Having coffee yesterday with a colleague, he quipped “everyone is just a developer” in the context of team building, recruiting and the growth of our team-members. While I agree with the sentiment, I had never simplified it to such a degree, and I really enjoy thinking about it this way.

The importance of this is multi-faceted:

Anyone working in the software industry will tell you that as an organization grows, so too does the bloat of the team structures. We go from having small teams where everyone does everything, to larger teams with scrum masters, technical project managers, program managers, product managers, and countless other titles that unavoidably overlap and cause confusion of clarity on the team.

Second, within the engineering sub-teams as they grow, our colleagues feel a constant pressure to distinguish and differentiate themselves amongst the sea of “software engineers” so many organizations fall victim to title saturation, or worse — title inflation. What I mean by this is we start creating specialized job titles for specialties within our teams. Frontend Software Engineer, Backend Software Engineer, Database Engineer, etc.

When you look top down at a growing organization it may not be immediately evident why I would caution against this, but having lived it both as an individual contributor and now as a leader, I can say unequivocally that several things are likely to result from this organization approach:

1. If you’re in an “agile development” environment as many (all?) of us are these days, your roles become fuzzy, and ownership and turf wars are inevitable. Who owns the backlog, the technical project manager, or the product manager? Who owns the delivery dates and communication to customers; the program manager, or the scrum master, or the engineering manager?
2. If you haven’t seen the film *Office Space* I encourage you to do so. It is a wonderful parody of a software company in the 90s that has many embellishments that are still applicable and guaranteed to entertain. One of the most famous being that the protagonist laments at several points that he has five bosses. When a single team is beholden to an engineering manager, product manager, project manager, program manager, and scrum master, this is exactly what you have created for your engineering teams, and as we learned in *Primed to Perform* – that is a major detractor to

performance.

3. Building on the performance factors, when many professionals feel they need to engage in turf wars for ownership of their own job due to overlapping responsibilities of a bloating team structure, nobody wins. Your Engineering Manager will fall victim to meeting over saturation regurgitating the same date projections and status updates to every *manager* on his team, and will lose focus and motivation to continue assisting the team in actually delivering what they spend all their time in meetings discussing and promising. Worse yet, your teams will inevitably fall into a culture of date obsession as a result of the turf wars, because those same managers will be forced to cling to the dates as the items they can "own."
4. Lastly, and most importantly, within the engineering group itself, an emphasis on specialization will create a "bus factor" problem that you do not want to deal with. That is to say, if your Database Engineer is the only contributor responsible for managing your data, and they happen to be on vacation or unavailable, their colleagues had better know the system(s) as well or your customers (and then *you*) are going to feel very real pain.
5. Building on the point above, specialization also breeds deeper ownership of technical domain, such that your collaboration will decrease, and problems will be solved in silos. If your Frontend Engineer is the owner of the application UI for example, they will need to be spending half their time staying up on best practices and tooling and the remaining half on actually pounding on the keyboard in order to be even half as effective as two individuals. How else will you avoid comfort zone bias and monolithic tendencies within your code?

In nearly two decades of working in this industry, I could itemize even more bullets but suffice to say, the simplified TLDR of this post is simply that "we're all just developers" and when our teams are built around that ideology everyone wins!

Now, that is not to say there is not meaningful value in program, project, and product management, as well as in the discipline of the scrum master role. Absolutely there is, and in disciplined organizations those roles bring significant economies of scale and value across multiple teams.

What I caution here is particularly with smaller teams, and in larger distributed teams where specializations are able to be disseminated amongst the team, I would almost certainly recommend that to hiring or creating a role for a specialization parallel to the team.

Build your teams such that everyone does a little bit of everything. Your UI designer will benefit in the long run from understanding how your API endpoints are built. The engineer formerly known as SRE will bring more value to your team if they understand how the data is modeled for your API, and how the javascript client consumes that data from UI to database to Kubernetes pods.

If you're lucky enough to be building something where your product role(s) parallel as engineers, consider yourselves a special breed of unicorn and relish the opportunity. If on the other hand, you're in a more common scenario where product manager is the "voice of the customer" then I would strongly encourage you to identify ways to make the product manager an integral part of the team such that they are just as familiar with the full-stack of your product as any engineer would be from a holistic perspective, and if at all possible involved in the day-to-day operations. Maybe they could even write some code?

Rather than have a single individual responsible for managing, scheduling and coordinating the scrum ceremonies as a Scrum Master, force your entire team to operate round-robin as scrum masters building those skills across the entire team. They will become more invested in the process, more collaborative and interested in improving your teams processes, and ultimately more efficient at all of the above.

In the end, we're all just developers.

Originally posted on LinkedIn:
https://www.linkedin.com/pulse/were-all-just-developers-jake-litwicki/?published=t