Maya Pegler-Gordon

Professor Johnson

Software Development Lab (CMSI 401)

Wednesday, October 7th

Maya Pegler-Gordon's Response to A Software Architect is Who You Blame

In Yegor Bugayenko's 2018 article, A Software Architect Is the Person You Blame, Bugayenko outlines the fundamental differences between a software engineer and a software architect. Bugayenko addresses the common misconception that the software architect title is merely a badge of honor or "status symbol" for long-term, senior engineers. He challenges this idea, stating bluntly that "this assumption is wrong." (Bugayenko) Bugayenko identifies being accountable for the project's successes and by consequence, the project failures as one of the important and distinguished responsibilities of the software architect role. The software architect is "who you blame" when things don't go according to plan (Bugayenko). Bugayenko describes the position as a sort of functioning "team guide," the team member who brings the project from it's "initial vision" and ideation process to a fully functioning code base and product (Bugayenko). The architect is responsible for setting priorities, assigning responsibilities, and providing mentorship and coaching to accomplish the agreed-upon set goals. Another distinction between a software architect and general engineer comes in the form of their authority, and role as an authoritative figure within the team. According to Bugayenko, the software architect has the last word, even above project managers especially when it comes to the code and technical project portions. Software architects must be "more than simply a senior programmer, more than simply a leader." (Bugayenko) Rather, these positions are held by individuals who are able to "stand as a gatekeep for quality and a guiding vision for their team." (Bugayenko) They must be able to understand their team, their strengths, and their weaknesses. These unique responsibilities and skillsets have led the tech industry to distinguish this largely overlooked role. Although the role is often overlooked or assumed to be similar to a senior developer, Bugayenko outlines a clear argument as to how they differ and why they are a crucial piece of a successful team.

After reading Bugayenko's article, I see some major benefits of employing software architects, as well as some of the consequences and added company responsibility that comes with the role. On the one hand, as Bugayenko outlines, software architects can offer a guiding voice throughout a company's product iteration process from ideation to release. Having a single individual responsible for maintaining the progress of a particular project offers a lot more guidance and organization to the team, all while minimizing the mishaps along the way. By employing to be the "the buck stops here" person, there is a more direct investment in the progression of a product launch and will effectively decrease the number of bugs, delays, and issues seen. In contrast, by giving a single individual all of this massive power and influence over the development, direction, and ultimate delivery of a company's products the company must take on the responsibility of being sure to hire the right people for the job. Rather than vetting individuals for specific technical skills and past experience, as they often do with general developers and programmers, companies must design an entirely new in-depth interview process that ensures those being hired into the role will take the company in the direction they want to be headed. Nothing would be more detrimental to a company's product than hiring someone with an entirely different vision and priority set to the company. In my response to Bugayenko's article, I will dive deeper into the major advantages and risks of hiring as a software architect for a company I have identified.

A major benefit of employing software architects comes with the nature of software development. In software development, bugs are bound to happen. Timelines get delayed, product priorities shift, consumer demands change course, etc. In a typical software team, these bugs and issues can be largely overlooked or swept under the rug to live in the backlog for months to come. After all, why deal with it now when it can be someone else problem. In a team with a software architect, having a designated person responsible for overseeing these timelines, product releases, and more will likely minimize the frequency of bugs, rollbacks, and crashes. After all, software engineers are human and humans are always more attentive to risk when it's their head on the chopping block. Given the nature of software development, software architects offer a unique role in ensuring the smooth rollout of key products.

Along with any authority figure and source of power, comes a lot of responsibility. If a team or company is going to give an individual the immense influence of a software architect, the role application and vetting process must be incredibly thorough. Typical software developer interviews which focus on a range of technical challenges surrounding data structures and algorithms, as well as behavioral questions, targeted to assess a candidate's ability to problem solve, collaborate, lead, and learn. As Bugayenko highlights, the responsibility of a software architect goes way beyond that of an engineer and requires a different set of skillsets, and thus a different interview process. These interviews must be formatted to assess similar technical competencies and soft skills, but also look for a more unique skill set. When designing these interviews I would filter for candidates who are able to identify strengths in other individuals to help their team contribute as much as possible, candidates who can take responsibility and blame, have demonstrated initiative, align their beliefs and values with the company, and are able to see the big picture. If the architect and great company or organization are misaligned on their values, I could see a major issue arising amongst companies who employ software architects.

I found Bugayenko's article very informative for the role of a software architect. Prior to engaging with the article, I was one of the individuals Bugayenko has a bone to pick with, assuming that software architect was just another rank in the never-ending ladder of software developer promotions. After learning more about their unique roles and responsibilities, I will be sure to reach out to any software architects on my team or adjacent teams once starting full time to pick their brain and learn about their perception of the differences between their earlier roles as developer and current responsibility set. Who knows, maybe Bugayenko has even convinced me to aspire to the software architect role!