Hemant Elhence: We've been practicing Agile development with many of our clients. We often observe that there is a fair amount of confusion in the role of product manager/product owner in the Agile context, and the kind of questions we often find people struggling with are things like, "So what really is the difference between a product manager and a product owner as it's defined in the Agile role, and where do you find these guys? What kind of a skill profile is expected for these guys to have? Do they come from the QA side? Do they come from the marketing side?" and so on.

Another question that often comes up in this case of the product management role, and a very common one, is, "When distributed development is pretty much the norm, especially now in the Agile context, how does the role of product manager get evolved in the distributed Agile development context?" so how do you divide that role? Then, of course, the core part of product management all along, and more so in Agile, is hardnosed prioritization of requirements or user stories, and why is that so hard?

We'll tackle these questions, and really what we would call demystify and dig deeper into the role of Agile product management.

I'm delighted to have Rich Mironov Mironov, who has written a book on this topic, to present this. Let me quickly introduce Rich Mironov, and then I'll turn it over to him. The mechanics of the webinar today are going to be, as is always with our monthly webinars, is we will do 45 minutes also roughly of presentation on this topic of demystifying Agile product management. Along the way, I may chime in with a question here or there.

You guys should feel free to send your questions. If there's something that just is not clear or you want to ask a followup, just type that, and I'll make a decision whether that question should be taken along the flow or should we wait until the end; and at the end we'll leave another 15 minutes of so for Q&A. We'll make the PowerPoint and the recording available also to our audience after the presentation is done today.

With that, let me introduce Rich Mironov. Rich Mironov is the CEO of Mironov Consulting. He has been leading and coaching product managers for more than three decades. He is a serial entrepreneur, real practitioner, and played the role of CEO or VP of products, VP of marketing, at a number of startups; and often finds himself parachuting into software companies and acting as VP of products.

He really is a practitioner, and glad that we have him to speak on this topic. He founded the first product camp and chaired the first product manager and

product owner track at the Agile Alliances annual conference, which is the biggest Agile conference. He is author of the book, *The Art of Product Management*, and the long-running Product Bytes blog.

With that, let me turn it over to Rich Mironov. Rich Mironov, over to you.

Rich Mironov: Thanks so much. Welcome everybody. Looks like a lot of folks on the webinar. I'd go through just a very brief agenda, and then maybe some level-setting because I've had some folks who wanted to put this in the right context.

I really wanted to pick up three of the questions that we get all the time. What's a product manager? What's a product owner? Why are we fussing with titles? Shouldn't this just be about getting the work done … so try to dissect that a little bit to figure out what the useful information is about what people do, a little bit of discussion about how we find and grow people with product skills, or how we train them up because I think that's an important part of this mix, and then, as we said, a lot of questions about the distributed model, so, "My headquarters is in San Francisco but my development team is in Bangalore. How do I manage this product owner or product manager problem?"

That's our three points for the morning. I'd also like to do a couple of sets of level checks. First of all, I'm very much focused on the problem of commercial software, or software for revenue. I think what we'll see is a lot of this discussion is less urgent and less important if what you're building is internally-consumed software, if it's IT, if the end users of your software are within your corporate boundaries. This is going to have a lot to do with how markets accept products and how that's important.

Second, there's a tremendous amount of great enthusiasm going on around lean startups and lean entrepreneurship, which I love. I was just looking at a video with Grant Cooper and Patrick Vlaskovits on Mind the Product, about the difference between being in a learning phase and being in an execution phase. Most of what we're going to talk about in the next 40 minutes, or 45 minutes, is about the execution phase; so you'll want a very different set of organizing principles for your product and UI and technical design folks if you're in the exploration or learning phase of figuring out what customers want. Keep that in mind.

The last thing we'll see is, if you're a tiny company, none of these fine distinctions matter. If you're a 20-person company, you have to hope for some one heroic person who's going to do everything we're going to talk about because there's no opportunity of delegation or division of roles. We're thinking of, again, commercial production revenue software, and probably in the medium and larger stages.

Let's dive in. By the way, there is a question box on the side; if you've got questions, just pop them in.

Here's my setup, which basically is you open up a job listing somewhere, you're going to find lots and lots of jobs for products managers. It's a job title; the org charts and the HR people understand this. It isn't necessarily Agile, and it tends to report somewhere off into marketing or engineering or up to the executive staff.

Product owner, coming right out of the great early work on Scrum and Agile, is part of a self-organizing development-oriented team. It tends to report up through the engineering or to the program office, and that's going to have a lot to do with some of the selection bias we're going to talk about.

Ultimately none of this matters because, whether we're talking about titles or roles or who does what, if you're trying to build software for commercial success, for revenue, there's stuff that has to get done; and if we spend a lot of time arguing about who does what, that's time not well spent in getting work done.

I'm going to try to make some distinctions in the next few slides between sprint-level work, which is the stories and backlogs and priorities and acceptance that, if you're in the Agile world, if you're doing Scrum or Kanban… you know you love … and the broader set of market challenges which is, "Can we figure out how we're going to sell and make money and engage with the market, and make sure that our products aren't stretched.

Again, if you're a small company, the answer is: you, the product person, do all of this; but as we scale up, we're going to see some distinctions in skill, in roles, and in how we organize ourselves.

There's our setup. I'd like to break this in a couple pieces. First is to ask the classic Silicone Valley HR organizational question of, "What is a product manager supposed to do?" Then we'll decompose that same question about product owners.

If you're a product manager, if you come into your software company every day and the badge you wear says, "Product Manager," here's mostly what you're responsible for … not just driving delivery of software but actual market acceptance. Whole products … if the market doesn't like it, it doesn't matter how good the software was. You're going to have lots more discussions about market segments, which is the different sets of audiences and their relative priorities in terms of revenue and acceptance; and you're constantly the question at the highest level, always building the right thing.

Obviously, if you're working on internal software, as a product manager you're mostly trying to drive acceptance and adoption of whatever your team is building internally, and you know … because we all know … that every internal project has competing priorities and has politics, and the internal product manager is often trying to drive internal acceptance of things that the executives believe are obviously right but the organization may not be ready to accept.

Putting that into a little more graphic model … and for those of you who've seen this before I apologize, I use this everywhere … I would put the product management function, or the people carrying that title, as serving through different sets of audiences or stakeholders. It's important … first of all, this isn't a hierarchy with product management at the top. This is really a product management at the bottom hierarchy; but in any case, there's three distinct sets of stakeholders and inputs and outputs.

The one we think of first, particularly if we're product owners, is on the development side; and there's a long list of things we might provide to the development team in an effort to get out the software that we're looking for. I've included a few things here, but take your pick … road maps and user stories and personas and priorities and requirements. These come in all varieties and shapes and sizes, often fitting the specific situation; but they're all wrapped up in the idea that we're trying to help our development brethren bring out the software which is as close to our collective vision as we can.

The product owner role is heavily focused here, as we'll see. The return code, the thing we get back from development, often we'd like to think of it as product but it's not. It's the technical product bits. It's the software and the manuals and the testing and the installation, but it doesn't include a lot of the things which are absolutely necessary for commercial products to succeed, as we'll see in this deck.

Then we turn to the right here, to our next group, and what we know we need, that marketing and sales people cry for all the time, is a different set of deliverables, including things like segmentation and messaging, future benefits, pricing, just to pick on a couple of those.

Segmentation is critical for our sales force because, otherwise, they don't know who the product is for, and they waste their time selling to the wrong people, or people we think are wrong. Certainly pricing is an absolute requirement. It's often make or break for a product independent of how good the software is. We've all seen a great product come to market and fail on the pricing side … qualifications, demos, etc. Notice none of these are identical to the technical

requirements that technical requirements
we provide to development, but they're necessary to sell.

Of course, what we get back from sales and marketing, what they politely call feedback or input, generally it's a slap to the head and phrases like, "What were you thinking?" In any case, we know that our sales and marketing people filter back some of what they hear from our end customers, but we also know they have tremendous selection biases. Sales reps are famous for only remembering the last meeting they were in. Marketing people are famous for warning the short three bullets with eight words each positioning and not the details of the product.

It's required as a product manager that you reach over the top of sales and marketing to talk with your real customers, your real prospects, win-loss analysis with people who didn't buy your product, rubbing elbows with the real users almost every day; maybe every day is not possible, but three or four times a week.

Finally, for our third set of stakeholders, the executives, they, in fact, want a completely different third set of activities here. What executives really care about … those of you who are executives, I think you'll agree … are the long-term strategies. "Tell me what next year's revenue is going to look like?" "How are we going to pick up another segment?" "What's the competition doing?" These are generally in units of quarters, or maybe years, because there's not much that executives can do in the very short term to move the plan.

As product managers we are doing the strategic looking out of where we're going, and in almost every case we're presenting proposals or business plans to build more product than our engineering team can deliver, and so we're asking for more resources. "If we just had two more X amount of teams, we could take another $20 million segment on."

What we get back from executives is … on a good day … budgets and staffs and targets for the proposals we have made. By the way, those budgets and staffs are not for product management, they're for development, so here we are representing … we're shilling for development in order to grow that team , and the targets that come back, of course, are the targets that are were in our business model as the claimed goals for that activity.

You can this is a three-cycle activity map here, and what's really interesting about it is there's nearly no one who can walk into a product management job this way and bring skills from all three of those areas. We're almost always bringing somebody in who has strong development skills and experience, or comes out of technical marketing, but it's rare to find somebody who can

navigate all three of those sets of relationships initially … a lot of mentoring, a lot of training.

Okay … catching my breath. That's the product management map. We could have drawn this map … and I have drawn this map … 20 years ago, long before the Agile Manifesto, long before we talked about lean startup, back in the bad old days. It would have looked the same. The challenge we face, and the hand-wringing I see almost every week, is as follows: When we turn this into an Agile model, we dramatically increase the amount of work, and the amount of input, and the amount of goodness that has to flow between product management and development.

We used to have the model that said, "Write a marketing requirements document in January, throw it over the wall; come back in October and hope you've got something good. Clearly, the Agile Scrum model means that we are doing hand-to-hand with stories and acceptance every single day in a much more intense model than we used to have.

I think of this as feeding the Agile beast. Here you're seeing somebody who, back in the old days of the steam engine, was known as the fireman, so that's the person who stands in the mountain of coal directly behind the steam engine and shovels coal all day in order to keep the train moving. It's critical because, if you stop shoveling, the train stops.

What I see happening to lots and lots of classically-trained product managers is that, when we put them into an Agile context, they get so much love and attention from development and they're so pulled into the need to be an Agile product owner that they spend nearly all their time working on the stories, and working on the acceptance, and working on the backlog; and they begin to neglect the other parts of the picture we drew.

It's inevitable because, if we've added 40, 50, 60% to the workload and we've done it on the development side, we're all going to lean that way.

That sets us up for the problem of classically-organized product management is harder to work in the Agile world, but it's still critical for success of commercial products.

Hemant Elhence: Rich Mironov, here, if a company was running a traditional model and they had two products managers for whatever size of development team they had, and they have transitioned to an Agile, are you saying that they would need

additional capacity given product management, so meaning, using these ratios, maybe three products managers?

Rich Mironov: Absolutely. Honestly, that's a hard case to make because, having run product management groups, the deliverables for product management are pretty non quantitative, sometimes they're suspect, and to go to the CEO or the division GM and say, "I need two more product managers because we're going Agile," often doesn't get a good reception. Often what we end up doing is we specify those additional people as product owners, both because that's where we need the help and because that's an easier hire, and we'll see exactly some of those issues come up.

I've struggled for years trying to figure how to quantify how much product management you need, and because it's really an interstitial middle management kind of role, it's pretty hard to stick numbers on it, but I think we'll come back and we'll try to do some estimation.

Again, I think … just editorializing here … I'm a huge advocate for Agile. I've worked on lots of Scrum teams. It's clear to me that better processes get better products built faster with happier customers and more revenue. It's just that a fair amount of that burden falls not just on the development team, not just on the Scrum team, but on the product management side where we now have to make some distinctions that were not made before.

Let's flip the coin and ask the product owner question; and pick any of the product owner books and here's what you'll read: that the product owner represents the customers' interests, backlog prioritization, requirements questions. You'll see the list of Scrum of Kanban activities … writing stories, reviewing stories, prioritizing backlogs, accepting work, and making sure that things are in value order.

Here we're feeding the hungry Agile beast. What's great about this is finally we're putting some more horsepower against the problem of week by week building the right thing, so it's really this intense level focus. Even more important, finally we have development teams that appreciate what this work is.

In the old model, product management was always viewed as folks we don't really know or care about, they don't help us so much, those marketing requirement documents are fuzzy and useless, and we're going to build the wrong thing. We're now getting, for the product owner, the real love that traditional product management never got from development, because mostly they weren't there.

I define this carefully and humbly as
product owner is really doing the portion of product management that development sees, understands, appreciates, and values. It's a capturing of nearly 100% of a product manager or some similar person to do the product owner role, often to the exclusion of the other things we're going to see. This is a great reason … by the way, I'm happy about this because what it means is we're finally putting someone face to face with our Scrum teams, with our development teams, such that we're building more of the right stuff … less waste, more velocity, more valued builds.

Most of the Agile models won't work without strong product ownership. I think this is terrific.

Hemant Elhence: There's a question here, if you don't mind, a very specific question that I thought it would be good to bring it in right now. The question is: Can product owners break the stories to address development and testing separately?

Rich Mironov: I'm going to park that for later because I think … my strong feeling is that stories should include both the development and testing. If you want to do whole stories that deliver whole value, I'm always suspicious of splitting testing out; but let's park that for later.

...

There are two words I highlighted here which I think are really important. The first one is the word "represents," and you'll find it in almost every intro Scrum and the product owner materials. I have a huge objection to the work represents when we're talking about commercial software, things we want to ship for money out to the world.

If I'm doing internal projects, if I'm heading up, as product owner, some work for the finance teams that's needs better reporting, it's obviously true that the finance team can nominate one or two people who represent the needs of finance and can look at the software and the reports we're building and decide what good looks and what they want.

When we reexamine that in the commercial space, we run into all kinds of problems of preferable and less preferable customers, and segmentation, and where we should invest our money. When we're doing commercial software, honestly, at the top I don't want somebody who represents customers; I want somebody who negotiates the goodness and the value of customers against my company's business goals. We don't give everyone what we want.

The other one is: almost every Scrum book talks about customer in the singular. We have someone up on a pedestal who represents the explicit need, and is the person who can confirm that we're doing the right thing. In the commercial space, that's just not true. We're always dealing with outliers and wrong segments, and people trying to apply our products to the wrong thing.

I'm always wary of those two words. Let's keep going.

If I draw my chart from before, clearly the product owner is going to spend a tremendously larger amount of time, I would say twice as much as the traditional product manager, in the core work of product owner with development, and you see them all here. Again, what we get back are the portions of the product that represent the technical output.

The other things the product owner does … again, if you open up the classic books … is they're running showcases. They're bringing in the core key customers every week or two, showing them progress, and getting the head nod that those showcase customers like or understand or approve of what we're doing, and they're doing course corrections based on those core customers.

Again, in the internal IT sense, I think it's exactly right. As soon as we go commercial, though, the problem that we have is that the two customers on the planet who are sufficiently committed to showing up for these showcases every two weeks, they aren't at all representative. They already love us. They already understand our products. They're probably much more in tune with the technical details. They're willing to skate over some of the hard startup issues. They're not average. If we let ourselves be led around by our showcase customers only, then we end up in a custom development cycle where we're building for our best customers instead of our average customers … something to watch out for.

I borrowed this years ago from Catherine Connor at Rally. This is a picture of her calendar when she was a product owner in the early days at Rally. What's important here is the only times that are not occupied by specific product owner tasks are the white ones. It's really hard to get out of the building. If you're working on B to B products that need you to be face to face with important customers … pretty hard to do. It's easy to fall into the model of cheating the Agile beast and thinking you know what the market wants.

Another way to think of this … and I borrowed this from Greg Cohen, by the way … is to think about which levers different folks have. When we read the product owner books, and we should, what we mostly focus on are product features … how should they work? … and the order of delivery of how we're going to sequence those things. When we look at the product manager set of tasks, and

we with that we get a super-set of that,
because now we should be … now always, but should be … worried about pricing and competition and distribution and services, and all of the things which make the broader whole product saleable and successful.

Really important because … not that someone who wears a product owner badge can't do this or doesn't do this; it's that, in general, what I see is we don't ask them to, we don't measure them on it, we don't train them up for it, we don't value it, and when we put them in the box, the smaller intents Scrum facing product owner box. We often leave the bigger box unfilled, and we end up with great software that doesn't sell.

Again, try not to get stuck on the title issue as much as the expectations and the skills we bring to the party.

Hemant Elhence: So Rich Mironov, is it still fair to characterize what historically people have felt, in the software product business, that, in this case, the product manager is primarily an external facing, many customers facing kind of a role; and product owner is primarily internal facing, meaning development facing, kind of a role?

Rich Mironov: Yeah, that's a useful thing. If I go back to the folks at Pragmatic, who designed some interesting role descriptions back … I don't know, must be 20 years … they talked about a technical product manager who is mostly inward-facing, a product marketing-focused person who is mostly outward-facing, and a strategy or director of product who is mostly strategy-facing.

The product owner role … and we've mapped those out … actually is an even narrower set of roles and distinctions and tasks than the Pragmatic technical product manager. We've sharpened that point even further. When we say inbound, we're usually speaking also about a lot of customer management and request things, and often support interfaces.

I'd say the old definition of technical product manager, or inbound product manager, is still wider and shallower than the product owner definition.

There's no perfection here. Everywhere I go these jobs are defined differently and these roles play out differently, but I'm trying to find the middle way, the center of mass.

I've set up the problem; of course, I haven't fixed anything. It's frustrating to people on both sides because, of course, we all want to do the right thing, and we all want to get the job done.

One last problem setup here. My friends in engineering would like to believe that prioritization is a quantitative, objective process, that one can sit down with spreadsheets and get it done; and, in my experience, that's just not the way it is … for a variety of reasons.

The reasons prioritization are hard … and some of them are listed here … that's me, by the way, on the right … the first is, at least in steady-state commercial products, the number of enhancement requests and demands that come in so far outstrip the ability to get things done in almost every case that you're going to discard the vast majority.

In Agile, of course, we don't say we discard them; we say we put them deep in the backlog, but that's just a euphemism. A lot of energy is spent explaining to people who have reasonable requests that they're not going to get those requests fulfilled. In equivalent terms, we tell them, "They're not in this year's roadmap," or, "We'll get to them when we get to them." Smart customers know what that means.

The second is: we'd like to believe that we can, for instance, predict the amount of revenue or customer satisfaction that a new feature or enhancement will make. The answer is: we're lying to ourselves.

Big error bars all over the predictions we make, and we make those based on gut feeling and emotion as much as we do on the science. We try our best. We also will tend to trade off the long term, trade away the long term for the short term because short-term demands are much more important.

Then last … and the Apple folks, I think, among other, have taught us that design matters and curating products matter … if you do all the things customers want you to do, you end up with these bizarre, crusty products that need to be broken up, that need to be curated, that honestly work less and less over time.

Anybody who's got the current version of Microsoft Office has seen 20 years of this process. They keep trying to jam more features in, and honestly it makes the product less useful over time.

These are really as much about people and organizations as they are about the facts, and if we take a mechanistic, computational, unemotional view, the Mr. Spock version of prioritization, honestly it doesn't work because the people matter, the customers matter, the organizations matter.

We've set up the problem nicely. Let's make sure we're going to get to some of the answers. I would describe this and … by the way, we've had a lot of transportation analogies here; we're talking about trains … when I think about

how the classic product manager fails in the Agile model, it looks like this. "I'm part time, I'm not engaged with the team. My stories aren't good. My backlog is lousy. I do a lot of hand-waving instead of writing good stuff. I'm pulled in too many directions, and I end up saying this thing too often, 'You guys are smart; go figure it out.'"

That's the essential sign of the traditional product manager failing to engage in the Agile model in sufficient quality, quantity and energy to keep Agile moving.

On the other side … because, of course, Agile is faster, we need a different kind of disaster here, so there's our airplane … what I see almost everywhere I go is that we pull product owners out of either technical roles or subject matter expert roles with no experience or training or hope or appreciation for what happens in the real markets; and so we end up mis-emphasizing or missing important connections, and we bring out products that aren't strong enough.

It's folks who didn't know anybody from marketing and sales. It's people who don't understand the company's strategy as well, and they're making choices that are local rather than global. As I said, they're confusing the two or three showcase customers with the big market.

This isn't a personal failure. I usually think of this as a mentoring and selection failure, because we're neither asking for these fields nor tracking these fields, nor training for them; we're trying to put people in who have them, and so it's our own problem.

Finally, the market failure … and this is more pre-Agile … we deliver great products that don't sell, and honestly, that's everybody's failure. We stick on old plans. Everywhere I go I see executives with the following belief, "I walked through engineering, and I saw people who weren't typing, so I don't think they're working," who don't understand the basics of how and why we build software. We assume static markets, it takes us two years to build the wrong thing, and there's our ship.

Again, the question is: How do we get stuff done?

Shifting slightly, let's talk briefly about skills. Again, I don't care what title you've got, I know that my company needs a set of skills. If I look at job descriptions, of which there's a lot … if you unpack 50 or 100 job descriptions on LinkedIn or someplace for what product managers are being asked to be, you'll see that primarily we want existing product management title experience, we want great verbal skills, we want technical skills, and a fair smattering of MBAs with a lot of subject matter expertise.

Notice this is an MBA meets bachelor of computer science balanced view of market versus engineering. It's not perfect. The reason I have the picture of the unicorn here is because, in fact, very few of these people exist and we always end up making tradeoffs on the margin; but this is what everyone is advertising for.

If we look at what development managers particularly want in their product owners, we see a different thing. We see subject matter expertise and a huge premium on writing good stories and understanding the software process, which is great. What we don't see is anyone asking for or pulling for market experience. You've got a lot of new product owners who believe in rational customers; I've not met very many of those.

We've got hiring managers who don't understand that the blocking and tackling and keeping interruptions from the team is so important, and this works really well for internal IT; but over and over what I see is: it's a bad fit for commercial products unless we give them the backing and organizational support they need to be successful.

What we'd really like … and that's, by the way, not a picture of me … we'd really like to put people in product roles who've got a balanced set of experience, maybe more than more than one product market. They've got to be technical enough to succeed. They've got to understand customers, and they've got to have the sort of people skills and political skills to understand that different audience members, different stakeholders, approach the world differently.

The pure engineers here often don't understand or even believe in the existence of sales quotes.

Hemant Elhence: I have one question. Given the nature of skills required of a product management professional, call them product owner or product manager, have you seen any good written test that we could use, just like if you were hiring a developer they could use some written test, a language test or problem-solving skills, and so on. Are there any tests that can be applied in the hiring process, or screen at least …?

Rich Mironov: I haven't seen any tests in the computational sense. I've written a lot about hiring product managers, as have other folks. You might point folks to my website or Quora. I'm always asking questions like, "Tell me about the last product failure you had and what you learned from it?" We're really looking for people with a balanced view who understand the meeting of economics plus technology plus markets; and that's a hard thing to find.

I'm looking for people and emphasize in all directions that have great communications abilities with both engineers and customers. It's a management job more than a technology job.

Let's talk a little bit about organizational models. How do we get out of this mess? The first thing to notice … and we know almost every development team lately is distributed. Ideally there's a whole team somewhere, not QA in one country, and architects in another country, and tech docs in a third country; but certainly they're development. They're far from wherever marketing and sales happen to be, and I tend to see that development is Agile, but the selling and marketing and market validation at the end is really waterfall.

We build in an Agile way, and then we throw it to the sales and marketing process to find out if it failed or succeeded. Some really good work among the lean startup and lean entrepreneur folks to improve this, but that's what I see. Products are more than code. We need to be making decisions about our products. If we shift our target market, it usually means changing features and priorities right away.

What about splitting this up? I've seen some good successful models where what we do is we set up a team where product managers, as I described them, are a little more outward-facing, a little more program or product-level, feature-level. The product owners and really assigned to one or two Scrum teams, and they're at the team backlog level. This requires tremendous sharing and communications. It's easy for the product managers and product owners to wander away and get the wrong thing, but at least it's a model, and I've seen it work.

Here are some pictures because we're short of pictures. In the startup or the minimal organization, what I've drawn here is the more technical work to the left in red, and the more market-focused work to the right in blue. We have one person … it doesn't matter what we call her … wears all the stripes, does all the jobs, and whether we call that person a product manager or product owner, the important word here is hero because this is a burnout job. This is a really tough position to do … 70% of your time with engineering and 70% of your time with customers.

I think, having been in this job, I know that this is an interim step to either success or failure.

The worst situation is this one, of course, where product owners and product managers don't talk to each other. They sit in different places. The product owners report up through some form of program or engineering. Product

managers are lightweight technologists, or they can't compete, and they sit in the marketing organization, they don't talk. This is a failure mode. Don't go here.

How do we blend this other ways? A few different pictures. We might have a senior person who does the product line-level strategy, and a series of product managers … or owners, don't care what we call them … who are assigned to match up against the engineering organization. These might be individual products or pieces of products, but here again we have one person doing the full job, everything we said for product owner and everything we said for product manager, struggle as best we can; and we have someone who is the coordinating theory to make sure those pieces fit together.

Another model that I like is to say that we have a more senior product manager who's more outward bound and more concerned with market issues, who's paired with one of more of what we have described as product owners. That's a collective or a cluster of people to handle the work for a product that's bigger than one person. That lets the product owners get good mentoring from the product manager, the senior product manager, on the things that they typically lack because these tend to be hires out of the engineering or the subject matter areas.

Again, lots of ways to draw this map. There is no perfect way. What's important is that we recognize the gaps we're going to have and we try to fill in some of those things.

Last, before we run out of time … a couple of clients of mine and I have spent a lot of time on delegating. This is really important if I'm going to be in a model where I have one senior product manager who is responsible for the overall product success, and a series of product owners who are okay. Often, though, what we have is one product manager, and the engineering team has put forward product owners for the various teams. This will work in a couple of examples; I'll describe one.

A client of mine out of Alameda, California, building very large virtual operating systems and virtual environments, they needed to do some major performance improvements on one of their products which, in fact, had, I think, eleven Scrum teams and one product manager. It was clear that that one product manager couldn't chase all of this down; and, in fact, the product manager had very little of value to say other than what the performance goals were.

The product ownership was much more reasonably given over to senior performance architects, senior software folks who knew how to squeeze every bit of performance out of the code, who could write the stories, who could

inspect the work, who could measure the results; because the project manager, honestly, added no value to those details and simply wanted the results.

In that case, we might delegate entire teams of product ownership to someone who's more technical or more equipped. If, in this case, the UI was central, we might have the product owner be the product manager, and the product manager is going to review every change to the UI.

In any case, regardless, this is a model where you need to plan, and think, and delegate, and be responsive, and have a working agreement. What stories is the product manager not adding value to? What are the places we're going to do that? And maybe most important for me … If it doesn't go well, the product manager owns the failure. You can delegate, but you can't delegate responsibility.

All of this needs to be in terms of … I, as the product manager, may choose delegate to non-product management project owners but I own the bag if it comes back. How do we thoughtfully do that?

Some takeaways as we get to the three-quarter mark here. I claim … and I think people will probably agree … that the textbook definition of product management is a clear superset of the textbook definition of product ownership. It's not better, it's not more important, but it's clearly it's larger scope, larger breadth. What that means is, if you're doing the whole product manager job, you're probably shortchanging the product ownership role.

Real product, successful products, need both great engineering and real market validation. They need really good economic analysis. It's easy to build products that don't make money, and there will always be unhappy, uncomfortable tradeoffs which don't simply work off the numbers. So someone has to be the person who makes hard decisions and owns them, and I don't care what your title is when you do that, but somebody's going to make those things.

Finally, when we need to split this up, product managers are probably going to be closer to customers and markets, and product owners are going to be closer to development. We know that creates gaps, so we need to build skills, organizations, and communications to keep that from being a failure mode.

A lot of talking from me. I think we're going to break for a few questions.

Hemant Elhence: Thanks, Rich Mironov. We have a number of questions here, but I'll give another couple minutes to the audience to put in their questions, the Q&A panel. In the

meantime, I'm going to make a quick intro for Synerzip, so if you can advance the page to the next one and maybe do a full build out.

For those of you who are not familiar with Synerzip, this is a quick intro to Synerzip in a nutshell. There are five points about us to keep in mind who we are. We are a software product development partner … emphasis is on partner … consistent with the Agile philosophy or collaboration over contracts. We are a development partner for particularly small to mid-sized, fast-growing, innovative technology companies. These are usually venture-backed companies. For each company who is a client, we put together a very dedicated, tailored and stable team for them which serves as an extension of the in-house engineering team. By working with us, our clients are able to reduce the risks related to software development and delivery because we use the Agile approach, and tailor Agile in the context of a client. If they're very early stage, of course, you need less discipline and more market feedback and responsiveness.

We are able to provide a significant cost advantage to our clients because we have a dual-shore development center. We have a US operation in Austin, Dallas, Houston, San Francisco, but also have a sizeable development center in India. We are able to put together teams and provide a 50% cost advantage.

The last point, is our flexibility. We provide flexibility to our clients by letting them convert the team that is working for them at Synerzip, whenever they feel like it, into their captive operation, if they feel like it. They don't have to, but they at least have the option.

The next page, if we get to it quickly, is just a glimpse of Synerzip clients. These are typical small to mid-sized software companies, no particular focus on technology, no particular focus on industry domain, but just typical small to mid-sized product companies that we work with.

That's about Synerzip. Let's talk now … take on some questions that are queued up here. If you're ready …

Rich Mironov: Sure, please.

Hemant Elhence: Let's talk a little bit first about … at what maturity of a company … at what point do you realize that … in the chart that you had shown a few pages ago, Rich Mironov … that you've outgrown and need to now expand and add another product owner, and not have the same person play the role of a product manager and product owner and be a hero … and so on and on the way it goes?

Rich Mironov: What I see … and let me start at the very smallest startups … if you have a startup of six or twelve people, you don't have a product manager. You have founders who are doing this work along with everything else. If it's a software company, you probably have eight engineers, or either people including techs and docs in the technical organization, one person doing the sales and marketing, and a couple of founder execs. There is no product manager.

I see a frequent failure mode when a startup gets to about 18, maybe 20 people. All of the informal communications, sitting around the table, start to fail. At that point, you really need somebody who becomes the traffic cop, who becomes the conscience of the product in the development cycle.

The first product, whatever, I see coming in at a company of, let's say, 20 people, because things are not working, communication has fallen down, not everybody is on the same phone call. We really need a backlog and a roadmap.

Beyond that, my metric, I think is … at least on commercial software … is you really need to be planning about a million dollars per year per engineer per product. So if I have a product that's taking ten people on the technical side to build, I'd better think that's going to be a 10 million dollar product someday; and that's about the right size for one product person.

Again, I'm probably meshing that altogether. That's someone doing all the product management, all the product marketing … 1-10, maybe 1-20 if they have two teams. Beyond that, they're out of control.

Sometimes on bigger products you might have, again, 5 or 11 or 32 teams working on what's really one product. You're probably going to have one product manager for each $10 to $20 million worth of revenue, and then you're going to fill in the number of product owners that directly cover your development team. Again, if you had seven teams and that's generating $20 million worth of revenue, you're probably out of business.

In any case, you'd expect a couple of product manager and maybe three or four product owners, all in the same organization. Nothing perfect here, different B to B, B to C; but that's my rule of thumb.

Hemant Elhence: Let's take another question which is you showed a page that talked about feature bloat, a lot of features being added along the way. There's been a discussion … some people are familiar with this notion of feature refactoring. Constantly, just in code, development teams are expected to routinely refactor code. Is there a notion … or how do you think about retiring old features which

are not being used, or refactoring and recombining features? Is that a role of a product owner at all?

Rich Mironov: I usually … again, in the distinction, I would probably throw that to the product manager side of the house; but you want to be doing that pretty slowly because customers take a long time, often, to understand their features.

A couple of times a year one needs to look at ones products with a clear eye and decide if it needs to be end of life, if it's not making its numbers. Sometimes products get big, and what we need to do is split them into two products because customers really do use them differently, where they have different audiences.

Refactor is probably not the terminology we use; repackage, re-price, reorganize. I'd expect you to do that maybe every couple of years on a successful B to B product, but to be thinking about it all the time.

Hemant Elhence: Let's take another one on prioritization techniques. Any advice on concrete prioritization techniques that product owners and product managers can use to logically force prioritization that you have seen work in the real world?

Rich Mironov: Sure, and there are a lot of really good techniques, lots of good books on this. I had the opportunity to be a business partner with Luke Hohmann some years ago who wrote *Innovation Games* and has a whole series of online tools around voting and gaming for priorities. I think they're wonderful. I think they're great. I wouldn't take those as gospel. I would never let my customers vote in features without a good set of strategic filters that I apply, but I think those are really good.

There's also a lot of good pseudonumeric ways of assigning value to features and stories. In general I see that at the feature level as useful; at the story level, not so much. If there's 10 or 40 stories that roll up into a useful epic or feature, you get no points for the individual stories; and I'm also very suspect of revenue projections at the feature level because mostly I don't think anybody can do it, but they're good placeholders.

Hemant Elhence: Let's take another one, which I'm going to rephrase it a bit. What should be the product owner/product manager's priority when they are balancing this between choosing between perceived customer needs versus a better engineered feature?

Rich Mironov: I actually think of this as a portfolio problem. I think trading an individual feature or story the customer wants against individual engineering change is really hard, and the numbers don't support it. Instead what I'd say is a balanced product

plan, or a balanced backlog, should spend some fraction of points every sprint, or every couple of sprints, on things that customers don't see but are important.

For instance, I'd like to say we're going to spend 15% of our points, if the product supports it, on fixing bugs and refactoring, and we're going to spend another 10% on architectural changes that we think are important, and we're going to spend 50% on specific customer enhancements.

There's some budget process so that you can trade off not one against the other, but you can look at your long list of bugs, or your long list of architectural changes, and prioritize the most important of those, and spend them out of a budget that doesn't really cross over into customer demands.

That's a layer stage argument. If you're in version 1 of your product, honestly you're only working on features that matter to customers because, otherwise, you're out of business.

Hemant Elhence: The portfolio argument you're making means you're applying portfolio thinking to the team's capacity, engineering teams.

Rich Mironov: Yes. I would say, if you have an engineering team that's working 100% on customer-requested stuff, I'm really worried about the health of your product. I would really want to break out fractional pieces of the capacity, whether you call it story points or whatever, to do the things we know we must do, that customers may not directly immediately perceive, because we expect this to be a long-running product that's going to have to survive.

Hemant Elhence: To come back to a question that we took a few minutes ago which is: Once you are ready to add additional capacity to product management function … let's say you add product. Initially you had a hero, one person playing both the roles; now you are adding two more product owners and expanding the team. Any guidance on how you should split the division of responsibilities in terms of each product owner has a different market segment, part of the backlog that they should tackle, or by some features sets, or by some modular design for our software? How should we divide?

Rich Mironov: Again, let's take the inbound and the outbound type. I think it's usually a good idea for the product owner-ish folks, the more inbound folks, to be aligned the way engineering is aligned, the way development is aligned, because they're going to write better stories, they're going to get better output on the one or two or three teams that are focused on a particular thing.

I usually expect that to be more technology focused, more feature and piece parts focused, which leaves the market segments and the outbound stuff to be catch-can among the non-product owners, if there is such a thing.

Often in a big company there is actually a product marketing, or a verticals organizations, and product management can pretend those done exist; but the most important thing, I think, is that the product manager, the senior person there, has a whole product view of not just the piece parts but, "How do we make the products overall successful and make tradeoffs between Scrum teams and between features?"

Hemant Elhence: Let's take one final question. There will be a few more we have to just leave. For UX work, do you see product owners playing any role in UX design or UX work, or is that completely ought to be left alone to US expert teams?

Rich Mironov: I think that's a lot about the individual people and their skills. I am UX deaf … the equivalent of that. I'm lousy at UX, so I would fail without a great UX person by my side. If you're a product owner or a product manager with great UX, then maybe you don't have to hire that person, or maybe you can share. I think of it equivalently as: can you build software without a software architect? How about if my product manager plays software architect?" I think we'd all see that as a failure.

There are enough hard skills to put in the box to be a great product owner, or to be a great product manager, that I don't think you can require UI/UX unless maybe you're in a very UX-intensive consumer application space.

I would lean toward having great UI/UX talent on the team; but if you don't have it, then you make do.

Hemant Elhence: Thank you so much, Rich Mironov. I think we've run into the limits.