Mark Schwartz

What Does it Mean to Lead IT

DevOps Enterprise Summit London 2017

What Does it Mean to Lead IT

DevOps Enterprise Summit London 2017

Mark Schwartz

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

The contents of this eBook are a transcript of the complete presentation given by Mark Schwartz, "What Does it Mean to Lead IT" at the DevOps Enterprise Summit London 2017. To view the original presentation, please visit https://www.youtube.com/watch?v=JXtt7Oo8JRA.

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

What Does it Mean to Lead IT

What does it mean to lead IT? This is kind of an important question for me, because I'm a CIO and I should probably know these things. It bears thinking, I think, now that we're agile and DevOps and so on. So, I want to talk a little bit about that. But first I want to tell you a little story.

So, when I finished high school, before I went to university, I managed to get a summer job as a programmer. It was actually at my dad's company, so it was easy. Easy to find that one. I was a COBOL programmer. I worked with a bunch of COBOL programmers on a payroll system for the company. And at the time, the company had just gotten its first computer ever. It was a Data General Eclipse and it had its own room, a computer room, which was air conditioned to the point of freezing. And it had these big, wonderful tape drives that were a little bit bigger than me, you know, sort of reel-to-reel things. I was very proud of myself when I learned to thread the tape through. And it had a glass door, you know, so people could see into the computer room, and what they would see is all these guys in white lab coats kind of shuffling around and threading tapes into the machines.

The programmers, there were three of us. And one of them was a guy named Igor. And Igor, he clearly…he really was a sweet person. He loved people, but he didn't have social skills, you'd have to say. Right? He also had this thing where he walked around with sort of a permanent frown on his face. You know, that…And you know, he was a brilliant assembly

language guru. When anything was broken, he would figure out a way to fix it somehow. And so, he would sort of stare off into space and come up with something brilliant and code it quickly.

Imagine now that you're an employee at this company, and you walk past the IT department on your way to the lunch room, and you look in the doors and you see these strange people doing their strange things. And if you're an executive in the company or a manager, you know…IT, those are the guys behind the door there who never seem to do anything on time and whose systems are breaking all the time. And when something went wrong, you would screw up your courage and go try to find Igor. And you would find the desk that he was sleeping under and wake him up. "Igor, Igor. The payroll system isn't working."

And he'd kind of go, "Interesting."

"Well, Igor, it's not interesting, actually. The payroll system's not working. Nobody's getting paid."

"Mmm."

Sort of like that. So, the manager…the job of the IT leader at the time, guy named Frank, was basically to get Igor to focus and produce code. I would say that would be the best summary of what he was there to do.

You probably think you know where this story is going. Alright, we don't have big room-sized computers anymore. We don't use tape drives. Only banks use COBOL these days. So, I'm probably going to tell you about how everything has changed since then. But if that's what you're thinking…actually my message is the opposite. I want to tell you how things are actually the same as they were in those days. And I think the way to see that, or the way to see a similarity, is to think about the

relationship that was born at that time between the IT organization and the rest of the company.

Clearly, if you were an employee of the company, a manager, IT was something really strange. And there were strange people, and they were doing strange things. And if you asked Igor what he was doing, he would go into this long-winded explanation of technical stuff that didn't make any sense. And altogether, the feeling that you got is: IT is something other. It's something different. It plays by its own rules. Anybody else, if they screw things up, they would get fired. But IT, there're always bugs and the payroll system crashes and somehow nobody…you know, they don't get fired. What is this?

I think a set of stereotypes was born. A stereotype of an IT organization that doesn't really care about the business…doesn't really know anything about the business. They're turned on by these crazy technical things. And it wasn't just a one-sided thing. On the flip side, the IT people would say, "Oh, the business, they're clueless. They don't know anything. We have to idiot proof our software because you know what's going to happen if we don't idiot proof it." They would say, "Oh, the business, they just want to…you know, scope creep every time we try to do anything." And so you had this sort of hardening of an us-versus-them atmosphere when it came to the relationship between IT and the business.

Another little story. I used to travel around the world a lot with a backpack, kind of bumming around the world. I did that for a number of years. And one thing that would happen often is, I would hook up with other travelers and we would decide, we've got to go to the local…We mostly traveled in developing countries because you can't really go for years and years without, you know, living on five dollars a day or

something like that. So, you're in some developing country and you've got to go get souvenirs, of course. So, you go to the craft market and you find something you like and the vendor immediately sizes you up and names what seems like a very high price. And course you have no idea what the price should be, right? So what do you do? They say fifty cents. And you know that's a really high price, so you cut in half. You say twenty-five cents. And then you haggle a little bit and you wind up with a price. And, of course, it's absurdly cheap anyway, but you walk away feeling like you've been ripped off because you're a tourist. Right? Whatever the price is. You really have no basis for judging.

So here's me as CIO going into the CEO's office. Take the CEO of the company that I worked for before the government. And he would outline this big project for me and say, "How long is it going to take?" And I would say, "About a year," or something. Well, he has no idea whether it should take a year, right? That's not his expertise. So, he says, "Too long. Six months." And we haggle a little bit and we agree on some number, seven months or something.

When you think about it, it's kind of absurd on a couple of fronts. One is that you're haggling over an estimate. You know, it's an estimate. How can that be negotiable, right? This is my best estimate. It's going to be about a year. But still, you would haggle, and then of course you were on the hook for delivering in seven months. Somehow it was a little bait and switch game where now your IT organization is required to deliver something in seven months when you originally estimated twelve months.

But what's really a little bit bizarre about it when you think about it is I said [the CEO] had no way of knowing how long it should take. Now, there's a fundamental assumption baked into that. When you think about

it, [the CEO] did have a way of knowing how long it would take. He could have relied on the experts in his company who were best in a position to judge how long it would take: me. Right? My whole reason for being there is that I'm an expert in these things. And so to judge how long it should take, you should call upon the person in your organization who best knows, and I'm that person, probably.

What I'm getting at is the attitude was really that I'm not part of the company, right? I'm an external organization, an external IT organization, and I am naming a price, and that's something that has to be haggled over, and haggled over in ignorance as opposed to just saying, "Well, you're the person who would know this best."

I think the dynamic that really developed is one where the IT organization is separate from the rest of the business. We've all observed this, right? We talk about IT and the business, whatever that means. IT, obviously, is part of the business, but you know, IT and the business. IT essentially is in the role of a contractor to the rest of the business. And the work of IT is transactional. You have the business, and the business hands over a set of requirements, and IT goes and does its thing, and then delivers the result to the business. That's a transaction. The unit of the transaction is a project, but it's a way that you deal with an external organization, essentially.

So, we would talk about requirements. The business gave IT its requirements. Think about that for a second. Do you go to somebody in your company and say, "This is what I require of you"? It's a little bit weird, right? But nevertheless, we would go to IT and say, "This is what we require." And then we would ask IT to deliver. Who else delivers? And then we would hold IT accountable for delivering on that schedule. These

are all words that we don't use with anybody else in the company. Do we say to finance, "The business is your customer, finance"? Or do we say, "Marketing, this is what we require of you"? Or you know, "Marketing, please provide customer service with a smile to the rest of the enterprise," or whatever. The IT organization has always been in this unique position where it was treated as an outsider to the rest of the organization, and so, of course, it made a lot of sense when we started to outsource IT. We were already outsourcing IT to an internal part of the organization.

I contend that a lot of the difficulties that we're having in pulling off a DevOps transformation have to do with this model, this contractor model of IT, where the underlying principle behind it is: we have these strange people, we have Igor in IT, and he is liable to go off and do anything to waste the company's time and money unless we figure out some way to control him. So, we're going to control IT by creating this project-oriented structure. And the waterfall is a great way to do this, right? IT commits to a Gantt chart. And then we can, even though they can never deliver on the Gantt chart, we can at least hold them accountable and tell them they did something wrong when they don't deliver. This is born, I think, of this feeling that IT people are just this weird, extra thing in the company that's kind of separate from everybody else in the business.

So, when we try to transform and move toward a more Agile environment, we're essentially coming back to the company and saying, "By the way, we're not going to have a Gantt chart, and the requirements might change. So, we don't know what we're going to give you." Well, the problem there is that it…under the surface at least, the conversation is really, "Well then, how can we hold you accountable? How can we control what you're doing? You're liable to go off and do just about anything. How

can we make sure that you're going to deliver on time, on schedule," whatever other paradigm of control you want to name. But the good news, of course, is that we are now…

Oh, and let me come back to the role of the IT leader in that case. Here I am, the CIO. I am told, essentially, and this…if you look at the literature on being a CIO and succeeding as a CIO, it says something along the lines of, first, you have to demonstrate that you can deliver value to the company by delivering projects on time and on budget. Second, you have to show that IT treats the rest of the business like a customer, provides good customer service. And once you do that, then you might be able to get a seat at the table and participate in strategic exercises. Right? The agenda here, I think, is pretty transparent. My job is to control these crazy geeks and make sure that they actually deliver. And I think this is deeply, deeply embedded in the way that we have always thought about IT within a business organization.

So, everything's good. We're now Agile and we're DevOps or whatever. And fortunately, all these things have changed. So now, for example, we have autonomous empowered teams. Yay. Okay. And we have a product owner and subject matter experts from the business, and they work together with the developers on the team. Yay. And the product owner is an expert on the business, so the product owner can figure out what the business needs and decides…makes good value decisions. Yay. And by the way, the product owner decides what is important to the business and then tells the developers what it is, and then the developers develop. Yay. And the product owner then takes what they develop and uses it to add value to the business while the developers have produced some software product. Yay? And by the way, we asked the developers to

commit to how much work they're going to accomplish in the sprint, and then we check to see if they're meeting their commitment at the end of each sprint. Yay? And we measure their velocity to make sure that they're being productive.

See, we sort of took that big picture metaphor for what IT is and we shrank it down to team size, essentially. We still have the business deciding on the requirements, now in the form of a product owner, and giving them to the technical people who then are charged with estimating how much they're going to get done in a sprint and then delivering it. What's the difference? I guess we've gone from big to little with a very similar control model. Or at least, this is the danger of mis-applying the ideas of Scrum. I'm not really gonna say that's what Scrum intended. But I think that the core of the problem here is that we're thinking of the product owner as representing the business, and the business decides what they require and tells IT, "We require this," and then waits until those things are delivered. And then goes and runs with it and delivers value somehow.

I think mostly what we've done is out of necessity because of this underlying need to establish control. We've shrunk the model down but we still have a control model. And really, if you think about the message of DevOps, inclusivity, single team working together…If you think about the message of Lean or Lean startup, I like to think in those terms, the business doesn't know what their requirements are. What they have is hypotheses about what's going to achieve the outcomes. And here you have a team that, in theory, is empowered to test those hypotheses and figure out how to create value for the organization. But in many cases, we're not letting them do that because we have our business experts. Not only does the product owner decide what's valuable, what's required by the

business, and toss it over to the team, the product owner also reviews what's produced in the end and says it's acceptable or not acceptable. Right? Once again, we're reproducing a model here.

If we're really going to take advantage of the power that is offered to us by Lean approaches, Agile approaches, DevOps, then what we have to do is find a way to get outside of this control paradigm or control way of thinking of things and regard IT as a part of the business working together with the business stakeholders to formulate hypotheses, formulate objectives, try to accomplish the objectives, and check, you know, iterate, see how it's working, and constantly improve.

So, I decided to write a book about it. That's kind of the way the story always ends these days for me. You know, I get interested in a topic and it's like, well, now I've got to write a book about it, dammit. So, you have it in your packets today, and I thought I would just sort of explain where this odd thing is coming from.

It's called *A Seat at the Table: IT Leadership in the Age of Agility*. And what I want to argue here is that we can't think of Agile DevOps as a way to run projects. You know, you still have this superstructure of IT leadership whose job it is to control the geeks and deliver predictability for the company. We can't not bring those ideas of Agility and DevOps to the leadership level as well and see how that means we have to change the way that we lead in IT. And I think what it comes down to is questioning just about every assumption. I mean, it's so deeply baked in the way that we think about IT, the way that we do governance.

For example, I said that the unit of transaction between IT and the business is a project, right? We give a command and then a project comes back. And we govern in projects, essentially. We decide go or no go for

projects. Well, why is that the right unit of granularity? It's the unit of granularity because in the waterfall model, where we were primarily concerned with control, we transacted in projects. But if we question that, if we say this is not about control anymore, maybe the granularity at which we make investment decisions should be a lot smaller. Maybe it should be at the feature level. Maybe we're talking about single-piece flow and decisions being made. Maybe it should be at a bigger level. Maybe it should be at the level of budget and long-term investment planning. You know, let's commit a certain amount of money to a general objective and then not bother the people who are delivering it right away. You know, this is the objective that we're funding. There are all sorts of ways that we can go with it, but we're stuck with this legacy of governing in terms of projects. Transact in project. Oversee things by dates and schedules and milestones and so on.

So, what I try to do in the book, and what I think we should all do, what we have to do if we're going to be IT leaders, is to question every one of those assumptions systematically and say, now that we're Agile, does this still make sense anymore? Are we still sticking to an old paradigm when we're thinking through how to oversee an Agile organization?

And I think in many cases we are. And I think questioning some of these assumptions when you're in an environment where power and control are such a core part of everything that we do, that it really takes a lot of courage to question those things and force people to rethink the underlying assumptions behind how IT is working with the rest of the enterprise.

So that is my message. IT leadership is about courage and about questioning these very, very deeply held assumptions, given that we now

know better ways to do a lot of the things that we used to do. That the IT practitioners are actually not in danger of going off and playing with the technology and doing all these crazy things. Almost every IT professional that I've met is actually trying to achieve good outcomes for the company. I don't even know where this stereotype comes from. In fact, I know Igor was…The thing he wanted most was to help the company and help everybody else. He just communicated in a strange way that gave rise to this assumption that he wasn't. So, if we are all here to try to add value for the company, if we are all actually concerned with adding business value, which I think we are, then we have to question the system that was set up on the assumption that we weren't actually trying to do these things.

So, I hope you will read and enjoy the book and walk away with a conclusion that there is this tremendous potential if we actually rethink that basic role of IT and how it fits in the organization, and work together with the rest of the organization to deliver value to the customers and the business. Thank you.

About the Author

Mark Schwartz is an iconoclastic CIO and a playful crafter of ideas, an inveterate purveyor of lucubratory prose. He has been an IT leader in organizations small and large, public, private, and nonprofit. As the former CIO of US Citizenship and Immigration Services, he provoked the federal government into adopting Agile and DevOps practices. He is pretty sure that when he was the CIO of Intrax Cultural Exchange he was the first person ever to use business intelligence and supply chain analytics to place au pairs with the right host families. Mark speaks frequently on the role of the CIO, innovation in IT, and Agile and DevOps approaches in challenging and low-trust environments. With a BS in computer science from Yale, a master's in philosophy from Yale, and an MBA from Wharton, Mark is either an expert on the business value of IT or just confused and much poorer.

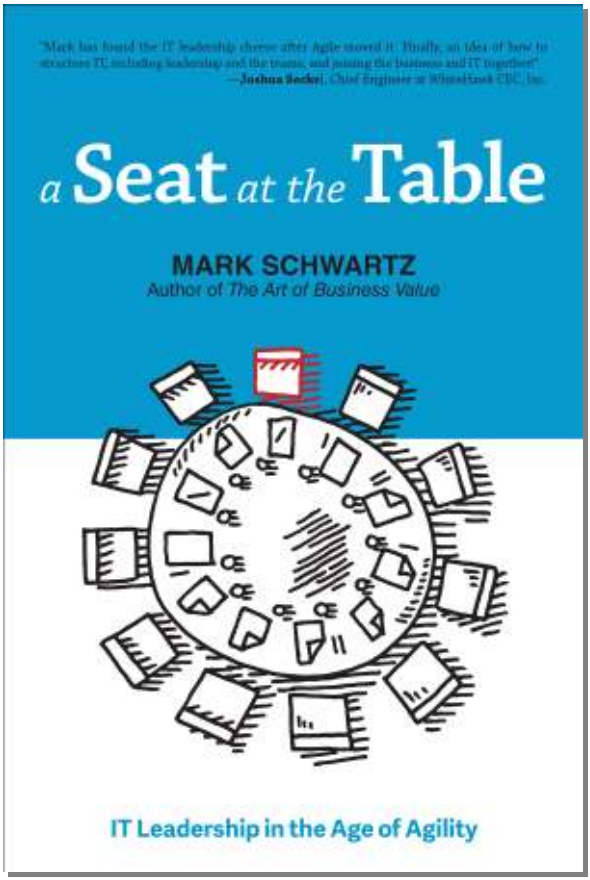
Mark is the author of *The Art of Business Value*, which—he is proud to report—has been labeled by his detractors "The Ecclesiastes of Product Management," and "Apocryphal." The book takes readers on a journey through the meaning of bureaucracy, the nature of cultural change, and the return on investment of an MBA degree, on the way to solving the great mystery…what exactly do we mean by business value and how

should that affect the way we practice IT? Mark promises that his new book, *Seat at the Table*, is more canonical and less apocryphal.

Mark is the winner of a *Computerworld Premier 100* award, an *Amazon Elite 100* award, a *Federal Computer Week Fed 100* award, and a *CIO Magazine CIO 100* award, which strongly suggests that there are less than 99 other people you could better spend time reading.

A Seat at the Table: IT Leadership in the Age of Agility

By Mark Schwartz



Now available in paper, eBook, and audio formats from all major retailers: https://itrevolution.com/book/seat-at-the-table/.