Title: Software Developers: How to have Impact? [WIP]

This post is Work In Progress, you can suggest changes to it directly in this Google Doc.

Can Software Developers Have Impact?

Yes

Why do I think so? (is this still in doubt?)

1. EA orgs need software developers
2. Developers are really useful for lots of things: They get high salaries, which indicates there is a lot of demand even given their high supply.
3. As a “worst case” (which is a pretty great case), you could probably earn lots of money, donate some of it, and have a profession that I personally consider really fun.

The 2 main things that determine your impact are..

1. Working on **an important problem**
2. Using **a good approach** to that problem

An relatively easy way to get “full points” on these is to **pick your favorite top cause**, and **work at an EA (or an EA vetted) org** that is working on that cause.

Most of the article will discuss how to get to that situation.

Other good paths also exist; I want to mention one of them:

Entrepreneurship

It could be a great path, but it’s out of scope for this post.

Is working directly more impactful than earning to give?

This is easily testable:
Ask the org that you’re considering applying to “would you prefer hiring me, or getting [as much as I’d donate]”?

I predict

If the org wants to hire you, they’d prefer that over getting **5x as much** as you’d donate, and they’d be all “**OMG YES**” about it.
But why guess what is easily testable?

If my prediction is wrong

If the EA org would prefer donations, then some of the sections here (like “which org?”) won’t be relevant for you, but others (like “how to build skill?”) still will.
Skipping sections is encouraged!

How to have a lot of impact?

The rest of the post is split into plans, and the intent is that you try the top plans first, and if they don’t work, go to the next plans

- Plan A [link]: Apply to EA aligned orgs
- Plan B [link]: Work in the industry: build skills and make money
- Plan C [link]: Learn programming, write a CV, and get your first job

Plan A: Apply to EA aligned orgs

Here are some very common questions:

Q: Which org? There are lots of options and I’m not sure which are the best / which would accept me

A: Orgs from the 80,000 hour job board

Here.

A: Apply first, filter later

It’s a common mistake to first filter the list of orgs **for a long time**, and then apply, without even knowing if those orgs will accept us. So I recommend first applying, and then based on who accepts us, filter.

A: Apply to your top choices first

If there are a few orgs that are so good that you’re not sure which one you’d choose if you were accepted even after thinking about it for 2 minutes (not 2 weeks!) : Start by applying to all of those. If they don’t accept you, go to your next choices.

Q: Am I wasting the org’s time if I apply even though I am [not qualified / applying to other places]

NO!
This is a big problem in the EA dev community: Many devs don’t apply because they are worried about wasting the org’s time, please be part of the solution!

You can, however, be transparent about the situation, for example:

1. “Hey, I’m applying even though I have less years of experience than your job ad requires. I’m writing this because I don’t want to waste your time, feel free to not interview me if you don’t want to”

2. “Hey, FYI I’m applying to a few other EA orgs as well, so it could be that I’ll end up going to one of those. Would you like to interview me anyway?”

I said both these things myself.

Q: If I don’t pass, when can I reapply?

This will often be written in the job ad, and will often be “you can reapply immediately”.
If it’s not written, you can ask (or get me to ask, if you prefer).
For example: “Hey, I’m considering applying, but I’m worried that I’m not ready for the interview. Do you prefer that I take time to study for the interview, or that I apply immediately and study only if I don’t pass?”
(I said this myself)

Q: Can’t EA orgs hire the next best person?

Probably not. There is a strange problem where EA Devs don’t apply because everyone expects that the next-best person is amazing or something like that. Be part of the solution!

Q: I’d apply, but [They’re not remote / the salary is too low / I don’t know how I’d feel working in a small team / …]

This is a blind spot for many orgs.
Consider telling them. I’m trying to aggregate this information myself, if you’d rather tell me.

Q: I’d apply, but I have a very specialized skillset / I’d apply, but I don’t have the specialized skillset

A common problem in EA is that each developer has a specialized skillset (like blockchain) and assumes that someone else would be better (like a django+react developer), and this ends up with lots of people not applying at all.
Please be part of the solution!

Plan B: Work in the industry, build skills and make money

Which skills?

If you want to work at a specific EA org

- Best plan: Check or ask if they have suggestions on what to learn in order to get accepted to work with them.
- Next best: Do something similar to what they do. For example, if you want to work on the EA Forum at CEA, try doing something similar, not something like optimizing the performance of a database at a huge org.

If you don’t really mind which org

There are more than 100 EA aligned tech jobs

Almost any skill you'll pick will have demand, including frontend, backend, ML / data science, building websites or apps, distributed systems, devops, information security, and probably more.

Skills I usually won't recommend:

- Very old unused languages (Pascal / Visual Basic)
- Very low level languages (c, assembly).
- I'm on the fence about recommending working on huge systems (with hundreds of developers or more)
    - Downside: EA orgs right now are small or medium. Some of the skills you'll gain from building something huge are somewhat different from building something small, and specifically the tradeoff between [building features quickly even if things sometimes break] and [not breaking anything even if features get built slower].
    - Possible upside: These orgs sometimes have great mentorship.
    - My recommendation in theory: Try first with an org that has both great mentorship and a skillset that is useful for EA, maybe you'll get accepted?
    - My recommendation in practice: You probably know what would be a good personal fit for you way better than I can write here. "Personal fit" means some things I wrote below about what would be best for you personally, it doesn't mean "everyone should go work in the most impressive place"

Will the skill that you learn have demand in the future or will EA tech needs change?
I observe that EA tech needs have grown over time, I can imagine them going up even more [link], and I can't realistically imagine them going down (assuming EA keeps existing). So my guess is that things that are needed today will also be needed in the future.
Also, my recommendations below will be something like "have an amazing career regardless of EA", so I don't expect you to be at risk for "wasting your time learning something useless". Use your judgment of course, but that is my expectation.

Bonus: Do you want to work at AI Safety?

I recently spoke to hiring managers at Anthropic and Redwood Research. They both really want to hire strong developers, **even if the developers have zero ML experience** and no AI Safety research skills.

What do they ask?
A significant part of the interviews in both companies are "build a small program in 1 hour". These are not leetcode questions (for example, they don't focus on the solution's complexity), but if you're good at solving leetcode questions quickly, and writing code quickly in general, then you'll probably do well in those interviews. Anthropic said this is similar to what they ask.

How to prepare?
Here are some ideas they suggested:

- Have a lot of responsibility in a small startup, where you have to do a bit of everything, including devops and product. You don't have to be an expert at anything, but being able to see the big picture is useful.
- Work with very strong engineers that give you really good mentorship, like at FAANG or a quant trading firm.
- Practice debugging distributed systems. Even a "map reduce" algorithm is a start.

I personally think that practicing writing tiny programs that actually do something in 1 hour would be useful. For example:

- A game like snake or so (that kind of mostly works)

- Leet code problems. Specifically if they require writing a lot of code, as opposed to being complicated mathematically.

Adding: Ought

“Even for internship we're looking for people who can be pretty autonomous at ML projects pretty soon”

So Ought are different here.

How to build skills quickly #1: Get mentoring and feedback

Ask about mentoring before applying

TL;DR: When applying, ask “will I have 1 hour every week where I can ask a senior developer questions and get feedback to help me improve?”
This will filter out lots of jobs, such as jobs where you are the only developer.
Also, some places may say they have much more than that.
Common mistake: Asking something very subjective
For example: “will I learn a lot?” or “will you help me improve?”.
Almost all hiring managers would reply with “yeah of course!” to those.

Bonus: Try understanding how good your future colleagues are
I don’t think I can explain how to do this, but it’s probably worth trying.
Bad idea: Believe the hiring manager when they say “we only have the best people!” (Fun fact: I used to totally believe these claims myself).

Other common questions about mentoring

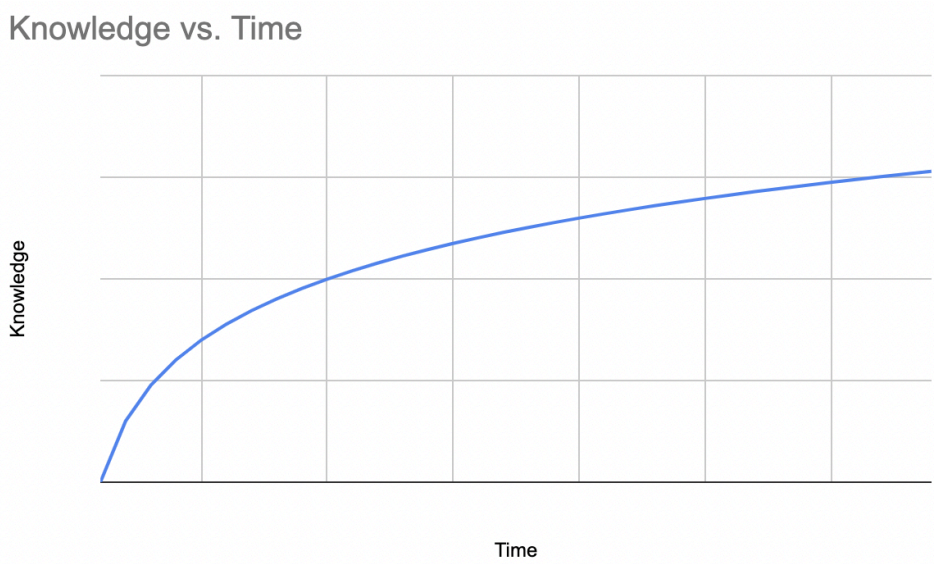
These questions are out of scope for the post, but I’ll at least mention them as having solutions:

- Q: How to get the most out of my mentor? What to ask them?
- Q: Am I wasting their time? Am I asking them too many questions?

Feel free to contact me about these.

How to build skills quickly #2: Variety

Here’s my made-up chart who’s point is “the learning curve is steeper at the beginning, when we’re learning something new”:



When we do something that we already know how to do, we often learn less fast.
Magic solution: Go do something new.

Q: Isn't it good to become an expert?

Yes. But this isn't the common situation.
"There is a difference between getting 5 years of experience and getting 1 year of experience 5 times".
For example: If you already know how to build a django backend and you're building more functions that you mostly already know how to do - then this is not "becoming an expert" and maybe it's time to switch jobs.

Alternative framing: "Do something hard"

Or, "**don't do something easy**" or "**don't do something you already know how to do**", if a main goal of yours is to build skill.
It's probably better to apply to jobs that are "over your league".

Common mistake: Apply to whatever you already know how to do because that's where you have the highest chance to get hired

Much better: Apply to the places that you most want to work at, and if you're not accepted, then consider applying to places you have more chances of getting accepted to. You don't need to apply to a field where 1000 companies are willing to hire you, you only need 1 company, give or take.

How to build skills quickly #3: Do something you like

There is an element of personal fit within software development. Some people like frontend, some like backend, some like ML, and so on.

Common mistake: Go for whatever has "the most demand".

There is lots of demand everywhere [link], and we don't want all EAs learning the 1 skillset that has the most demand and leaving the others empty, probably.
There's an advantage to each person doing whatever they're best fit at.

It matters if you're excited. How do you feel about your job when you come home?

If you come home and think about the exciting interesting problem you have at work, this is very different from coming home after another boring day at the work that you know you "must" do. The difference in your skill in 5 years will be significant between these situations. Pick the former!

Q: Switching jobs: Will anyone accept me? / I worked here for so little time, would anyone want to hire me now?

Consider trying: apply first, without quitting your current job, and see if you're accepted. Or even cheaper: Send your CV and see if anyone invites you to an interview.

Q: Switching jobs: I need income stability

Yeah I endorse income stability. In most situations I recommend first finding a new job and only then quitting.

Common mistake: Staying in a job that is bad for you

It is common for me to hear people staying in a job with a bad boss, or not enough learning, or something else that is a really bad fit, but they stay because of one of the reasons above, sometimes the option of leaving is almost unthinkable ("quitting after less than a year??").
I am not saying all cases are like this. Sometimes problems can be solved, sometimes there are other relevant circumstances, but it seems to be useful to people when I share my prior, so here it is.

Questions about the interview process

These questions are out of scope for the article, but I want to put them here for you to know that **these are common questions, often with "textbook" answers**, it's not just you, and I invite you to talk to me about them.

1. Q: **Interviewing is not fun!** A: Yeah there are things to do about that!
2. Q: How much money to ask for? / How to negotiate for salary?
3. Q: How to optimize a job search? I want a better job next time!
4. Q: Should I learn lots of [leetcode / best practices / architecture / …] ?
5. Q: How to improve my CV?
6. Q: Should I polish my [blog / side project / article] before applying?
7. Impostor syndrome!

Plan C: Learn programming, write a CV, and get your first job

See this post.

Appendix: Can Software Developers Do Harm?

Yeah eh please don't destroy the world or anything thank you!

Some ideas on how to check if your current workplace might do harm:

- **Best:** Ask 80k, or post a question in the Forum
- **Nice**: Ask your boss if they think your work might advance the world's ML capabilities.
    - "yes": Worry.
    - "What? No, we're building video games": You're probably fine.
    - "What? No, we're building video games for the My Little Pony brand": Eh, worry again?
- **Nice**: Ask your boss if they think your product is doing harm
    - "No! I know that everyone thinks so but it's not true! My reasons almost never convince anyone who is not getting a paycheck from the company, but it doesn't matter!": Mmm.. maybe worry?

Thank you

[people] for reviewing this post!
You can suggest changes to it directly here.
If you write a post on one of the topics I mentioned as out of scope, let me know and I'll link to you