An

AppSumo

Production

THE LEAN STARTUP ACTION VIDEO

Stop wasting time on tasks that dont matter, learn to be lean and efficient.

with

Eric Ries

Author of "The Lean Startup"

*TRANSCRIPT

Paul: Hi, everyone, I'm Paul Hontz from the Startup Foundry. We have Eric Ries here today to talk to us about the Lean Startup methodology. So, Eric, introduce yourself.

Eric: Hey, everybody, I'm Eric Ries. I guess I am the founder of the Lean Startup movement, whatever that means. I spent something like ten years as an entrepreneur before I started then blogging on blog called Startup Lessons Learned and writing about this concept called Lean Startup. The fundamental thing I learned as an entrepreneur is that most of the work I was doing really didn't matter. I kept building products that nobody used. I kept having companies that were not successful. I really came to believe that the myths that we hear about what makes a great entrepreneur – perseverance, vision, determination – not that those things are bad, but those things are not sufficient for our companies to succeed. If we're going to get better as an entrepreneurial industry at improving our chances of success, then we need a new and more scientific method to do the work that we do every day.

Paul: Okay. So in this video – it's about an hour or so long – what are people going to be able to do after watching this?

Eric: Well, I'm hoping that at some point during this video you'll have a moment where you say, "Oh, my God. I need to stop watching this video and go out and do something immediately." Because most of the problems that entrepreneurs have are urgent, but they don't know it. This isn't going to be about, oh, make sure you're a better designer or make sure you have better product vision. Most advice, I feel like is of the form of just like be more awesome, and if you're not awesome enough, too bad for you."

But this isn't what we're talking about. In Lean Startup, we fundamentally believe that today, right now, there are things you could do that would increase your business' odds of success dramatically.

Paul: Okay. The way that we're going to do that is by learning build, measure, and learn, how those three things interact in that cycle.

Eric: Yeah, it's a continuous process of, we call it continuous innovation, where we're constantly doing this thing of build, measure, learn as fast as we can. So that's the fundamental flywheel, the engine that powers all startups.

Paul: Great. Let's jump into this.

Eric: Okay.

Paul: In this section, we're going to talk about validating your idea and learning, figuring out which ideas you should be going after. So if you're like most entrepreneurs, you a have a hundred different ideas, a list of different startup thoughts, and all this crazy stuff. How do you lock in on one idea?

Eric: Well, the truth is that every entrepreneur in their heart has some kind of vision of where they want to go. Some people call the vision the idea, and some people call the product the idea. Different people think about it different ways, and I don't think that the distinctions are that important. But I call it the vision of where you want to go, which is, at the end of the day, if I'm successful, sure I'll make money, sure I'll be famous, all that good stuff, but how will I know that the world is a better place because my company existed than if it didn't exist? So really it's connecting to what's that benefit for customers that I really care about.

Often it's quite indistinct. It's not like I know exactly when I start out building digital cyber cash for PDAs, it's not exactly clear that I'm going to wind up as the online payments provider for people auctioning beanie babies. It's not important that you know the future, but if you look at all the different pivots that PayPal did along the way, what you see is there is a consistent feeling that there's something wrong with online payments, with the way currencies are made, with the idea that it should be easier and more convenient for people to exchange cash online. So the vision is that statement of something to do with making the world better around payments. It starts out quite vague, and it really only gets specific as we learn more.

It's really important to start with that, because there's a lot of science in the Lean Startup. It's a scientific method for testing ideas and for discovering how to build a sustainable business. But even science requires a great deal of intuition, because a scientific experiment is only as good as the hypothesis you formulate. How do you come up with a great scientific theory? You don't just go into a spreadsheet and turn the crank. It re-

quires a lot of testing and experimentation to develop that internal intuition to find a good idea.

Paul: Okay. So take us through some of those experiments that startups can run to figure out if the vision is correct.

Eric: Yeah. So the truth is that everything a startup does, every activity is already an experiment. I saw a blog post recently that said something like, "Netflix is very famous for testing a lot of different things." They test pricing, they test different website flows, they test different offers. They're constantly testing. But this blog post was saying, "But Netflix didn't test the idea of sending people DVDs by mail. That was their vision and they kind of willed it into being." I just thought that was so funny because we focus on the micro experimentation as if that's what it means to be a scientist, it's just dealing with little test tubes in a lab. What I think we lose sight of is the fact that Netflix itself was an experiment to discover if it's possible to make a profitable and sustainable business around this concept of DVDs by mail. If you look into the story, you see a lot of things have changed along the way. It took them quite a while to figure it out.

So what we want to test, we want to figure out what are the risks that we're actually taking? What, to borrow a phrase from Randy Komisar, I call the leap of faith assumptions. So if you're building a business that is going to be powered by advertising some day and you want to grow through viral growth like Facebook or Hotmail did, then you have two really big leap of faith assumptions. One is, can I build a large collection of people's time and attention? That's the asset that I'm creating. Can I do that basically for free through viral growth? Secondly, if I have a massive asset of people's time and attention, can I make money by selling that to advertisers? So just by analogy, it's like saying, "Can I find a diamond mine where I can extract diamonds?" And a second is, "If I had a lot of diamonds, would I be able to sell them to anybody?"

Now, in theory, both of those are important leap of faith assumptions. If the answer to either of those questions is no, you have no business. But which one is the riskier assumption, selling diamond or finding diamonds? Clearly finding diamonds is where the risk is, in that particular business. So in that particular business you want to figure out, okay, my most risky leap of faith assumption is really this thing about viral growth. Let me go spend my time experimenting and testing on that. Maybe later I'll get to this question of how much is the attention worth? Or maybe I'll do a little bit of that testing in parallel, but my energy will be focused here's where the risk is, here's where it makes sense to de-risk.

Paul: Sure, okay. So we looked at that from a free service that's going with advertising. What about a paid service? Let's say Company X will charge a $30 monthly fee for their service. How do you measure that?

Eric: I basically believe there are three fundamental types of what I call engines of growth, each of which is a standalone feedback loop that helps startups grow. This can get a little bit complicated and abstract, so we'll try to make it concrete and then we can come back to it later to get into the details.

But the short version is you have viral businesses, like we were just talking about. They grow fundamentally though each person being a catalyst for other people to join. Then you have the paid businesses that grow through paid advertising, where they make money for every user and they reinvest in advertising. You actually have a third category of sticky or retention based businesses that have high customer lock-in or very long retention periods, like World of Warcraft or a cable service, something that is fundamental and people want to stay signed on and they rarely sign off.

Each of those three engines of growth has a different feedback loop that powers it, but what they all have in common is they're all sources of what I call sustainable growth. The rule of sustainable growth is really simple. It simply means that new customers are a result of the actions of past customers, either because the past customers invited them or because you're using revenue that you made from past customers to get them, or because the new customers are the past customers coming back because they can never stop coming back. Each of those sources of sustainable growth is different from like a one-time PR campaign, even different than one-shot SEO optimization, or doing a big publicity stunt, or even just blowing a lot of money on a Super Bowl ad. Those are all unsustainable sources of growth, and we're just going to ignore them for the purposes of what we're talking about.

So let's take the paid engine of growth to really answer your question. In that case, the speed with which a company will grow is dependent really on only two variables. There is how much money do I make per customer, and how much does it cost me to get each customer? So if it costs me $1 to get a customer, but I only make $0.80, then my business is going to shrink, not grow.

Paul: Right.

Eric: But if it costs me $1 to acquire a customer, but I make $1.10, then I'm going to have 10% growth per cycle, and cycle time of course will vary depending on the kind of advertising. That's the same rate of growth as an enterprise sales business, where it costs $10,000 to acquire and service a customer, but I make $11,000 in revenue. So it's what we call the marginal profit, how much extra money do I get per customer that will determine everything about the company's rate of growth.

Paul: So you're factoring that in when you're validating your idea?

Eric: Exactly. So if you want to build a business like that, you just want to find out the answer to that question as soon as possible. How much is it going to cost me to acquire a customer, and how much am I going to make per customer?

So if you notice all of the engines of growth, including this one, the key numbers that we need are all per customer metrics. They are not gross or what I call vanity metrics. So when we look at startup board meetings or you read about startups on all of the startup blogs, especially certain tech outlets, all the numbers you see are just like vanity metrics one after the other. Listen, I'm all for using vanity metrics to make your competitors feel bad by putting them in a press release, like that's a time honored tradition, I understand that. But we get confused when we start to use those vanity metrics to guide our own business decision making.

So if you have a million customers, so what? Why is that even a good thing? That's like the old joke about the grocery store. If you losing money per customer, but making it up in volume, that's not actually a good thing. You're busy destroying things. If all you care about is growth, like a cancer, than just create a Ponzi scheme. The fastest way to make a high profit fast, fast growth is just to take money and you return it make to the new people as suckers. So if we're trying to do something more than a Ponzi scheme, then the crap we read in the press is not going to cut it for understanding whether our business is actually creating value or not.

The cool thing about per customer metrics, they do not require large amounts of money or numbers of customers to test, because even with 10 or 100 customers, we can still be measuring how much money do we make per customer? What's the retention rate per customer? What's the viral spread per customer? If I have a super viral product, a product that is so viral people can't help but contaminate all their friends with it, then all I have to do is expose 10 people and I'll immediately have 10 more. Well, if I think I have a super viral product and I expose 10 people and I get zero more, is that a statistically significant result?

Paul: So with the Lean Startup methodology, you're trying to iterate as quickly as possible and test those ideas. So what are some ways that you've found that help to optimize ways of testing your ideas? So going with your example, are you saying a unit test? Are you planning things out in advance, or do you just release and let it go and see what happens?

Eric: This is a really tricky one. We talk about the build, measure, learn feedback loop. So the true answer is that this is the continuous process that is happening all the time, and we want to go through the loop as fast as possible. But notice that that is not the same thing a just going as fast as possible in general. There are a lot of people who think being a startup means hustling and going crazy and just being very frenetic and having a lot of action, and that's actually wrong. All that action can be as distracting as helpful.

So we want to be very rigorous about figuring out, "Well, what actions do we take to actually help us get through that build, measure, learn feedback loop as quickly as possible?" Even though we write the build, measure, learn because that the temporal order of the actions, our thinking needs to work in reverse because we've got to start with, "Well, what do I need to learn?

Then in order to learn that, what would I need to have measured? Then therefore, what do I need to build to get that measurement?"

So take the example of a product that's going to have an online service component, where people are going to sign up for the product, okay, very simple.

Paul: With paid or . . .

Eric: It doesn't matter. [inaudible 0:13:36] of engines of growth. For almost all online services there is a sign up step.

Paul: Right.

Eric: There is a very simple question, which is, "Okay. When people get to our landing page, how many of them are going to sign up to do our free trial?" If people won't do the trail, you're doomed. The business is totally toast. It was just one of the leap of faith assumptions. But most companies that have an online service, like this critical number to their business' success is buried in Page 97a of Appendix II in the spreadsheet in the back of the business plan. It's just like, "Ah, yeah, Number of Customers Come In/ Percentage That Sign Up." Well, let's say it says 10% of customers will sign up in that box right there. Instead of being a tiny little box in Page 97a in two point font, that should be like a huge banner on our wall in red font. That's like, "Hello! We need 10% of customers to sign up or we don't have a business."

When you write it that way, you start to feel like, "Is the right approach really to build my product and then see if people will sign up for it?" I think the answer is just obviously no, no, no. That's exactly backwards. If what we need to learn is will people sign up, then the thing to do right away is to measure that today. So if you're listening right now to this video and you have an online service business that you're planning to release in a month or three months or six months. People talk to me about this all the time and it's like, "Well, we're almost there. We're in pre-alpha. We're three more stages away from our release." It's like, "Stop what you're doing. Stop even listening to this video and immediately get that online service sign up page up there and start to try to see can you get people to pre-sign up for the thing before it's done?" Because you're probably right, probably people are going to flock to it and you're going to have a 100% conversion rate, every person will find your offer irresistible. But wouldn't it make sense to just double check? What's the harm of double checking?

If you're not exactly sure what the product is going to be, that's okay. Just talk about the customer's problem and sell magic. We call this the Magic Test. You just say, "Hey, do you have this problem? Wouldn't you like to have this problem solved for $49.95?" Like, solved basically by magic, just describe how awesome their life is going to be after. Sign up here. If you can't sell magic, you definitely can't sell your product, because magic is the best possible product. So if that's not selling, again, it doesn't mean give up, but it means, "Hey, now we have something to think about. Maybe something's not quite right."

Paul: Okay. So what you're saying to startups is, "Even if you're not for enough to release something, put up a landing page with an option of signing up and sell your magic, and if you see people clicking through on it, you know other people have a problem. So you've solved that problem and you're learning from that."

Eric: Yeah, you're starting the process of learning. Now, of course, the next thing that happens is people will say, "Okay. I put up a landing page and 25% of people sign up and so I'm golden. I learned something." I'm like, "Who cares? 25% compared to what? In the business plan, it says 10% of people are supposed to sign up, but you just pulled that number out of a certain orifice." "Well, it's 10% because if it's 10% and then they each pay us $10, then we make $1 per customer and we believe that it will cost us $0.80 to acquire a customer." So each of these terms is a term in an equation that yields some kind of profitable business. So, "Okay. You sign up the first one at 25%, that's awesome. Well what's the second one?" "Oh, I don't know." It's like, "You don't know? Hello, red flag! Are you kidding me! You need to know all the terms."

So we call this establishing the baseline for all these input numbers, that are all critical leap of faith assumptions in that business plan. We don't have to obsess about every one. They're not all equally important, but we do want to, as quickly as possible, find out, "Well, where are we right now? What's the level of customer demand for our solution out there, right this second?"

The answer might be zero. It might be that we have something that's so new that no one even knows they want it. That's okay. But now we know some facts, and so we can start to plot strategy based on facts instead of speculation. If it truly is too new for anyone to understand, then we need to get working on educating people about why they need it or maybe we're on the wrong track. So now our situation is polarized, as they say in poker. Maybe we've got a great hand or maybe we're bluffing. We don't even know yet, but we'll go find out. So that's the kind of thinking you can do once you have real information about what the status quo is.

I just want to clarify one thing. This is not market research. The goal here is not just to do an abstract assessment of what people might want. You can do market research indefinitely. Lean Startup is about using actual product development as an experimental tool for discovering what people want. The reason why it's better than market research is it immediately puts us in a position to do the next iteration if we're right. So if we're having people sign up on a landing page and now we're like, "Okay. What do we do next?" Well, gosh, maybe the next thing we want to do is go talk to people and figure out why aren't they paying us as much money as we thought. Well, who should we talk to? In market research, it's like, "Well, let me go commission a new study and get a new focus group together." It's like, "F that. We've just got people signing up. We've got 100 people signed up." Well, that's 100 human beings that we can go talk to right now to figure out what's wrong or maybe even figure out what's right.

But there's another reason why this is better than market research, which is that oftentimes the thing that goes wrong is something you can't even anticipate that it even was an issue at all. So like when Zappos was getting started, instead of creating warehouses and distribution channels and creating another dot-com disaster, the founder actually went to a shoe store, took digital photographs of the shoes, put them online, and offered them for sale. If somebody bought the shoes from him, he would literally walk back to that shoe store, buy the shoes, and mail them to the person. So it was a classic minimum viable product, where he's using manual labor on the backend to fulfill the orders. Now not only was he able to gauge the level of interest in buying shoes online, in classic MBP style, but he also but himself in a position to have unexpected crazy stuff happen. Like maybe the way he's pricing the shoes doesn't work for online, or maybe people buy the shoes and return them, or maybe the selection is all wrong, or maybe there are people who just have an irrational fear of putting something they bought online on their feet.

Those are the kinds of issues that just might not come up in a casual conversation. If you're just talking to people, you're not going to find that stuff out. People might say, "Oh, I'd love to buy shoes online," but actually be afraid of it. They might say, "Oh, I'd be happy to pay full retail," but actually they need a discount. Or they're exactly the opposite. They might say, "No, I definitely want discounts," but then maybe the discounts make the shoes look cheap. So I could go on and on about all the things that might go wrong. Now you can spend your days as an entrepreneur worrying about what might go wrong, but there's always an infinite number of things that might go wrong. So forget that. Let's go find out what's actually going to go wrong, as soon as possible. So we put ourselves in a position for stuff to fail, and that then puts us in a position to learn.

Paul: Great. I actually heard you speak a few weeks back, and you shared a really interesting story about an instant messaging plugin that you were working on. Can you go into a quick 30 second, 2 minute version of that story, and walk us through the mistakes you made with that and how you could have avoided some of those by using this idea?

Eric: Yeah. I'll just preface this, I'll try to give you a short answer, but it's challenging because this really happened to me and it still hurts to think about. I'll preface it by saying that most of the time that startups fail, it's not because their strategy is stupid. It's just because the strategy is wrong. It's not that it didn't follow the good principles you'd read in a great book on business strategy. It's just the facts that it was based on were empirically false. So I'll give you this example.

So we created a company called IMVU in 2004, and we wanted to take the 3D avatar technology for virtual worlds and online games and bring it to the hot new social technology of its time, which was instant messaging. We had no idea social networking was on the horizon. So we were trying to make avatar based instant messaging. We were really into strategy, and we spent a

lot of time at the whiteboard, plotting all the boxes and diagrams, and today we would say getting our business model canvas in order, and we did all that stuff.

Here was our analysis. Everybody knows that instant messaging is a network effects business, which is a fancy way of saying, "If you're the only person with a telephone, it's not worth anything. But the more people you can call, the more valuable the telephone network is." Everyone in the world was already on an instant messaging client at the time. So that meant that getting people to switch to a new instant messaging client was going to be really hard, because if you want to switch, you've got bring all you friends with you. So there are high switching costs, and therefore instant messaging is an industry characterized by high barriers to entry.

When I give this analysis in a MBA classrooms, people usually nod along and they're like, "That sounds smart. That's really good." Then I say, "Okay. You think that's smart, check this out. Here's our strategy for getting around this problem. We're going to build an instant messaging add-on that interoperates with every single IM client and allows you to just automatically transform your boring 2D instant messaging into our awesome 3D instant messaging. So it's got one whole D extra than your regular instant messaging." It just seemed really brilliant because people wouldn't have to learn a new client, they would have to learn a new buddy list. But even better, you would have to use the product with a friend, which meant our product would spread virally automatically, because every time you chat with someone, it automatically sends a URL right to them through IM to download our product. So just like PayPal colonized eBay, just like YouTube colonized MySpace, we were going to colonize instant messaging with our awesome stuff.

Now that sounds pretty smart. We used to meet with VCs, and we would tell them we we're going to do this 3D instant messaging. The first thing they would always say is, "You should really make it interoperable with the existing networks." We were like, "Yeah, we got that." We were like, "We totally nailed it."

So we spent six months building this product. I mean, killing ourselves. At first we were like, "Okay. We're going to ship this thing in 180 days." We thought 180 days was like minimum viable product territory, although now it seems ridiculous, it's so long. But look, we did a really good job. It was really hard. We had to get 3D avatar technology, downloadable client, eCommerce, is all used to generate content before selling the clothes to each other, for their avatars. This was a huge product that had its own blogging and social networking features, online photo uploads, photo galleries, the whole payments infrastructure. There was a lot that had to happen in six months, okay. I was the CTO of this company. So it was my primary job to make all this code work.

We get to day 179, and we're about to release and we are terrified because look, the truth is this product sucked. It was just as likely to crash your computer as it would be to give you some kind of awesome 3D avatar experience, because look, we had to cut a lot of corners, as I say time, quality, money, thing too. Well, we didn't have that much money, and we were running out of time, so quality had to go. We were just worried that when we shipped this thing, customers would be like, "Ah, this sucks. Your brand means nothing." They'd tell their friends that IMVU had low quality. But we were like, "You know what? Let's just ship it anyway and we'll get the feedback and we'll make it better every single day. So it will be okay."

So we had the courage to actually ship the product, and in some ways it was actually a relief that none of the bad stuff we thought was going to happen happened, because no one would even download the product. People literally would not use it at all. They wouldn't even try it to find out how bad it was. So my professional reputation was protected because these people didn't find how it would crash your computer. But it was sort of like, "Wait, what is going on here!" I had killed myself in the last six months to build this product for customers, and now they're not even finding out whether it works or not. So we're not even getting any good feedback.

If you look at what did we spend our time talking about those six months. I know you guys watching would never do this, but maybe you have a friend who has this problem. Go to your friend's meetings every day at their startup. What's being debated? I'll tell you. What features absolutely have to be included, and which ones can we live without? What order should we work on the features that customers are going to love? Which customer should we listen to and which one should we ignore? Etc. That's what we

did. We spent all of our time talking about that stuff, and yet none of it mattered. Like it didn't even matter which bugs we fixed and didn't fix because customers didn't even find out if the bugs were there. That was pretty depressing.

Paul: Okay. So if you could do this all over again with that exact same product and you were just put back in time, what you have done differently?

Eric: Well, the first issue is, the first way that I analyzed this at the time was, "Do I even really need to be present? Why was I even on this team?" Because my code that did all this IM interoperability didn't accomplish anything. It didn't matter. So maybe the right thing to do would be like, "Well, the team could have lived without me." But it's like, "Okay, no, no. That's not true, that's not true." Make myself feel better. I was like, "If we hadn't built this product and showed it to customers, we never would have learned is it an important thing? Namely, that customers don't want this IM add-on." We can talk about why if you're interested, but the basic idea is that customers didn't want to use it.

So I was like, "Okay. That was good. At least we learned this thing." But the thought that really bothered me was if the goal was to learn this thing, why did it take six months and thousands and thousands of lines of code? What if instead of supporting different 12 IM networks, we had supported only 6 or only 3 or only 1? Would the level of learning have been any different? No. People don't want to use the thing. Everyone said, "I don't care." Instant messaging add-on is a deal breaker despite our brilliant strategy.

Well, gosh, that's a lot less code to support only one network. But then, get this. If people won't even download the product, why do we need to build the product at all? What if we just created a single webpage with the classic magic test on it? Here's a screen shot of the awesome 3D avatar experience you're going to have and here's how it works and download now. Would we even have had to create page two where we apologize, the product doesn't exist yet, or would a 404 have been sufficient. Because when you can't get people past page one, it doesn't matter what's on page two. So that's how I wish we had approached it, is instead of thinking about what would be cool to build, we approached it from what do we really need to learn? And we're backwards to that.

I want to be super clear. If we had run that specific experiment, it would have been a negative result. People would not have downloaded the product, but it wouldn't have been the end. First of all, that would only have taken like 12 hours, so we would have had 179 days left to figure out what to do. It's not like we would have given up. It's just we would then have been able to start asking better and more reasonable questions of customers.

See, when we brought customers in to use our product out of desperation because they wouldn't sign up, we're like, "They're not signing up. What do we do? What do we do? Okay. I guess like last resort, we'll talk to some customers." We bring in these customers, they would say, "What is this?" We were like, "It's IMVU, an Instant Messaging add-on. You should download it." And they'd be like, "What the hell is that? What is an Instant Messaging add-on?" We would be like, "Ah, it's this cool thing that interoperates and all." The customer would look at us like, "What? No thank you. I don't know what that is. My friends have never heard of that. I'd rather not do it." And we're like, "Look, we paying you to be here. It's a usability test, so you're going to download this thing. Okay." If we coerced them, they would do it. Then they would create their avatar and they loved that part of the process, being able to chose what your avatar would look like and how it would be dressed. This was way before James Cameron's movie, so avatars were new to most people and they thought, oh, that's pretty cool.

Then we would get the software installed on the computer, and we would be like, "Okay. Now it's time to invite one of your friends." They would be like, "No thanks." Like, "No, no, no. You're in the usability test. We're paying you. Invite one of your friends." They would just say, "No, not going to do it. You could pay me as much money as you want, I'm not doing it." "Why not?" They're like, "I don't know if this thing is cool yet, and so if I invite one of my friends to something that's not cool, then I'm lame." It's like on a list of what makes software mission critical for a 15-year-old, not looking lame in front of your friends is actually really high on the list, which we had no idea about.

So we kept trying to figure out how do we overcome that problem. We thought, oh, we just need to make the software easier to use. Because everyone says,

"What should startups do?" Focus, good design, high-quality products. So we believed all of that stuff too. We said, "Okay, yeah." We kept making the product easier and easier and easier for people to use, but because it was the wrong product, we were just making it easier for people to discover they didn't want to use our product. So that just helped them churn out faster. It was all a total waste.

It took us months and months of having those conversations to finally really understand it's not ease of use, it's not that we're missing some features, it's that this whole strategy is fundamentally flawed. We took care to explain the strategy to customers that they'd see how brilliant is with the network effects and the switching costing, and they just go, "Listen, old man, what is your problem? I don't want to do this. Just give me a standalone IM network," which we eventually did and when we made that pivot, eventually things turned out a lot better for us.

Paul: Okay. So what a startup could pull from that story is you want to see if you even need a page two as soon as possible, instead of waiting the six months and building the perfect system or a system that at least hits all of your main goals.

Eric: Yeah.

Paul: What do you think of things like LaunchRock? Is that helpful for a startup? What are your thoughts on that?

Eric: Well, I think that if you use the LaunchRock system, it's basically a landing page generator.

Paul: Right, that will collect email addresses.

Eric: Someone once posted a tweet that was something like I've got this cool new startup that does – I can't remember, it was three or four about what it did – sign up here. It was a friend of mine, so I was like, "All right. I'm want to sign up." I go there and the LaunchRock page is like, "This is going to be a super hyped awesome startup." It said nothing about what the startup did. You couldn't figure out any information. It's like, "Sign up here to be the first to get it." It was just like, "What are you going to learn from that experiment?" Like, "Yes, people want to part of some super hype thing. No kidding." But the kind of people who are going to sign up for that are just going to be random early adopters. That's not helpful.

So instead what I wish they had done was really make that landing page about the problem they're trying to solve and really made it that magic test so that the people that sign up are really people who are like, "Yes. Oh, I'm so excited to do this," and not friends of friends. You don't want to get friends signed up.

People think that to make a startup you want to get big as quickly as possible, but I just think that's fundamentally wrong. You want to learn as quickly as possible. So having random TechCrunch readers thinking your thing is the new hotness, like, I don't learn anything from talking to those people. Have you ever read TechCrunch comments? People are supernaturally cranky. So they're just going to tell you your idea sucks. Then what are you going to do? So you don't want that kind of attention. What you want to do is really figure out who are the people that could really benefit from this and how can I get into dialogue with them as quickly as possible? If LaunchRock helps, Amen.

Paul: So LaunchRock is good if you're using it for the magic test?

Eric: Yeah. Same with Unbounce or Performable, or there's a number of tools that can do it quite good, that kind of test.

Paul: Okay. Great. Now let's talk about the building phase. That's kind of the next section here. Can you give an overview of what the build phase looks like? And again, this is a continuous loop.

Eric: Exactly.

Paul: But can you give us a quick overview of what this looks like for a typical startup?

Eric: Yeah. Again, I'm really going to keep reminding us all that tactics are not principles. They are two different things. So the specific tools that we use to build products are going to vary industry to industry, but the principles are the same. The short version of this is, because we spend all this time talking about what are we trying to learn versus not learn, we can now go a level up and say, "Well, why is it so important?" The reason it's called the Lean Startup is by analogy to something

called lean manufacturing, which is the most important business philosophy in the last 30 years that has really changed how manufacturing and a lot other kinds of business is done. The key to it is to eliminate the activities that companies do that are wasteful and only focus on the ones that create value. A lot of those tools have been honed over decades to help us really make work very efficient, but the assumption of all traditional business is that we already know what we need to build. So you're building the next version of the thing you built last year, like a new automobile or a new sequel to a video game or just a new incremental improvement over your previous thing, whether it's a toothbrush or Microsoft Office, it doesn't matter. Fundamentally, you know who the customer is and what problem they're trying to solve.

What we need to do is come up with tools that can be lean and efficient, but at discovering who the customer is and what is needed, in terms of what we need to build. So that's where all this stuff comes from. So let's take a specific example.

One of the most common building tools in Lean Startup is called split testing or A/B experimentation. This is just a very simple technique that comes out of direct mail, direct marketing where you show half the customers one thing and the other half a different thing, and you just measure the differences in response rate. It's like a classic clinical double-blind controlled trial where the customers don't know they're part of an experiment. Traditionally, you do it test little things, like optimum copy design, better buttons, what cover should my book be, that kind of stuff. In Lean Startup, we take those experiments directly into the heart of product development and we say, "Look, if you're actually building a product and delivering it to customers, wouldn't you like to know if the features you're adding are actually making the product better or worse?" Pretty simple. Most people already assume, they know in their heart that the features they're adding are making the product better, but I beg to differ. My experience working with hundreds of startups is that most changes make most products worse. So we're trying to take those techniques and really bring them into our everyday work in building a product.

Paul: Can you give a specific example of a change that's negatively influencing a product? Or that had negatively influenced a product in a startup.

Eric: Yeah. I've got to do it without naming names. I want to respect the confidentiality of the many people that I work with.

Paul: But even just being able to describe what problem they were trying to solve and something like that, just to give a concrete example?

Eric: Sure, sure, sure. Well, let me start with some mistakes that I've made just so I can be a little more specific and then actually get into the details. I'll go back the issue I was talking about IMVU in our early days trying to make the product easier to use. I remember one experiment really well. Here's the idea.

Since it's a social product, it's a new social environment. It's challenging to figure out what to do, and we wanted to make sure we offered a really good first time user experience. Everybody knows the right thing to do in a startup is to make sure every customer has a great first time user experience. So we were obsessed with that and we decided, all right, here's the experiment. We're going to hire paid customer service reps to sit in our office and chat with customers all day. For the people in the experimental group, when they log into IMVU for the first time and say, 'Okay, I'm ready to chat, we will automatically route then to a customer service agent. They don't know it's a customer service agent. We just have professional people who give you a great tour of IMVU. Some people we'll just randomly match them with other people on the system.

Now that was a complicated project to build. It was expensive to bring the agents in. It was complicated to do the routing, and we ran the numbers and it was just amazing. The behavior of the two groups was 100% the same. They were equally likely to use the product, come back to the product later, buy from us, chat with their friends, everything was basically the same. Now you might say, "'Well, since everything's the same, at least it wasn't harmful." But no, that is harmful because you're spending all this money and energy without getting any benefit from it, and that really is the very common case.

I'll give you one more. We had a new person join the company who wanted to redesign our god awful registration process. This is years ago when we had a very Web 1.0 registration process, like crappy radio buttons,

really poor. The idea was we would make a beautiful Web 2.0 Ajax, visual. this is a 3D avatar product, so you need big, bold, beautiful pictures. So there was a three-step registration process. It was like, if I remember correctly, give us your information, customize your avatar, download the product, three steps. This person was new, and we were trying to indoctrinate them into this experimental approach, the phrase Lean Startup had not been coined yet, so we didn't know what to call it. I was the CTO. I was in charge of engineering. So I basically would say, "All right, new person. I believe you that your new design is going to be awesome, but let's just double check. Let's get the data to prove it, and that way everyone in the company will know how awesome it was." I always made that offer to people because I know every time you roll out a new registration process, something like this, there's always something that you didn't realize about it.

So we did the experiment, we split tested. I had asked, "Hey, can we split test each individual page by themselves?" It was like, "No, that's a deal breaker. It's got to be the holistic full experience." "Okay. It's only a three pages. I can live with it." So we do the experiment and it comes back that the registration rate on the new thing is 20% lower than the old one. So the beautiful Ajax, 3D, enriched is 20% worse than the old one, which was devastating for our team. Actually, the first impulse of most designers when that they get that kind of data is, "Well, it just looks so much better, let's ship it anyway." I was like, "Are you kidding, ship it? No, no, no." Here's the thing. I want to really emphasize this point, the reason you don't ship it anyway is actually not because we care so much about squeezing every last once of performance out of our registration process, it's because when we fail, we have the opportunity to learn, if we take advantage of the opportunity.

See, I actually believed in the hypothesis that this new Ajax registration is better, but if it has a lower conversion rate, that means that there's something about it that has been lost that we had in the old one that we don't realize. Therefore there's something about our customers that we don't understand, like maybe they find beautiful designs repellent because they are countercultural renegades or something. Like, boy, wouldn't you want to know if that was the case? Anyway in this case that wasn't what was going on. It was really simple. It was so simple it's amazing that we didn't see it. So after it failed, then I was allowed to say, "Okay. Yeah, let's test each page one after the other, old versus new." What we did when we did that experiment, we found that the first two pages were awesome and they had 20% improvement in the registration process, from getting people from page to page. But then we were giving up all those gains and losing an additional 20%, so we're losing 40% of people on the third page, the download page, which was actually the simplest page. It's just a giant page with this huge download button, with beautiful chrome and 3D specular highlights, just to make sure you don't miss out on the download. Once we had that negative result, then we knew the right questions to ask and we were able then to talk to customers.

We had been doing focus groups with this design all along. We had lots of customers involved and our customers were always like, "Yep, it's great, it's awesome. Love it, love it, love it." Once we knew there's a download problem, our tests got a lot smarter. We'd bring customers in and we'd watch them go through the process. instead of asking them, "Do you love this page?" We just watched what they did and we noticed this funny thing, which was that we would be like, "Download the product," and they would pause and they'd be looking around the page. We're looking at our watch, like, "Dude. Okay. Click the download." Finally we would get frustrated, and we'd be like, "Just click the download button." We'd try to be neutral. We'd be like, "So why aren't you clicking the download button?" The person would be like, "'I can't find the button." We're like, "Why not? What about this giant thing?" She's like, "I know, there's this huge banner telling me to download, but it doesn't tell me how to do it. I feel like an idiot." We're like, "Well, why not just click? What's the worst that could happen?" If you've ever talked to a normal person and asked them what's the worst thing that could happen if they randomly click something on their computer, then you do not understand normal people. They're like, "Are you kidding me?"

Paul: They're terrified.

Eric: Right, terrified. You never ever click unless you know exactly what's going to happen, at the pain of losing all of your photos of your grandkids. That's the kind of mentality. So once we had that experience, it was so obvious what we had done wrong and we know how to fix it. When we fixed it, all the numbers went up. But the insight was more than just best color for

the button. The insight was, "Oh, we think about how we use our computer different than this person, and we need to be aware of that in our design." So that was actually a very important moment there.

Paul: Okay. So that loops back to the learning, which we just talked about. So how do we avoid some of these problems, like building a feature that will negatively influence the product? Is there any way that you can avoid some of those landmines, or does that just loop back to learning from the feature?

Eric: No, you absolutely can. Look, the myth that we have is there are these visionary designers out there that know all the answers and can just get everything right the first time, and that's not true. It's not like Steve Jobs physically hand assembles every iPod himself. There are a lot of people that have to work together, and they make a lot of mistakes along the way, and they have to figure out what the customers will want. But it is true that there are people who are really good product designers. There are people who have an intuition for a specific set of customers and how to make their lives awesome and what they want and how to build for them.

So that's great. If we have those skills in-house, then we want to empower those people to make those decisions, because they're going to be more likely to be right. We're going to do ourselves is double checking just to make sure. Because right now we live in a world where people are putting on a lot of theatre about how they're a great design genius. How do you prove that you're the next Steve Jobs? You kind of dress up like Steve and you make inscrutable comments like Steve and you try to play the role of a designer, but that's actually very stressful. Janice Fraser, who leads kind of the lean user experience movement talks about how that puts a lot of pressure on designers to get everything right the first time, and you're not allowed to admit that you make mistakes.

I just think that whole framework is really unhelpful, especially because there's a way to get better at design over time. It's the same thing that we learned as a civilization with the scientific method. That's why when I use the word experiment, I mean science experiment, not just throw stuff on the wall and see what sticks. A science experiment, where you make a specific prediction about what's going to happen and then test that prediction and then learn, is valuable (a) because you get the learning, but it's also valuable for your building intuition because as you use your judgment more, and more, and more, and reflect and get real data about how good your judgment is, you're actually training your product intuition. You're training your judgment to get better over time. So when I work with teams that adopt these tools of building, especially if they adopt them team wide so that everybody has the opportunity to learn, they actually get better at building stuff. So they improve their engineering productivity, which is a very counterintuitive result, but it's straight out of the lean playbook. This is why we want to use systems and tools that take advantage of the innate creativity of every person on our team.

Paul: Okay, got it. Could you walk us through a typical workflow of how you decide if a feature should ship in a product or not?

Eric: Yeah. Let's actually get into the real detail of this, because a lot of people are familiar with the agile technique of story cards, which is a great way for planning out what you're going to do. If you use the Scrum system, then you have this thing called the product backlog, which is like the unified set of all the things you plan to do in the future. If you're using a system like that, it's really cool. Each card represents a small amount of work, like a day or two amount of work. It's called a story card because it is written from the perspective of the customer. So it's a story about what the benefit to them will be rather than some technical mumbo jumbo.

So the typical thing to do is we'd set up a big whiteboard with three sections on it – product backlog, things that are in progress, and things that are done. So you would just move the cards along one after the other to get a sense of how it's going. You can do this on a whiteboard with 3x5 cards, which is actually the method I recommend, but there are also, if you have a distributed team, you can use tools like Footfall Traker, which is also pretty good. You move the cards one after the other, and the deal of Scrum or any of these agile systems is that you don't interrupt people once they start a task. Once you put a task in progress, it stays in progress until it's done. But as long as you obey that rule, you're allowed to rearrange the product backlog as much as you want, so as, what they call, the product owner or the person in charge of figuring out what

needs to get built, rearranges.

That system works great as long as the product owner knows what customers want, because it puts all the burden on the product owner to get this all right. Now, we talked about the designer pressure to get things right the first time, you have the same issue here with the product owner. In entrepreneurship, we're, by definition, operating in conditions of extreme uncertainty. So we don't know the answers to these questions, and so all of our story cards are actually experiments.

So my recommendation to teams it is to use a system called Kanban, which is based in the Toyota Production System where we constrain the number of cards that can be in each bucket. So if we have a team of five developers, then we only have five cards in progress at any time. We never have ten cards. So we don't want people multitasking because that really diminishes their opportunity to learn and allows mistakes to creep in. Then we also add a fourth column to the Kanban board called validated. We do this very simple system. Cards flow from one set to the next. So we plan what we are going to do, we build it, then it's in progress, then once it's built, it's in stage three. But then it doesn't go to stage four until we have figured out whether it was a good idea to have done this task in the first place. That might be validation through a split test. It might be validation through a customer usability test. It might be validation through cohort analysis. It might be validation through any method that the company believes in. This is agnostic.

But it's constrained. So if we have five people on the team, then there are only five allowed in that bucket. So if start just building a lot of crap, what will happen is very quickly the validated bucket will fill up with stuff that is not yet validated. That will cause all of the previous buckets to back up, and eventually the team will grind itself to a halt. The first time this happens it's really frustrating. We have these engineers being like, "Dude, my job is to build stuff. I want to pound on my keyboard, but I need a story and I'm not allowed to get a story from the backlog, so I'm stuck. I'm wasting time. This system is inefficient and I'm going home." You've got to be like, "No, no, no." The day your become an entrepreneur you rip up your business card. It no longer says programmer on it. Now it says entrepreneur, and the definition of entrepreneur is a person who does whatever it takes to make the company successful. That' it. That's your job. So if your job is to go talk to customers, to clear out that validated bucket, do it. It's like, "I don't want to talk to customers, that's like the worst part of engineering." It's like, "Hello, welcome to entrepreneurship."

So maybe the first time this happens, it'll happen in fits and starts like you do too much stuff and then you have to go talk to all these customers. Then, of course, it been a couple days since we shipped the feature, so actually which customers do we talk to. I'm like I don't remember. It's much harder. The reason we go through this exercise it that it very quickly trains every person in the company to look at a story card critically to say, "Wait a minute. (A) do I actually believe this is going to make the product better or not?" Like in a lot of agile shops programmers become code monkeys and they're just like, "Whatever I code, whatever you say. I exercise no higher rank function. I'm just a catalyst for code." I think that's really dumb. You're hiring talented, creative people and then you're telling them to suppress their creativity and just do what they're told. It's like what a waste. Second, it causes everyone to say, "I don't know how to validate this idea. So why should I build it?" That is the win, because if you run an experiment with no hypothesis, it's actually a waste of time. You'd be better off not doing it.

So I know all this Lean Startup stuff, it sounds like more work. It's like, "Ah, I've got to do all this extra work to validate and talk to customers and look at metrics." But the goal is actually make you do less work, because you stop doing, my prediction, 60% of the tasks you're currently doing that nobody cares about. So imagine working in a company where every time you make a change to the product you have confidence that you're actually making customers' lives better. It's a lot more fun.

Paul: Okay. That's really good, thanks. That helped firm it up a little bit. So we have all this information. We have a product, we've been talking to customers. We need to measure this, what metrics do you use to look at if something is successful or not?

Eric: Okay. Let's say you go to a startup board meeting, not your board meeting of course, but a friend's. The friend has been working hard for the last six weeks, or a month, I mean, how long was it since the last board meeting and they want to show progress to

their board. What do they do? Well, everyone knows that what you want to do is show the board graphs that are up and to the right since last time. So this creates a lot of what I call success theatre, which is the work we do to make it seem like we're being successful no matter what. So a lot of people are just trying to measure as much stuff as possible so that every board meeting they'll be able to find something that's up and to the right, so that they can show their board they're making progress. This leads to Eric's Rule of Frugal Analytics, which is no matter how bad you're screwing up at all times, there is a graph in Google Analytics that is up and to the right. We just measures so much stuff, there's all this information. You always find something that's up and to the right. That is success theatre. That is the path of vanity metrics, and down that path lies a lot of danger.

So we talk a lot about measuring stuff in Lean Startup, but our goal is not to measure as much stuff as possible. In fact, it's exactly the opposite. We to find the few key, actionable metrics we must measure in order to connect our building activities to our learning activities.

Paul: Okay. So, as I startup how do we identify those? What metrics are actually important and what's just fluff?

Eric: Yeah. So there are two parts to the answer. Let's start with, what makes a good metric? Then we'll talk about the specific metrics that matter for your particular business. For metrics, there's a heuristic I use called "The Three As." A good metric is actionable, auditable, and accessible. What that means is actionable, you look at a report with a number on it and the number's going up. You would know what to do to get more of that to happen. So here's a classic vanity metric. Our website had 650,000 hits this month. What the hell does that mean? Seriously, what is a hit? Have you ever been in one of those product prioritization meetings where the first 20% of the meeting, half an hour sometimes, is taken up debating what the units on the graphs mean? I'm like, "Well, does a hit include graphics? Does it include Flash objects?" No one has any idea what a hit is. Second, let's say that last month we had 600,000 hits and this month 650,000. Is that good?

Well, there's a bug in human psychology that vanity metrics prey on, and the bug is this. If the number of hits has gone up this month, I know what caused it. It was whatever I was working on this month, obviously. If I'm in engineering, it's those cool new features that we launched, duh. If I'm in marketing though, it's that cool new marketing campaign that we did. If I'm in QA it's like, oh, we higher quality standards. If I'm in customer service it's that new customer. Everyone always believes that when the numbers go up, it's because of whatever they were working on at the time.

Let me ask you this question. When the numbers go down, what caused that? So let's say the number of hits goes down to 500,000.

Paul: Well, clearly that's a seasonal thing.

Eric: Right. That's actually what they'll say, seasonal effects. I've never heard anyone be like, "Our numbers are up because of seasonal effects." No, it's always down because of seasonal effects and/or it's those idiots down the hall. We made great features in marketing, but the stupid engineers screwed up the features, or vice versa. So everyone is learning bullshit. They're learning their own private reality where thing always makes the numbers go up, and those morons down the hall are always making the numbers go down. So then is it any wonder these departments wind up at each other's throats, because it's like, "Our good people against those stupid people." That's how these kinds of wars start. It's really stupid.

So you don't learn anything from vanity metrics unless they're actionable. So we have to really get underneath to say, "Well, hold on. What is the cause of these numbers changing?" That's when people start to be like, "Oh, well we launched something on September 2nd and so on September 5th the numbers went up, therefore cause and effect." It's always like, "Hello, that is like classic astrology." Right? I say, "On September 2nd, Mercury was in retrograde. So I say, that's why the numbers went up. Now prove that I'm wrong." Nobody can prove that it's a completely non falsifiable hypothesis. If you really dig into the data and try to figure out, which customer saw what, and frankly most teams do not know what causes their numbers to move. I think that's a terrifying place to be. I understand why people are so scared of metrics. It feels like voodoo. You're just like, "Wow, good things happen to me some months and not other months." We've got to get out of that situation.

So an actionable metric is one where you can understand what caused it. That's why split testing and cohort analysis are so important in Lean Startup. We want to know that if we launch a new feature or a new marketing campaign, we want to know for sure that it had an impact on customers, not just, "Well the next day it went up." That's the issue.

The second key attribute of metrics, the second A, is that they're auditable, because what we want to use metrics for is to learn and learning is just a pleasant euphemism for killing people's pet projects. If you've ever tried to kill someone's pet project, you will know just how hard it is. You've got to get a clean headshot or the zombie will come back. So what does it take to actually get someone to believe that their pet project is a bad idea? It requires a report that is not only actionable, but believable. See, I used to go around with metrics all the time to try to convince people to do things. If I was giving them bad news, they would always say the same thing. It's like, "Well, that report is not accurate. People love my feature, but the numbers are down because there's a bug in the reporting code." You're like, "Bullshit."

Paul: Right.

Eric: Prove that the numbers are accurate. Most systems, like Google Analytics, you can only prove that the numbers are consistent with each other, but that doesn't mean that they're consistent with reality, and more importantly that doesn't mean that anyone will believe you that they are.

So auditable is really remembering the key phrase, "Metrics are people too." Every number is actually a human being somewhere, and if we can turn the numbers back into human beings, then we have a way of checking to see if the data is accurate. So if I say, "Listen, people don't like this feature. That's what the metrics say." And you say, "Oh, yeah, which people don't like it?" I used to hear this question all the time, "Which people? Everyone I talk to loves it. So who are these people who don't like it?" I can say, "These people. Let's click on this button," and boom. "These 100 people all stopped using our product because of your stupid pet feature." "Well, let's go call them." "Listen, all I want, as a purveyor of metrics, is to be right about the truth. I don't care about your pet feature if it's the best thing since sliced bread. I just want to know the truth of it. So let's go get the truth, let's go talk to the people." Well, all of a sudden people's skepticism about metrics evaporates when they have to go talk to people. They say, "Ah, man. All right, fine. I'll just believe the report." So it just gives people a way to be like, "Oh, I guess the report is accurate because it is checkable." But sometimes people will actually do that checking. Sometimes the follow-up conversations are actually the most important part of an auditable report. So it's not just, "Did they really not like my feature?" But, why?

And last, the reports have to be accessible. If you have to ask a data analyst to run the report for you, it is not accessible, because now you have this person being your central point of failure. You have to go talk to them, make sure they understood what report you want run, get the date range. Then of course they're the high priest of data, so whose report gets run before whose other report, you know?

Paul: Right.

Eric: They have a lot of opportunities to influence what happens in the company. In a lot places, they become the de facto product manager because they control the facts. It's like no priesthoods in a lean startup. That is not how it works. Every person has a right to independent access to the data. So like at one company I worked at called Grockit. I worked with them to do this lean transformation, and they had done a really smart thing, which is they had a system that would automatically email a PDF attachment of all the key data to every team member every day. So you come in, in the morning and there's just this email waiting for you that's like, "State of Grockit, how's it doing?" That included not just the overall gross metrics, but every experiment that is currently running and how it's doing. So people got over their fear of being wrong because now it's like, 'Look, you're going to run an experiment? Everyone's going to know every day how it's doing. If it's screwing up, people are going to know." That's good. We want that. So just by making the reports accessible everyone in the company feels empowered to use data in their everyday decision making.

What we did at IMVU was is we had a single web-

page that people could go to get data about all of our experiments. It was just a simple form. Everybody in the company had access to it. It was actually part of our production environment, we considered this a part of the product. So people could produce a standard one-page report with the answer to any particular experimental question that was in a format that everybody else could understand. That's the last key piece of accessibility. If people don't understand what the report means, it's useless. That's why website hits are such a bad metric. Right? What the hell is a hit? I don't understand what that is, but everyone understands what people are. So you say, "Look, 50,000 hits disappeared this month." They're like, "Whatever." But if you're like, "50,000 people stopped using our product." It's like, okay. That's like a giant stadium full of people who are not using our product. So anyway, that's the kind of key to accessibility is to make sure that you use very simple metrics, no ratios and weird indexes, simple metrics denominated in units that everybody understands.

Paul: Okay. How do startups that have very limited time, very limited resources copy those same principles and use those metrics?

Eric: The key is to build everything incrementally. So it's easy to convince yourself that you need to world's best, fanciest recording system that's going to take weeks or months to build, or you need a fully general experimentation system that allows you to do anything that you could possibly imagine, or that you absolutely have to have this crazy deployment system that automatically fixes all the problems. I mean talk about a lot of that stuff in Lean Startup, but if you google for continuous deployment, you can see talks about teams that have built stuff like we built at IMVU that does this automatically. But the key is to not be intimidated and not to convince yourself that you have to do everything all at once.

The way that this works is it's best build everything out piece by piece as you need it. So the mantra is, "Just in time, not just in case." So when you need something a little bit more sophisticated, make it a little bit more sophisticated, but don't go crazy trying to build a great system. I built IMVU's original experimentation system, including the cohort analysis, the reporting, and how to do the splitting, in a couple of days of PHP. It was like 300 lines of code. So don't start with anything more complicated than that.

Paul: Okay. All right. So really simple and really accessible, like you said. Okay.

Eric: Oh, one more thing about that. The more complicated the mechanism by which the data is collected, the more room for error and the less believable it is. So a lot of engineers have this love of summaries of summaries of summaries as the data gets massaged to like 29 different systems. So like, the master data's over here, and the report's over here, and there's like system, system, system in between. Well, I show you a report on stage five and if I find one weird thing about it, I'm not going to believe the whole report. I'll be like, "This whole thing is wholly inaccurate." You're like, "No, it's just that one thing." How are we going to know? Well, we have to go trace it back.

It's much better to do the inefficient thing and derive your reports directly from your master data. You know, live, in production even, if possible. "That's so inefficient. You don't want to burden the master database." "I know, know." But think about how auditable that is. If there's a problem, I just print out the SQL statement that generated this number and I go run it against the master database by hand. Okay, now I know I've got it right.

Paul: Let's talk about the specific metrics that we're actually measuring. What would a company that's dependent on just tons and tons of users and going after advertising dollars, what's a metric that they can look at?

Eric: Well, again, it's really important to remember the difference between the viral engine of growth and the sticky or a compounding engine of growth. So for a product that causes people to sign up as a side effect of other people signing up, that's the definition of a viral product, really the only number that matters is the viral coefficient, which is I bring in ten customers, how many more do they bring in? It's important also to make sure that the product actually creates value for the customers that come in, but that's not going to determine how it grows. So whether you charge them money or do advertising, there's an effective growth of the product, and that really is the most important question for product/market fit.

On the other hand, you have products that do the stickier compounding engine of growth. They may also have huge numbers of customers, and they can also use advertising too. It's just that they have this kind of lock-in or a recurring model. So I'll give you an example. I happened to have this experience. It was really surreal for me. On the same day two different entrepreneurs asked me for advice about the exact same problem. Here's what the two companies did. One is like a competitor to eBay that does online trading and collectables for like Anime figurines and stuff. One is an enterprise database company that sells like a competitor to stuff that's made by SAP and IBM and Oracle.

So these two companies supposedly have nothing in common. They don't serve the same customers. Their metrics are wildly different. One is trying to make a few cents in transactions off trading like eBay does, and the other is signing contracts with the world's biggest companies for hundreds of thousands of dollars. But they both have the same problem, which was that their growth had unexpectedly flat lined, and they couldn't figure why. I looked at the graphs and their vanity metric graphs looked exactly the same. Everything was going up where it was fine, and then it just flat. The advice they wanted for me was, how do we get more growth? Now everybody knows how to get more growth, more advertising, more marketing, more product features. If you're not having enough growth it's because you've got to get more people in the system.

So growth to most people implies outbound activities. But using the engine of growth framework, you say, "Well, both of these businesses depend for their survival on some form of addiction to the product." In the case of an eBay type product, it's the network effects of buyers and sellers, like if you're a hardcore fan of beanie babies and you want to buy and sell them, you've got to go to the one place where the best deals are, and you can't just not login today because what if today of days is the ultimate unloading of some attic somewhere with all the best beanie babies. You have to constantly check it, so you're always reengaging. That's because of the network effects of eBay. So for an eBay competitor, the same dynamic.

Well, think about an enterprise database. If you build your application on an enterprise database, you're not going to quit. There's no quitting. Once you've start a licensing fee, once you're on the program, you can never get off. That's vendor lock-in. That's why you evaluate that enterprise vendor so carefully, because once you decide, okay, I'm building an Oracle or this new thing, you're locked in for the duration.

So these are both sticky engines of growth where what matters to their growth is how does the compounding interest work? What it is the natural organic number of people coming in to the product, subtracting off what's the natural churn. Sticky products should basically have no churn. World of Warcraft has basically no churn. eBay has very little churn or it did in its high-growth product/market fit days. An enterprise database should have very close to no churn. So it's organic growth minus churn. It's your compounding rate. If you ever look at a compounding interest table, you'll see that even if you compound 5% per period, period after period after period, it can become quite exponential quite fast.

So what's the problem in both of these companies? I asked them both to do this analysis. What is the churn rate compared to the growth rate? Both of them, and this is very common for sticky products, it just so happened that they were growing like 60% per period, but they had 40% churn or something like that. I don't remember the exact numbers now, but it was like the growth is being exactly balanced by the churn, so that it looks like we're flat lining. So human beings interpret a flat line meaning everyone came and they stayed, and no one new is coming. But in a sticky business that's not what's happening at all. Tons of new people are coming in and tons of people are churning out. You can think of a water tank where the outflow is exactly balanced with the inflow.

So in a situation like that, because we are actually measuring the right thing, it was really easy to see what to do, not more marketing, less marketing for God's sake, fix the churn rate. Really focus in on why are people leaving and how do we prevent that from happening. I'm pleased to report both companies have done that, and both are doing much better because they just focused on how do we actually get people once they start to stay.

Paul: Gotcha. Okay. What's something that someone watching this video right now could close this browser and go in and do, immediately, in regards

to measuring their startup and figuring out the right metrics to use?

Eric: Well, if you had to choose one thing that you're probably not doing, is you're probably not doing any form of cohort analysis. So it's always a good idea to start with. If you're ever not sure what tool to use, this is an easy one because almost nothing bad can happen. Cohort analysis is just a very fancy term for thinking about the different groups of people that use your product and looking at their data independently. So, like in the early days of IMVU, we were doing our marketing budget on $5 a day. That's literally all we could afford. In those days, Google AdWords, the minimum bid was $0.05 per click, so $5 a day bought us 100 clicks. We thought when we first started doing that, "Oh, 100 people will come into our product, 50 of them will buy it, and we'll make a ton of money. How great." Of course, you guys already know the answer, pretty much zero of them bought anything. But every day we kept doing the $5 a day. So we had this nice renewable source of audience, every day 100 new people would come in and that gave us an independent report card for each day. So we got to a point where there was a long period of IMVU's history where basically 1 out of every 100 people who came in would buy the product, so 1% conversion rate.

After a while, and this went on for months, every day we're making the product better and better and better, fixing bugs, adding features, doing all this cool stuff, but the conversion rate was still 1% every day. It's like, "Wait a minute. Did the people that came in today, did they call up the people yesterday and they're like, 'Okay, guys, how many of you bought?' Okay, 1% of you. Okay. We'll do the same." No, these people never meet those people. So what is going on that their behavior is so consistent cohort after cohort? Well, that probably means that the product is actually not getting better over this period. We think it's getting better. We're fixing bugs and adding features, but our fundamental notion of what makes the product better, clearly our customers don't agree. That was very hard to hear. We did not want that information. That is an insight you could only get through cohort analysis because you're looking at each group independently, and if their behavior is consistent, it means that everything you are doing doesn't matter.

So wouldn't you want to know? It's like, "Pssst." If you're in that situation, I understand you don't want to tell you're investors and I understand you don't want to tell your spouse, but wouldn't you want to know if every single thing you're doing matters not one ounce to your customers? I would want to know.

Paul: Yeah, that's really great advice. So just get out there and hit the street. Talk to actual people.

Eric: Yeah, and measure what they do, not what they say.