Making the World Smaller: Facebook, Internships

http://www.goldsborough.me/mws/2020/08/18/19-07-45-making_the_world_smaller-_facebook,_internships/

Peter Goldsborough Sat Oct, 17 02:25

This anonymized email correspondence is part of my series on "Making the World Smaller". See the introductory post for context.

Hey Peter! I read the article on your site about your Facebook internship, and was in awe when you said you wrote 30k LOC.

I'm currently interning at Facebook too, and have been trying to be more productive, but even though I'm happy with how much I'm outputting and I think I ramp-up decently fast, I'm nowhere near the area code of 30k LOC, and I'm really curious how you were able to write so much.

I'm entering into my midpoint review and have completed most of my summer project, but feel like I haven't written much. I didn't really care about LOC much before coming here, but my mentor has been hinting how prevalent it's used to evaluate interns so I'm trying to be more productive.

I actually plan to drop out of school if I get a return offer, so though I know LOC isn't a perfect metric, I want to exceed as many metrics as I can. Since I don't plan to have a degree, both for my self-esteem and for whenever someone points out I don't have a degree, I want to be able to point to this point of my life and say "hey I killed that, I'm good at building software, I have the knowledge and ability" by anyone's metrics.

I'm curious about your internship, the scope/scale of your work seems massive, were you able to work 8 hour days? How quickly did you ask for help to get unblocked instead of diving into docs? How did you write 30k lines, and how many of those lines were your main summer project vs stretch goals?

Sorry for all the questions, but I'm really interested in your experience and thanks.

Let me try to answer your questions. Three years is a bit of time so I don't remember a lot of details of my performance. I don't even remember that I wrote that much code. For one, it really is a stupid metric and you certainly shouldn't base your self esteem on your LOCs. I've been 2x intern manager now and yes, LOC is one metric we have to compare intern productivity in calibrations. We kind of have to use it because we only have 12 weeks to evaluate interns and so we need every signal we can get. But let's be clear that LOC depends heavily on what language you're using and the complexity of your project. If I tell intern A to write some tests for a class they may generate 1000 LOCs in one

week. Another intern B I might tell to understand ShardManager settings and figure out how to configure some load balancing constraints, which will amount to about 5 LOCs in Configerator after a week of learning. Intern B probably learned more and became a better software engineer in that week than did intern A, and intern B may have landed a 10% decrease in cpu utilization across the tier which will earn him major impact while nobody gives a shit about those 1000 LOCs of test code — they're just good to have. So don't delude yourself that writing 30k LOCs is the only thing you need to get right.

On that point, intern performance has 5 axes for a good reason. Software engineering is not just emitting code into an IDE. Productivity is one axis, LOC is one way to measure productivity. The other axes are just as important, in particular Initiative & Independence and Communication & Collaboration is where great interns set themselves apart from good or mediocre interns. Quality of Work and Learning Speed is also very important. Here are some examples of how you can set yourself apart in these axes:

- Initiative & Independence: Think outside of your project, think outside the box. Be proactive. Projects are never set in stone. Become a member of the team, not just an intern. A member of the team wouldn't take task after task in his/her project. A member of the team would question what is the best way to arrive at some solution, and what else can be done outside of that solution to help the team and drive the mission forward. Do things that people don't ask of you. Suggest ways that the project could be altered to overshoot the project. Own your project. In the last few weeks of my internship I was effectively driving my own time. I had things on my mind that I wanted to do to help the mission and my manager no longer had to tell me what to do. This axis is the one that sets rockstar interns apart from the other 3988 interns each summer.
- Communication & Collaboration: Again, become a member of the team. Not an intern. By the end of the internship people should treat you as an FTE and not an intern. That, literally, is also the point of the internship. You're expected to be an E3 engineer that could be put onto the team the day your internship ends. That also means communicating and collaborating with your peers. Participate in meetings, participate in brainstorms. Bring ideas to the table. And post, post a lot. Post all the time until people tell you to stop (which they won't).

Quality of Work is important too and it kind of connects to learning speed. You're expected to be able to pick up things quickly and that includes style guides and suggestions that people make to your code. Try to never receive the same comment on your diff twice (w.r.t. coding patterns). In my case I was already kind of an expert in C++ and so I knew more C++ than most people on the team and I also have a greater sense of code quality than most software engineers so my quality of work was good.

In some sense LOC is not a goal in itself. A large number of LOCs will come from you performing above average in every single axis:

- Quality of work will mean your diffs receive few comments and land quickly, reducing your iteration time
- Learning speed will mean you will understand other codebases, libraries, issues quickly and spend more time executing on tasks (writing code),
- Initiative & Independence will mean that you will know what needs to be done to complete the next task, maybe because the task was your idea. So you will spend more time coding and less time asking your mentor what to do next,
- Communication & Collaboration will mean that you will communicate problems effectively with your peers and use your peers' expertise and knowledge to bounce ideas off of them before you write some code. Because you communicate effectively and early, you will not spend time rewriting some code after putting up a diff and another engineer telling you that your approach is completely wrong which you could have found out by talking to them for 5 minutes before you went ahead with coding.
- Productivity in the end just means that you've done all of the above things right, and now all you need to do is emit characters into your IDE that achieve the goal.

I don't remember how quickly I unblocked myself. I certainly didn't work 8 hours a day, I can tell you that much. I probably worked from 10am to 8pm, sometimes midnight. I worked sometimes on the weekend. I definitely burned myself out in the end. But I don't really regret that. You have three months to impress and land a return offer, so my philosophy tells me that it's better to spend 12 weeks working my ass off 14 hours a day and then taking a week of vacation in the Bahamas (I wish) than it is to coast through the internship at a comfortable pace. This is not what most people will tell you, but it's just a life choice you have to make at one point. I was never the smartest kid, I got ahead by working harder. So one factor why I landed 30k LOCs of code is certainly that I made 50% more time to write code where others were doing idk what.

I'm not sure your idea of exceeding as many metrics as possible will really validate you as a software engineer after you drop out. I don't know if you already figured this out from my cv or post or whatever, but I dropped out after my internship to join Facebook full time. At the end of my internship my team manager pulled me into a room and offered me a full time offer if I wanted one instead of a return offer. I have three years of experience living with this decision now, and I have no regrets for sure, but I can tell you that nobody checks your performance later on to evaluate your worth… and I don't think if you write "Wrote 30k LOC" on your CV as a way of making up for lack of a degree will work out in the way you expect. And it will not mean that you're good at "building software" or that you have "knowledge" or "ability". So my point is that while I commend your

motivation to exceed, to validate yourself as a software engineer, to impress people, I think your strategy of achieving any of that by writing more code is misguided. There's a lot more required to exceed, to validate yourself and to impress people. I outlined the axes above and you should strive to exceed in each one if you want success and validation.