Episode 13: Don Reinertsen (Part 2)

Episode Transcription

Mik Kersten:
Hello and welcome to the Mik + One podcast, where I sit down with industry leaders to discuss the project to product movement. I'm Mik Kersten founder and CEO of Tasktop and bestselling author of Project to Product: How to Survive and Thrive in the Age of Digital Disruption with the Flow Framework®.

Mik Kersten:
On today's episode, I continue the conversation with Don Reinertsen, Product Development Consultant and the author of three bestselling books on product development, Developing Products in Half the Time, Managing the Design Factory, and most recently the award winning book, and one of my all-time favorite books, The Principles of Product Development Flow.

Mik Kersten:
If you haven't already tuned into part one of our conversation, I highly suggest listen to that first, as this builds on it. I hope that you enjoy the second part where Don shares even more of his amazing insights. So, with that, let's get started.

Mik Kersten:
I'd love to get your thoughts on this, the flow time clock, the one that we optimize to encourage our customers to optimize too, it will start when work is taken into the value stream. You touched on this a bit so I wanted to dig into it because basically, for us, it's the role of product management and product management leadership to determine this work. It's what you say no to. In terms of the full requests that a tech company has, there could be on open source projects, we were doing about 1% of the incoming work requests. So we were saying no to 99% of things, because on popular open source projects you'll have say six people who are supporting a part of the Linux Chrome that's used by a good chunk of the planet. So there's basically infinite requests, and your lead time clock is meaningless.

Mik Kersten:
That said, if you've got work that's been defined as relevant to a market already, as you were pointing out, if you exclude that and you say, we're just doing great on delivery, but you exclude the things that are actually meaningful to the market, of course you'll get it wrong as well.

Mik Kersten:
So, my view on this is to make a distinction. This is why the Flow Framework tried to make this so core, you have to measure all of them. You have to measure your lead time. You have to measure your flow time, which is when work's actually been taken into the value stream, even just for design and ideation. And those clocks don't stop until you've delivered economic value to the market, which then of course you have to measure through a metric, like a gross margin change over time.

Mik Kersten:
I guess, in your experiences, this difference between flow time and lead time, is there anything else that you want to share in how organizations can think about this? Because I think, again, the problem I see is that there's been this excuse made, well, it's a fuzzy front end. I hear your terms thrown around a lot. We're just not going to measure anything upstream of development.

Don Reinertsen:
Well, historically what we always used to do is we used to divide it into what we call pre-development and the development cycle, and the knife edge checkpoints on that were typically pre-development started when we logged in the opportunity with an opportunity number. When it became visible to management within our system. So, unless you got issued an opportunity number, then it wasn't in the system.

Development started when we opened up a charge number to charge time to the project in a classic engineering organization, because it was pretty much guaranteed that you were not making a serious effort if you didn't have any place to charge your time to. Opening up a charge number was a very good proxy for, 'has the organization gotten serious about it?'

Don Reinertsen:
And then we used to end the development cycle with the first revenue producing unit. When the first unit went out the door that actually produced revenue. And then those were, it was worth breaking down those two pieces of pre versus regular development, in the sense that there would be a lot of people saying... They'd start putting their team to work without having a charge number, knowing they would be paid for the thing and things like that, because they were trying to make their development cycle metric look good.

Don Reinertsen:
So it became, 'well we haven't really started yet because we didn't have a full team, I was working with a halftime team at the time, or I didn't have a budget, or marketing had not signed off the requirement document' and things. So, there was a tendency for the metric to produce some dysfunctional behavior if there was a way to cheat on one particular milestone.

Don Reinertsen:
Breaking the end-to-end thing into individual pieces, I tend to use the historic term cycle time as opposed to lead-time because lead-time has a specific meaning in manufacturing, which is totally different than the way we're trying to use it in product development. But whatever label you put on it, there's an overall time from when did you first have the opportunity to spend money on this until when you first get an actual economic return.

Don Reinertsen:
It's useful to have some break point in that big chunk of time as, how long did it take from spotting the opportunity to having a fully staffed team working on it? To just start trying to boil the pot and knowing the time between when I started seriously investing money until when I got a return out of it. That's the way I've historically thought about it.

Don Reinertsen:
I think with just some crude capture mechanisms, like it isn't an opportunity that somebody has written it down in the opportunity book and given it a number, simple things like that often produces visibility on the ideas and those then, once you start capturing it that way, you can end up taking a look at it on a regular cadence and say, okay, what new ideas came in during this week? Which ones do we think we need to pursue further?

Don Reinertsen:
Interestingly, in effect, there's a self-imposed filtering process at that front end, of people will stop proposing ideas if they never end up getting accepted by the organization. The character of what is not assigned an opportunity number to it, a lot of times the idea will come back the next week, wearing a different color hat and try to get into the process then. But I think it's quite healthy because what happens is by the things that the steering committee rejects as opportunities, it is in effect communicating to people what business do we think we are in.

Don Reinertsen:
It's a great strategic question always used to be, okay, which customers are we trying to serve? What type of value are we trying to provide to them? And who are we choosing to compete with? The essence

of strategy is customer, company, and competition. At least the old McKinsey way of thinking about it was.

Don Reinertsen:
People in the organization have to have some sense of who are we trying to compete with? What are we trying to compete with them for? On what basis are we trying to compete? And I think in the way you respond to what people consider to be opportunities, you end up saying, well, we're more into project management than we are grocery delivery. Grocery delivery is a great opportunity for someone out there, but we're not going to take that into our pipeline. There's economic value there, but not for us.

Don Reinertsen:
I would say in general, most of the tech companies I run into tend to have weakness in that first stage of the process before they commit to do something.

Mik Kersten:
Yeah, absolutely. I think we're seeing that weakness. And again, I think a disciplined way of measuring it, uncovers that weakness.

Don Reinertsen:
Yes, once you put metrics on it, I would ask people, what percent of your development process is fuzzy front end? How much have you improved the fuzzy front end in the last two years? Their answer is, probably a lot, but I don't know because I don't measure it.

Mik Kersten:
This fascinating thing that's still ringing in my mind, that you said at the start, is architecture margin, and then connecting that to what you said about the air-to-air missile. That thing has visibility built into it, because if it didn't, it would just be going straight in one direction. Do we need a visibility margin and the way that we build products? Because even your point on charging something to the right place, that's a margin, that made something more visible because effort was taken to make sure, well, this connects us to this part of the P&L, or this revenue bucket, or something else. So, you're just making me think that organizations need to think of a visibility margin in how they built software because not measuring economic value or outcome is no longer an option. So, do you think we should think about this as a visibility margin in how we structure our delivery organizations?

Don Reinertsen:
I think if you say something is an opportunity, you got to communicate to the team, how much gas am I willing to put in your tank? The biggest opportunities are going to get the largest amount of gas and things like that. If you go completely around having no signal... Now, in most cases, the signal is, how many teams do we have working on this? How many scrum teams are working on the same thing? But you have to have some mechanism to tell people this is the rate at which we're willing to, in effect, burn money to qualify this opportunity.

Don Reinertsen:
And again, for a lot of projects, that first step is trying to figure out if there's a $100 bill in the envelope. I tell people the mistake most people make is they look at the project as if they're going to be investing for the next three years in it and justify the project on the basis of that three-year investment. I say no, your decision right now is you're going to buy information and how much money does it take for you to get a better idea of what the terms of this bet are? Can you find out if there are real customers that have this

need? Can you find out if there's a real technology that will solve this problem? And can you do that without doing three years' worth of development? The answer-

Mik Kersten:
You need to be able to measure. A sales leader, if they're exploring new territory and their field is not great with visibility, because everything is stuck in their notepads, guess what, they'll make sure that there's an overhead applied to have those people that are working to make sure there's visibility so that they can quickly learn this region is a reasonable region or it's not.

Mik Kersten:
I think it sounds to me like whether the goal is gross margin, or to produce economically useful information to determine if there is an opportunity here in three to six months, not three years, we need that kind of visibility, just the visibility baked in, by accepting that these things need to get measured.

Don Reinertsen:
As soon as you are operating in a stochastic world, instead of a deterministic world, if you want to get certain outcomes, you're going to have to build margin into the system. In fact, the other interesting thing about that is you start focusing on... In measuring the health of the program, you're not measuring whether you're on schedule or whether you're on budget or whether you're you're on performance. You're measuring how much margin do I have left in this program?

Don Reinertsen:
When I was in the Navy on submarines, there's a concept on submarines called reserve buoyancy. The term is reserve buoyancy. And reserve buoyancy essentially measures, how buoyant would the submarine be if something broke and you started flooding water in? What weight of water could you take in and still be able to get back up to the surface? The amount of reserve buoyancy is determined by you have ballast tanks and things and high pressure air to blow it and things like that. But you always have to have margin. Submarines that don't have any reserve buoyancy ultimately disappear. They end up at the bottom of the ocean. Because while your propulsion plant is propelling you, you can maintain a depth and things like that, but if you have negative buoyancy, then you just drop like a rock to the bottom of the ocean.

Don Reinertsen:
But I was talking with one development team many years ago, and the leader said, "We have what we call a schedule bank. That's the number of days we are ahead of our original plan. And in our weekly meeting, we always ask the question, have we made a withdrawal or a deposit to the schedule back? Do we still have margin left on there?" And the idea is, don't deplete your margin early in a program, because bad things are going to happen. Bad things happen more frequently at the end of the program than at the beginning of the program, and the last thing you want to do is to have burned up half your margin when your 50% of the way through the program. That's a disaster.

Don Reinertsen:
I think one of the real useful metrics is, how much margin do we have in the program at this moment and what has happened in the last week? Because that's very predictive. There's another interesting analogy in terms of physiology and shock in first aid. You probably know what shock is, Somebody cuts himself, they're bleeding, hypovolemic shock. There are two different forms of shock, one is what's known as compensated shock, and one is uncompensated shock.

Don Reinertsen:

When you have compensated shock, what happens is, even though you've lost blood, your heart rate has increased, your breathing rate has increased, your body has shut off blood flow to unessential organs and things and it is still maintaining blood flow to the brain because that's considered extremely important.

Don Reinertsen:
Now you can't tell that the person is in shock by looking at their blood flow because their blood pressure is still high. They haven't lost any blood pressure. But what has happened is their heart rate has gone up, their breathing, their skin is becoming pale. You've got indicators that the margin is being consumed and you pay attention to those indicators that you're consuming margin now, which means you're heading to an unsafe condition. And then you get into what's called decompensated shock, which is where your body can no longer compensate for the loss of blood. Then they deteriorate very quickly. They lose blood flow to vital organs and they die.

Don Reinertsen:
But a team can be like that, where all of their performance metrics are looking good, but people have moved from working 40 hour weeks to working 70 hour weeks and things like that. Well, that's a sign that the team is consuming their margin and things, and that the team will not be able to handle an additional shock of some form without the whole thing blowing up.

Don Reinertsen:
The fact that your metrics have not deteriorated is actually a sign that none of your metrics are measuring the resilience and reserve capacity of the team. Are the team using every hour of the day to work? Are they de-prioritizing training and things like that? I think you have a set of metrics that tell you how much margin the team has, or how much margin the project has, that are traditionally absent in the metrics we use in managing.

Don Reinertsen:
You're keenly aware of this every time you walk through a development lab and talk to the developers, because you'll see all of those indicators and things like that, but they're typically not present in the management system. They're present in the interpersonal contact with the people working on the project.

Mik Kersten:
I'll tell you a really quick story on this, but we're basically now into this realm of looking at the interaction for you between tech debt between again, the team health, and I think this reverse buoyancy concept is si key.

Mik Kersten:
I'll tell you a quick story where our teams are now so big. I used to, when we started as a company 13 years ago, I used to be able to get a read on the team on how much capacity they had left and in their own buoyancy after this massive product launch, after some quality or scaling issues, whatever those things those shocks that hit the teams.

Mik Kersten:
And this is actually what inspired me. We were measuring employee net promoter score all along, but I realized once we got larger, once our product portfolio grew, measuring how happy developers were, was not quite meaningful. When I started measuring how the interactions is, [inaudible 00:19:03] the happiness of one product value stream, everyone involved in it, not just the developers or testers or the support staff, the designers, everybody, all the product managers, everyone. We saw exactly this, it's basically, here's just a really quick pattern, we've seen this in many other organizations that we measure

it. Technical debt is that the footwork piles up and up, features get blocked, so there's more and more pressure on that teams, which causes flow load, so WIP to go up, and when that next shock comes because of some customer problems, some launch problems, some delay, there is no more reserve buoyancy left.

Mik Kersten:
And then a fascinating thing happens in the way that we measure, which is the employee happiness drops, the employee satisfaction drops, because of the 70 hour workweeks, because of frustration, because of interpersonal problems. And so what I've realized is that, and especially now in this world where observing teams and social interactions directly has become much, much more difficult, we actually have to measure that. And the way we measure it is, through COVID how our employment promoter scores are affected. And knowing that if flow load goes up, because all of a sudden people are dealing with their young children at home and less productive, and if we don't factor that margin into how we plan, we're going to end up with exactly the problem that you're describing.

Mik Kersten:
So it's fascinating to me that it's this dynamic system and I think... I read all your posts on reinertsenassociates.com, but your going to gemba post, it's sufficiently complex organizations and systems, we just need a different way of going to gemba to see these things. And if we don't, and I think it's endemic in IT organizations right now, that the reserve buoyancy capacity is almost nothing and it just got worse.

Don Reinertsen:
Let me just correct. It's reserve buoyancy.

Mik Kersten:
Reserve buoyancy. Thank you.

Don Reinertsen:
I don't want to propagate a new term here, but yeah, no, I think it's really interesting. I know when I talked about in that first…

Mik Kersten:
Reverse buoyancy sounds terrible, so I repeat that again.

Don Reinertsen:
That's okay. But the go to gemba, I talked about how Dave Packard and Bill Hewlett used to go out and talk to the engineers. In general, they were really exceptional engineers themselves, and so the things they would talk about is like, where do you think the risk is in this program? What do you think are the biggest challenges you're going to have to overcome? What's your plan for solving that problem? If that solution doesn't work, what other plan do you have to do it? And things. They would try to get inside the engineer's head about where is the... Because value added… In processes that generate information, which is what product development is, the information is generated by reduction in risk. It's classic information theory. It's altering the probability of events and things like that.

Don Reinertsen:
If you're really trying to excel at information generating, you're trying to excel at stripping risk out of programs. And so, understanding where engineers believe the risk is and what is being done to get rid of that risk, is just an extremely important consideration. I worked one time with a company that did

aerospace satellites and things like that, and the experience guy in charge of the project told me, he said, "I view my job in designing a new satellite as simply a problem of risk reduction. We start with a very large amount of risk when the program begins, and at the end of the program, we want to have all of that risk has gone away. The way I sequence my activities is I do the activities that eliminate the largest amount of risk per dollar spent, and that will invariably get me to a successful program if I tackle the biggest risks I can reduce most cheaply."

Don Reinertsen:
Because all of these satellite launches, they're just one-time deals. Most of the cost is the design of the satellite, not the building of the satellite, and it only happens once. So, you get really good at approaching it as a problem of risk management.

Mik Kersten:
I think again, the product development innovation process is, as you say, all about that. Just one last question, if we can, because I think this has been such a... There's been a lot of discussion about technical debt, but I've noticed, and of course I've made technical debts a core part of the Flow Framework, but to me, so little economic analysis of it and the one I saw you present, the one I studied it when you first published it on your blog, on comparing technical debt and the framework, I think it's such an important concept that changed some of my thinking on it, and I think needs to change others thinking on it.

Mik Kersten:
So, if you could just give us a really quick summary on that to leave people with that in terms of how they need to work that into their own economic thinking on product development.

Don Reinertsen:
Earlier in this podcast, you used the term deferred work. That is certainly my preference, because I consider it to be less value latent. I'm deciding not to do certain work at a particular time. I think the minute you end up calling it... You call it a technical debt for a reason, because you people know debt is bad and you have to repay debt, things like that. But in reality, if you want to think more clearly about it, you would say it's totally different than financial debt. You don't necessarily have to repay it. Sometimes the funds you use to repay it are much cheaper than the funds you would be spending today.

Don Reinertsen:
A classic example would be, you think you can use a piece of open source code on your design, but it's not yet ready, so you're going to just [inaudible 00:25:19] up something that would substitute for that open source solution, and then later on, when the open source one becomes available, you'll switch over to using that open source module.

Don Reinertsen:
Well, would replicating that open source model at the time be a good choice? No, just get something that is simply workable with the minimum amount of effort for you to proceed in with your system integration and the rest of your design, and then recognize that you will be able to retire that debt at a much lower cost when that open source module becomes available.

Don Reinertsen:
I think people have a little bit of a dog whistle reaction to technical debt. I tend to think of it as it's a business decision that you're going to make. I think a classic example I always like to use is, you're 99% complete on the project, you've got 1% that you thought was really important and should be done, and the

question is, do I hold the other 99% of the revenue stream hostage until I finish that 1%? The economics clearly favor of, okay, I won't perfect that particular feature, I'll perfect it later on. Not I won't perfect it ever, but I won't perfect it now.

Don Reinertsen:
I've seen example after example of people introducing products where they had a feature almost complete, but it wasn't complete to their quality level, so they didn't even announce the presence of that feature in the product until it became working completely to this. Where people say, look, that feature is not important enough to delay the entire introduction of the product. We'll leave it in the product, it will be a hidden feature, we'll get some people to try it out and things like that, but until we can meet our quality standards, we are not going to make that claim on the product.

Don Reinertsen:
People in medical devices and medical instruments do a really good job of that in the sense where they say, we will always meet our claims when we put a product out into the marketplace, but we will modulate what claims we make based on the perfection of our technology. We will say, you can use it for this. We're a 100% confident you can use it for this particular application. As soon as we become 100% confident that you can also use it for this, then we will make the claim.

Don Reinertsen:
They don't go through this notion of saying, we need to be able to meet all of the claims we would like to make on the product, they end up saying, when are there sensible economic choices to back off from our claims in order to deliver a very large value to our customers?

Don Reinertsen:
I think it's an economic choice. I don't think technical debt is always bad. I don't think it's always good. I think people ought to think of it as an economic problem, not as a philosophical choice.

Mik Kersten:
Terms aside, I think I encourage everyone to read that post, Technical Debt: Adding Math to the Metaphor. I'll give you a quick example, Don, of how we think of it. Because the pitfalls that you talked about have been such a big problem for us and for others I've seen in the past, we will not take on any tech debt reduction unless it increases feature flow measurably, there's a hypothesis obviously, but we're able to measure hypothesis, within a 12 month window. Because otherwise what's happening is that we're again, we could have waited that six months and some open source replacement, some new Amazon AWS service is now available for this thing.

Mik Kersten:
It goes, I think, back your points on risk. For us internally in product development, there's too much risk of reducing too much tech debt too early, because it can be superseded by something else that's changed in the technology landscape. We actually see it, every single release planning activity, we look at tech debt reduction as a way of increasing architectural margin. Sometimes we're actually happier with a thinner architectural margin when we know there might be some big announcement coming from some technology stack that we depend on.

Mik Kersten:
I think back to your point and the point on reserve buoyancy, we actually manage that constantly. How much architectural margin do we have? Do we need to do a whole bunch of tech debt work to increase the architectural margin when we don't know quite how things will go? But we never assume, and we

used to, by the way, that we can have a clear picture of architectural margin for the next two years. Because we can't. Things are too uncertain. The risks are too high in making those assumptions and we're better off pivoting midstream than doing that.

Mik Kersten:
I knew I would learn a whole bunch of foundational things that I thought I knew but didn't understand well enough from this podcast. And that's another one, is that in the end I think that a lot of this does go back to risk reduction and those information three principles, and basically how we reduce risk in product development by bringing in more of the learning earlier into the process.

Don Reinertsen:
I often point out to people, you've all seen those Standish studies saying that 50% of the features you're putting in this product are never used by anybody in the entire life of the product, and things like that. Why do you think it makes economic sense to eliminate all technical debt in all features, if half of them... You're polishing features that nobody is going to use.

Don Reinertsen:
Obviously I think the world we're moving to is saying, well, let us get it out in front of the customers, start seeing signals that we've got issues that they want us to correct, and correct those issues quickly. Let's decide what debt we need to deal with from customer poll. But I really like the idea of eliminating technical debt as a way of providing design margin.

Don Reinertsen:
The hardware analogy would be, I've got a bunch of subsystems, I've got a power supply of a certain size, some of the subsystems are drawing more power than I intended, and things like that. If I'm going to need more power in other areas, that's going to break the system. Well, why don't I take some of those power-hungry subsystems and reduce their power draw on it? I didn't take the time to optimize the design of those things in the original design, why don't I do that now, build myself more architectural margin in the design, so that I can tolerate other contingencies that are guaranteed to appear in the course of the program.

Don Reinertsen:
The other thing I often point out to be is that notion that these are bad things come at you through a memory-less process. It does not remember that in the first half of the program, I gave you a lot of bad accidents, therefore, I will reward you in the second half of a program by having things be really easy and good. No. Assume that you're going to continue to have bad accidents in the second half of the program, and if you've lost your reserve buoyancy, fix that before something bad happens.

Mik Kersten:
Amazing. Don, thank you so, so much. There's so much wisdom in here. I'm sure many of us will be digesting this podcast over and over, and definitely I will. Thank you so much for your time. How can people find you and the work you continue doing online?

Don Reinertsen:
Well, I guess the first challenge is spelling my name, but no, they can ask somebody else in their organization. Have you heard of a book called Flow? And they will probably point that out. The likelihood of finding somebody in an engineering organization today that is aware of the book is reasonably high and of course you can goggle the book. You can go to my web, reinertsenassociates.com. I would say

most people are becoming aware of my work through secondhand word of mouth, as opposed to sitting down on Google and say, let me Google product development and see what comes up.

Mik Kersten:
Well, and I strongly encourage everyone listening to find that person in their organization, because if they don't find something familiar with your work Don, they risk, I can assure them, the risk is too high. These are foundational things. I think you've inspired so much follow-on work, but I encourage everyone listening to go straight to the source. So, with that, thank you so much.

Don Reinertsen:
I'm always happy when somebody says, I wish I had read that book 10 years ago.

Mik Kersten:
Yes, exactly. I was very frustrated and I started reading it, but I hadn't. I'd like to save others the pain and thank you so much for summarizing this. I do want to just emphasize for everyone again, we just scratched the surface of this incredibly dense and foundational book, so I do encourage you to search for Don's other work. And again, thank you so much for doing this Don.

Mik Kersten:
Once again, thank you so much, Don, for joining me on this episode. If you enjoyed this episode, please leave a review and you can follow me in my journey on Twitter or using the #MikPlusoOne. Don's Twitter handle is @dreinertsen, if you want to reach out to him or find out more information on his books.

Mik Kersten:
Don't forget I have a new episode every two weeks, so hit subscribe and join us again. You can also search for project to product to get the book and remember that all of the proceeds go to supporting women and minorities in technology. Thanks. Stay safe. And until next time.