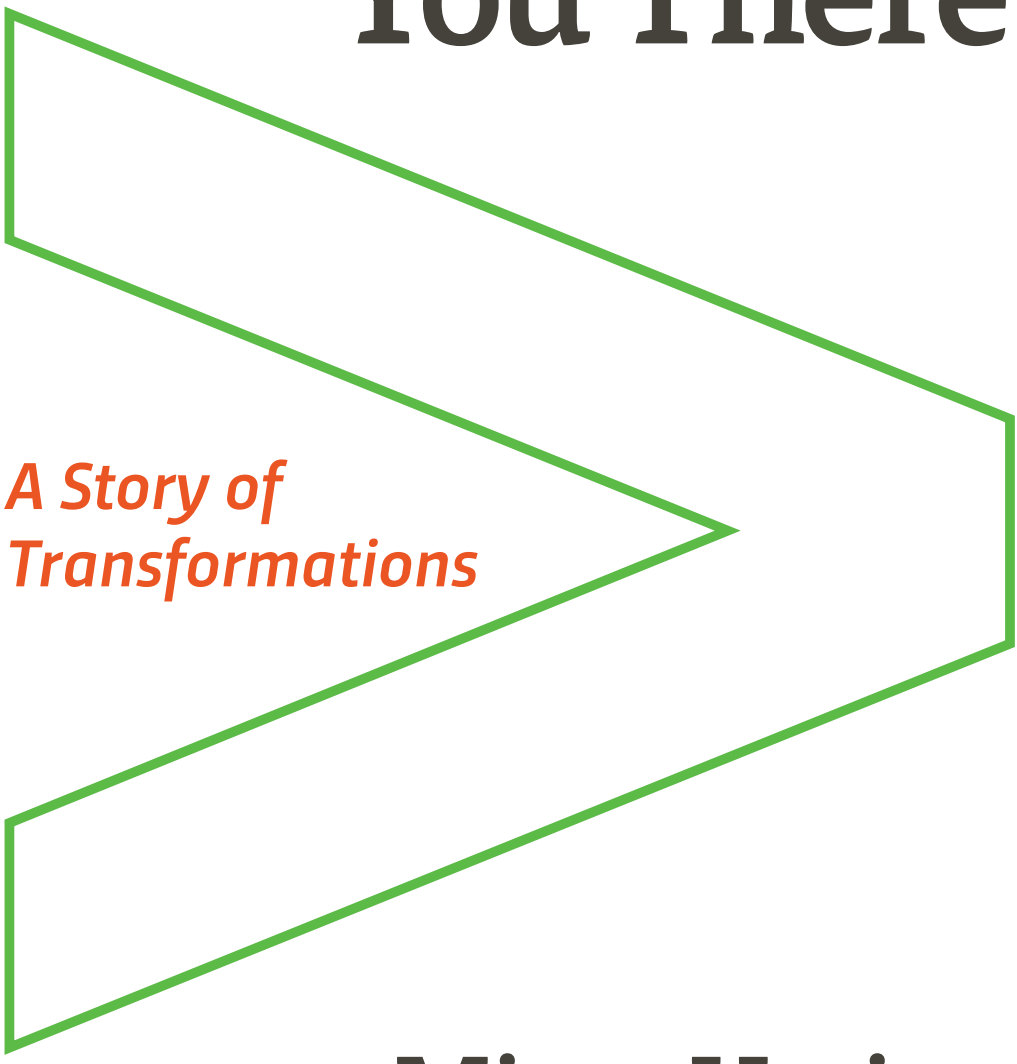
# What Got You Here Won't Get You There



*A Story of Transformations*

Mirco Hering

*DevOps Enterprise Summit San Francisco 2017*

# What Got You Here Won't Get You There

## A Story of Transformations

*DevOps Enterprise Summit San Francisco, 2017*

Mirco Hering

IT REVOLUTION

25 NW 23rd Pl, Suite 6314

Portland, OR 97210

The contents of this eBook are a transcript of the complete presentation given by Mirco Hering, "What Got You Here Won't Get You There: A Story of Transformations" at the DevOps Enterprise Summit San Francisco 2017. To view the original presentation, please visit https://www.youtube.com/watch?v=iuk1RZuWz-I&t=7s.

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 itrevution.com

# What Got You Here Won't Get You There: A Story of Transformations

Thank you, guys, for coming to my presentation. I do understand that I stand between you guys and drinks, and at the risk that you all are going to get up and leave now, I am told that in the exhibition hall they're already serving beers. I'm actually now physically between you and drinks, you could actually leave now and have beers. So, thanks for staying.

I'm going to talk about, well, the talk is called "What Got You Here Won't Get You There: A Story of Transformation." I really want to share a bit of my experiences and what I've learned along the way. You have to have a slide on which company you work for. I work for Accenture, I assume that many of you guys would have heard about us. I'm looking after our Agile and DevOps practice in Asia-Pacific, or as I like to call it, just good delivery. Unfortunately, good delivery doesn't sell, so Agile and DevOps it is.

What do I do at work? So, I have a team of pretty passionate change agents that are working with a bit of our clients, and we're really trying to solve our clients problems. The other day I was talking to one of our clients, and he asked me why I'm still in the job that I've been with Accenture for about fourteen years or so. And it's really to be able to solve problems now, and with Agile we can connect with our customers. It used to be you'd get the requirements document, which is 500 pages long, and all you're doing is implementing that thing that you might or might not

know what it is about. And with Agile and DevOps I think we really started to break the barriers.

I also blog at *Not A Factory Anymore*. If that title sounds weird, you will realize as part of this talk why I call it that. And then, the coolest thing I saw today was the first physical copy of my book, which I'm also handing out afterwards. (audience applause) Thank you. It is really kind of a nervous, scary thing where you're really worried that people will call your baby ugly. So be kind!

Alright, so I do have to admit that I stole the title "What Got You Here Won't Get You There." I'm not sure whether you guys have read this book. It is absolutely brilliant. It describes kind of, for business, why in your career you might get stuck when the things that used to work for you don't work for you anymore. So, I stole it and wanted to acknowledge that.

What I'm here for is, this is the core problem that I'm struggling with, alright? So, I came into the workforce fifteen years ago, and I believe that we are still solving the same problems. When I was working for Daimler-Chrysler ages ago, we built controls and schedules to deploy software into production, we had some kind of monitoring that had been built into the applications to log some stuff out, and here I am fifteen years later we are still talking about deployment automation. We still seem to be in a situation where, like one of the speakers this morning said, "where source code is being emailed around."

So, what happened? I guess I wrote a blog about it but let me give you the cliff notes. Here's where we were [see Figure 1]. This is a pretty generic DevOps maturity model, but it's just for illustration purposes. So, we were actually kind of in this place where fifteen years ago you had to do automation because it was the only way for you to become more efficient.

You didn't have any kind of shortcuts to take cost out of your delivery. So, what happened then is that we started to see more packaged software, which means we didn't have to worry about this anymore, it's just customizations, the product will take care of everything. We had the opportunity to take work and just move it to a different location to become cheaper. And we had these shortcuts that we could leverage. And then we started to get into locked-in environments where you don't even see the code anymore.

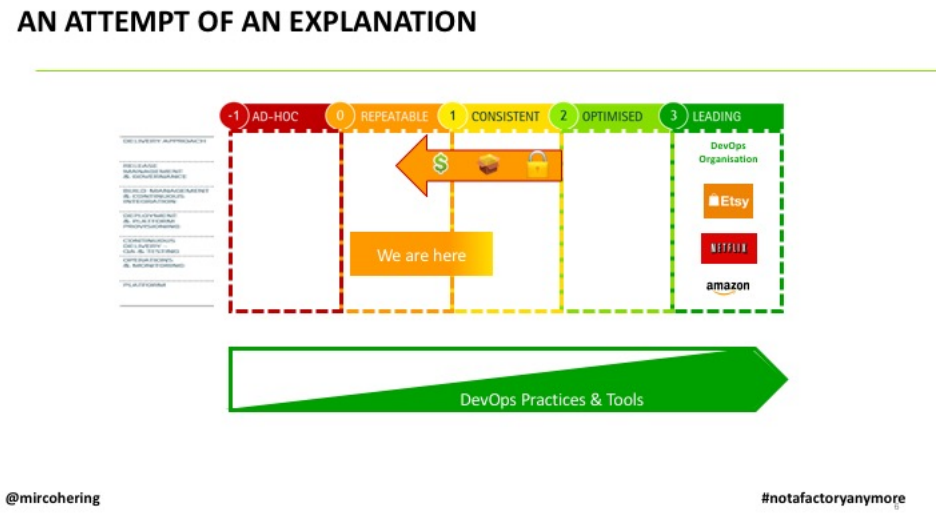



**Figure 1**

I think that's part of where we regressed over time, and we are now starting to realize—well, "now" for the last few years—that we need to bring some of that stuff back. That we've run out of shortcuts. That the packaged software requires a hell of a lot of customization for what you need to really be different in the market. And we all know that in the mobile and digital market you need to have a differentiated experience for your customers. So, we needed to start to do something different, and

that's where we ended up with DevOps practices and tools that are now coming back, and we are revisiting a lot of those things that we used to do.

To me, the problem of why we got here, and why I think they are changing now, is that it's a mindset problem. It's not that we consciously made these decisions. It's not like we said "Oh, yeah, forget about automation." We just had, or we convinced ourselves that there is, a faster way of getting the answer. That mindset shift, and a lot of the things that we've learned, came from manufacturing or from factory models, and I think that's what I wanted to explore a bit with you on why that's not helpful.

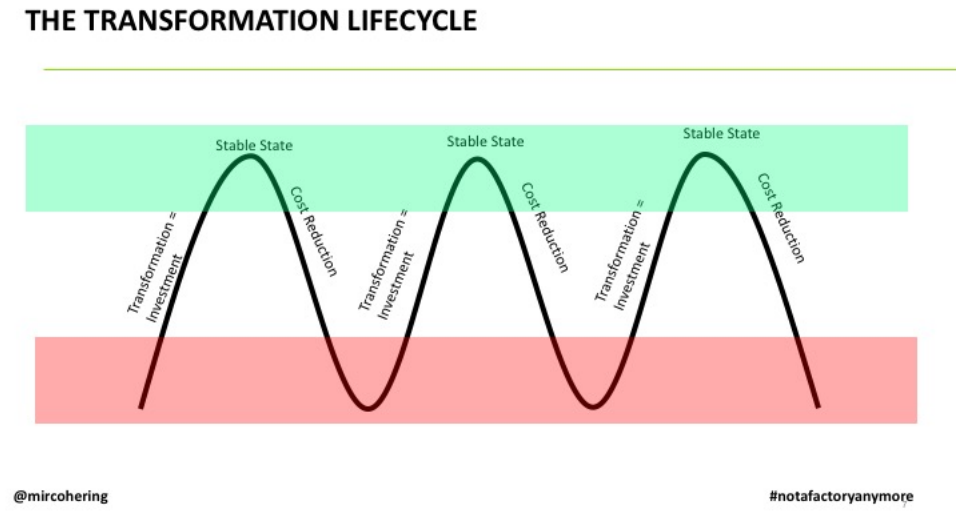



**Figure 2**

Before we get into that, let's look at the mental model for all this. So, this [see Figure 2] perhaps looks familiar to you. This is a transformation lifecycle. It basically goes from, "We have a problem, so we're wanting a 2–5 year program to get to the end-state." When we get to the end-state, surprise, surprise everyone says, "We're done here, so let's start cutting

some cost because we don't need these big teams anymore." So, we're going to start to lean it out. We start accruing some of the technical debt because we don't really want to pay it off at this point, and we start to deteriorate.

Until, three years down the track [when] a new CIO comes in, and you say let's do a transformation, let's implement this next big thing. And we are going through the cycle again and again. Now, companies like mine make a lot of money out of this, and it's a pain for you guys, right? So, really what we need to get to is this model, right? (See Figure 3.) And it's quite challenging, because if you go to this, these 2–5 years were okay in the past. They weren't great, but they were okay. If you now have 2–5 years for this cycle, you're going to be out of business. Because then you kind of digital disruptor can be much quicker than that.

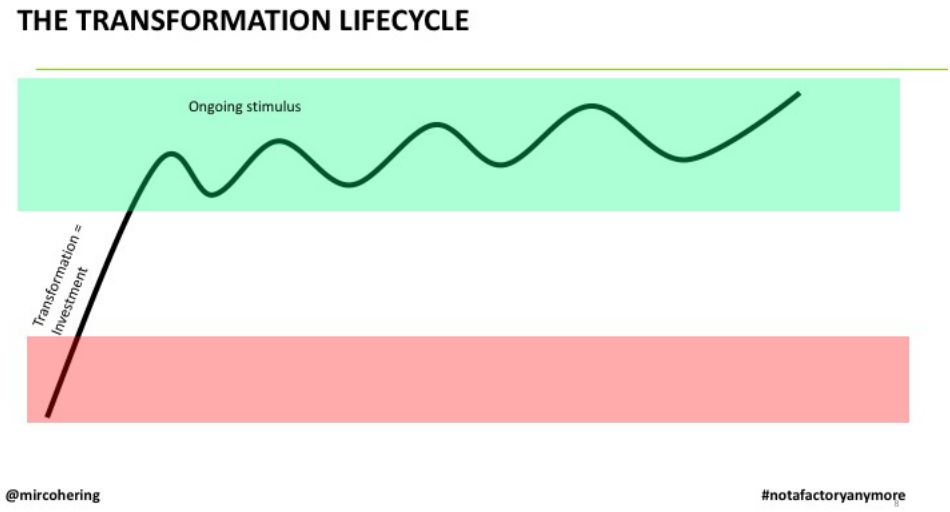



**Figure 3**

So, you won't get a chance to do the next SAP transformation over five years. You have to really get into this model where (a) we appreciate that

there's no end-state. Where if you think there is an end-state, you're dreaming. It's going to continue to evolve, and you have to think about what that does to your architecture, to your tools, and to everything else that you do. And then (b) you need to find a model where you can actually have this continuous ongoing stimulus from somewhere. And that's obviously a lot of my job, by bringing the ideas from across different clients to my clients, so we can share. Or going to this conference and bringing your executives here so you can learn and figure out what's happening and what the next stimulus is that you need to give to your organization.

So, this kind of anti-transformation transformation is really the transformation you should do, as weird as that sounds. So, let's explore a little bit mental models.

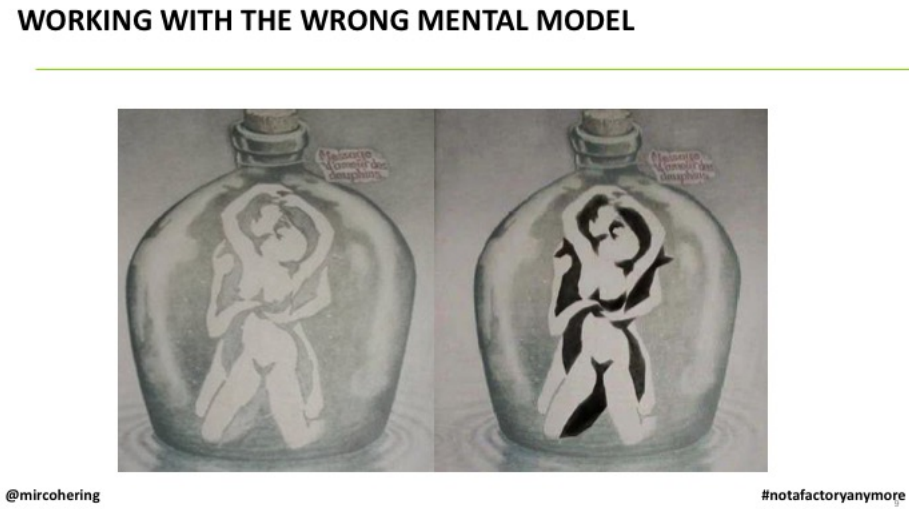



**Figure 4**

I will ask you what you see in this picture (Figure 4), and you don't have to speak out loud, but I know what you guys are seeing. All of you are

seeing nine dolphins, correct? You won't believe it, but if you show this to elementary school children they do see nine dolphins first. And that's good that they have a very different mental model to you. I don't know what it says about you guys that you saw something different. But I've heard that it shows something different. That is really the risk of mental models. Why? Because you're seeing something and you're doing an interpretation of it, and that's going to inform your worldview and the actions that you take. Not in this case, please don't, but in general.

Here, I really like this quote, (Figure 5) because it really describes where we are at, what we have, an amazing power at our hands with the technologies and tools that we have and the automation. We are still working with management processes that came in the mid-20's and based on management principles from the 1900's. And that creates a lot of friction.

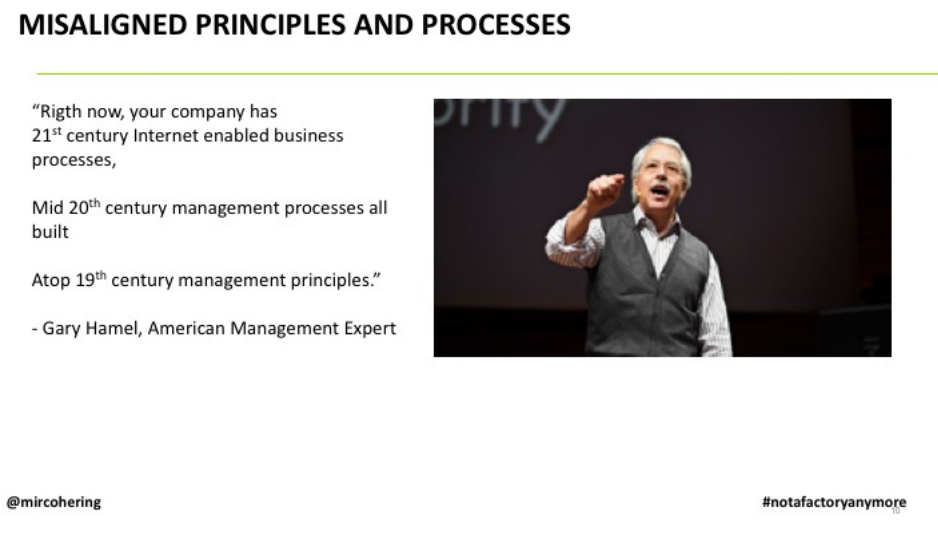



**Figure 5**

So let's look at this in a little bit more detail. I want to start because I can sometimes get, well, hate mail. Why do I talk about not a factory anymore? Don't you know about Toyota and all the good stuff they're doing? I do know about them, and there's a lot of good stuff that we've taken from them. What I really struggle with is that we are treating it as if it's a sweat shop, you know? Like a 1900 Charlie Chaplin *Modern Times*-type factory, and that's the thinking that I don't like.

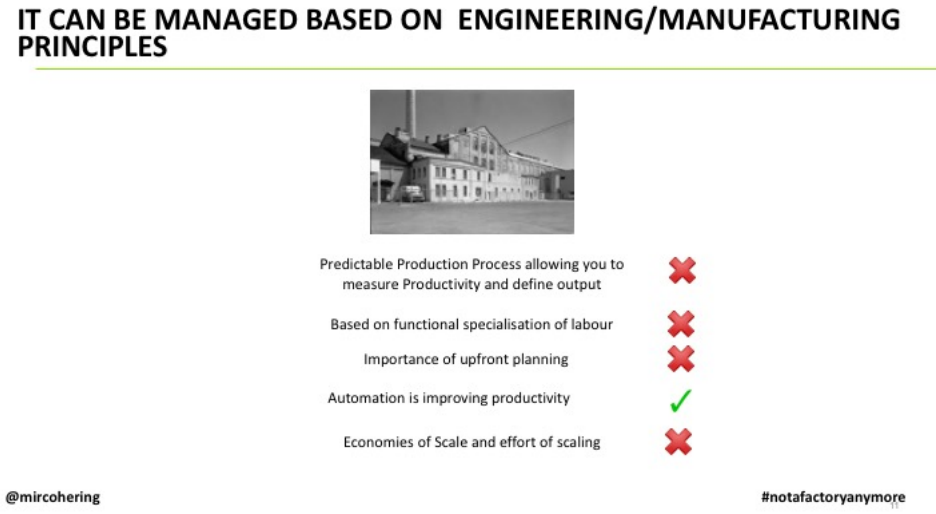



**Figure 6**

Let's have a look at what that meant. (See Figure 6.) The first thing is that in manufacturing you have a predictable process. What that means is, when we change the process we can, to some level, guarantee that we get a good outcome, yeah? And we can measure the outcome. Who here has tried to measure productivity in IT and has succeeded? Exactly. You can't. We are not doing the same thing twice most of the time. At least, all of the projects are new. There's some services that can do that, but in general, you just can't. And if you are fixing the processes, and we had, I'm not sure

if you saw Damon's presentation, the idea that we could put more processes in place to fix a problem is ludicrous. You just add more and more Band-Aids, more and more checks, more and more process steps into it. And it doesn't guarantee you the outcome.

That, obviously, is not actually applicable in IT anymore. As we get the next one, the functional specialization of labor, the idea that we can define something down to the smallest step that someone can execute, like an assembly line. And I was an engineer when I started my career, and I hated nothing more than getting a tech design, which was pseudo-code that I translated into code to hand it over to someone else who then told me I made a mistake. That doesn't feel right. But that's kind of where we got to, and in manufacturing that makes sense where you have the same guy putting the same screw in, well then the saying's true because we're doing the same thing again and again. In IT we don't, alright? So, that doesn't apply either.

Importance of upfront planning, right? If you want to build a new factory, you do have to plan upfront because you have to set up the location, you have to build the factory, you have to bring in machinery, and then you can start building something. Or, if you want to change what you are producing, then you need to change the configuration of it. So, it was necessary to plan up ahead. In IT, a few years back, or certainly when I started, we had the same situation still. If I wanted to run and get a new solution, I had to order some servers, they take three months to be delivered, they get put into a datacenter and it takes another two months to set all the cabling right. We install middleware. You get it, right?

You have to plan up ahead. Nowadays, you go into AWS and you stand up an instance and you have a new company very, very quickly set

up. And that is different. That's why we don't necessarily require much upfront planning. If you set up your company in the cloud, don't do what someone did a couple of years ago where they then deleted their S3 buckets and went out of business. That's also not something that used to happen much. You had to kind of burn down the factory to make that problem.

Automation is improving productivity. That's the one thing that actually applies. So all the good automation ideas and that's where some of the Lean thinking inspires…that's absolutely applicable.

And then economies of scale. I want to expand my production, so I'm going to take one factory and build another factory that looks exactly the same. That works. In IT, I don't know whether you have had that Agile proof of concept, you know, the first guy should be more agile and they're super successful, so how hard can it be to scale? You all know. It actually doesn't work like that. Right? Because the complexities of people and working with each other, the social interactions, the ambiguity that exists, no matter how well you write your designs, there's always ambiguity. And because you're not talking to the same team, the same dynamics, you will get a different outcome. I don't know whether any organization has spent, like, two times one million dollars to see what comes out of two different teams, but I'm sure it will be different. And if you have two times one million dollars, let me know. I'd be happy to run the experiment.

So, should we burn down the factory? That's kind of where my blog comes from. And in the book that I talked about [*DevOps for the Modern Enterprise*], there's a whole appendix that goes deeper into this. But, to me really, that mental model needs to stop. Because as long as we have this mental model, it pushes us into adding more process, relying on the wrong

principles to achieve the outcome. And any transformation you will do, and as you have heard in the beginning, an Agile transformation or a DevOps transformation is not something you can do, you have to be on this ongoing journey. But you have to stop breaking those principles.

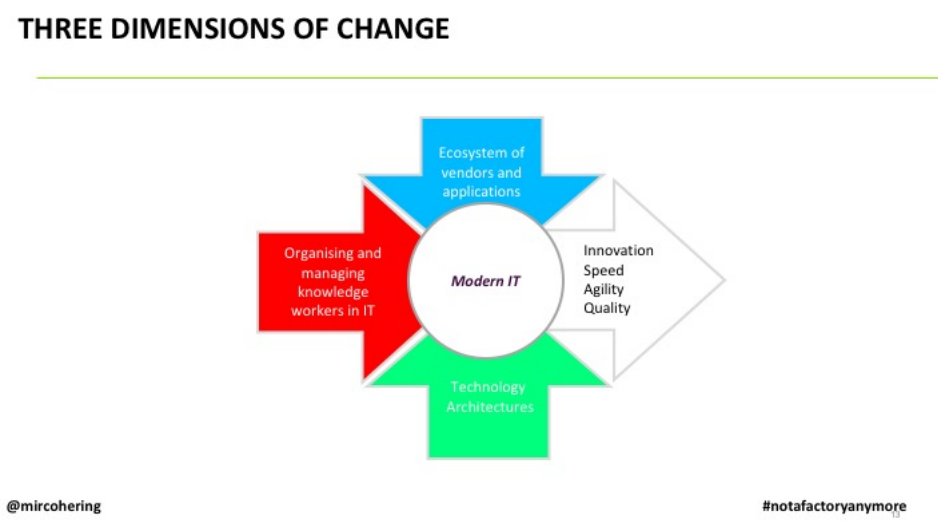



**Figure 7**

Alright, so here's the model that I use (see Figure 7), and it's really three dimensions that you need to look at. So you need to look at, of course, the technologies and the technology architecture that you're using: the tools, the principles, the practices. That makes sense. I think most people have understood now that you need to do something on the people side, the culture side, and that's really, really important as well. But then you need to look at your ecosystem, because you're not alone. One of the biggest frustrations that I have and that might be idiosyncratic, no one ever talks about the culture shift or the culture challenge that you have because you're working with other organizations. People tend to talk about Agile, that it's really important that we have the right culture, and we're on

this cultural transformation, and then they talk about what they're doing in-house and that's great. But how do you make sure that your vendors, your SI's, are actually adopting to the same culture or working in the same way with you? Because that's a real challenge. I don't know many organizations out there that don't have any partners. They will be very few. I can certainly tell you that we do business with lots of organizations, and that means those ones definitely have someone in there.

I had this moment a few years ago when I started doing CIO workshops to talk about these transformational ideas. At the end of the talk, the CIO would came over and say, "Mirco, that was a brilliant talk. Thank you so much. I need to introduce you to someone, because they really need to hear that." And they said, "Let me introduce you to your colleague over there." I didn't really have a good answer to that, because it's kind of embarrassing at that moment, like, why are we not doing all of this for our clients? Until I realized that it's actually not set up that way. The contracts that are in place, the incentives that are in place, prevents people from doing something. And that's true not just for us, that's for every other vendor as well. They're doing what you incentivize them to do. So, we need to change that ecosystem and that culture if we really want to make a change. Or, you need to in-house everything, which is not terribly practical.

These are the three dimensions, I will double-click into them a little bit just so you can get a flavor of why the factory mentality and mental model is not helping us.

The first thing I do is actually talk about production processes. (See Figure 8) Say if you've done an MBA or something, and you look into production theory, there is a formula that tells you about batch files and

their holding cost, transaction cost, and then there's the three batch sizes. That makes sense, right? Now, holding costs are something we can't do a lot with in IT. That's basically you have all this functionality that you have to deploy to production, that's all those values that you don't get, it's the cost of delay, that's ultimately your holding cost.

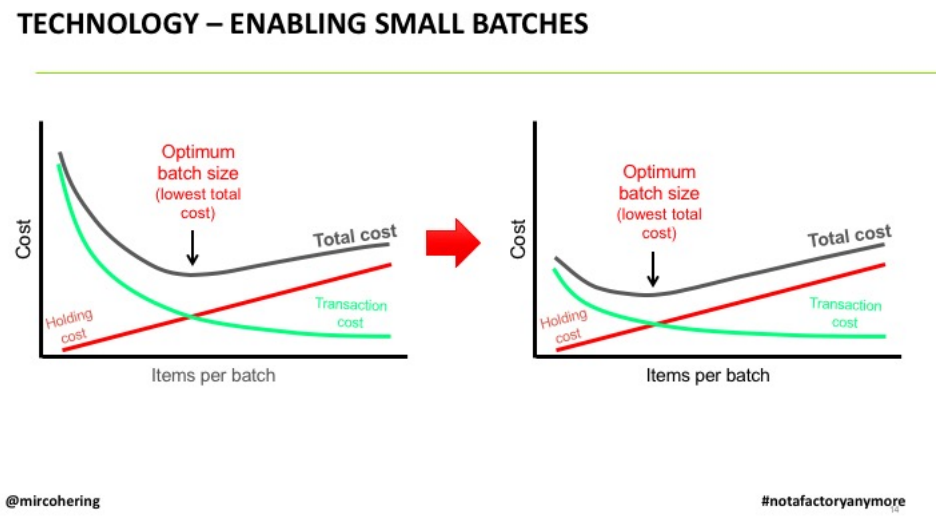



**Figure 8**

The transaction cost is everything we do to release something. So, our governance processes, our deployment processes, our environment standards, our development, our testing. Now, we all know we want to have smaller batch sizes, and we know that new technologies, the cloud and all the automation, enables us to reduce the transaction cost. There's a corollary here which is if you don't change anything, you can't have smaller batch sizes. So, if you're going on an Agile transformation, a pure Agile transformation, and you're not doing anything else and you have pretty inefficient processes before releasing, you have manual testing, it's going to be pretty painful, it's going to hurt you, and you're going to start

increasing your batch sizes again. I've seen a few organizations who have gone down that path.

You need to change those things. You need to look at your governance process that was great when you had nine-month releases. They need to be different when you're doing monthly releases. And you need to radically change that, rather than saying, "Okay, how can I take a day out of it?" And you really need to rethink your overall processes. Think about everything that goes into the transaction cost so you can reduce that over time.


**Figure 9**

The second thing is governance. Now, again, I'm not going to ask for hands but just think about it, who here is running status reports based on PowerPoint or Excel sheets? So, what does it mean? If you have a three-layer organization—you probably have more layers than that—that means someone is asking a developer on what the status is for that team lead, and then that person will provide that status to someone else who condenses it

into an executive status that then, at some stage, gets delivered to the governance board. And if you have more layers you have more than that.

So that means any information you look at is at least three days old, and everyone has changed it a little bit more green along the way. So you have this green tinge on it that makes it look good. That is nonsense in our world. We've been talking about big data in analytics for so long, and you're creating so much data in your process. All those tickets, all the work items, the user stories, the Jenkins lines, the whatever. So much data, and it's nicely isolated in those tools so no one looks at it. The one thing that we've done in my team is we've created a dashboard in Splunk, but any technology will do, where we take all the data and aggregate it and have real-time live data about our systems and our projects.

One thing that you see there on the left (see Figure 9), which is a little too small so let's blow that up a bit, if you're working in an organization that does a lot of Agile, this is my Agile program view. Yes, all Agile teams have different amounts of iterations or sprints, and they're different sizes, and they apply different value, but if you normalize the time and the scope, you can plot them on this line and then you can see when projects are falling behind. If you're through 60% of your sprint and you've delivered 10% of the scope that is working—working, tested in an environment, not just code—then you know you have a problem. And you can quite easily identify the project here that you need to look into. And the size of the bubble means the overall amount of money we're investing in that program.

Now, that is live data you can then double-click and you get into version one or whatever, and again, there's all these different tools out there. As an executive, it's pretty hard to look at them because you have to

look at five different ones. So here we've aggregated that all into one view (see Figure 9). And you should absolutely be able to do this. I mean, you're doing it for the business, why don't we do it for our team. And again, it's that mindset shift and the speed that we need to look at these things that is making a difference.

We have another pretty cool one, and you can't really see that well, but the bottom, the blueish lines, they actually show you a release score, which there is a number of factors that go into that from 0-100 based on late changes, known defects, how many builds have failed, and then what the impact on production was when we went live (see Figure 9). So, you can actually now look at it and say, "Okay, if we had a huge impact in production, what happened before and what was our release score," and you can tweak that. And there's no magic bullet. I can't give you the formula that will work for your organization, but you're meant to be learning. You're meant to be looking at this and tweaking it.

Then we look at the people dimension (see Figure 10), and this is probably a pretty standard view of how you go through this. And the idea here is really that you need to provide the right context to your people. As I said before, as a developer, when all I see is the technical design, all I can do is translate that. I don't know what I'm doing. I'm just implementing it. One of the phrases that I hate with a passion is "build as designed." That's an out for people to say, "Okay, we did what we were asked, we didn't have to think, we're done here." Contractually, that's sometimes good for us, I will admit that, but that's not what we're here for. So, we need to provide the context. And that means you need to fly people in or you need to bring people together, up front, to talk about the business challenges you're trying to solve in IT.


**Figure 10**

Don't give them the design or give them a long document to talk about. That doesn't create context. Context is talking to each other, planning things together, and breaking the scope up into what you need to do. It's really important that you get away from project managers who assign tasks and chase people around. That's not your job. Your job is to manage what happens across the different teams, the dependencies, the outside.

Let's look at the ecosystem (see Figure 11). This is probably the thing that I don't see enough of in the blogosphere, in the community. The very first one is evaluating software vendors. I've done many projects where we look at software products and which tools we should use. And we're pretty good at looking the functionality but we tend to undervalue the engineering.

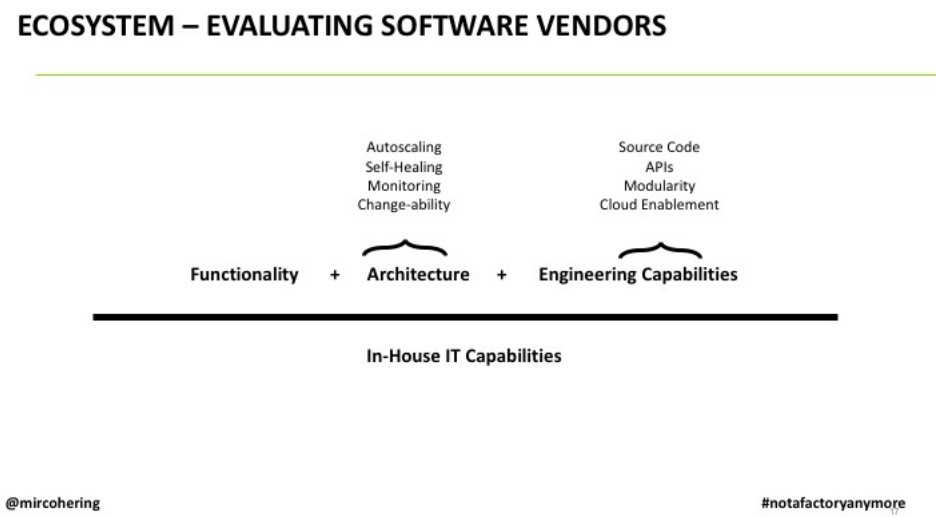



**Figure 11**

You have functionality as one function, then you have the architecture— "This is a product that I'm using," the SAP, the Oracle, the SalesForce, the Pega, whatever you use—what are their capabilities for auto-scaling, for all the DevOps practices that you are talking about? Because, they don't necessarily come out of the box and you need to ask these questions. And then you need to understand, how important is that for me? How would I deal with the failover for those.

And then, what are the engineering capabilities? Can you actually get the source code, can you enable path load development, can you do branching and merging on those things? Do they have APIs that allow you to drive the automation? I had a tool earlier in my career where I needed to use a UI screen-scraper to press buttons to do deployments because there was no API for this to be enabled. That's nonsense. We need to look at those things. We need to look at the modularity and cloud enablement to make sure that you can actually break these things up. I said before

there's no end-state. That means, each of these products will be replaced. So, you need to know how you would do that. Because in three, five, ten years' time, you will do it. And you don't necessarily want to do that at a big bang.

And then you look at this from another lens and you say, what actual capabilities do I have? Because if you don't have a lot of IT capabilities, perhaps the other things don't matter as much. But if you have strong engineering capabilities, then you want these things to be done. I think, as a community, as an industry, we haven't done enough in this space because we haven't asked these questions. If I'm a vendor and I can build more functionality, which you're asking for, or I can build better engineering, which you're never asking for, what do I do? I build more functionality, other than believing in the good of the world and doing the right thing. I think, as an industry, we need to ask for those things a lot more than we do. My frustration is that some of the SaaS products, for example, are actually pretty bad in this. You'd be surprised. I'm not going to name names, I'm on video.

The next one, and this is my favorite exercise that I run, so let's see whether this works here and whether the animation is correct. Evaluating people is something that everyone cares about. You want to know whether you're getting good money, good value for your money. The traditional way of doing this is looking at a vendor and we look at what their day rate is, how much does an engineering day cost me? There's Vendor A = $100, Vendor B = $80, so I'm going to go with Vendor B in absence of any other criteria to look at.


**Figure 12**

Okay, now, this is how work looks like in every project of some size (see Figure 12) There's more junior people to more experienced people, and the junior people are cheaper, and the more experienced people are more expensive. Now, the more junior guys are probably going to do things that are a bit more repetitive, things that need less creativity. They might be doing the password resets, they might be doing the deployments that are scripted so they can run through certain processes.

When you automate that, you automate at the bottom (see Figure 13). Now, what does this do? It actually drives up the average day rate. Now, I had this conversation with a CIO and he said, "Yep, Mirco, I just had one of the vendors that works for me come back to me with a 30% ADR reduction"—average daily rate is ADR, if you don't know that—"and now, after this, I'm worried. Because clearly they are not automating things, they're looking for more junior guys that they can deal with. And I want to do exactly the opposite. I want to have higher skilled people that do more

automation, that drive the overall cost and the total cost of ownership down." And this is really one of those counter-intuitive things where the whole industry is looking at it the wrong way. And we haven't really figured out better ways of doing this.

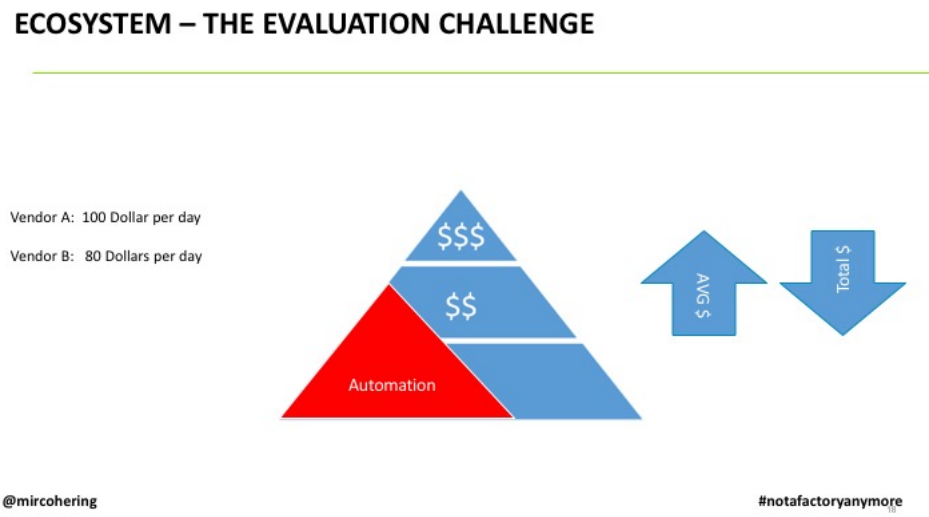



**Figure 13**

Now, the next one, you're evaluating your own team, your operations team. One of the things that always comes up, again, I'm working on SLAs, how will you know that I'm doing a good job? SLAs are something that I find inherently dangerous. Here is our two SLAs. A password reset, a small task, shouldn't take longer than two hours, and production Sev 2 incident, which is annoying but not super critical, will take me a day. If you have a password reset and a Sev 2 coming into your team, what will you do when you have these SLAs? You're going to do the password reset because that's on the clock. For the other thing you have the full day. And that creates a dynamic where, this is just two and normally there's a whole set of these

SLAs, and they create a dynamic where you're actually not getting the outcome that you're after.

What you're really after is that over time each of these things take less time and are more efficiently solved. And SLAs are really kind of putting a number in there. And good organizations are not taking the end-state number, they're looking at learning and getting better.

Here's another example. First time resolution rate and resolution time. You absolutely want that to be right. We want to have a high resolution rate and it should be really quick when it hits your service desk. You heard Damon talk about how there's a lot that we can automate in operations. And you get this kind of tree here (see Figure 14) where there's actually very few things and only exceptions that take your team.

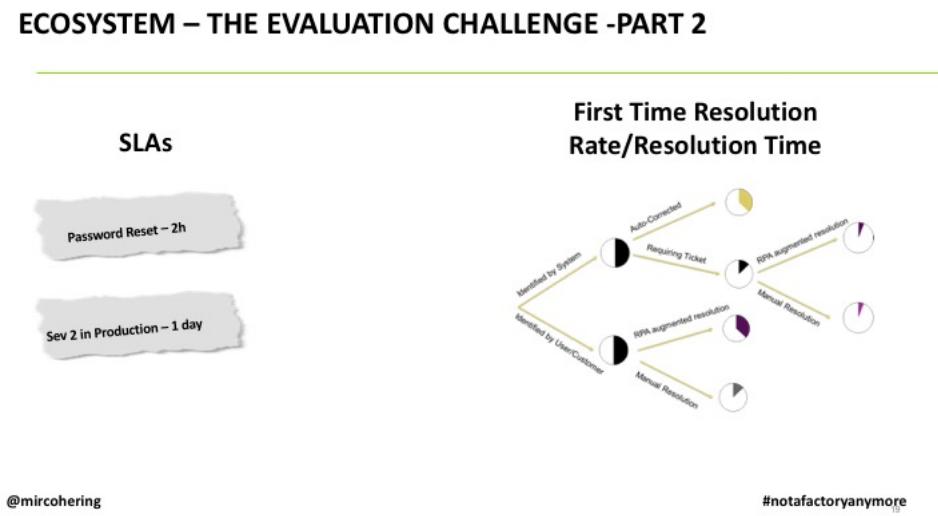



**Figure 14**

Now again, what does that do? If all the simple stuff is automated, I get the really complex stuff. Now, that drives up the time it will take to solve and it will drive down the first rate resolution unless I start measuring all

the things that have been automated, which you probably won't create tickets for. So again, you have this completely counterintuitive dynamic that plays out in organizations that we have created for ourselves. It basically makes it really hard to happen in these situations, where if I do the right thing I will start hurting these metrics, and I need to go back to my client and I need to say, "Hey, we need to change this. The contract you put in place, it doesn't, basically it doesn't incentivize us to do these things. We really need to think about this differently."

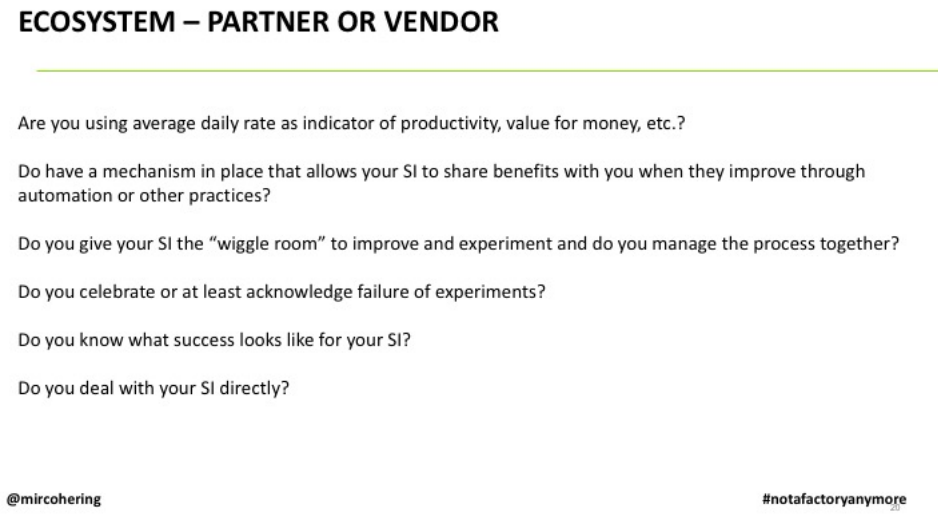



**Figure 15**

Then, and you will have these slides, but then I think that the other bit where everyone talks about we're now partners, we're not vendors, but really we still treat them as vendors. Because when we talk about failing and learning, we really don't want that. We pay money for you, you can't fail. Now, if I can't fail, that means I can't innovate. I can't experiment with something. How do you solve these problems? And this is my self-

assessment for clients to say, are we really working together to actually become a better organization as a partnership?

I can't give you the silver bullet. I can sell you snake oil in all flavors. But, at the end of the day, unfortunately, it looks like this (see lower line in Figure 16). We're going through this process where we're going to get better, and then you get through this peak where you believe you've done it, and then from there on you're going to fall a bit back, but hopefully not as far as it says, as we showed on the transformation slide. And really, actually, it doesn't look like this. Even that is not correct. It really looks like this (see upper line in Figure 16). We're going to go through this again and again and again and again. And we need to be resilient to the drawbacks and the setbacks that we have. And we can't just declare success too early or punish people when we're moving backwards. You will have deployment failures. You will have environment outages. It's a matter of learning from it and getting better.


**Figure 16**

So I am, ugh, twenty seconds left, so there are still some problems. I haven't solved these. I'm a lot better at telling you all the things that don't work. These are the things that I really think we should think about (see Figure 17). The whole partnership model. How do we find out if the relationship is working for you? It's a bit like in a marriage, how do you know that it's working for me and for you? Anyway, I leave that as a thought. And then, last but not least, how do we get into the anti-transformation transformation?

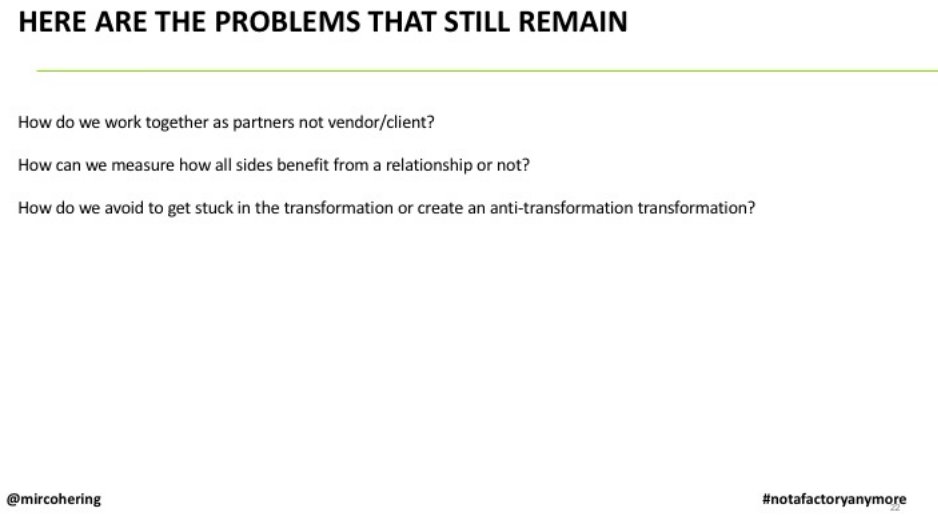



**Figure 17**

If I could bundle this up and sell it as snake oil, I'll be on my island, enjoying time off. Thank you guys. I hope that was something that you enjoyed.

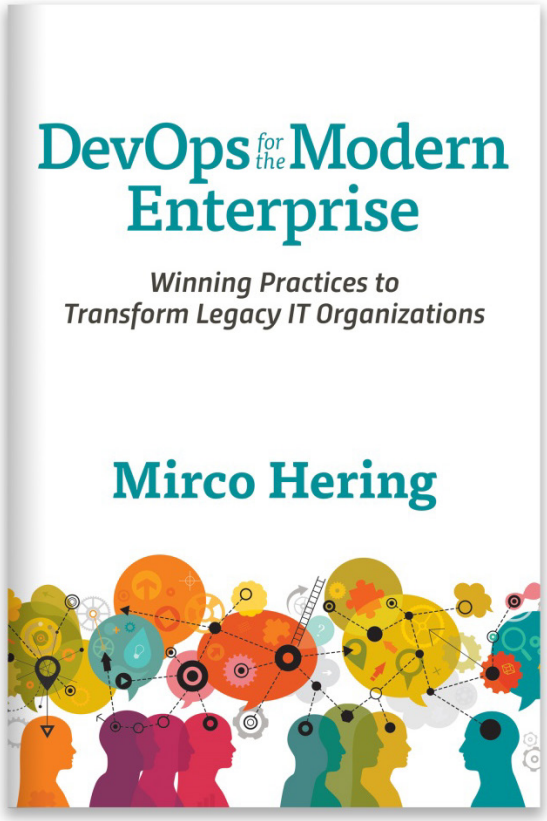
# About the Author

For over ten years Mirco Hering has worked on accelerating software delivery through innovative approaches (what is now called DevOps) and six years ago started experimenting with Agile methods. He supports major public and private sector companies in Australia and overseas in their search for efficient IT delivery. Mirco also blogs about IT delivery at NotAFactoryAnymore.com and speaks globally at conferences about Agile, DevOps, and organizational psychology.

Follow Mirco at Twitter @MircoHering.

# DevOps for the Modern Enterprise: Winning Practices to Transform Legacy IT

By Mirco Hering




Now available in paper, eBook, and audio formats from all major retailers: https://itrevolution.com/book/devops_modern_enterprise/.