hurricane Katrina, they closed the school of engineering.

Q: And did you go on to graduate school after that?

Brian Gerkey: I did. So I got interested in robotics at Tulane. There was a young assistant professor there named Jim Jennings who had just come out of Cornell out of Bruce Donald's group. So he started a robotics lab at Tulane and some of my friends were helping out in his lab. We were all undergrads. He had a couple of grad students as well and just so as an undergrad I started to sort of be on the fringes of a group that was hacking on robots in his lab. Ended up

doing my senior thesis under his supervision with robots and when it came time to graduate, I looked at my options which were to go get a real job or stay in school and I've done real jobs before and that didn't seem like that much fun, so applied to graduate school and ended up in the interaction lab at USC down in Los Angeles.

Q: And what were some of the projects that you were doing in Tulane with the group?

Brian Gerkey: It was mostly on multi robot coordination which turned out to be a theme of the work that I did in graduate school. So the particular – let's see. My thesis was actually on handing off tasks between robots. So the idea is that one robot was trying to do something and these were pretty simple mobile robots so that the thing that it was trying to do was like push a box across a room. So it would get the box pushed partway across the room and then be called off to do something else and hand off that task to another robot which would come and finish it, but also we did a lot of work on the middleware. So these robots came with not very good software, so we spent quite a lot of time building up the software infrastructure that allowed you to program the robots to do what you wanted them to do.

Q: What kind of robots were you working on?

Brian Gerkey: These were RWI B14s. They were trashcan robots that stood about this tall, this big around.

Q: And what year did you finish at Tulane?

Brian Gerkey: Ninety-eight.

Q: Ninety-eight. And so what kind of robotics was going on at USC when you got there?

Brian Gerkey: So when I got to USC, I joined Maja Mataric's group there. She had recently moved there from Brandeis and before that she was in Rod Brooks' group at MIT. So she was really at the forefront of behavior based robotics, in particular with multi robot systems. So the focus of her lab was how to program groups of robots to get interesting behavior and that was very much in line with the kinds of things that I had been doing and that I was interested in. So I joined up there. At the time she had about four of these robots that were called R2s. They were the follow-ups to the R1s that she'd used in her thesis work and we were programming those to do various things, but we were starting to get more sophisticated robots. We got the Pioneer 1s and Pioneer 2s from Active Media and those became the sort of workhorse mobile robots in our lab.

Q: Were there specific applications that you were interested in?

Brian Gerkey: Well, in academia, applications are often secondary or tertiary or not in the picture, so for me, I was more interested in the science of organizing teams of robots and that was something that I always struggled with was what's a motivating application. If I'm going to organize this group of robots to do something, what should I have them do that actually makes it worthwhile to organize them? We tended to use placeholder tasks. So if I imagine that these robots might be useful in a warehouse, well, I'll have them push a cardboard box across a room and say that that was kind of like material handling or if I imagine a security robot, well, I'll have a robot that patrols and goes to fixed points around a pen and the idea is that you pick those tasks and they have all the same features as that task that you think is useful in the future will one day have.

Q: And what were some of the technical and scientific challenges that you were working on in those tasks?

Brian Gerkey: They kind of fell into two groups. So on the one hand you had the scientific challenges. So if you're coordinating groups of robots then the biggest issue is scalability. How do you design a decision-making process that will get the behavior you want from a potentially very large group of robots in a reasonable amount of time because you know from the start that it doesn't make sense to take in say all the sensor information from all the robots and then centrally plan what they're all going to do and then disseminate commands for problems of computational limitations and communication limitations that you just know that's not going to work, so you have to come up with algorithms that make that decision process efficient and that was the scientific part of what I was doing. I borrowed a lot from operations research and using linear programming techniques, sort of things that have been around for half a century in a different context. They turned out to be useful in these multi robot problems as well. But then there was a whole other set of problems which was more on the technical side which was just keeping the robots working and making them programmable so that the robots often arrive with software that was subpar and the hardware was flaky and so a lot of the issues you dealt with were just keeping the robots up and running and providing a development environment that would allow you to be productive.

Q: So what was your thesis work on? What was it called?

Brian Gerkey: My thesis was called "On Multi Robot Task Allocation." I started with on and felt kind of old-timey. So it started out as trying to build a system to do task allocation for robots as a way of getting robots to coordinate. So I have a bunch of things that I want done. I have a bunch of robots. Who should do what and how should that evolve over time? I ended up building one or two such systems and then I had an epiphany one day when I had just produced a

video of my robots collaboratively pushing a box across the room and there was this particular sequence where I wanted to show fault tolerance. So the two robots are pushing this box and I run into the video and I pick a robot up and take it out and then the other robot realizes that and automatically takes over, so it starts pushing the right side and the left side. It does it by itself and I can bring the other robot and I can put it back and then they act as a team again. I thought that was really cool.

This was in 2001 and then not long after that I found a video from Lynn Parker's thesis work again with Rod Brooks at MIT from 1994 in which her robots did exactly the same thing. She had two robots pushing a box. She came in, picked one up, took it out, and then actually put a different kind of robot back in. Then I said it's been seven years. What did we learn? And that was disappointing for a while, but then I had the idea to try to understand, well, what's really at the core of her approach versus my approach. Not the details of the systems that we built. We could try to compare the code or the robots, but that's not really meaningful. What's really at the core of the two approaches and that's where the operations' research literature turned out to be very helpful and I owe a lot to Mike Wellman who's a professor at the University of Michigan who was on my thesis proposal defense committee who put me onto the fact that what I was really looking at was a well studied problem in the OR community and that there were these well understood ways of looking at these problems and characterizing solutions to them and even bounding how good the solution could be and then I could come up with a framework, but I said, well, it turns out that my approach and Lynn's approach are actually equivalent in a certain way and you could say things about the performance of them and so on. So that all together was the thesis.

Q: And how did you get in touch with Professor Wellman?

Brian Gerkey: How did I get in touch with Professor Wellman? I had been reading his papers. He's a computer scientist and verges into economics, so I'd just been reading his papers and when it came time to put together a committee to give me advice on the thesis idea that I was proposing, I got in touch with him by email and he said, "Yeah, I'll help out."

Q: And who else was on your committee?

Brian Gerkey: On my proposal committee it was of course my advisor Maja, Gaurav Suhkatme, also at USC, Deborah Estrin who had just moved from USC to UCLA. She does a lot of work in networking and embedded systems and Milind Tambe who does multi agent systems work at USC in ISI.

Q: So earlier you said you were eager to get to stay in academia, but how did you wind up working in a private company?

Brian Gerkey: That's a good question. So from grad school I went on to do a post-doc because that's what you do after grad school and I worked with Sebastian Thrun at Stanford for a couple years and then from there I went to SRI which is just around the corner from here and did research there for about two and a half years and then near the end of that time, Eric Berger who was a student at Stanford out of Ken Salisbury's group and had been a co-designer on the PR1 came by to chat one day when I was at SRI. We had met when he was a student at Stanford and I was a post-doc there. I think he might've been an undergrad at the time and he came by one day and said, "So we're doing this new thing at this new company and we're going to build up all this open source robot software infrastructure," and that's a thing that I had been doing a lot of completely independent of my thesis work and he said, "Do you want to come help us do this?" and I said, "That sounds pretty cool." So came over and met Steve and Scott and the rest of the team that was here at the time and then came on board and that was about three and a half years ago.

Q: What kind of work were you doing at Stanford with Sebastian's group?

Brian Gerkey: In Sebastian's group at Stanford I was still doing multi robot work. So I was sort of following on from my thesis work, trying to wrangle teams of robots to do interesting things. Spent a fair bit of time working on a collaborative search problem where you have a team of robots that you want to have run through a building and search it in such a way that they find anybody who might happen to be in the building. That turned out to verge a little bit into computational geometry and was a pretty interesting thing to work on. I left right at the time that the autonomous driving stuff was coming up, so I didn't work on that.

Q: And you did similar work at SRI?

Brian Gerkey: I did do similar work at SRI. Well, SRI did some multi robot work and also some outdoor robot work. We had an interesting DARPA project that involved using where you had to get a robot to start from whatever location it happened to be in and get to some GPS designated location and along the way you had to use only stereovision to avoid all the obstacles and get there and that's where I first worked with Kurt Konolige who is actually here at Willow now and we worked together on that project. He was doing the stereovision and visual odometry part and I did the controller part, so how do you actually decide exactly where the robot's going to go given the information you have from the cameras.

Q: We talked to Bob Bolles over there and he showed us that room full of all these little robots and I guess they did something last night.

Brian Gerkey: Yeah. Oh yeah. I know Bob very well. Bob was also on a larger project. So that was the centibots project with all the little amigo bots and that happened just before I got there, so I wasn't working on that. That was part of the SDR program that DARPA ran which I had a peripheral role in when I was at USC.

Q: And how did you get involved with the open source community? You mentioned you had done it before you came here kind of on your own.

Brian Gerkey: Yeah, so when I mentioned the Pioneers, the robots that we used when I was in grad school, they came with some software that it was a particular behavior based interface for programming robots and it was fine. It was a reasonable way to talk to the robot, but like any software, it had bugs and it had limitations and unfortunately because it was a closed source piece of software, we couldn't do anything about it and it was made all the more frustrating because it so happened that the author of that software was actually a fellow grad student of mine in the lab at USC. He had done it under contract work for the company and he's right across the room, but I can't get access to the code. I can't fix bugs. I can't make it do something different. So like any good computer scientist, I said I can write my own and not live with these limitations. So together with a visiting student named Kasper Stoy, we wrote something that we ended up calling Player which was an open source middleware for robots. It's a server that runs on the robot. It has inside it all the pieces you need to talk to all the devices, the lasers and cameras and the wheels and so on. It eventually came to contain algorithms for doing things like localization and mapping and planning and things like that and we released it open source. We posted it on SourceForge. This was 2000-2001 and ended up building a pretty substantial community around that project. Beyond the device server Player came Stage, which is a 2D robot simulator and Gazebo, which is a 3D robot simulator. Those all were packaged together as part of what eventually we called the Player Project and still there are thriving communities around all those pieces of software.

Q: What kind of people constitute that community?

Brian Gerkey: The Player community is primarily people doing research and education. Player was really designed for robots like the Pioneer. It's a fairly simple mobile base and it tends to have a camera and maybe a laser on it, a laser range finder, and so Player's well designed for running in that environment. So people that have a bunch of those robots in their lab will use it to support their research and people who have a bunch of robots that they want to use to teach robotics will use it in classes. That's the main use. A couple of companies picked it up and

shipped robots with Player in it, but then they tend to be shipping those robots to people who are then using them for research or education. So it's only sort of one degree removed.

Q: And do you find a lot of people writing additional code that's compatible and trying to build a larger library around it?

Brian Gerkey: Sure. There were a lot of contributors to Player. The model that we had, it was very much following the Linux model where the way we looked at it, we had essentially built a kernel. There was this server that had all the infrastructure in it for exchanging messages and what provided new functionality was drivers, so if there a new IME came out or a new camera came out then you could add support to that by writing a driver that knows how to talk to that piece of hardware and originally, before making a new release of Player, I used to actually test it. So I would run it on our Pioneer and make sure everything worked, but pretty quickly because of all the drivers that got submitted from around the world, it had code in it for hardware that I had never seen and still have never seen. So it got to a point where I on my own couldn't test it and had to rely on the community to do quality control on more of a continuous basis.

Q: And what did robot operating systems look like prior to – what else were people using at that time?

Brian Gerkey: Prior to Player, I don't want to make any broad claims that could be falsified, but my experience had been that they tended to be libraries that you would link your program against and they had embedded in them the ability to talk to certain devices and that's certainly the way that IU was. That was the software that we originally used in the Pioneer and so the innovation with Player was that we made it work across the network. So there was a server component on the robot that would actually talk to all the devices over serial lines and USB lines and fire wire and whatever, but you wrote your control program, the thing that decided what the robot was going to do on the other side of a network connection, and that meant you could write it in any language that you wanted, for example. That was a big advantage. So there's still Player client libraries in C and C++ and Java and Lisp and Ada and there was one in Tickle for a while, so it basically made the development environment a lot more flexible to have that network abstraction embedded into the middleware.

Q: And so I think that brings us practically to Willow and you getting into Willow and the kinds of things you started working on when you were here. So could you tell us a little bit about that?

Brian Gerkey: Sure.

Q: What was it like?

Brian Gerkey: Yeah, so like I said, Eric and I had interacted at Stanford because he was actually using Player on a robot that he was building there, so when he and Keenan were moving over to Willow and they had the idea that we should build a next-generation robot software infrastructure, he came to me and said, "Well, will you help us build this?" and that's what became ROS was the thing we started building. We were working closely with Morgan Quigley at Stanford who had been writing something he called Switchyard and that was a thing that he'd built to use to counter some of the limitations in Player. So we took some ideas in Switchyard and morphed them and iterated on them. There was about a year there where if you had come to ROS and tried it, then I apologize profusely because it was terrible and it was terrible in a different way every day, but eventually we settled on what we still feel like is a pretty good design for a piece of robot middleware.

Q: What were the problems you were running into in those early days that made it be terrible in different ways from day to day?

Brian Gerkey: Well, I think it's the problem that you have on any new projects. There's such a big space of designs that you might try and especially when you're building essentially a toolbox. You don't know exactly what it's going to be used for, so it's very hard to make the tradeoffs in a principled way because you don't know exactly who's going to use it for what, so you're trying to be general, but you're also trying to be efficient and easy to use and all these other things that tend to trade off against generality. So what would happen is that we'd go down one path for a while and then convince ourselves that that was probably not a good way to do it and then throw it all away and then write a new one and go down another path for a while and then throw that one away, but the hope is that you get closer and closer to a good design as you go through that process.

Q: And what were some of the concepts and design elements of Switchyard that you were able to leverage in the new ROS?

Brian Gerkey: Well, I'd say the biggest thing that ROS borrowed from Switchyard was a peer-to-peer setup. So Player had been a client server architecture which is easier to write and easier to use, but not as flexible as a distributed peer-to-peer architecture. So that was one piece that we borrowed from Switchyard, this idea that you're necessarily setting up a distributed competing system that is composed of many nodes and that those nodes talk to each other about the connections between the nodes are according to whatever information needs to be shared between them. We also borrowed an idea from Switchyard which really came from Unix before it which is that in building complex systems, a good way to make progress is to build small tools where each tool does something specific and does it really well and you design tools to be composed together to give you the behavior you want. So if you're familiar with UNIX, it's full of command line tools that you can string together in interesting ways to do very powerful things

and that was the design philosophy behind Switchyard and also ROS. We don't build a monolithic system that does a particular thing. We give you a Swiss army knife of all kinds of things that can be put together.

Q: And in terms of the fact that robots drive around and run into things and break things, what are some of the issues and concerns about security and reliability in an operating system for a robot?

Brian Gerkey: Well, I'd say the biggest concern in terms of reliability is that at this stage in robotics, we're still working out the bugs even in the basic components and so it's important to design the system so that when one thing fails it doesn't bring down the entire robot and that sort of naturally follows from the peer-to-peer setup. If one node in my system crashes, there was just a bug in it, I don't want it to tear down the whole system. I want to have the ability to bring that piece back up and have everything keep going. So certainly from a reliability point of view, just dealing with buggy code is really important. If you talk about the fact that robots can run into things, that's not the sort of thing you can protect against at the low level, right? That's very much a high level idea that has to be embedded into the decision-making process that is at a different point in the system and also running into stuff isn't necessarily bad. Maybe you meant to run into something. Maybe you need to run into the door to close it. We often have this problem with manipulation, right? We spend most of the time controlling the robot arm trying not to hit stuff. We have a lot of software that makes it so that I can get my hand from under the table to over the table without hitting the table, but then if I want to pick something up, then I'm colliding with this pen to pick it up and so that's a purposeful collision. So it's not the case that running into things is always a bad thing. It's very much context dependent.

Q: And so you mentioned when you were developing Player that you envisioned it being used with kind of small simple robots like some of the ones people were using in education and research at the time. So when you were developing ROS, did you have any considerations about particular types of platforms?

Brian Gerkey: So when we were building ROS, we were designing it for the PR2. So the PR2, we knew the basics. We looked at the PR1 and we knew basically what the PR2 was going to be like. It was going to have a lot of sensors. It was going to have arms. It was going to have the sensors were going to be actuated. There was going to be a laser that was going to be tilting. There were going to be cameras in the arms that are moving with respect to the body. So we took that into account when making design decisions about ROS. For example, early on we decided to put in an abstraction for dealing with coordinate transforms. So this is a really important and totally tedious aspect of robotics which is that if I wanted to incorporate say two lasers scans where one came from a laser that is on the base of the robot and one came from a laser that is in the robot's hand, for example, then I need to know where the base was and where

the hand was at a particular point in time and then I need to pick another frame in which I'm going to incorporate them and then do some math to transform the two and that's the kind of thing that tends to be done by hand again and again and again in everybody's program and it's the kind of thing that everybody gets wrong. Personally, if I try to do that kind of math by hand, I take a random walk through the space of trig functions until I get it right. So early on we decided that's crazy. What we should do is have a consistent way of dealing with coordinate transforms and in fact, every piece of data that is published should have a frame attached to it. We should say this scan or this point cloud or this image or whatever it is was acquired at this time in this frame and then there's a library that knows where all the frames were with respect to each other over time and then can do things for you automatically, like take this point cloud that I took in the coordinate frame of my hand and show me where it is in the coordinate frame of my head.

Q: And obviously the PR2 is a much more complex robot than the ones that you were thinking of before, so how did some of that affect how you designed ROS or how you went about it?

Brian Gerkey: Well, the complexity of PR2 really came through in the design of the software in that we knew that we wanted the system to be flexible. We wanted to be able to let the user decide how to either exploit or hide the complexity of the robot and that meant letting the user decide which parts would be up at any given time, which sensors you're using, which actuators are you using, and also have full control over how the the robot was being commanded. Do you want a high level interface where you, for example, tell the hand where to go? You say I want the hand to be here and it computers inverse kinematics and runs some fairly sophisticated controllers and gets you there. Or do you want to command the joints individually and say apply a torque here and then apply a torque there and then apply a torque there? You can go in at all levels.

Q: And can you tell us a little bit about some of the initial testing of the program and if anything interesting happened at those points?

Brian Gerkey: Hardware or software?

Q: Both.

Brian Gerkey: Well, I'll tell you about software because I can't talk as much about hardware. So I can tell you that in the early days of ROS development we didn't do a lot of testing. We started building up pieces and started building up pieces very, very quickly. In a short period of time we had a lot of functionality. We had the basics for controlling the robot and we were starting to have it navigate. We were doing some control on the arm. Perception pieces were

coming together and we started to have trouble with stability. Pretty quickly we had a big code base that on any given day may or may not compile and long story short, we ended up coming around to very standard software engineering techniques where, for example, you write unit tests and you write regression tests and you have a continuous integration system that is watching the code that you're checking in, checking it out, compiling it, running tests on it, and giving you some confidence that you didn't break it with that change that you just made at two in the morning.

What turned out to be very important for us was whole system regression tests. So we've got this very complex hardware system and we've got this pretty complex software system as well. It's very hard to test every little piece of it as much as we would like. It's just there's only so much time that you can dedicate to testing. You're never going to get complete coverage and even if you test every piece, then if you really want to be complete, you need to test every interaction between every piece and all the different ways you might put them together and that's pretty much impossible. So our proxy for doing that complete coverage which we can't do is to pick a few tasks that are representative of what we want the robot to do, make those tasks work in simulation, and then have a regression test that runs inside the simulator. So for example, once we had the robot opening doors, we made sure that it would open doors in simulation in a 3D physics based, pretty realistic simulation, and then that becomes a regression test. Anytime somebody checks in a change to the code that is involved in opening a door somewhere on a server in a room in the back there's a simulation spins up with a PR2 on one side of a door and it's got its arms tucked and it runs the whole door opening sequence where it looks for the handle and then reaches out an arm and opens the door and pushes the door open and then drives through and we can make that into a test by saying within some period of time that the robot end up on the other side of the door or not and it turns out those kind of tripwire tests, does it no longer get through the door, are really important. When the light turns red on that test, you know something went wrong. You don't know what, but you can look at the changes between when it was green and red and then figure out what happened.

Q: And do you try to use that sign of testing and reliability for community generated code or integrating things that other members of the open source community are developing for ROS?

Brian Gerkey: So to improve quality or support quality assurance in the community, we do two things. One is technical in that if people release their code, we have a release system so you can decide to release your code to the world, then it runs on our automated servers where it will run all the tests that you defined and give you reports, did they pass, did they fail. Of course that's not very helpful if you didn't write any tests and we can't write the tests for you. Well, we could, but I don't think that's the best use of our time, so instead what we do is try to build a culture of testing, which is, I think, unusual in the research community in my experience. There's not a big focus on writing tests. So we're trying and with some success to set an example for what's expected for good software in robotics, even coming out of a research lab.

It's so much better to have basic tests that show that it doesn't just break out of the box than it is to have no tests at all and I think we're getting some traction. I mean even grad students in university labs are actually writing tests for their software which is something I never did as a grad student in a university lab.

Q: And how are you raising that kind of awareness with people?

Brian Gerkey: Well, I guess we're raising awareness about testing mostly just by our own example. If you go in and look at code that we've written, you'll find a good measure of testing in there and we let everybody know that the best way to protect yourself against changes in the system is to have tests in your code. So if you've written tests and you released your code and somebody else makes a change that is going to break your piece and you have a test against it, then the light will turn red when that other person made a change and then you've got a chain of responsibility there and you can get back to the person who made the change. If you didn't have a test that demonstrated that your thing is no longer working then we can't do anything about that.

Q: And have you found people trying to port ROS to other platforms than PR2?

Brian Gerkey: Oh yes. So we designed ROS for the PR2, but from the very beginning we wanted it to not only be for the PR2. We knew that the PR2 was going to be a really great robot, but it was not going to be the only robot in the world and probably not even the only robot that we build. So we wanted to make sure that it would run on other robots. Our mantra early on was we wanted the string PR2 to appear as little as often in the code base. If something was PR2 specific we'd say this is the PR2 thing, but we wanted to have as few of those things as we could. We wanted to build general capabilities. So at this point it kind of depends on how you count it, but ROS runs on dozens of different robots, dozens of different kinds of robots that are made by robot manufacturers, but also just a wide variety of one-off robots that hobbyists and educators and academics have built.

Q: What are some of the crazier ones you've seen?

Brian Gerkey: Well, I mean in terms of unexpected ones, there's the PR2 and then you could look at robots that are kind of like the PR2. We had some good experience collaborating with folks in Japan to get the ROS navigation system running on the HRP –

Q: I think it's four now.

Brian Gerkey: Four, yeah. It was the one that was on wheels. They had a version that was on wheels, but in any case, that robot, it's kind of similar to the PR2. The Care-O-bot from Fraunhofer in Germany is kind of like the PR2 and they've done a great job of supporting ROS on that platform and then from there you can go to simpler robots like the Pioneers of the world, but then there are some that we just didn't think of at all like quadrotors. People are running ROS both onboard and offboard quadrotors, running ROS on little tiny embedded computers. There's a really active group at ETH in Zurich that has run ROS on a wide variety of things from these crazy little two-wheeled robots that have magnetic wheels and are used to drive inside pipes to inspect them, to boats that go on the surfaces of lakes. There's a guy who instrumented a surfboard to get acceleration information and used ROS to acquire and then plot that information so he could see exactly what his board was doing as he was coming down the wave. So certainly some areas that we didn't plan for and that's what's really exciting now. I mean early on we were really counting users. How many people are on the mailing list? How many repositories are out there? How many packages have been written? And we still count that stuff because that's interesting in terms of measuring impact, but now more than that we're really looking for different kinds of users, different kinds of people. Are we now seeing ROS used on a completely different kind of robot? Are we seeing ROS used in a commercial context as opposed to an academic context?

Q: Roughly how many users are you at?

Brian Gerkey: Counting users on an open source project is hard because they don't have to tell us that they're using it. So we try to have proxies for it. Our main mailing list has about 1000 users on it, so it's at least that many, but in my experience, most users won't join a mailing list. They just use the software and they come in through other gateways to talk to people, different forums and so on. One proxy that we use is how many unique visitors hit the installation page every month because we figure if they're reading the installation instructions, some of them are going away and installing it, and that's about 10,000 a month right now. So that's some indication of how many users there are. We see peaks at certain times of the year. They're using ROS in classes in universities, so you see a peak of installations around the beginning of the academic year, for example. I would guess it's tens of thousands, maybe hundreds of thousands. It's hard to say.

Q: So what's the next evolution of ROS and the next challenges you face developing it?

Brian Gerkey: So where I see ROS going next is getting ROS onto more computing platforms. So we started off really targeting Linux and that's still where we put most of our effort and Mac OS X is also a good platform to support. We're partially supported there right now, but the real prize will be Windows. I mean so in terms of getting a lot more developers involved, having good Windows support I think will be important and that's something that we're making steps

toward right now and the reason that we would do that is that what I'd like to see happen is to get people who don't have robotics backgrounds using ROS. So right now people who have robotics backgrounds who are very interested in robotics don't have any problem picking up ROS and it's great for them and I think we're doing a good job as a community of satisfying that need, but what will be really different is when people who have good software engineering skills and have creative ideas for what you might do with a robot, but don't have a robotics background can come in and start using ROS to start prototyping their ideas for what a robot might do and so there is some work for us as a community to do to make ROS more accessible to people who are coming in. They didn't go through grad school in a robotics program, they're not experts in computer vision, but what we want to do is provide a toolbox that lets them put the pieces together to try out their crazy idea for what a robot might be useful for.

Q: Great, and we always wind up with this question. So for young people interested in a career in robotics, what kind of advice do you have for them?

Brian Gerkey: For a young person who's interested in robotics, I'd say that first of all you're living in a lucky time. You're lucky to live in this time, I should say. I think things are really taking off right now like we've been hoping that they would for quite a while. I would say get as much computer science education as you can, get as much math education as you can. Those are the two pieces on which most of the work in robotics is built these days, but then I'm a computer scientist, so I'm not going to tell you to go get the mechanical engineering expertise, although that would clearly be useful.

Q: Thank you. Great.

Brian Gerkey: All right.