What Hackathons Teach About Making Products

Justus Perlwitz

2018-05-04

Contents

I like hackathons. Hackathons are competitions that run over multiple days — typically a whole weekend — and in which the participants — programmers and designers — solve a challenge in order to win a prize. Typically, there are a few challenges that are all based around some theme — blockchains, sustainability, democracy, and so on.

Not only are hackathons interesting because of the subjects that are covered in the challenges, they also become an interesting study in startup archetypes. The majority of participants are in some way affiliated with startups or entrepreneurship in general, and everyone is represented fairly. From genius backend programmers to brilliant product designers. From loud and obnoxious brogrammers to people who don't know how they got here in the first place.

The teams that those individuals make up are equally interesting. There are teams that diligently fulfill all the tasks in hackathon, only to create something that not even they can fully understand or explain. There are teams that create beautiful concepts for world-changing products, only to be unable to answer how they'd implement it, when asked by jury. Some teams hit the nail right on the head, and produce something that is well-designed and compelling. Something that makes you think: *Why didn't I think of that?*

Of course, what all those teams have in common is their ability to produce an enormous amount of output — code, design, concept, and so on — in a short amount of time. Yet, the result is not always *good*. Or rather, just combining a lot of skills won't create a whole that is greater than the sum.

When I read The Lean Startup for the first time, I was impressed by the noble intentions of the author, Eric Ries. He describes his approach to creating startups as one that *reduces waste*. If you combine all the designer hours, developer hours, business development hours and so on that accumulate during a hackathon, you would expect only great results.

And at the same time, if we are not seeing great results, we are ultimately faced with a lot of waste. Waste of energy that was needed to create failed concepts, waste of energy to stay awake for more than 24 hours, as is typical in a hackathon, waste of time that could be spent to get some well-deserved rest before the daily work-life drudge starts again on Monday.

Of course when I talk about waste, I am not only talking about the specific case of hackathons. They are an easy example to talk about, since they have a very well specified deadline, clear evaluation criteria, and require a lot of concentration in a very short time span. They become a case study in project management and team interaction.

And if we successfully figure out how to make hackathons work, we can apply this approach on a bigger scale. Hackathons become a way to understand how to model project management in real companies as well.

But unlike most companies, hackathons have the potential to show how to make project management work without complicated middle management. Management rarely exists to make the job of employees easier, but mostly to protect the company from their employees. Hackathons, on the other hand, give us a safe space where we can experiment with new approaches without fear of repercussion. After all, if our little weekend experiment fails, we will still walk away having learned a great lot about what *doesn't* work.

So let's head back to the drawing board and think about how to make great teams that can build great products. But first, I would like to introduce some constraints:

- We talk about product teams that contain only developers and designers;
- We create products that have well-defined target audiences and serve a specific purpose;
- We model our approach to not require a team lead, but will create an environment of shared responsibility and leadership.

Then, I would like to describe my ideal designer and developer. A designer should have a broad range of skills in product and visual design. It's less about mastering all skills, but having the ability to get the team to 80% done in a short amount of time. Similarly, I would expect the developer to bring a broad toolkit centered around web technologies and be particularly good at understanding architecture constraints and quickly estimating the feasibility of product ideas.

My impression at Hackathons so far was that the team with the most convincing argument for why their product as a whole, when finished, will work has the highest likelihood of winning. Simply creating a form with a few buttons that randomly beeps when the presenter remembers to click the right sequence during the product pitch is not enough. As a matter of fact, the prototype comes secondary to the concept. The concept and the team's approach to implementing it matter the most, as this is where the real money lies.

So for the next few weeks, I want to explore the following questions and come to some kind of conclusion in the form of a hackathon product blueprint:

- How can I develop a strong product concept with market appeal?
- How can developers and designers work together in a complementary way?
- How can we share ownership and leadership equally?