INTERVIEWS

An Interview with Laura Nolan

RIK FARROW

Laura Nolan has been writing software since teaching herself to code in 1996. After earning her computer science degree at Trinity College Dublin, she worked as a Software Engineer for several years before joining Google as a Site Reliability Engineer in 2013. Currently, she is Tech Lead for a team of SREs working on Google's Edge Network in Dublin. Laura is also co-chair of the USENIX SREcon Europe/Middle East/Africa Conference as well as one of the (many) authors of the O'Reilly SRE book. When away from the keyboard, Laura can often be found traveling, hiking, scuba diving, or just enjoying a scotch and a good book (Longmorn and space opera are her particular weaknesses). lnolan@google.com

Rik Farrow is the editor of *;login:*. rik@usenix.org

I first heard about Laura Nolan when she spoke at SREcon Europe in 2015. I watched the video of her talk [1], and that forms the basis of some of my questions.

Laura has since been a co-chair of SREcon, and we've had email discussions about potential *;login:* authors. Some of the people she suggested have written articles related to SRE for *;login:*.

I thought it was time to get to know Laura better, so I asked her if we could do this interview.

Rik Farrow: I watched your presentation at SREcon Europe about distributed consensus algorithms. I have enough systems background to know what that means, but perhaps you'd like to explain that in the context of SRE?

Laura Nolan: Well, distributed consensus is a really important building block of a lot of practical distributed systems. Any system where you need to get a group of processes to agree on something, as a whole, is solving distributed consensus. I think it's important for SRE to know something about it because it is a difficult problem, and trying to solve it in ad hoc ways can lead to some very surprising outcomes. So that talk, and the distributed systems chapter of the O'Reilly SRE book [2], is really trying to focus on showing what distributed consensus problems are and discussing the operational aspects of distributed consensus systems: their performance constraints, failure modes, monitoring, and so on.

Generally speaking, I think we need to raise our level here as a profession. SREs are working at the sharp end of distributed systems. Most of us don't have a solid background in them though; most engineering and computer science education is still pretty light on distributed systems content. We are figuring out these things on the fly all the time, and we're not being systematic enough about it.

Software engineers have design patterns...we need distributed systems reliability patterns!

So I'd love to see more distributed systems content at SREcon next year, building on some of the great content we had such as Theo Schlossnagle and John Looney's distributed systems workshop [3] and the reliable RPC talk and workshop presented by three Googlers (Grainne Sheerin, Lisa Carey, and Gabe Krabbe).

RF: In your talk, you mention using Paxos. I interviewed the primary author of Raft, Diego Ongaro, who said that one of the reasons for creating Raft was that Paxos was difficult to reason about. Yet Google seems to have settled on Paxos. Could you tell us the reasons for using Paxos instead of Raft (or ZAB)?

LN: Well, historical reasons is one part—Google was using Paxos heavily since before 2006, when the Chubby paper was published. The Raft paper [4] was published around 2014, and ZAB was also after Chubby. I'm not involved personally in any of this infrastructure at Google—I used to be on a team that ran a lot of Paxos-based data stores but have moved on—but as a software engineer, I will say that making that sort of change to any software system is expensive, and always more expensive than you think. Effort to rewrite a system is one thing, but then you have to test it, which is particularly onerous in the case of this sort of system, with many subtle failure modes. You'd need a really compelling reason to do it, and Raft and ZAB don't provide any immediate technical benefits over Paxos.

RF: How much opportunity have you had to learn new things and move into other areas? How does that compare with other places you have worked?

LN: Google certainly affords plenty of opportunity to work on different things and learn—this is one of the benefits of being a large organization. Mobility is encouraged reasonably strongly—not every few months, but if an SRE wants to move every few years that's seen as positive. It avoids teams becoming too siloed and set in their ways.

In the almost-five years I've been here, I've worked on three major areas that were all very different—big data stores and pipelines were the first, then an internal development project, and now I'm working on a team whose main focus is the reliability of our Edge Network and peering. So that's been a major change for someone without a network engineering background, and I've been working hard on "knowledge upload" related to both networking generally and our network specifically.

Even staying on one team, though, things do change—the major tools we work with evolve and are replaced, for instance; new projects are developed, and SRE teams will begin to support them. One big change in the last couple of years has been the introduction of Golang as the programming language of choice for most SRE automation projects, so everyone's learned Go.

RF: You mentioned that Golang has become the language of choice. What do you think of Go, and what were you using (or liked using) before Go, for contrast?

LN: I like Golang. It is relatively hard to shoot yourself in the foot with it compared to many other languages. It scales fairly well, has great concurrency constructs, and I am a fan of stronger typing. Generics would be nice, though, and I miss my ternary operator!

RF: John Looney has written about psychological safety on SRE teams. When I read his article, I felt like my entire work life would have been different if the teams I had worked with were more like the ideal John writes about. What has your experience been along the lines of psychological safety?

LN: I think that psychological safety is really important. My experiences as a member of teams have definitely been on a continuum from very psychologically unsafe to quite safe. It's certainly much more pleasant and way more productive to be in a team that is safer. Unsafe teams will burn you out faster than anything else, and burnout is the curse of Ops work.

Google SRE is pretty good, as these things go, in particular with respect to blame. We had an incident a few months ago that I think illustrates this nicely. We had an internal mishap that caused us to have a pager storm, and without going into the details, the trigger was a junior non-engineer who erroneously did something involving a mailing list. I know for a fact that multiple SREs specifically reached out to that person's manager afterwards to tell them that the incident was not that person's fault. None of them knew the individual personally, they just didn't want them to experience any negative consequences for the incident. I thought that was pretty great. And, of course, we've fixed the root cause of the pager storm too!

Tanya Reilly, an amazing SRE, gave a talk at LISA where she discussed what to do when someone breaks something in your system or finds a bug: you thank them for finding the gap and then you go fix it. That is for me an essence of psychological safety: no blame, no acrimony—just making the systems better.

On the flipside, I think the worst thing for psychological safety is the engineer who thinks they're smarter than everyone else and is constantly negative and critical about other people's ideas. That sucks the life right out of a team. Nobody, no matter how much of a rock star they are, is worth the kind of damage that causes. Don't be that person! Far better to be the engineer who makes everyone around them better than to be the one that makes everyone around them miserable.

RF: While at a WiAC meetup during NSDI '17 in Boston, one of the participants said that women have to work twice as hard as men do just to be noticed. You've worked in several organizations. What's your viewpoint on this?

LN: Different people have different experiences. I don't think I've ever been overlooked due to being a woman, but then I am assertive and outspoken, and it would be hard not to notice me. I did, however, once have a hilarious performance review (before I was at Google) with a male manager of mine where he spent about a solid hour telling me about how I talked too much. There was not an ounce of self-awareness there as I couldn't get a word in edgewise to actually discuss this issue properly. I didn't stay much longer in that organization.

I think there may be more truth in saying that there are some expectations for women that don't exist for men. There's a common antipattern where women end up doing a lot of the emotional labor in a team, even if it's not in their job description—things like organizing team events and recognizing occasions, and so on. Another huge thing is women engineers taking on things like taking minutes, organizing meetings, project management, building relationships with partner teams—some people call it being "the glue." Teams actually really need the glue to work well, but it can be under-recognized compared to coding because it's harder to measure. This also goes for men who are "glue" types. I also feel like it can be harder to get technical things done sometimes as women—I've seen things like excessively picky design and code reviews aimed at women more often than at men.

References

[1] L. Nolan, "Distributed Consensus Algorithms for Extreme Reliability," SREcon15 Europe: https://www.usenix.org/conference/srecon15europe/program/presentation/nolan.

[2] B. Beyer, C. Jones, J. Petoff, and N. Murphy, "Monitoring Distributed Systems," in *Site Reliability Engineering* (O'Reilly Media, 2016).

[3] J. Looney and T. Schlossnagle, "Distributed Systems-Reasoning," SREcon17 Europe: https://www.usenix.org/conference/srecon17europe/program/presentation/looney.

[4] D. Ongaro, J. Ousterhout, "In Search of an Understandable Consensus Algorithm," in *Proceedings of the 2014 USENIX Annual Technical Conference (ATC '14)*: https://web.stanford.edu/~ouster/cgi-bin/papers/raft-atc14.