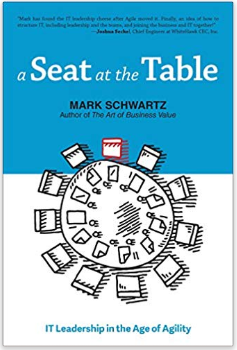
A Seat at the Table – IT Leadership in the Age of Agility – Part 3

By Mark Schwartz

“Courage, I say, is the value most needed by Agile IT leaders.”  
 – **Mark Schwartz**

# Last Time in Part 2

**Enterprise Architecture:** The job of IT leaders is not to execute projects on behalf of the business; it is to steward the asset that is the total of all of the enterprise’s IT capabilities—an asset that has functional capabilities (how it is used today) but also latent capabilities (how it will support future agility and how it will offer options in the future).

**Build Versus Buy:** In a world where IT capabilities were delivered as a single “product” at the end of a project, a “product” that then only required a bit of maintenance now and then, the economics of IT delivery often favored buying a product off the shelf. But when we view IT capabilities as being in a continuous state of transformation, when we see them as tightly integrated into an EA, and when we realize that the costs and risks of custom development have been radically reduced, the economics often now favor custom development.

**Governance and Oversight:** Governance has traditionally been viewed as a filter; a way of allocating scarce IT resources among many competing projects. But in a world where IT is integral to strategy, it makes more sense to begin from strategic objectives and produce investment themes that accomplish those objectives. When combined with Agile and Lean practices, this approach can focus IT planning, reduce risk, eliminate waste, and provide a supportive environment for teams engaged in creating value.

If you missed last month, you can find the handout for Part 2 on the Agile4Defense GitHub page at: <https://git.io/JeaO2>

# Risk

*The presence of uncertainty is the simple reason why Agile approaches work better than plan-driven approaches—it is also the reason why a good IT leader will often have to make “wrong” decisions. An IT leader adds business value by adopting an intelligent attitude toward risk.*

**Risk is the chance of a negative impact resulting from uncertainty.** We can reduce risk—often at a cost—but there is generally no way to eliminate it.

* Almost all of our decisions have potential negative consequences, if only the opportunity costs of not having chosen a path that would have turned out to be more profitable.
* As an IT leader making decisions under conditions of tremendous uncertainty, your choices will often turn out to be wrong.

**Agile and plan-driven models have very different ways of dealing with uncertainty.** Plan driven approaches, even Waterfall, have always acknowledged that the future is uncertain. But this uncertainty was treated as a potential aberrance—a risk or eventuality whose impact required mitigation.

* The basic plan would remain valid, and the risk of the unexpected could be managed within the plan by itemizing potential risks, actively working to prevent them, and, if necessary, building slack into the plan to manage the potential impact of the risk on schedule.
* Such an approach assumes that the risks are bounded and can be itemized, that we can know the risks in advance, and that we have a way to (try to) control them.
* The relationship between uncertainty, risk, and change is far too complicated for such control when delivering IT systems, where complexity is overwhelming and the number of potential failure modes is high.
* In weighing the things known for certain against the uncertainties of the future, we are sure to find that the uncertainties dwarf the certainties, and that the uncertainties are not small potential deviations from the initial plan—they are the very substance of the project.

**The Agile way to deal with uncertainty is to create options and then “buy” information to more accurately assess probabilities.** Traditional teams attempt to drive out uncertainty by planning and analysis. Agile teams tend to drive out uncertainty by developing working software in small increments and then adjusting.

* Waterfall plans are made at the moment of greatest uncertainty—the beginning of the initiative.
* Working in an Agile way, we can leave options open and be prepared to accommodate a range of scenarios as the future unfolds.
* Jez Humble and his co-authors suggest that instead of using detailed planning to manage risk, we instead use experiments.
* When we encounter a risk, we should think of something we can do that will help us gain information to mitigate it. We build something, measure results, and thereby learn enough to cope with the uncertainty. “The purpose of measurement,” they remind us, “is not to gain certainty but to reduce uncertainty.” Uncertainty is still a given, but we want to drive out as much as we can until it stops being cost-effective to do so. We can take this as yet another definition of what it means to be Agile.

**Why do people feel like there is risk in an Agile approach?** They are worried about the schedule for delivering what they think of as FOC. The perceived risk is that the project will be “finished” late.

* This, as we know, is based on the outdated idea that we define the scope of the system ahead of time and keep working until we deliver it. That is precisely what we do not do in an Agile approach.
* Instead, we focus on reducing the risk of deploying functionality later than “as soon as possible.” We deploy small value-adding chunks, continuously.
* In the Waterfall, we were worried about getting FOC delivered soon;
* With an Agile approach, we go this one better—we make sure that we deliver each individual piece of functionality soon.
* If it becomes truly necessary to hit a schedule milestone, then we can adjust our scope to be sure that we do so.
* The importance that we have attached to the timing of FOC is just another example of the misconception that IT delivery is about delivering discrete, finished products. We want our FOC because we think that is the point at which we are finished investing and just need to “maintain.” But unless we want functional and technical debt, a lumpy EA, and a need to do an expensive and risky transformation effort, FOC is in no sense a “final” operating capability.

# Quality

*It is difficult for IT to gain a seat at the table when IT is always failing, but on the other hand, an IT leader who is reacting to statistical noise—failures that he or she has already chosen to accept—is destroying business value. An IT leader must have the necessary technical skills, make impeccable decisions under uncertainty, and then have the courage to face the consequences.*

**In the plan-driven model, quality was easier to understand.**

* We specified what the system should do, and then measured quality as adherence to that specification.
* The opposite of a defect was “working as designed,” even if the design was poor.

**In the Agile world, though, it is a more complicated matter.**

* Clearly, we do not mean adherence to specification, but do we mean spare-no-expense-best-in-class? Probably not that, either.
* We are constantly making quality decisions, especially in a Continuous Delivery model, as we decide whether the quality of each individual feature is adequate for the feature to be deployed.
* It is easy enough to say that the feature is ready for deployment when it passes all of its tests. But is quality a simple yes-or-no attribute? If it passes its tests, is it high quality? And how can we assess the quality of something that is unfinished—that will later be added to, incrementally?
* Perhaps we should separate the concepts of deployability and quality. A feature is simply deployable when it passes its tests; that is a yes-or-no question independent of quality.
* Once the feature is in production, we can assess its quality based on its fit to need, or perhaps its actual success in accomplishing the business outcome for which it was intended.

**Fail often, fail fast—but fail well.** Why in the Agile world do we talk about the need to fail and to fail often? This is a linguistic ambiguity—we are talking about a different kind of failure—in fact, a kind of failure that is the opposite of defects and outages.

* Trying things out is a way of learning in the Agile world; it is a kind of feedback cycle that lets us make good decisions in the normal course of work.
* Let’s say that we are deciding between two different open source products for building a piece of the system and do not know enough of their impacts to make the choice. In the old-school way of making the decision, we would do research, debate the choice, maybe do a proof of concept.
* In our new world, we can simply implement one of the approaches and see how it works for us. We could even try the other approach as well and compare. We are comfortable with the idea that we will break things in testing, because the end result of doing so is that we can make a decision more quickly and with more good information available.
* “Failing” in this sense is simply an efficient process we use to select among alternatives.

# Shadow IT

*Agile ways of working support a community approach to IT, where IT leaders achieve their objectives by mobilizing the skills and passions of a broad community and encourage the members of that community to work together across organizational silos in a way that values skills and contributions.*

**Shadow IT—rogue IT, IT that is out of the control of the IT organization.** It is what has saved IT up to this point. It is a powerful phenomenon that we have not yet learned to take advantage of, caught up as we are in the contractor-control model of IT. Shadow IT is what happens when the IT organization is unable to meet the needs of a part of the company, perhaps due to capacity constraints or to the governance process’s limitations.

**A picture of today’s emerging workforce.** These characteristics of today’s emerging workforce reinforce the changes I have been describing, making them almost inevitable.

* **Respect for skill:** Your new IT workforce will respect co-workers who have impressive technical skills, and will have little respect for people who don’t. Skill is demonstrated through hands-on “shipping” of product. Management for the sake of management is not respected.
* **Get things done:** The hierarchy must be flattened. Layers of management get in the way of goals. The employee wants the shortest possible path to shipping code without needing layers of approval. Management should be close enough to the action that they can demonstrate understanding—witnessing employees’ contributions and removing impediments.
* **Cross-functional and team-based:** It used to be natural for ops specialists to do ops, developers to do development, and testers to do testing. Now, crossover skills are increasingly important. On a delivery team, employees will help each other across roles in the interest of shipping code. If there is a backlog in exploratory testing, people who normally do development will help test. Software engineers will oversee their code in production and help make changes to the infrastructure if necessary to improve performance.
* **Fairness and social responsibility:** The workplace must be fair. Arbitrariness provokes negative reactions. If someone needs to be on-call to solve problems (“wear a pager”), then everyone should share in that responsibility.
* **Focus of roles is changing:** The software engineer role is increasing in importance. Tests and infrastructure are now both represented in code; with SDN, soon even the network will be. Infrastructure can now be tested, like code; it can be placed in version control.
* **Technology matters:** Most IT people are technologists; they need to work with technology, to touch technology, to walk and talk technology—at least some of the time. They are in your IT organization because they enjoy technology; they are proud to be technologists. It is time to shake off the last traces of the attitude that this is something to be ashamed of or that they need to be controlled, learn to speak the language of the business, or start wearing jackets and ties.

# The CIO’s Place at the Table

*IT leadership runs the business along with the others who run the business. The seat at the table is earned by being at the table.*

The role of senior IT leadership has always been framed in terms of control—the underlying theme of interactions between IT leaders and the rest of the business has been to “keep your people under control, provide good customer service, deliver what you say you will deliver, and you will be rewarded with a seat at the table.” This way of thinking has become less and less tenable as we ask IT to step up and play a more central role in the company by driving its digital agenda.

Agile approaches hold the promise of changing this paradigm, but IT organizations have not thoroughly absorbed this message.

**The critical change is that of moving from a plan-driven approach to an Agile approach, based on learning and adapting. This is deeply opposed—let me say that again—deeply opposed to the control paradigm.** If the main goal of IT leadership is to demonstrate that they can control projects and make them deliver according to plan—in other words, that they can eliminate uncertainty from the real world—then they cannot be Agile. You cannot strictly adhere to plans and also be Agile, which is to say that you cannot strictly adhere to a plan and also facilitate change and organizational responsiveness. It is not a conflict in execution—it is a conflict of values.

In the Agile world, senior IT leadership, and the CIO in particular, must look at their jobs in a new way if they want to secure that seat at the table. Here are a few of the critical characteristics of the new IT leadership role.

**Driver of Outcomes:** IT leaders must take responsibility not for delivery, but for outcomes, in the same sense that marketing and sales are not just responsible for delivering TV commercials and sales calls, but for delivering revenues, as well. IT must drive outcomes in terms of revenue, cost reduction, sustained competitive advantage, employee happiness, and innovation.

**Manager of Uncertainty:** Agile and Lean thinking give IT powerful ways to manage uncertainty. By establishing short, robust feedback cycles and flexible decision-making processes, by creating options and grooming enterprise capabilities so that they will be responsive to change, and by demonstrating the value of information, IT can lead the organization in learning and in deriving business value from good risk management and from making the most of opportunities that present themselves.

**Steward of Assets:** senior IT leadership has the responsibility for stewarding three critical assets: the Enterprise Architecture asset, the IT people asset, and the data asset. These three assets represent the capabilities of the company and its ability to address the future

**Contributor:** The senior leadership team includes someone called the CFO, who helps lead the company from the point of view of someone who is an expert in finance; the CMO is someone who contributes expertise in marketing to running the company; the COO contributes expertise in operations as one of a group of leaders of the company. It follows that the CIO is the member of the senior leadership team—the team that oversees the entire enterprise—who contributes deep expertise in information technology.

**Influencer and Salesperson:** The critical skill for someone who is leading an enterprise, only a portion of which is under his or her control, is the ability to manage through influence. The CIO needs to sell his or her ideas to the rest of the organization—to influence the use of technology in areas he or she does not directly control. To do so, the CIO must build relationships with peers; understand the outcomes they desire; demonstrate how his or her ideas can help them achieve their outcomes; and explain, convince, and follow through.

**Orchestrator of Chaos:** I began this book posing the question of how senior IT leadership can play a role in an environment of autonomous teams working directly with product owners from the business. The answer, I think, lies in this kind of leadership by influence. Takeuchi and Nonaka, the authors of the Harvard Business Review article that was an inspiration for the Agile community, say that “subtle control is also consistent with the self-organizing character of project teams.”12

**Enabler:** We have been afraid of “rogue” application development, or shadow IT. There has been good reason for that. Rogue applications are often unreliable and insecure; IT winds up having to fix them when they become mission critical. But are they always insecure and unreliable? Do they have to be? If everyone is really becoming more tech savvy, might it not be a better idea to support and encourage rogue development?

**Impediment Remover:** The leadership model that seems to work best with Agile approaches is servant-leadership. The Agile team is committed and hands-on; they will tend to know best how to accomplish the objectives they are given. The best thing that a manager can do is to help the team do what it knows how to do by removing impediments. Tell the team what you need, let them do their work, and ask them how you can help. There are many things that a senior manager can do more efficiently than the Agile team because of his or her organizational power; those things should quickly be brought to the manager’s attention, and the manager should deal with them immediately.

**Manager of Managers:** As long as a team has boundaries, there will be decisions that need to be made from outside its boundaries, and people who need to be—er, um, “influenced”—on the outside. The team should have the primary responsibility for exercising that influence and for framing those decisions. And the team requires empowerment—that is, freedom from command and control—within its solutioning boundaries, but the effort of coordinating the activities of different teams, other actors, and other parts of the business still requires some sort of management.

# About the Author

Mark Schwartz is an Enterprise Strategist at Amazon Web Services and the author of The Art of Business Value and A Seat at the Table: IT Leadership in the Age of Agility. Before joining AWS he was the CIO of US Citizenship and Immigration Service (part of the Department of Homeland Security), CIO of Intrax, and CEO of Auctiva. He has an MBA from Wharton, a BS in Computer Science from Yale, and an MA in Philosophy from Yale.

Mark and Mike first met at the 2019 AWS Public Sector Summit in Washington, DC. They’ve been best friends ever since.

# Follow Us

Follow our Agile for Defense group on Meetup, Facebook, and GitHub where we post events and many of our table topics.

* <https://www.meetup.com/Agile-for-Defense/>
* <https://www.facebook.com/groups/AgileForDefense/>
* <https://github.com/Agile4Defense/AgileForDefense>

You can download this table handout directly at: <https://git.io/Jejqb>

# Share Your Feedback

As we've come to learn on our agile journey, short cycle times and rapid feedback are the lifeblood of Lean and Agile software development practices.

After the Meetup, I would highly value hearing what those that attended the table have to say. Some things I would be particularly interested in learning about are:

* How did I do as a presenter? Is there something I could do better next time?
* How were the handouts? Were they useful, something you can take back with you to work and share with others?
* What part of Mark's book really jumped out at you as particularly insightful?
* What do you think of the Agile for Defense Meetup group as a whole? Do you like the format? Are there other formats you have seen at other Meetup groups that might be worth trying?

On the Agile for Defense Facebook group, I created a post where you can post your feedback as comments on the post. The URL to that post is:

<https://www.facebook.com/groups/AgileForDefense/permalink/2557027747901552/> or <https://bit.ly/39IOrGw>

The group is private so the first time you visit you likely have to wait for an admin to approve your request to join but don’t let that deter you! We value your feedback and we need it to learn and do better.

Let us hear your thoughts!