Why I Strive to be a 0.1x Engineer

Posted January 25, 2016 under XP.

There has been more discussion recently on the concept of a “10x engineer”. 10x engineers are, (from Quora) *“the top tier of engineers that are 10x more productive than the average”*

Productivity

I have observed that some people are able to get 10 times more done than me. However, I’d argue that individual productivity is as irrelevant as team efficiency.

Productivity is often defined and thought about in terms of the amount of stuff produced.

“The effectiveness of productive effort, especially in industry, as measured in terms of the rate of output per unit of input”

Diseconomies of Scale

The trouble is, software has diseconomies of scale. The more we build, the more expensive it becomes to build and maintain. As software grows, we’ll spend more time and money on:

- **Operational support** – keeping it running
- **User support** – helping people use the features
- **Developer support** – training new people to understand our software
- **Developing new features** – As the system grows so will the complexity and the time to build new features on top of it (Even with well-factored code)
- **Understanding dependencies** – The complex software and systems upon which we build
- **Building Tools** – to scale testing/deployment/software changes
- **Communication** – as we try to enable more people to work on it

The more each individual produces, the slower the team around them will operate.

Are we Effective?

Only a small percentage of things I build end up generating enough value to justify their existence – and that’s with a development process that is intended to constantly focus us on the highest value work.

If we build a feature that users are happy with it's easy to count that as a win. It's even easier to count it as a win if it makes more money than it cost to build.

Does it look as good when you its compare its cost/benefit to some of the other things that the team could have been working on over the same time period? Everything we choose to work on has an opportunity cost, since by choosing to work on it we are therefore not able to work on something potentially more valuable.

Applying the 0.1x

The times I feel I've made most difference to our team's effectiveness is when I find ways to not build things.

- **Let's not build that feature.**
  Is there existing software that could be used instead?
- **Let's not add this functionality.**
  Does the complexity it will introduce really justify its existence?
- **Let's not build that product yet.**
  Can we first do some small things to test the assumption that it will be valuable?
- **Let's not build/deploy that development tool.**
  Can we adjust our process or practices instead to make it unnecessary?
- **Let's not adopt this new technology.**
  Can we achieve the same thing with a technology that the team is already using and familiar with? "The best tool for the job" is a very dangerous phrase.
- **Let's not keep maintaining this feature.**
  What is blocking us from deleting this code?
- **Let's not automate this.**
  Can we find a way to not need to do it all?

Identifying the Value is Hard

Given the cost of maintaining everything we build, it would literally be better for us to do 10% the work and sit around doing nothing for the rest of our time, if we could figure out the right 10% to work on.

We could even spend 10x as long on minimising the ongoing cost of maintaining that 10%. Figuring out what the most valuable things to work on and what is a waste of time is the hard part.

Website

Your Website

Your Comment Here…

Submit Comment

More Articles

Admin