

How Are Architects Made?

Bett Correa, Verizon

For all you readers who want the inside dope on how to be an architect, you've come to the right place. Here's a real-life story of the evolution of an E2E architect who has shared her lessons learned with us. There's clear insight here. She describes her realization that it's not just about knowledge of the domain or technical skill, but also about communication and learning. —*Linda Rising, associate editor*

ARCHITECTS DESIGN SOFTWARE by making decisions about how the software should work. At Verizon, each system has its own group of developers, business analysts, testers, project managers, and, of course, architects. A new system springs to life when existing systems require too much change to their functionality to accommodate new requirements.

I'm a customer experience architect, or end-to-end architect (E2E), at Verizon, which means that I study how the current set of systems work to meet our customers' expectations. I also work on our own customer touch point systems. I use contextual design methods, such as sitting with users while they do their work and taking the resulting

feedback to reengineer our processes and tools.

E2E architects must bridge the gap between teams, understand each system's limits, and consider each team's politics. When the business submits a new requirement, the E2E architect must create a design that implements that requirement across multiple systems, "sell" that design to various teams, and teach both developers and testers how the new system will work.

In this article, I'm going to describe my journey to becoming an architect and the skills I learned along the way. I hope this article will inspire you to encourage team members who have an interest in architecture toward becoming architects.

Mastering the Golden Triangle

Herd the cats, as I often think of my job, requires a careful balance among technical knowledge, domain knowledge, and communication skills.

I started honing my technical knowledge in my teens, when I spent hours coding in Perl and JavaScript, but I didn't put much time into design back then. In the start-up companies where I worked as a developer after earning my AS, I was more of a "code as fast as possible and hit compile to see if you get lucky" kind of a person, and I usually didn't. I never realized that there was a discipline with best practices built around architecture. When I went back to get my BS in computer science, I started playing with software design tools, but the curricula was tailored more to creating algorithms. Eventually, after working at several other start-up companies, I made my way to Verizon, where I focused on learning the minutia of the telecommunications domain.

I learned communication skills the hard way, after spending most of my time hiding in my cube, coding. My boss asked me to join client calls, but I told him I'd rather not because I was shy. I joined Toastmasters in an attempt to fix my communication problem and really liked the format. I started to feel more comfortable speaking in front of audiences and running meetings.

I soon started working on systems requirements, which was close to design but not quite the same: I was primarily documenting what others told me the system should do. I worked on that project for a few months before meeting my future mentor, Danilo, the E2E architect for a Voice over IP (VoIP)

product. He was the first person to introduce me to the idea that you could do design full time. That was the point at which I became obsessed with learning how to become an architect.

Clicking into Place

My mentor told me that I needed to learn the systems and understand how they worked together. He also told me to get to know the system leads because

giving something more security but allowing it to take longer to do a specific function. One decision might work well in most cases but fail in others, so I had to figure out in how many cases it would fail and whether the work-around for those few cases would end up being expensive. Slowly, I started to build my design experience, which eventually led to an official job offer as an architect.

My design decisions were based on trade-offs.

they could help me learn as well. I started in the VoIP area and quickly became an expert in many of the related systems—people soon began asking me how things worked. I spent many hours with the switch manufacturers, learning how soft switches worked, and with the various business teams, trying to understand the market, where we were going as a company, and our different project goals. I read white papers and took classes in our online university. I found that most people were generous and helpful if you asked and were willing to work hard. All those hours paid off, because I could talk with ease about the details of how the system should work with both developers and business clients.

One day, my mentor gave me small pieces of a larger design to do, which was very different from the activities I'd done up to that point. I had to decide several parts of the design, such as which system should do which function when two systems were equally equipped to handle that function. Many times, I found that there wasn't a single right answer: my design decisions were based on trade-offs, such as

In my first year, I worked on the designs for 25 separate projects, which was a fantastic experience because I got to see so many different parts of our VoIP and FiOS (Verizon's fiberoptic service) infrastructure up close. I was able to get groups of people from a variety of backgrounds to work together on solutions to technical problems. I found that by learning the details of the domain and listening very closely to all the stakeholders, I could create a design that worked—as well as get stakeholder buy in and negotiate with those who had opposing viewpoints.

Pulling Together the Best Practices

Our team was growing, so after working in the group for a year, my boss asked me to put together a best practices presentation for the newer architects. That presentation turned into a speech, which eventually became a book (*You Can Be a Software Architect*, 2012). I started teaching how to become an architect and mentoring young software developers in this particular area. I loved being able to pay it forward.

In a nutshell, if you're a manager who wants to develop more architects,

- identify members of your team to mentor,
- give them pieces of design to do,
- encourage networking by having networking events,
- introduce the whole team to design thinking by involving them in the design process, and
- encourage and reward formal training in design practices.


If you're interested in becoming an architect yourself,

- find an architect to be your mentor,
- look for projects you can design, even if on your own,
- go to networking events, and
- attend formal training in design practices.

In my experience, two things didn't work. Until I actually started "doing" architecture, I didn't understand how to do it. I had a lot of grand ideas, but once I started working with the teams involved, I realized that I didn't know anything at all. Without practice, there's no way to know if your high ideals around design will actually work

The second thing that didn't work was doing architecture without training. Formal training via books and classes are important to being an effective architect. So many powerful techniques on design are taught in classes or in books, such as contextual design, in which the architect or business analyst watches and documents the worker doing the work in his or her natural environment.

Computer science is an ever-evolving area, and training is a must. Contextual design was a big eye opener because it gave me

a clear viewpoint from which to do my designs. Instead of just trying to meet goals for requirements, which can often lead to any number of design decisions, having a very clear focus on customer experience leads to designs that are accepted by users and meet their unrequested needs. Collecting requirements in a vacuum without contextual design inputs leads to systems that don't answer user needs. Architects will always be in demand as long as software is needed. It's a challenging but exciting job. 

Acknowledgments

Thanks to Rebecca Wirfs-Brock, who helped me shape this article.



ABOUT THE AUTHOR



BETT CORREA is a customer experience architect at Verizon. She recently wrote a book called *You Can Be a Software Architect* (2012). Contact her at bettworld@gmail.com. She also cohosts a weekly software architecture podcast at www.architecturecast.net.



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

computing|now

GET HOT TOPIC INSIGHTS FROM INDUSTRY LEADERS

- Our bloggers keep you up on the latest Cloud, Big Data, Programming, Enterprise and Software strategies.
- Our multimedia, videos and articles give you technology solutions you can use.
- Our professional development information helps your career.

Visit ComputingNow.computer.org. Your resource for technical development and leadership.



IEEE  computer society

Visit <http://computingnow.computer.org>