

Mik Kersten



How Value Stream Networks Will Transform IT & Business

DevOps Enterprise Summit Las Vegas 2018

How Value Stream Networks Will Transform IT & Business

DevOps Enterprise Summit Las Vegas 2018

Mik Kersten



25 NW 23rd Pl, Suite 6314
Portland, OR 97210

The contents of this eBook are a transcript of the complete presentation given by Mik Kersten, “How Value Stream Networks Will Transform IT & Business” at the DevOps Enterprise Summit Las Vegas 2018. To view the original presentation, please visit https://www.youtube.com/watch?v=E5VP3ioSRU8&list=PLvk9Yh_MWYuzV60LtpXOIPNnd6p67dWSN&index=2&t=0s.

eBook published 2018 by IT Revolution Press.

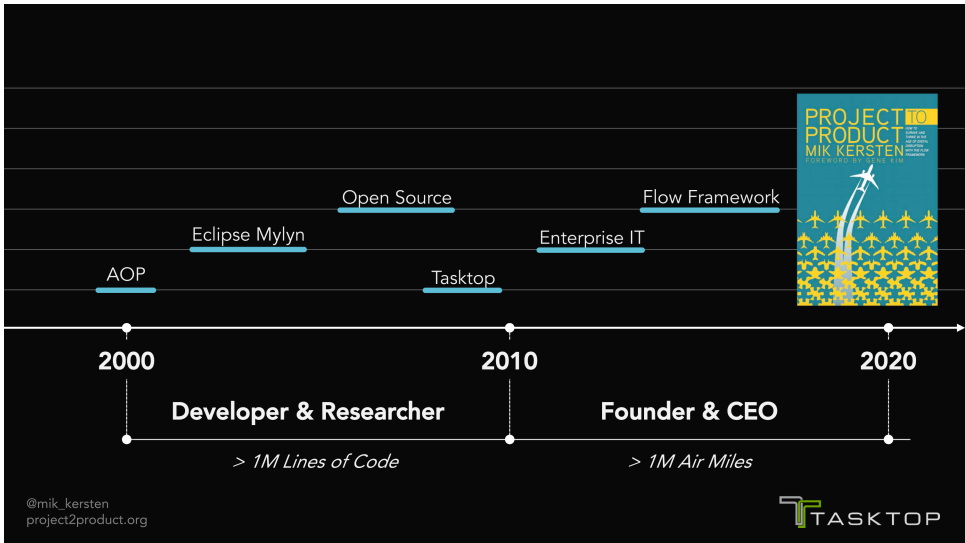
For further information on this or any other books and materials produced by IT Revolution Press please visit our website at itrevolution.com

This eBook is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Hello everyone, I'm just thrilled to be here today with all of you and [to be] sharing the story behind this book, *Project to Product*. My name is Mik Kersten. For my entire career I have been on this journey to help transform how software is built. The first ten years of that journey I spent as a developer and just writing a ton of code, because I thought the problem was actually with the code, with the way that the code was disconnected from the value stream. I actually ended up writing over a million lines of open-source Java code that's still in use today in frameworks such as aspect-oriented programming that you might know from Spring, and tools like Eclipse, my own project that influenced how developer tools work and so on. And as Gene [Kim] said, what I realized is that if we do remove those impediments from developers and connect them [developers] to the users that they're supporting, to the flow of business value, we get the sense of focus and flow and even joy. And I actually got my PhD in software engineering on exactly that topic.

So when my supervisor and I realized this, we decided, okay, let's try to scale this, let's try to make this work for the very large scale. And then we thought, where better to look for those very large scales than enterprise IT. So we started a company called Tasktop to do just that. I learned some very different lessons about what it's like to work in enterprise IT and at that those kinds of scales. Then it was like an open source. I actually noticed a hundred fold productivity difference between how quickly open source developers—I was managing a community of hundreds of open source contributors—could create value compared to what I was seeing in these very large scaled enterprise IT organizations. And I realized the problem is actually much deeper.

I think this morning Gene gave us the inspiration that we can no longer target just improving the lives of ten million developers. We now have to think of forty million IT specialists. So I can quantify this past decade of my life not with lines of code, but with traveling over a million air miles—which I can assure you is nowhere near as fun or rewarding as writing a million lines of code, but it did teach me a whole lot of valuable lessons by allowing me to meet with hundreds of IT leaders who are struggling with this problem and struggling with this weird and broken layer that we’ve put between technology and the business.



To try to bridge that layer, to bridge those gaps, I created this thing called the Flow Framework that I’ll share with you today. And that Flow Framework is what’s featured in *Project to Product*. The goal is to give us a common language between the things we’ve learned over the past ten years of Agile and of DevOps, and bridge that gap between the way that the

2 How Value Stream Networks Will Transform IT & Business

business looks at technology and looks at surviving and thriving in this age of digital disruption.

So for me, the journey started at Xerox PARC. It's 1999. I just got my undergrad degree from the University of British Columbia in computer science and in anthropology, and I had this opportunity to work at Xerox PARC, which is amazing to me. This is the computer science lab where they invented things like the graphical user interface, you know, little things like the mouse and object and programming was pioneered there as well. And I was just having the time of my life because I was getting to work on these super cool new programming tools that were supposed to bring business requirements actually closer to the developers and express them in the code. I started working basically eighty-hour weeks and trying to do this because we have a thriving user community, and I loved what I was doing. And then after some time the RSI [Repetitive Strain Injury] started.



So I got this really bad tendinitis, and then that progressed into some nerve issues in my arms. And I tried, you know, mouse thing with my other hand and so on. But at one point my boss told me that he'd seen several careers end this way, and he suggested some paid time off. This was the absolute last thing I wanted to hear. I was having the time of my life building these tools and coding, and I was far from burned out. It was just my body was not holding up, and I really, then at that point, decided to figure out why this was happening. Why was it that I couldn't keep coding these longer hours with us having so much fun doing it?

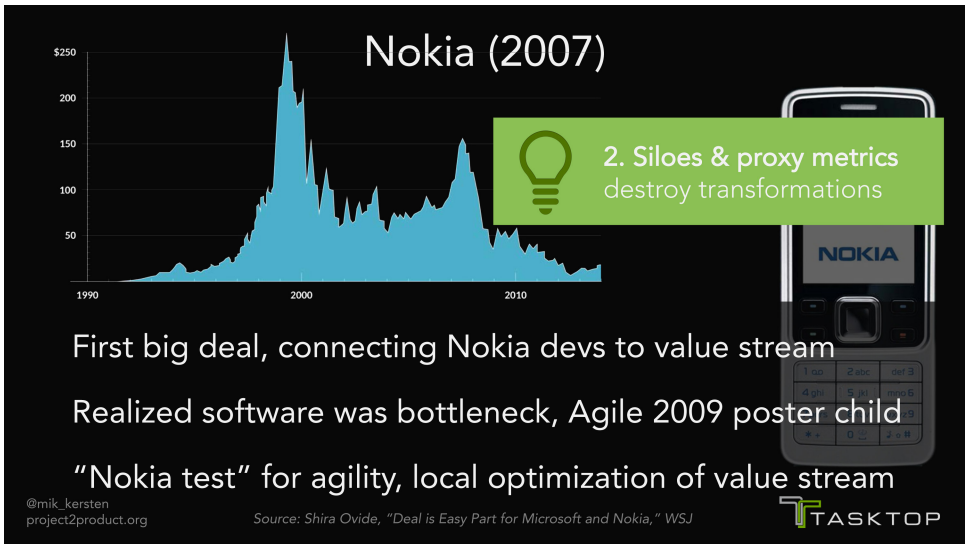
So I started analyzing my own work, and the most surprising thing to me was that it wasn't the coding that was giving me the RSI. It was actually all this thrashing I was doing with all these windows which were invented at PARC. Those felt a little ironic, but the windows, and those windows had stack traces, they had tickets, they had instant messaging, there was email because we were tracking user stories and email at that point, and by the way, we'd done all our continuous integration, continuous delivery automation. I wasn't even thrashing that way. It was actually all the communication in terms of what we were supposed to be doing and how misaligned our software architecture was to that flow of value stream artifacts. so I thought, "Okay, if it's a misalignment, well, we can make a software tool to solve that." So that's why I did this experiment of creating Mylyn. It actually did fix my RSI, because it bridged this gap between what I was working on and the structure in the code.

Other people seem to have this problem as well, so we started seeing millions of monthly downloads, and it was kind of the foundation for my PhD thesis. I realized there was something really fundamental around this weird disconnect between the software architecture and the value stream,

and that was my first epiphany [which is] featured in the book—that these fragmented value streams kill personal productivity. In my case, they actually caused me physical, bodily injury, because I just kept wanting to code. In other cases, of course, you'll end up with burnout, with people who are unmotivated because they just can't get work done the way that they want to.

So now fast forward a little bit. It is now 2007, and that's what my phone looks like. It was...I thought it was a beautiful phone. And it turned out that Nokia had started their Agile transformation. So just like you've heard Jeffrey's numbers or say earlier this morning, [Nokia] realized that they were about to get disrupted. They knew about the capacitive screen, the large screen of the iPhone, a year ahead of time. They realized that they better get better at software. Clearly this company was incredible at manufacturing physical devices, but the screen was going to get so much bigger and going to involve so much software that they realized that software delivery would be their bottleneck. And they realized this in time. They started adopting Agile practices. They started hiring more developers. They did all the things that your company is doing today. And it appeared like they were doing it the right way.

I was invited to speak at the Agile conference in 2009 about how to connect Agile and open source, and I was amazed because every room I went into there was a consultant on stage saying how Nokia was a poster child for enterprise Agile. But then I started working with Nokia, and I realized that the truth was very different from the way that they were being described because what Nokia was doing was...basically just the way that the business was looking at the Agile transformation was how many teams were Agile, how many teams were trained on Scrum.



So the measurement of the success of the Agile transformation was this proxy metric. If you've seen Jeff Bezos' letter to shareholders in 2016, he talks about proxy metrics: metrics of activities, not metrics of result. Or for John Smart's talk about this as well. And this doesn't get agility, basically it tested whether [Nokia] had optimized a very narrow segment of value stream development. You may have heard Kevin Fisher mention this morning that when Nationwide looked at their end-to-end value stream, only two and a half percent of the time was spent actually in development. So now imagine Nokia could have hired twenty times the developers, could have hired twenty Ken Schwabers to help them on Scrum, and it would not have helped this problem. As a result of this, the business not realizing this, Nokia actually completely lost the market, the mobile market that it created.

And this was really my second epiphany, that the silos...that looking at the value stream, this end-to-end value stream of bringing business value to your customers, and measuring proxy metrics. Some of those proxy

metrics are very important. Chain successor is important. Development cycle time is important. But [they're] not enough to make a transformation successful. We cannot measure these proxy metrics and these silos. These are the things that destroy and derail transformations, as we saw with Nokia.

So now it's 2016, and we should have learned all these lessons. But I'm at this bank, this is a top twenty-five bank by revenue, and they're in their third transformation. So the first two failed in very similar ways to Nokia. The third one was not just an Agile transformation, there was an Agile and DevOps transformation. So I thought, okay, this must have a better chance of success. And they actually were taking the approach of automation as well, so there was some fundamental positive differences. The amazing thing was that it was such an important transformation that there was a billion dollar budget for this particular transformation.

2016

Business

IT

3. Project Management & Cost Centers are the wrong model

Top 25 bank, 3rd transformation, now w/ DevOps, \$1B

Project management layer between IT and business

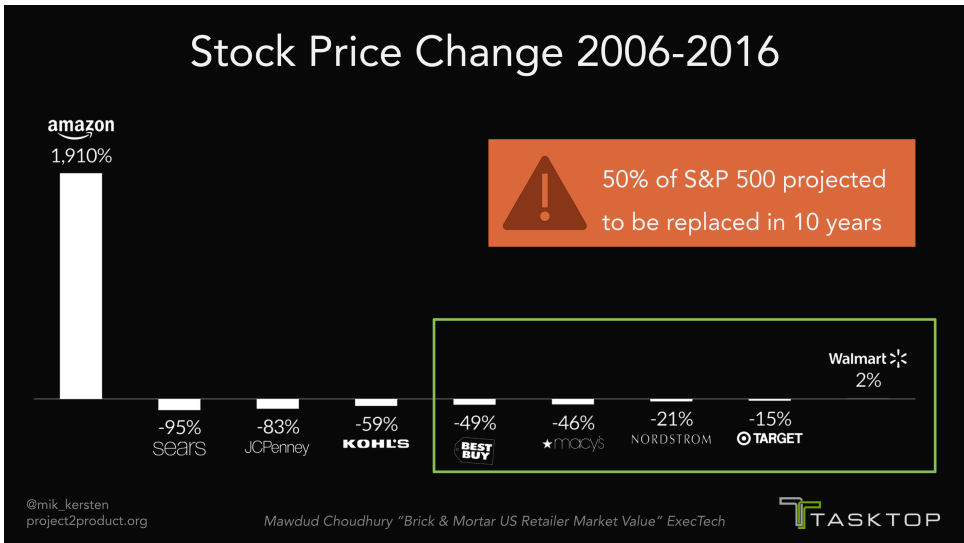
2 years later, IT predictably delivering even less than before

So everyone was pushing for making this thing work. Everyone wanted this thing to succeed. And then I realized that it was going to fail, and this was two years ago. It's now happened. I noticed that there was another very big silo. So there was this...IT was doing the transformation basically within IT, and there was a chief digital officer and those kind of people who were brought in, of course. They were creating these amazing dashboards for this bank. So you imagine these amazing car dashboards, because it was separated. Those dashboards would never work in any of the regions that they were planned for. The software architecture was never going to support it, and the silo that was separating the business side, who is wanting to do the transformation, and IT was completely getting in the way of this.

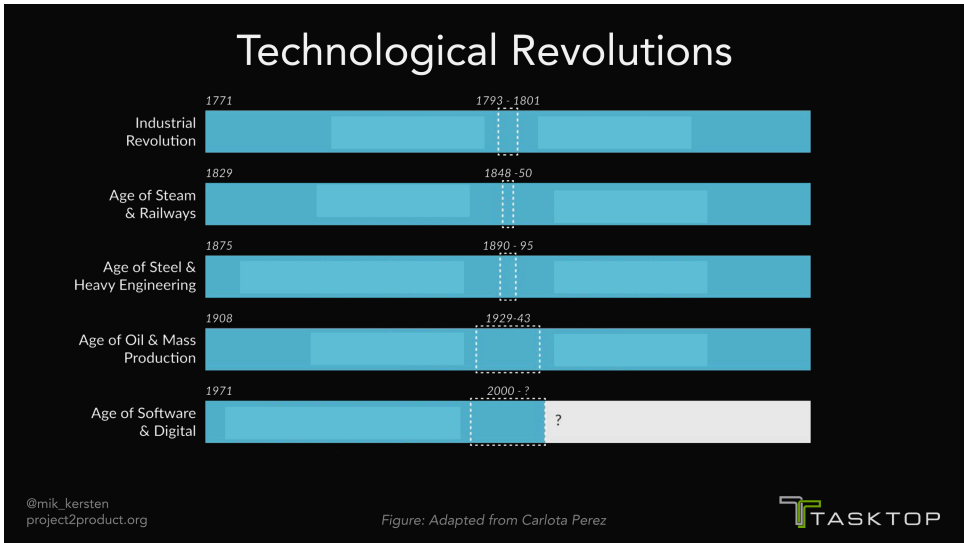
So the transformation was being managed to, of course, cost, because it was a project management transformation. And so, yes, of course the costs were reduced at the end of the two years. In addition, after interviewing the people the next time [I visited] everyone said the same thing: our ability to deliver value was also significantly reduced of course, because it [the transformation] was just managed to cost alone. So this layer that was put between the transformation—the technology side—and the business side completely got in the way. It was the wrong lens. It's the wrong layer for this thing to succeed. And that led to my third epiphany, that project management and cost centers are completely the wrong model if you want to become a software innovator. There are these things that completely derail these transformations, and we have to replace them with something better. We have to do that quickly, because other companies already have.

So this is the stock price change between 2006 and 2016 of some retailers that you might know. Now, the retailer that's got that big bar, that

1,910 percent stock price increase, that retailer, Amazon, has completely aligned its business strategy to its software products to its software architecture with microservices to its organizational structure with two visa teams, and they've completely out-innovated everyone else in the process. Of course, some of that stock price is AWS, and there's more to the story. In addition, it's great to see that some of the companies that are speaking here at this conference are actually making steps in the right direction. For example, Target's prices are near its all-time high, but this is not just retail. We've got at this point, this world where some organizations have done this, they do not have this project management layer that's assuming a high degree of certainty for what will happen for the next few years of the market. They've become software innovators, and the change is now so rapid between those who've managed software delivery at scale and other organizations that at today's churn rate, half of the S&P 500 will be replaced in the next ten years. So this is happening, and it's happening quickly.

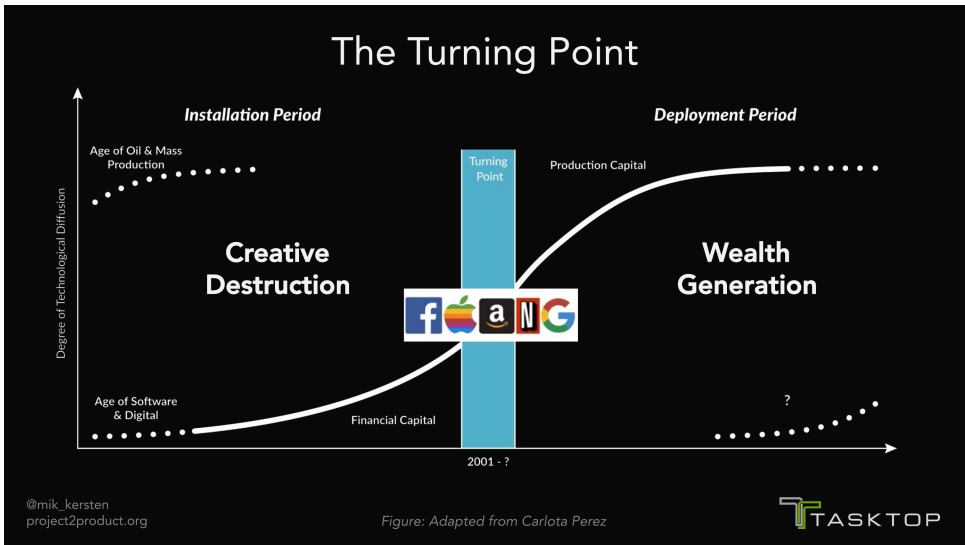


And as I was wondering about this. I was wondering, has this ever happened before, because I think for all our careers we've been used to all this crazy change, JavaScript framers coming out every two weeks and so on. But have we seen this rate of change? And Gene introduced me to the work of an economic historian by the name of Carlota Perez. She has studied technological revolutions, and she studied the last five technological revolutions, starting with the Industrial Revolution, the Age of Steam and Railways, Age of Steel and Heavy Engineering, Age of Mass Production, and where we are today, the Age of Software and Digital.



And a really fascinating thing that she discovered was that these revolutions come and have two phases. There's an Installation Period and there's a Deployment Period. In between them, there's a Turning Point. So in the Installation Period, there's big frenzy because some new means of production becomes very cheap. So you might have mass production. This is the period in Detroit over a hundred years ago where there were three

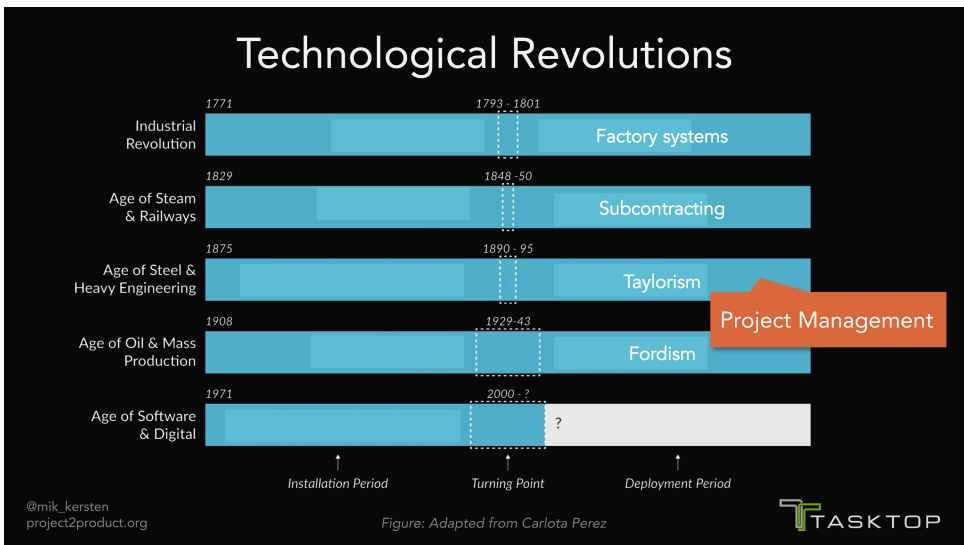
hundred car startups...over three hundred car startups in Detroit. And this [was] fueled by financial capital, today [it's] venture capital. And as another example of that, there's a conference on Fintech in Las Vegas today. There are over two thousand Fintech startups today. So these companies are mastering the new means of production. And what's interesting, is that this causes a tremendous amount of creative destruction, which has happened in every one of these technological revolutions, and then some masters of the new means of production emerge.



So we can now call those the FANGS: Facebook, Amazon, Netflix, Google, you can include Microsoft, Alibaba, fd.com, and so on. These companies have mastered the new means of production, and they're amassing great amounts of wealth. And I think the key thing right now is to help all the other companies in the world economy move into this wealth generation that happens when we have the dissemination of this new means of production. So it really becomes a question: how do our companies

survive this turning point? What did we learn from all those tech startups? What did we learn from the tech giants? What are they doing differently than enterprise IT organizations who are still responsible for the majority of the world economy are doing today? And how do we have our company survive this mass extinction event that happens in a Turning Point? Because we are currently in the middle of that Turning Point.

So if we look at those revolutions, each of them has brought with it a new managerial discipline. There've been new financial systems discovered—things like factory system sub-contracting—then Taylorism, and then Fordism. What's so fascinating, and from conversations with Carlota Perez this became clear [for me], is that Taylorism is actually where we learned how to do really interesting, amazing projects. The Hoover Dam, not far from here, that created Gantt charts, which is an amazing thing. But you could not create an economy of mass production that way, where mass production actually manages Lean methods and products and value streams.



What I've noticed is happening in these companies is that a lot of them were actually founded and established before the start of the Age of Software, before the 70s. We're still using management techniques from two ages ago to run our businesses and our strategies. And so I want to look at whether this is...how did this happen? How did companies, mass production companies, survive the last Turning Point?

I actually looked at a company that we've worked with very closely: BMW. They're clearly one of the masters of mass production, and they made it through that last Turning Point. What did they do? How has their business connected to production? And could we learn anything from that?

I had this amazing opportunity to visit the BMW Leipzig plant where...this is René Te-Strote on the right. He's actually speaking with Carmen DeArdo tomorrow and can tell you more of the story and more of what BMW is doing in their Agile transformation. But the thing that is so amazing to me about this plant is that it's got the complexity...not only the architectural elegance but the complexity of this plant is tremendous. There's a new car delivered off the end of the production line every seventy seconds, a new 1 or 2 series model. And those cars are actually delivered in the same sequence they were ordered. It's called just-in-sequence delivery. With an ecosystem of twelve thousand suppliers putting parts into those cars, and of course, many suppliers providing the different robots and parts on the production line.

This actually...we don't have an excuse of complexity in enterprise IT. This is somewhere near that same scale of complexity, but no one there is talking about project management. They're living the Lean principles that...they're living the Lean principles that here are listed from Lean

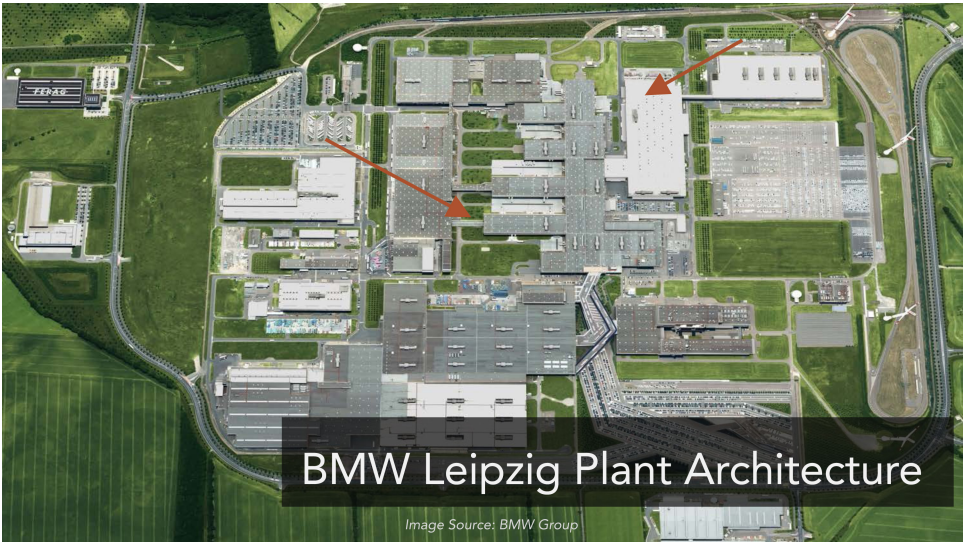
thinking. So those principles are precisely specified value by product. Identify the value stream for each product, make value flow without interruptions, and let the customer pull value from the producer. Because in the end, products are about customer pulls. That's what we're meant to deliver. And so in this plant, for example, everybody knows where the bottleneck is, the CEO knows what the bottleneck is with the limiting factor of the planters.



Meanwhile, all those meetings I did traveling, I asked many, many times, “What is the bottleneck of your software production?” And IT executives or business leaders will just give vague answers or a blank stare, because we're not even sure about what the units of production are in software in terms of the way that the language of the technology side and the business side. And BMW is so amazing with architecting and everything around these value streams. You can actually see the value stream architecture from space.

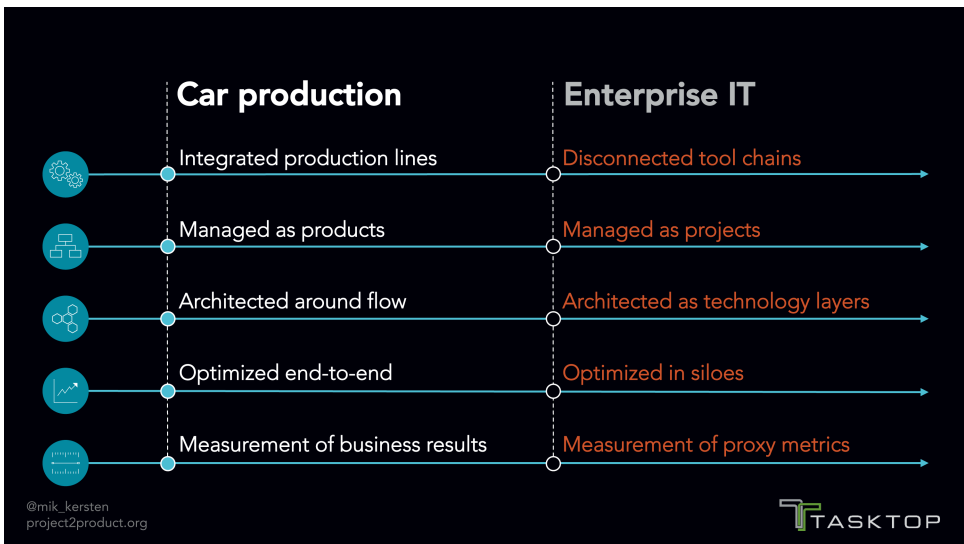


So this over here is the 1 and 2 series building. This is a very long high-automation line because it's, again, delivering those cars every seventy seconds. No manufacturing step can take more than seventy seconds. The cars weave in there, and this is extensible. They can add new manufacturing steps by extending those buildings.



BMW did not know how much demand there would be for their electric series, the i3 or i8. This made it a very configurable line that i8...the time to create an i8 is actually for each step of productions thirty minutes. There's much less automation then in the 1 or 2 series. So everything is configurable, as these value streams in the business understands the product lines and the value streams.

And let's compare this to what we're doing today within enterprise IT. In car production, we've got these beautifully integrated production lines, not these disconnected toolchains. Everything is managed as products, not as projects. Everything's architected around flow, not technology, layers, front ends, back ends, and so on. Everything's architecturally in these product lines. Everything's optimized end to end, not optimized in silos, and everything, of course, is managed by business results, not this proxy metrics of one aspect of the silo.



So if we look at business value flow, because fundamentally I think that's the problem, we've not agreed upon what production, what productivity is in software delivery. We don't have a common definition. If we look at business flow at the BMW plant, it's about quality cars that deliver "sheer driving pleasure." Interestingly, those cars are designed in yearly cycles, even though they're delivered every seventy seconds, so there's a difference between what they're doing and what we're doing. But the creative and the manufacturing processes are decoupled. They're actually in separate buildings, and of course all of this flows across a linear line. The line stops; everything stops. That's a little bit different than what we see.





In IT and technology and software, we delight our customers, not with releases, because those should happen continuously now, but with new features. They're pulling new features, they're pulling new value, our customers are. Those things are designed sometimes on a daily basis, weekly or daily basis, and the creative and the manufacturing process are one.

So that's a pretty fundamental difference. And the flow is not linear. It's not a batch process of delivering a feature. It's a flow across a complex value stream network that includes all the specialists, designers, developers, support staff, and operations involved.

So the Flow Framework aims to answer this core question that what flows into software delivery. We need a common definition of productivity and flow, and it's these four units. Features: that's new business value that's pulled by the customer. Defects: quality improvements, also pulled by the customer. Risks, so the reduction of risk; security, availability, compliance, and so on, pulled by people like risk officers. Debts: technical

debt improvement, and these are pulled by architects, by developers, and so on.


What flows in software delivery?

 Features <i>New business value, pulled by customer</i>	 Risks <i>Security, availability, compliance, pulled by risk officers</i>
 Defects <i>Quality improvements, pulled by customer</i>	 Debts <i>Technical debt improvements, pulled by architects</i>

*Flow Items are MECE**

@mik_kersten
project2product.org

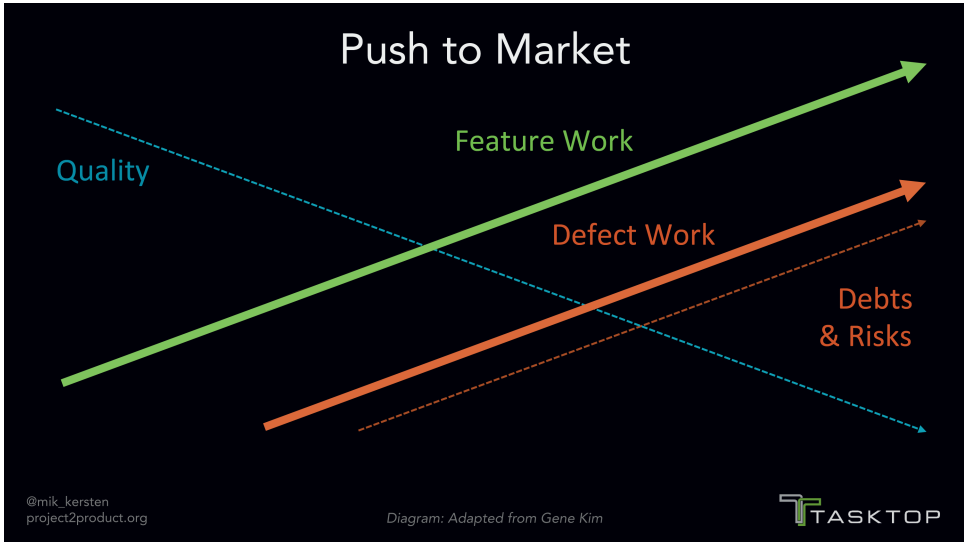
*Mutually Exclusive and Comprehensively Exhaustive



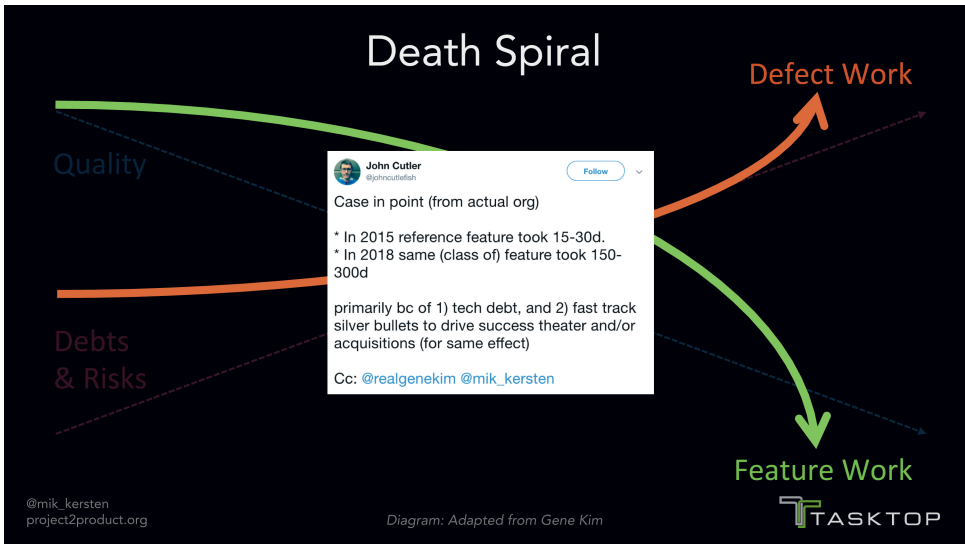
These four flow items are mutually exclusive and comprehensively exhaustive, which means they represent all work that's being done. You can have these very specific taxonomies, for example, in the scaled Agile framework, which has a very, very nice taxonomy that we use. All of those types of work items map into these four flow items.

I'll give you an example of how we can use these to express a common language between the business leaders and the technologists. So this is...imagine your last big push to market. Of course, with a big push to market, it's features, it's your ability to deliver great features to your market, to your users, that's probably going to define the success of that product. As you do that without push to features, there's a cost because it's a zero sum game between those four flow items, which means that you take on additional debt and risks. And the result? Quality, of course, goes

down as you do that, and defect work goes up and up and up to the point where you can easily get into this death spiral of—[I've] been there more than I care to admit—where defect works keeps rising. This is where we can actually get the burnout.



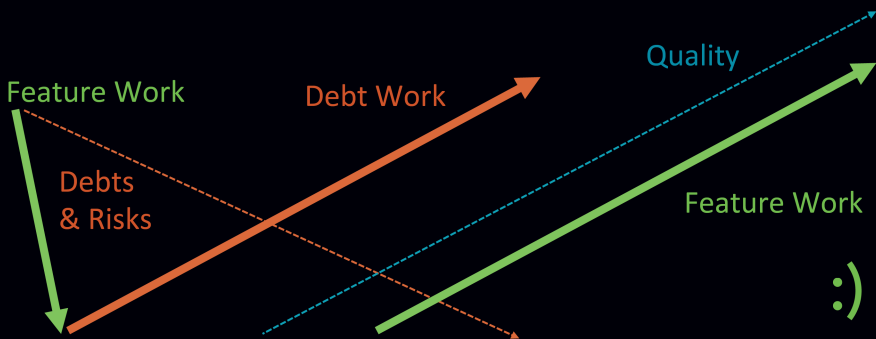
Developers want to be delivering features. They want to see the impact of their work on your customers. But of course they don't get to do that. And that's where we actually measure, as you'll see, things like employee Net Promoter Scores drop as less of their work is done. And this is exactly the situation that Nokia got itself into. And just as a case in point, this is by John Cutler. This is the kind of things that of course we've all experienced. In 2015 a reference features took fifteen to thirty days. In 2018, the country's probably now bigger, 150 to 300 days. And of course, the technologists understand why this is happening. They know they need to invest in technical debt reduction. The problem is the company, the organization, the businesses not understanding that.



Now, contrast that to the security standard that Microsoft witnessed in 2003 with SQL Slammer. So what Bill Gates did is [he] sent this fascinating memo back in 2003, before security is front-page news, and said across our value streams we're going to basically drop the feature work; we're going to reduce our debts. And then the debts were reduced, and this was of course the trustworthy computing initiative. As a result of that, Microsoft assured its place in the future and the way it's been able to do things like move to the cloud, where, in the end, they're going faster than ever and more securely than ever. So this is exactly what Nokia, at a business level, failed to do.

The whole goal of the Flow Framework is to provide this common language, the thing, of course, that the tech giants who have CEOs who were previously software developers already innately understand, but to make this [thing] more accessible to our business leaders and to come up with a common language and framework between the technologists and the business leaders.

Debt & Risk Reduction



@mik_kersten
project2product.org

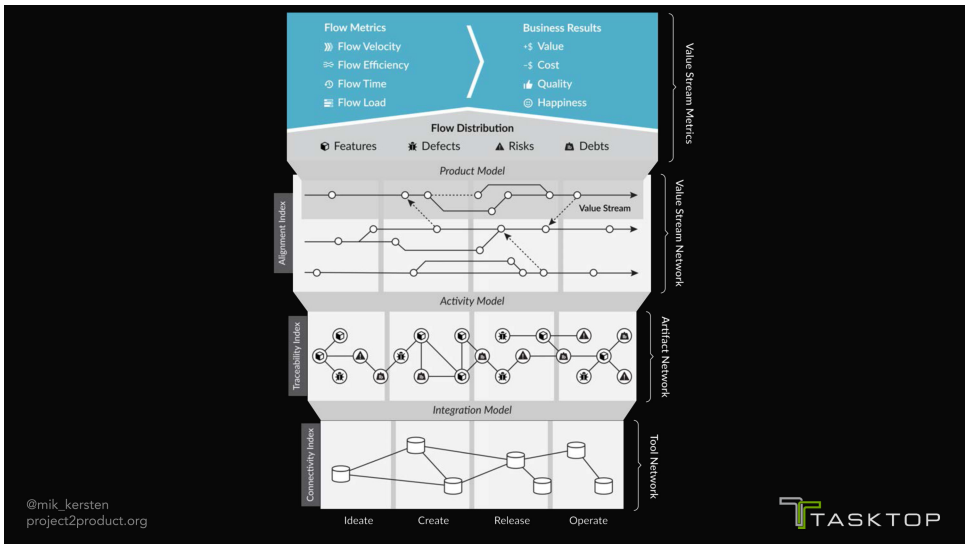
Diagram: Adapted from Gene Kim

TASKTOP

So I'll just give you a really quick overview of that. This idea of the Flow Framework is that you've got this flow distribution and you're measuring for flow metrics, how many of each of those flow items you delivered for a sprint, a release, a year, and so on—the efficiency of that delivery. Because we can measure that in the value stream network, the time...how long it took for a feature or a fix to get to market and flow load. If you're familiar with Dominica DeGrandis's book or a Donald Reinertsen's work we know that putting too much load on a value stream causes things like burnout and actually decreases productivity, so we've got a measure for that.

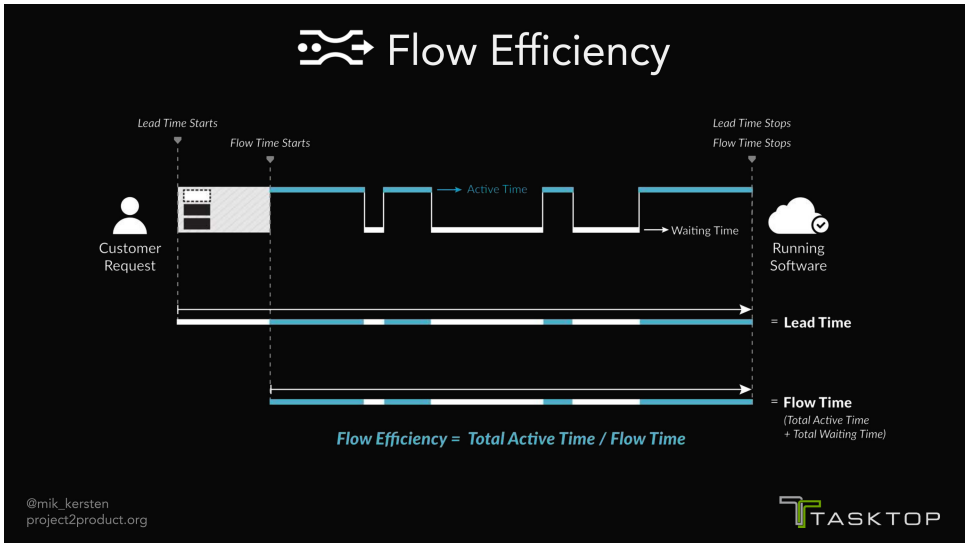
And the key thing is you do this for each value stream. You don't do this for operations. You don't do this for one feature team. You don't do this just for a Scrum team. You do this for a value stream, which tends to be bigger than feature teams, and you correlate that to business results that have to be specified for that value stream, so those business results can be revenue. If it's an internal product that you're doing, the internal product

value stream, such as an internal developer platform or API, can be an adoption metric. You measure cost, but you measure costs for the value stream across all the different specialists involved in building a value stream. You measure quality and you measure happiness, but you measure happiness with measures like employee Net Promoter Score for all the people working on that value stream because then you'll have an early warning indicator that if you put too much load on one set of teams, you could cause them to burn out and cause developers to start leaving because they're so miserable or you didn't give them the chance to reduce technical debt.



This is just a high-level sense of the Flow Framework, and this is how you'll get to end-of-life all those products. Project management, things go on forever. If you see business results dropping while you're adding more features, maybe it's time for that thing to die. And on top of this, we can—later on these metrics are covered in the book, such as flow efficiency. So

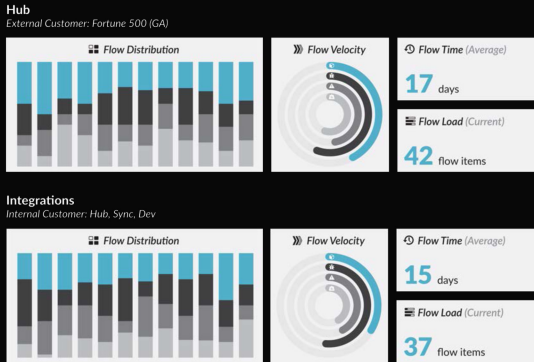
for example, we measure end-to-end flow time rather than, again, just looking at development cycle time, or code commit to code deploy. We really need to take the customer's perspective of flow, which has to do, of course, with the end-to-end activity ever since that feature was submitted, that defect fix was submitted, we measure there.



And we can create these dashboards using those flow metrics, the key thing being those dashboards are all for each value stream. So you've got this level of visibility. This won't tell you if you bring the right features to market, but it will actually show you when you've got a bottleneck, when you've got one of your time fees, such as too many dependencies between a value stream, slowing a single one of the value streams down.

So the question becomes, how can we do this? And how can we really go from silos and proxy metrics to this connected value stream network that we can measure, that we can optimize, that we can manage, and that we can see?

Flow Metrics

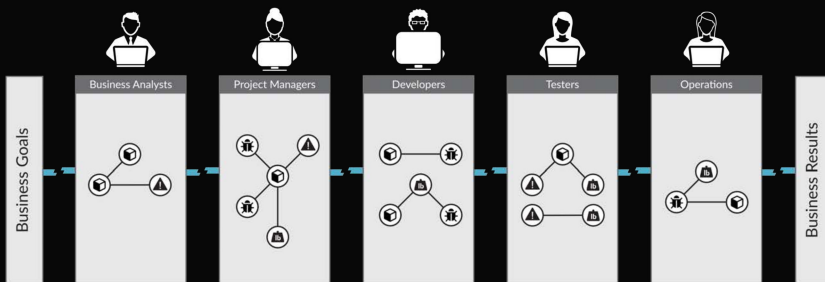


@mik_kersten
project2product.org



And the idea here is that you basically take all these different tools and create a tool network out of them. You stop thinking about individual tools. You create this abstract model on top of that, that you can actually measure, and you can measure across flow on that model. We call this the artifact network, [which] sits on top of your tool network.

From silos and proxy metrics

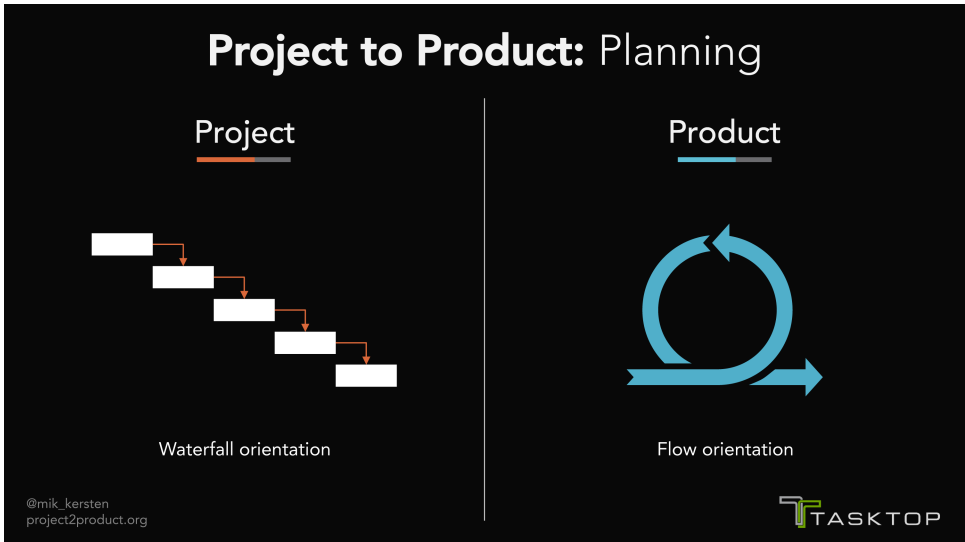


To a connected value stream network

@mik_kersten
project2product.org



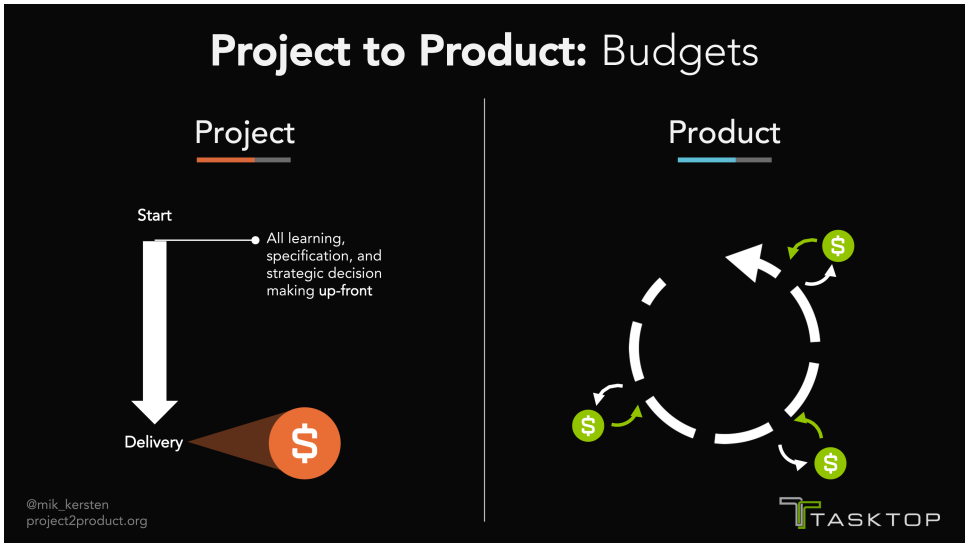
You can then measure flows across that network, and you can measure those flows and map them into your product value streams and measure the flows in those product value streams. So you can go from this waterfall world of projects, where, again, you're assuming that you know exactly what you should be living for years on end to this flow orientation, but measure the results of that flow orientation.



You can go from this world where everything has to be specified up front to what some very innovative companies, including tech giants, things like what we're doing at Tasktop, which is you create a product development budget, but you allow the reallocation of budgets between product value streams on a much shorter timeframe than a year so you can adapt your market for wherever you are getting those business results.

And of course, this key thing is that you stop bringing people to the work; you bring work to the people. So you allow those teams to form stable value streams that are bigger than just feature teams. Feature teams

are a great start, but you give the entire value stream the ability to have autonomy, mastery, and purpose, and you allow them to set the flow distribution. Those teams, the product value stream teams, will know when they should produce more tech debt, for example, to reach your north star, whereas where they should run faster on feature delivery.



So, just as a really quick example of how we do this at Tasktop, because the Flow Framework has absolutely changed how I look at managing software delivery, we just connect our values through the network. We've got all our customer requests, similar to BMW. Where their's are coming in from customer orders, ours are coming into Salesforce, they're going into product management tools with target process, then to Jira, which is connected to the contractors, then to our service desk, and so on.

much that flow load would affect the happiness of this team. And the fact that I'm not even sure we completely communicated to them that they'd be able to reduce it at some point in the future, even though we were thinking it. And this...actually this became a real problem. But I can now see the problem. I could also see that the team who was unhappiest the year before, who actually were given the time to re-architect their continuous delivery and their entire test infrastructure, is now the happiest team.



Being able to see this, this is fine when you're like us, and if you are looking usually at six of these, imagine this across a portfolio of hundreds of internal and external products. This becomes a really big deal and a really important tool. So my advice to people kind of more on the business side is ensure that you're defining the product portfolios and these product value streams and that the metrics are being tracked, and empower the delivery teams themselves to allocate flow distribution. They're the ones

who know when they need to reduce tech debt. You just need to listen when they need to be given that chance.

Again, this is why we need a common language. This is why the leaders of the tech giants know when to listen to their delivery teams. Sometimes you'll actually need to do things differently to hit a north star. For example, you might need to do a major risk reduction and major investment and things like GDPR, and you need to accept that that's going to take away from the flow of, say, feature delivery.

To technologists, create your value stream network by connecting all those tools. Again, abstract over that tool network. Define your value stream architecture. This is as important as your software architecture. That's probably one of the main things I've learned. And then use those flow metrics to identify bottlenecks. You can then dig in and see, okay, is the problem that this team is actually...their testing pyramid is inverted? Is the problem that they haven't...they've said they've done it, but they've actually not done their CICD pipeline properly? This will kind of be your early warning signals, or you look at where to extract a platform because everyone's got dependencies on this one internal component or a team.

So, quickly, the ideas that is we go from project and cost centers to these product value streams; from silos and proxy metrics, the flow metrics and business results, fragmented value streams to this integrated value stream network.

And so, the book will launch on November 20th. I'll be signing some copies of the book tomorrow. We're trying to document some more best practices around this at flowframework.org, or you can follow the book on projecttoproduct.org.

And then the main ask I have is that Gene and I had been trying to understand how best to present this to CEOs, to our boards, to our executive team, so they can understand where to invest in helping make us all successful and bringing our companies through the Turning Point. So Gene had that cool idea of those system dynamic diagrams that you saw. If you have any ideas of how we can better present this to your organizations, please get in touch and let us know. And thank you very much. I hope you enjoy the book.

About the Author



Dr. Mik Kersten started his career as a Research Scientist at Xerox PARC where he created the first aspect-oriented development environment. He then pioneered the integration of development tools with Agile and DevOps as part of his Computer Science PhD at the University of British Columbia. Founding

Tasktop out of that research, Mik has written over one million lines of open-source code that is still in use today, and he has brought seven successful open-source and commercial products to market.

Mik's experiences working with some of the largest digital transformations in the world has led him to identify the critical disconnect between business leaders and technologists. Since that time, Mik has been working on creating new tools and a new framework for connecting software value stream networks and enabling the shift from project to product.

Mik lives with his family in Vancouver, Canada, and travels globally, sharing his vision for transforming how software is built.

Project to Product: How to Survive and Thrive in the Age of Digital Disruption with the Flow Framework

By Mik Kersten



Now available in paperback, eBook, and audio formats from all major retailers: <https://itrevolution.com/book/project-to-product/>.