the art of boring™

EP48 | Technical debt

Rob Campbell	00:38	Welcome everybody! Today we're going to probe what might on the surface seem like a pretty dull topic, but it's actually something that I find fascinating as it pertains to either running a business or the ramifications to us as investors. The topic is something called technical debt. To help us explore this, this is going to be a podcast in two parts. First up I'll be speaking with my colleague Dwight Pratt, who works in our business technology group. With Dwight, we'll really define what technical debt is and help us understand what it means—at least from his perspective. Then we'll pull in Karan Phadke, who's one of our equity analysts, to talk about how tech debt considerations influence his work in assessing companies and their competitive advantages, management quality, and ultimately how we incorporate such assessments into our decision-making process.
Rob Campbell	01:22	So first up, Dwight, welcome to the pod.
Dwight Pratt	01:24	Thank you, Rob.
Rob Campbell	01:25	Dwight, you are the Head of Information Security and Infrastructure at Mawer, which is quite a hefty title. Can you tell us a little bit about what this means, and maybe just a bit of your background?
Dwight Pratt	01:34	So I've been an IT professional for around 20 years. I've worked in varying industries, so from telecommunications, to healthcare, to now finance. I've had the opportunity to work at the ground level as an analyst myself. I've also led many IT teams, so I've worked with network professionals, systems analysts, solutions architects, [and] security professionals like myself. I've also had the opportunity to work with several developers. Currently, like you said, I lead the Information Security and Infrastructure teams here at Mawer, which means I'm responsible for securing our firm from cybersecurity attacks and basically managing the underlying infrastructure, which all of our applications run on. That's me in a nutshell.
Rob Campbell	02:21	Seems like a pretty important job, and must have been a fascinating career just as a lot of this stuff—I'm thinking particularly about cyber—just really [came] to the fore the last five, 10, 15 years.

Dwight Pratt	02:32	Yeah, definitely. It's definitely been something that has been my focus for the past, I would say 10 years, heavy focus on cyber with, I guess, all the cyber-attacks that's been happening in the media. We definitely have to be on our toes and on the ball with all the protections that we have in place here.
Rob Campbell	02:50	Can you give our listeners a sense [of] just how big of a threat is this? Obviously people have read about things in the news—I don't even know if you can share this. Like, do you know how much our organization is targeted? Or organizations like ours? I mean, why would we be targets for these types of individuals?
Dwight Pratt	03:04	Definitely financial organizations are massive targets. An article I read—I think it was last week—talked about how financial organizations are being targeted by phishing attacks. For those of you who don't know what phishing attacks are, it's basically an email crafted to fool the recipient of that email. So, usually it will have a link that you need to click on and then that link will take you to some nefarious website, which usually requires you to enter some credentials. If you're unfortunate enough to do that, then that person now has either access to your system or access to your personal information that you use to log into your banking sites and so forth. So, financial organizations are targeted heavily, obviously, because there's massive financial gain. It's definitely something that as a financial firm we have to be very aware of and always ready to fight.
Rob Campbell	04:01	Got it. Maybe that's a good segue into today's topic on technical debt. I mean, I know that's not just a cyber thing, but it's an interesting angle to introduce: what is technical debt?
Dwight Pratt	04:11	The way I would define it is, it's actually a metaphor that software developers use. When they design code, sometimes they have to make short-term compromises in that code, whether it be they have to get it out to market really quickly, or they're under some time constraints or financial constraints. So it started as a software development metaphor, but it doesn't only deal with software, it can also deal with routine maintenances that you have to do to your systems. Those get delayed for whatever reason, or maybe you have a weak IT system architecture. Maybe you have outdated systems, for example, or maybe just your IT processes are suboptimal. It can be a combination of code or just the poor management of your IT systems.
Rob Campbell	04:58	Okay, so “technical” in the sense that it’s coding and IT-related and “debt” in the sense that you’re not running at an optimal level, you’re having to make choices whether explicitly or implicitly that result in a debt of sorts.

		And that debt can either be, well your system just doesn't function as well as it should, or a debt in the financial sense that we'll probably talk a little bit later with Karan about, as to, well, you might actually have to invest in the future to catch up.
Dwight Pratt	05:22	That is exactly correct. That's a very good explanation.
Rob Campbell	05:24	I know you touched on this a little bit, but just broadly speaking—what are the general categories of technical debt that we or others might run across?
Dwight Pratt	05:32	So, I would classify technical debt into three categories. First you have what's known as deliberate technical debt. Often IT folks will know that there is a right way to do something and there's a wrong way. Maybe I wouldn't say “wrong,” maybe a quick way to do something. However, sometimes because of time constraints or pressures to be first to market, they will intentionally choose to do something the wrong way and deal with the consequences at a later time.
Rob Campbell	06:03	So, kicking the can a little bit?
Dwight Pratt	06:05	Yeah, so that's what I would say is the first one: deliberate technical debt. The second is what I would classify as accidental technical debt. This is the type of debt that's unintentional, usually a result of an oversight by a person or persons that have designed or built a solution. When you design a solution, the goal really is to build it so that it not only meets the needs of today, but it also meets the needs of tomorrow and it can adapt and if you need to scale up or scale down, you have the ability to do that. We all know that there is no perfect design and—
Rob Campbell	06:44	—The world changes.
Dwight Pratt	06:44	—exactly. And sometimes you realize after a solution has been in place and you need new features, that implementing those new features is not as easy as you hope they would be and it may require a complete overhaul of your system. That's accidental. You didn't really intend for that to happen, it just happened.
Rob Campbell	07:03	All of a sudden you wake up and you realize, “Oh gosh, we've got some catching up to do.”
Dwight Pratt	07:07	Exactly. Exactly.

Rob Campbell	07:08	What about the third type?
Dwight Pratt	07:09	The third kind is what we would call “bit rot” technical debt. So bit rot is technical debt that happens over time when a system slowly becomes more complex because you've made so many incremental changes to that system over time, and that usually gets exacerbated when you have several different people who are also making those changes. You have a developer that maybe first built the system, and then somebody else will have made a change to that same code, not really understanding what the original code was intended to do. Then, after a while, the system slowly starts to get worse and worse and worse. And then the original design that it was intended for is no longer what it was intended to be. So that's what we would call “bit rot.”
Rob Campbell	07:57	I've got this image in my head of band-aids on top of band-aids.
Dwight Pratt	07:59	Band-aids on top of band-aids.
Rob Campbell	08:00	Casts and crutches…and [the] thing still works along, but doesn't quite get there.
Dwight Pratt	08:03	Exactly.
Rob Campbell	08:04	What's interesting about it—and I think this is just, again, so interesting about technical debt—is this idea of a choice, or a trade-off; that first example of deliberate technical debt. It's like, okay, we know we're doing this, but that might not mean that it's the wrong answer to take on some deliberate technical debt. And as I think just from a management perspective, the trade-off is: what do you invest today, versus what [do] you invest later? And there could be really legitimate reasons to continue on with some of this “bit rot” debt, or whatever it is. Can you speak to that just a little bit more? And maybe from your perspective as somebody in the cyber-world where, maybe you've had to make some of these decisions?
Dwight Pratt	08:42	Yeah. Well, I mean with the pace of technology today, you need to be able to adapt very quickly. So, when you have technical debt that’s poorly managed, this could result in software companies becoming obsolete very quickly. Technology companies need to produce innovative solutions that are rapid paced, that provide great user experiences; consumers want something that’s fantastic.

But if you're unable to evolve rapidly due to poor quality code or to poor IT infrastructure, this can be very detrimental to you as an organization. The same applies to those companies that fail to keep their IT systems patched, for example. We've seen over the past few years a lot of companies that have had cybersecurity breaches due to just not keeping up to date with security patches.

Rob Campbell	09:33	Which seems pretty basic?
Dwight Pratt	09:35	It is a very basic thing, but like you said, sometimes deliberate decisions are made to not do something. Maybe it was, "We're not going to do this patch because of the implications that it's going to have on our application," or "It's going to cost us too much time to do this." So, deliberate decisions are made to not do that.
Rob Campbell	09:57	I would also think that that gets exacerbated if there are really short-term financial pressures on an entity or a company or, I'm even thinking, governments being a pretty good example of that—where there are rollouts and big software upgrades might be politically just less appealing. But of course, as soon as the software breach or the cyber breach occurs, it's like why didn't you do this in the first place? What were you thinking?
Dwight Pratt	10:18	Exactly [laughs], that's always what happens. The pressure is on for a lot of these organizations, and sometimes poor decisions are made. And it usually results in unfortunate things happening to these organizations.
Rob Campbell	10:28	Interesting. I know we've talked about the, like you said, "poor decisions," but who out there in your view and in your work does a really good job of managing tech debt?
Dwight Pratt	10:37	Well, one company that I believe manages technical debt really well is Microsoft. They are a perfect example of a company that creates deliberate technical debt. And I'll give an example: Microsoft likes to be first to the market. They will intentionally release new product with known issues, but what they'll do is they'll fix the issues after the product has been released, because that allows them to be first to the market with that product.

Now, what happens by being first to the market, is this puts pressure on their competitors because their competitors are saying, "Wow, Microsoft has this new product out, we need to compete with them." So now they are under pressure, and they now have to hurry to get their product out to the market. Which, in turn, could result in technical debt for them because now—[laugh] you see the cycle right?—the pressure to be "second" to the market.

		[Which] causes you to make decisions: sometimes you have to not do this particular line of code because you need to get it out there quickly. And as a result, this causes you now to have technical debt.
Rob Campbell	11:46	That's a competitive advantage: taking on tech debt to be first, but as a result, putting everybody else in tech debt. And then, you're putting something out to be first, you're aware that it's not perfect...what does Microsoft do at that point? I mean, obviously they don't want an imperfect product, out there.
Dwight Pratt	12:01	They have a very good process. What they do is they release frequent updates. And what's even better, is they've developed a quite an effective feedback mechanism whereby they learn about bugs and things that are wrong with their software and opportunities for improvement from their end user community. So their end users will actually say, “Hey, we've discovered a bug in your software,” or “Hey, we know that there's a security hole here.” And on a monthly basis, Microsoft has what they call “Patch Tuesday” (which is the first Tuesday of every month), and this is when they would actually release a bunch of security fixes that were discovered either internally by their own staff, or by their user community. They [then] release these patches, which then consumers can deploy to the software to fix those bugs. They're quite good at managing deliberate technical debt.
	12:51	For example, we actually use Office 365 and Microsoft Azure cloud platform[s], and they are rapidly deploying fixes and enhancements—sometimes on a weekly basis. I'll log into my version of Office and I'll see a new feature that I didn't see yesterday. So they're quite effective. And I think the reason why they're so effective is because they have a strong foundation that they've built—an underlying foundation that they've built—with their software. And due to their scale, they're able to reinvest a significant amount of engineering and development [so] they develop new and better products.
		It's very hard for others to compete with Microsoft, and this helps them to ensure that they maintain a competitive edge over the other companies that are trying to keep up with them.
Rob Campbell	13:40	Got it. I want to explore that foundation a little bit more that you spoke to, because I'm not sure that I understand exactly what that means. Are you speaking to the actual infrastructure or code base that underlines Azure or a lot of these other products? Or are you speaking to just, like, [the] cultural foundation within your organization? That “Hey, we're going to be first, we're going to improve, we're going to have this open mindset to solicit feedback”—those types of things?

Dwight Pratt	14:01	It's the underlying infrastructure. I guess one of the things for us here...we're not Microsoft, but we do try to manage our technical debt here at Mawer. One of the things that we like to focus on is creating the right architecture and building a strong foundation. And I'll explain that a little bit more.
		Think of when you're building a house, I'll use a prime example—a few years ago I bought a new home. It was a custom build and as we met with our builder for the first time, we were presented with the blueprints for the house. The blueprints they show you where the walls are and where all the electrical and the plumbing and all that kind of stuff runs. It was at that time that I had the opportunity to actually make changes to where I wanted things to be.
	14:47	And I noticed that the ceilings in the basement were eight-foot ceilings. I thought to myself, I remember doing my basement in the past and once you put the drywall on the ceiling, you tend to lose a little bit of height—maybe a few inches. So I thought, “Hey, could I have nine foot ceilings? I wonder if that's something that's possible.” What the builder told me was, “It's actually good that you told us this before we pour the foundation, because when we pour the foundation, we need to pour it at a level that's high enough so that as we start to frame in your basement ceiling, [it] will actually be nine feet.” But because I was able to catch that at the—I guess the “architecture” stage, we were able to actually build the right level of foundation.
		So when you look at software and IT systems, it's important to architect it well from the get-go, have the right underlying foundation. It makes it a lot easier to make changes to that code or to those solutions later on down the road where you don't have to redo your entire foundation.
	16:01	Imagine you're doing renovations in your house and each time you had to do a renovation...
Rob Campbell	16:05	—You had to go back—
Dwight Pratt	16:06	—You have to go back to the foundation and change the foundation! That gets pretty expensive. So you actually don't want to do that. You want to make sure that the foundation is good, it's architected really well. Making those renovations or those code changes or infrastructure changes then becomes a lot easier.
Rob Campbell	16:23	Just going back to Microsoft—did they just sort of luck into that strong foundation? Or was this something that they had planned out years and years in advance?

Dwight Pratt	16:29	I think it's been years. I've been using Windows since Windows '95 and I remember Windows '95 when it was released, it was... filled with bugs, a lot of issues. But I was also a beta user, so I had an opportunity to provide feedback to Microsoft.
	16:47	I was telling them, "Hey, you know, I found a bug here. Here's what's wrong with it." So I think it's embedded in their culture and it has been embedded in their culture for as long as I can possibly remember. So yeah, definitely strong processes built into that organization.
Rob Campbell	17:01	Just building on culture before I let you go: I'm wondering if you could speak to the flip side of it. We've talked about how there can be consequences to having poor coding or poor infrastructure, but I wonder what that means for people who work within a team or at an organization where there is a lot of technical debt. If you're constantly having to fix bugs or run patches, I mean—does that not just become frustrating as an employee? I guess I'm trying to draw a line between some of the cultural impacts that go beyond just the actual software itself.
Dwight Pratt	17:29	Oh yeah, definitely. I don't want to say "fortunately" or "unfortunately," [but] I have worked in organizations before that ha[ve] had technical debt—whether it be deliberate technical debt or unintentional or bit rot. And what I've found is that having that much technical debt, or technical debt that is managed poorly, usually leads to employee burnout because employees are constantly having to put band-aids on things and fixing solutions and over and over. And with IT folks—developers and systems analysts in general—to have to troubleshoot day in and day out, the same problems over and over...it's just not something that a lot of IT people like to do. They feel like they're spinning their wheels, and usually what it could result in is people eventually just leaving the organization.
		IT folks—they want to be innovative, they want to produce solutions that are cool and helpful to the business. They don't want to be spinning their wheels on a daily basis trying to fix technical debt. So, definitely it will and can impact the culture of an organization, of an IT organization, and I've seen it. I've been in organizations where that has happened before.
Rob Campbell	18:45	Well, thanks Dwight! That's given us great perspective on what technical debt is and your experience over your career with it. Thanks for coming on.
Dwight Pratt	18:52	It was my pleasure. Thanks for having me.

Rob Campbell	18:55	So, for the second part of the podcast, I'm joined now by one of our equity analysts on our global small cap team, Karan Phadke. Karan, we've had a conversation with Dwight about technical debt from his perspective, but I'm really curious for your perspective from an investment standpoint. So, maybe that's a good place to start. From where you sit, looking at companies and interviewing management teams, what is technical debt and why does it matter to you?
Karan Phadke	19:18	Right. So from my standpoint, technical debt is a concept that obviously stems from software development. In that context, that refers to the accumulated costs and long-term consequences of using just poor coding techniques, lack of maintenance, and obsolescent architecture.
	19:36	More recently, I think a lot of folks have caught onto the importance of intangible assets from IP or marketing. But the other side of that would be intangible liabilities, and that's where technical debt sits. It doesn't have a number assigned to it on the balance sheet, but it's still very much like actual debt, in the sense that you can kick the can down the road, but at some point, you do have to pay it back.
Rob Campbell	19:59	So there is a hidden financial cost to companies that can come from these poor coding techniques. In terms of that financial impact...and ultimately as investor...the impact on stock price performance—how does that actually manifest itself? Have you seen that?
Karan Phadke	20:14	I think from a business perspective, having too much legacy infrastructure and a weak code base at some point does make it difficult to improve, iterate, and customize your product offering for the customers. And that can obviously lead to higher customer churn rates, weakening competitive position, elevated expenses; and at some point, you may need to have a massive overhaul—either through an investment in people to update your stack, or actually just purchasing an entirely new technology platform. So, I think it's a leading indicator for financial and ultimately stock price performance over a five, 10-year period.
Rob Campbell	20:53	So at some point in the future, higher expenses: either higher R&D expenses and compressed margins, or some impact that way in terms of your performance.
Karan Phadke	21:00	Yeah, exactly. Especially on that free cashflow line, which is sort of the output of all the intangibles that make up a business.

Rob Campbell	21:09	Got it. It strikes me so far we're talking about poor coding techniques and my mind automatically goes towards tech companies, but is tech debt just a tech company issue?
Karan Phadke	21:19	Yes. I think two points there: the first is it's not just poor coding techniques, it would also be just the architecture that you use in general for your technology stack, as well as just your org structure. So, how you develop—whether it's agile or not, for example. To your point, it's one type of off-balance sheet liability. Non-financial liabilities as a genre applies to all sorts of companies. So as an example, if you have very bad employee relations, you could view that in some sense as cultural debt. Or if you abuse your customers by overpricing, that could be a form of customer debt. Then the same thing on the supplier side: if you're a big company and you squeeze them too hard, that's also a form of liability that at some point might come due. And also a flag within technology: it's a concept that can be applied both to new and old companies, as well as first and slow growth companies—in the sense that [it] remind[s] of the saying "Garbage trucks only smell when they stop moving" so you can have a fast-growing, youthful company that's accumulating a lot of technical debt, nonetheless.
Rob Campbell	22:26	You can be on the right track, but you can get run over if you just sit there. Can you speak a little bit more specifically? Are there examples of companies that you can share with our listeners that have suffered poor financial performance? As it relates to tech debt or some of these other kinds of debts that you've just spoken about?
Karan Phadke	22:41	Sure. A couple examples come to mind—we don't own any of them in the global small cap fund, but I have briefly looked at them. The first one would be a company called GBST in Australia, and they produce financial software. They had to rehaul their entire code base in a huge R&D project over the past five years, which really crushed profits. They've mostly gone through that transition now and [are] actually potentially being taken private, but at a price well below their all-time highs.
	23:10	Similarly, there was a company called MYOB, an accounting software firm that went through a transition from on premise to cloud software in private equity hands. And they went public not too long ago and we actually purchased them in global small cap because they had an entirely new technology base that was refreshed and great for customers and funnily enough, they were also purchased by private equity from us shortly thereafter.
	23:36	I think there's sort of a common pattern, in the sense that public markets are quite unforgiving to businesses that generate a lot of earnings and then stop generating them because they have to reinvest.

And often that sort of transition from an old to a new product architecture requires private ownership because it can be very costly and time-consuming to successfully implement.

Rob Campbell	23:58	Fascinating. From your perspective as an analyst, what are the specific types of things that you look at when assessing a company's tech debt and what might be problematic or what might be just...sort of normal levels of technical debt?
Karan Phadke	24:10	I think the way we try to get some indications around the level of technical debt is, as you may know, and some of our listeners too—we have a team of developers and data specialists within the research team called "The Lab," and they're sort of our go-to source for both tools and technology expertise. So, analysts can work with them to design surveys focused specifically on this topic, and they sometimes sit in [with] us for calls or conferences to really "get under the hood" as it relates to technology issues.
Rob Campbell	24:40	If I can just interrupt you for a second, when you mean surveys, are you actually going out and surveying companies? Or what is that?
Karan Phadke	24:46	Yes, it would be a sort of a question list of both qualitative and quantitative metrics that we can ask a whole plethora of industry participants, whether it's the management team, competitors, customers, suppliers, and so on. From a tactical standpoint, I think one of the biggest things you can do is just have an actual product demo with a sales or R&D person. I think seeing what the product looks and feels like says a thousand words and you can quickly gauge how the application or solution is delivered, customized, and integrated.
	25:19	Beyond that, obviously, as I mentioned—it's helpful to speak with actual customers and competitors. Again, just to verify the longevity of the firm's product from a different angle. Then, on the financial side, basic comparisons of margins and R&D spending across comparable group of companies is helpful. Other empirical metrics that you can track include a frequency of product release cycles, system uptime, and engineer turnover. I'd qualify all of this by saying yes, technical debt is one piece of the puzzle that gives you some clues around the fragility of a business model over the long run, but you do have to balance that with other considerations like valuation.
Rob Campbell	25:58	There're a couple of things that come to mind for me that relate back to our conversation that we just had with Dwight with regards to frequency of release, engineer turnover, and productivity—those types of things.

My question now, though, is based on my conversation with Dwight—who is very clearly full-time involved in these types of things—but is this something that really just resides at the chief technology officer level? Or is this something that you're talking about with CEOs and CFOs?

Karan Phadke 26:22 I think, increasingly, technology as a capability or distribution sort of…competence, is becoming important in all types of businesses. I would question the premise that the chief technology officer sits separately to the rest of the business, and when you view it from that context, technical debt in my opinion is a sweet topic because it does dictate the future capital allocation requirements for a business and also gives you some clues around the firm's culture and long-term vision in general.

I like to see management teams that can provide specific answers on how they've set up the engineering team and development process, how they balance speed versus scalability versus quality, and how they prioritize which features or products to ship. And again, the qualifier being that specific, tangible answers are a lot more valuable than higher level talk about, "Oh, we're quote-unquote agile" without actually getting into the details.

Rob Campbell 27:24 Okay. Are there any KPIs that we look for in that regard? Or is it just company specific?

Karan Phadke 27:29 Beyond, obviously, the financial KPIs (I mentioned some of them earlier), but frequency of product release cycle. So, a company that can iterate and have a new update every two weeks for example, versus every year, can tell you something about the industry that they're operating in, as well as the company itself. So, I think for a KPI like that, you have to compare it within the industry. If you're selling enterprise resource planning software to utilities, you don't need a biweekly release cycle. You should be comparing your release cycle to other companies that sell within that vertical. If you have a much more frequent release cycle, it could be an indication that you're a little bit more flexible and product focused.

28:09 The other one would be system uptime. So again, if you're providing core financial software in a government-regulated vertical, you need 99.99% system uptime because you can't afford any lapses, so comparing those across competitors is helpful. And then finally, engineer turnover, too. I think this also has to be normalized by country and by segment. So, if you speak with a company in the U.S., based in Silicon Valley doing consumer software, you could have 30, 40% engineer turnover, which could be fine compared to other companies in that space. But if you're speaking with some company in Japan or New Zealand that does sort of old-school niche, vertical market software, then maybe 10 to 15% attrition is a better comparable.

Rob Campbell	28:56	In terms of some of these KPIs that you mentioned…I mean how are you accessing this data? Is this publicly available? Do you have to ask the companies themselves? Did they disclose it? Are there industry groups that publish this sort of research? How do you access or get familiarity with [laughs] essentially what you just talked about?
Karan Phadke	29:10	Some of this information you can have just through conversations with different stakeholders and the value chains, so, asking a few different customers, competitors, former employees to get just an indication whether you're roughly right or roughly wrong. There's also LinkedIn for example, or job posting sites. You can scrape them to get sort of a KPI monitor or a tracker that you can play around with, too. I think again, we're not trying to be to-a-decimal place accurate, we're just trying to get a rough indication of which companies are sort of…better or worse, and then trying to fit that in with the rest of the puzzle to get a mosaic view of both the intangible assets and liabilities that a company may have.
Rob Campbell	29:54	Karan, in terms of the global small cap portfolio, are there companies that we have in the portfolio or that we've bought recently where one of the reasons to buy them is, "Hey, we think that they do a pretty good job of managing their tech debt?"
Karan Phadke	30:05	A couple of examples come to mind. These are not that recent, we've been owners for a little while in them, but one would be Technology One and another is Bravura Solutions. Both companies have, over the past few years, emerged with state-of-the-art cloud enabled software platforms, which they're switching their existing customers on to, and they both spend strongly on R&D and have innovation-led cultures. Those are two technology-first software type businesses.
	30:32	We also own Vitech software in Sweden, which we've held for many years, and they acquire smaller niche software firms and will have the capacity to invest in them if required just to extend the shelf life of the products. Then finally, there's one called Kainos, which is a UK-based IT consulting firm that helps the government digitize processes like passport renewals. As you can imagine, government has particularly high levels of technical debt, so someone that can service them is a good opportunity.
Rob Campbell	31:03	That's another angle too, I mean we talked with Dwight about Microsoft and what they do to help their clients with technical debt, but are there other companies in the portfolio beyond Kainos that help their clients manage tech debt?

Karan Phadke	31:14	I think Kainos obviously is the most obvious one because they are a consulting firm, but we also own what we call "value-added resellers." So, these are companies that will act as an outsourced IT function for smaller businesses. If you take Mawer, for example, we may not have biggest IT team, and we may need some advice on what software and hardware we require both for our application software standpoint, but also security. Value-added resellers act as sort of the advisors and middlemen that can help SMEs with this and in some sense help them manage their technical debt.
Rob Campbell	31:53	We've talked about tech debt and what you look for in companies; we've talked with Dwight about some of the stuff more from an architecture perspective that we've considered; and, you alluded to this earlier, but I want to go back to it—stuff that you're doing as a research team to manage the research team's tech debt, or improve the tools we use. I know we've spoken a little bit about this, but I'd just like to marry those two concepts if you think it's appropriate. What are some of the things that we're doing?
Karan Phadke	32:18	I mentioned a little bit about our Lab, and what they really do is push on the flywheel as it relates to tools and processes that we use within Research. But I think a more important part about investing is obviously just evolving and learning because the market changes over time. And the main intangible, or "technical debt" you could call it, for investors in my view, is actually having strongly held dogmatic beliefs that become concrete in their minds. Culture is incredibly important and culture that encourages relentless open-mindedness and flexibility, in my view, is one of the best tools that we can develop to prevent technical debt from accumulating in our mental software, so to speak.
Rob Campbell	33:00	Well, it wouldn't be an Art of Boring podcast if we didn't end with a comment about culture. Clearly permeates a lot of what we do as investors. Karan, thanks so much for joining us! Hope to have you on again soon.
Karan Phadke	33:10	Thanks for having me, Rob.