# Working with Engineers

Bryant Chou, CTO – Vungle
10/16/12

Software Engineers have the unique ability to be in two different mental states at the same time. Creating and fixing. We spend roughly 10-15% of our time creating, and the remainder of our time fixing. This is a mentality that is not unique to software engineering but difficult for non-software people to understand.

All engineers have the difficult position of constantly being afraid of failure. Civil Engineers fear for their bridges' structural stability during seismic events. Software Engineers are constantly afraid of bugs that are in their code, impacting business. This is usually the one role in a Silicon Valley startup company that can make and break your company. What about lawyers? Can't they structure a poor deal that hurts the company? Surely, but there are no lawyers in the Silicon Valley that can recuperate lost revenue when your SaaS goes down. When treated appropriately software engineers will go above and beyond the call of duty, creating stellar companies out of lackluster ones along the way.

Because of this, engineers that work in small startups live in a state of anxiety. We fear the worst, and when shit hits the fan, we have to drop everything and address it immediately. Or the whole company is at risk. We wear many hats. There are not enough engineers and capital in SV[1] startups to afford engineering teams that specialize. Early stage companies must work on everything that needs to be worked on. We also work 24/7. For times that we don't spend in the office, we all wear pagers[2] that will ring us every time our servers hiccup.

The one way engineers can assuage a constant state of anxiety is to know that they have a very well written piece of software. But how is good software written?

There are two simple ingredients to a well written piece of software. The first is the engineer. The moniker of "rockstar" engineer[3] is often overused in SV, but it's very well known that said engineer is worth 10 or 20 average joe engineers. A SV startup can hire as many rockstars as they want, but they would ALL be ineffective if the second ingredient were missing: flow.

Flow can best be described as a stream of consciousness. For example, writers have flow. Stephen King has said that he has written an entire book over the course of an evening over a bottle of wine and some psychedelic fungi[4]. This incredible level of productivity is

[1] *Silicon Valley*

[2] *smartphones (not those archaic black things doctors still unfortunately have to carry around)*

[3] *Rockstar engineers are so 2011. It's all about ninja engineers now.*

[4] *Circa 2012, software engineers prefer beer instead of wine, and BAWLS energy drink instead of mushrooms.*

well documented in the psychological academic community and is very easy for even a freshman college student writing his term paper to understand[5].

However, the difference between software engineers and authors of fiction is 15x. Fifteen?

Though engineers and writers both use the computer, software engineers must take into account *at least fifteen* different streams of consciousness and have the ability to deliver all fifteen streams succinctly and accurately into their programs. For example, this is a list of common things a software developer may have to think about whenever they are coding:

1. Concurrency
2. Speed
3. CPU usage
4. Memory usage
5. Database usage
6. Data integrity
7. Timing
8. Language best practices
9. Object Oriented Design
10. Modular development
11. Production implications and the difference between development environments
12. Multi-tenant implications
13. Multi-server implications
14. Unit tests
15. Testability of code
16. Continuous Integration
17. Deployability
18. Configurability

Granted, the engineer may not be thinking about ALL of these things at once, but these are all essential ingredients to achieve less stress AFTER code is delivered and running in production.

Why is it important for a developer to have good flow during development? It's because bugs are costly. To find a production bug may take hours, days, or even weeks[6]. Bugs are avoided by strong and experienced software engineers managing fifteen streams of consciousness at the same time, laying down code without interruptions. This saves enormous costs down the road, and will make engineers much happier as a result.

*In the 1960s, a high level executive at Bell Labs took his daughter to work with him. He began showing his daughter around the premises, and they eventually arrived at the research and development floor of the building. When his daughter saw computer operators sitting in isolated,*

[5] *Millennials, known for the various ways they distract themselves on the internet (ie. Reddit), have probably experienced the sensation of flow a lot less than their predecessors. However, they are much better at multi-tasking (which is impossible to do when flow-ing)!*

[6] *At Veritas, I was told of a story of a team of engineers spending 3 months to even locate a bug*

*soundproof rooms by themselves he asked them why they were sitting alone. He replied, "they're engineers honey, they need to have their own space so they can think about very hard problems."*

To help people understand what interrupts flow, here is a list of very common interruptions to software engineering flow:

- Loud sounds or loud people
- Interruptions
- Barking
- Slow or down internet
- Sales people on calls
- Hunger[7]

Despite this constant state of anxiety, engineers at startups are at the end of the day, very proud of their work. They have accomplished things that no one else have ever done before. They have delivered value to the company, it's customers, and it's investors. With all that said, we studied to become engineers, and we remain engineers because of the 10-15% of the time we can spend *creating* not fixing.

## Appendix

### How to tell if an engineer is approachable

Does the engineer have headphones on?
Does the engineer have notes strewn about the desk?
Does the engineer look deep in thought?
Does the engineer seem like he has tunnel vision?
Does the engineer look at his monitor that is opposite you[8]?
Can your question be answered through email?

If yes to any of the above, it's probably not a good idea to approach the engineer at the time.

### How to effectively communicate with engineers

This could warrant another post, but it can be summed up quite succinctly here. Engineers require facts and tangible information that relates to them. Also, engineers' brains are stuffed with so much technical information and knowledge, that they are usually lost of the day-to-day facts of the business. Most will not know which clients drive what percentage of the company's revenue. That is why it is very important to ask yourself two things before even approaching an engineer.

1. How urgent is this matter

[7] *A common analogy in Silicon Valley is: "Engineers are like mushrooms. You keep them in the dark, feed them shit, and watch them grow". This is pretty much true, but they prefer pizza/catering over fecal based fertilizer*
[8] *Assumes a multi-monitor setup*

2. If it isn't urgent, can it be asked via email[9]

When posing a question to an engineer, it's important for the engineer to have context. A lot of engineers are business savvy and will often propose an alternative, easier solution than one you were imagining. Explain who the customer is, why this request is important, or what will happen if it doesn't get addressed.

## How Engineers Estimate Time

Engineers are notoriously bad at estimating time. The best way to convey this is via the following "It'll take" table:

**It'll Take** Table

| What Engineers Say | What It Really Means | |
|---|---|---|
| A few minutes | An hour or two | The engineer usually has to wrap up whatever he's working on, before context switching. |
| About an hour | Several hours | The engineer knows what to do, but thinking it's something that isn't very important, will procrastinate for an hour, or take a lunch break before starting. |
| A day or two | Two days | This is usually the most accurate response. It's usually a task that requires enough time for the engineer to dedicate a full day – which means he'll not do anything that day, opting for dinner with his girlfriend, then spend all day & night since he doesn't like to leave work feeling like he didn't accomplish anything. |
| A few days | 1 week | |
| A week or two | 1 week - 1 month | These are projects that require a lot of thinking. A hardworking engineer will |

[9] *Unless you know the engineer doe not respond to emails, or if you've never sent them an email before. If you've never sent them an email before, ask your peers if the engineer is responsive to email*

| | | |
|---|---|---|
| | | usually finish in a week, but engineers why either a) don't think it's that important or b) are lazy may take up to a month and take advantage of the fact that they have several weeks to get it done before people start asking him about it. |
| A couple weeks | Never | Usually there are too many things going on that relegate a project as large as this to the BOB[10] (backlog of blackholes) |

[10] *Rockstar engineers feel like they can build Facebook in a "couple weeks", but that usually fails when they are reminded that it's actually not that easy*