On Being a Generalist

Adrian Kosmaczewski

2022-06-10

There is a lot of discussion online these days about the relative benefits (and drawbacks) of being a generalist software developer.

I have always been proud to be a generalist; and I have for a long time. Of course, I have sometimes (somehow) specialized in some technology stack that interested me: .NET from 2000 to 2008, iOS from 2008 to 2019, and Cloud Native things like containers and Kubernetes clusters since 2019. But I always kept an open eye for whatever else was going on in the software world.

This very blog is a reflection of that; back in 2005 and 2006, even though I was working as a .NET developer by day, I was blogging about Ruby on Rails, Ubuntu, VBScript and C++ by night. I can't help it. That's the way I roll. These days I'm interested in Rust, TypeScript, and even BASIC!

It is precisely this same curiosity what took me to start a magazine like De Programmatica Ipsum with my friend Graham; who, by the way, has also been writing and talking about generalists lately. Being a generalist allows me to see connections and trends across technologies, things that might not be immediately evident to people immersed into a single tech stack every day.

This is the reason why I enjoy so much reading past issues of Byte, Dr. Dobb's Journal, or keeping up with IEEE Software. Software is infinitely interesting. After 25 years of practice, I still find it fascinating.

Being a generalist has its perks: it game me the possibility of migrating across galaxies, and not only once, but quite a few times in my career. It allowed me to manage multi-language projects like Fortune or Conway from begin to end. It made me appreciate software history; a lot, even: reading how previous generations did our job makes me understand that we've been re-inventing too many wheels, needlessly.

Of course, being a generalist is not always great; in particular, I have to say that it has been indeed very difficult for recruiters to help me find a job in the past. For most of them, I'm not an easy-to-place resource (I know, that phrase sounds horrible). Most of the jobs I have found in my career, it was through my own contacts, or by applying directly to the hiring manager. Recruiters have a hard time with my profile; they definitely prefer specialists. Please understand that I don't blame them, and I can totally understand. I just happen to know the limitations brought by my choices.

Do I recommend other developers to be generalists? Yes, of course I do. I think it's a very rich and enlightening approach, one that will open your eyes to the beautiful world of software and its various facets. It has allowed me to work with people in various fields, with very different backgrounds, and with very different technologies.

What kind of jobs can you do as a generalist software engineer? Well, for starters, it's a great way to become a consultant, particularly in small-to-medium organizations. In such environments you need to be able to take lots of decisions that will have a long lasting impact, and having perspective definitely helps in those cases. Customers need guidance to start new software projects, and being a generalist helps a lot.

You can also be one of the first developers in a young startup, where you need to do a little bit of everything, and you need to coach others to get things done. This is another place where generalists are an asset, in my opinion. Not only will you have to write the code, you will have to choose the programming language, the CI/CD workflows, the methodologies, the testing and deployment harnesses, etc. Being a generalist helps a lot in these cases.

Finally I think that being a generalist can also help you become a good CTO; the best CTOs I worked with, in any case, had perspective and experience in various fields.

But if you do choose to be a generalist, be aware that your market value might erode a bit ; not everywhere, but in some select places you might be interested to work in. If that's the case, you'd better follow the instructions of the excellent "Positioning Manual for Indie Consultants" by Philip Morgan, become a super specialist in some tech stack, and enjoy the benefits that approach could bring.

In short: there is no "better" approach, but simply ones that work better for you, in your context, and following your own taste. You will have to find a way, and I suggest to suppress all those voices around you, telling you to specialize or perish. Develop your own taste, find what you like about software, and follow it.

PS: you might have noticed that I have not talked about the famous "T-shaped skills" approach; it's up to you to decide whether or not it's worth a road to follow. I, for one, am not so sure about it, and still prefer to grow my overall curiosity.