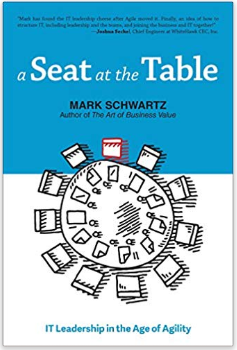
A Seat at the Table – IT Leadership in the Age of Agility

By Mark Schwartz

“Mark Schwartz is a rare combination: a deep thinker who has also applied lean, Agile, and DevOps principles at the highest level, leading an extraordinary Agile transformation in the US Federal Government at USCIS. In this book, he shows how modern IT leaders succeed by driving business outcomes rather than operating an order-taking function. This shift in organizational mindset is critical to any successful technology transformation but requires substantial changes in behavior at every level, and Mark’s thorough analysis will prove invaluable to leaders who must execute it.”  
 – **Jez Humble**, CTO, DevOps Research & Assessment, LLC

# A Few Themes

**Getting in Our Own Way:** First, that we have locked ourselves into a frame of reference that is getting in our way as we try to become Agile. This frame of reference includes the notions of project, systems, application, investment, architecture, skill set, and accountability. We have, to be honest, made a jumble of these concepts.

**IT as an Asset:** Second, that the business value of IT is more like the value of an intangible asset, which I will call – despite some disconcerting connotations of the term – the Enterprise Architecture. The asset view of IT will substitute for the outdated project view in my vision for what IT leadership must become.

**Uncertainty and Risk:** Third, underlying all of these changes – all of the problems with plan-drive approaches, all of the advantages of Agile approaches – is a confusion about how to deal with uncertainty and risk. What I call the “contractor-control paradigm” – is really about trying to make risk go away, when risk really the essence of what we do.

**Complex Adaptive Systems:** Fourth and last, that the business should be thought of as a community, or perhaps as a Complex Adaptive System, which needs to be led and managed through an inspect-and-adapt, feedback-and-vision-oriented approach because of its complexity.

# A Nimble Approach to The Table

**Agile in One Paragraph:** Agile thinking simply says that we should empower small teams to inspect and adapt rather than stick to a plan. Lean thinking gives that small team ways to speed up its inspecting and adapting process to maximize its impact. Continuous Delivery and DevOps place the entire value stream in the hands of that small team so that it can optimize the whole – a term of art in lean thinking – and be empowered as a team to own the entire value delivery process. <Bow/><Applause/>

**Brining Lean Principles to Software Development:** Kanban is David Anderson’s approach to bringing Lean principles into software development while driving fear out of the transformation process through incremental change. Anderson provides four rules for implementing Kanban:

1. Start with existing processes
2. Pursue incremental, evolutionary change
3. Respect the current process, roles, responsibilities, and titles
4. Encourage leadership at all levels

# Planning

**Espousing the Wrong Values:** The real reason we should reject the plan-driven approach to IT is that it espouses all the wrong values.

* What is the value of rigidity and executing according to plan if you could be improving on the plan or responding to changing circumstances instead?
* What is the value of hitting your spending target if you could be spending less, or if a marginal dollar of unplanned spend will bring a return of much more than a dollar?
* What is the value of telling knowledge workers to follow a plan when you are hiring them to use their brains to figure out how to do things best?
* What is the value of adhering to a plan that was made at the beginning of a project, when uncertainty was greatest?

Business value is destroyed only when we substitute extensive planning for execution and when we substitute execution according to plan for thinking and adapting.

**A Better Way to Plan:** Nevertheless, planning is important. In my role, I need to decide whether to allow an initiative to begin, which of course depends on whether I am satisfied with its plan. I might ask these questions when I review a team’s plans:

* What are the business outcomes the team is trying to achieve? What kinds of activities do they think will achieve them? Why are those outcomes valuable to the business? I want to make sure that the team has a clear vision, that they understand the business’s intent, and that they will make decisions within the context of that intent.
* How will the team determine the specific requirements—that is, determine what work they will do? Note that I am not asking what the requirements are, but rather how the team will discover them. I want to make sure the team has a good basis for making value decisions.
* How is the team planning to work together? What skills are on the team? How will the team communicate? How do they plan to retrospect and continually improve their process? I want to make sure they can operate as a team and learn together, and that they are committed to continuous improvement.
* How will the team seek feedback on its work? How will it solicit feedback and guidance from management? How frequently will it engage management? I want to make sure that we have an understanding on how my input and feedback will enter into their process.
* What are the key risks to delivery? What assumptions are contained in the plans? How will the team move quickly to test those assumptions and gain information to manage the risks? I want to understand their plan for learning.

# Requirements

**Requirements simply don’t exist:** A requirement is a constraint. It is a way of saying “create value this way, rather than other ways.” Really, a requirement is a constraint masquerading as a decision.

* What we have traditionally called a requirement may better be thought of as a hypothesis.
* We can think of the project team and its sponsors and stakeholders as embarking on a voyage of discovery and innovation, during which they begin with a goal in mind and empirically find and test hypotheses.

**A Better Way – Desired Outcomes:** We must replace the notion of requirement with that of a desired outcome. What gets tossed over the wall for the team to solve should not be a set of requirements: it should be an objective, an envisioned outcome that would add business value.

* The best framework I have found for working with desired outcomes as requirements is Gojko Adzic’s *Impact Mapping: Making a Big Impact with Software Products and Projects*.
* The age of IT organizations hiding behind requirements—“just tell me what you need”— is gone. IT leaders must instead take ownership, responsibility, and accountability for accomplishing the business’s objectives. The IT leader must have the courage to own outcomes.

# Transformation

**Mistakes have been made:** Transformational projects are evidence that a mistake has been made.

* The worst of these transformational projects are so-called “modernization projects”—transformational projects undertaken to bring technology platforms up to date.
* There is a deeper problem at the root of this dysfunctional transformation cycle. It lies, I believe, in our distinction between the development of a system and its operation and maintenance.
* Dividing our IT spending into development and maintenance buckets leads to some ineffective ways of making decisions. A nod to the ancient Greeks and the tale of Theseus’s ship.

**A Better Way – The Strangler Pattern:** Theseus’s activities fall into what the software world now calls the strangler pattern: a way to incrementally modernize a legacy system as defined by Martin Fowler.

* Instead of building an entirely new system, we take a small piece of the legacy system and rebuild it in a way that lets it interoperate with the rest of the legacy system. We launch that piece into production and have users use it seamlessly as if it is part of the legacy system.
* Then we take another piece out of the legacy system and do the same thing. And another. And another. Until eventually there is nothing left of the legacy system—it has vanished, piece by piece, like Alice in Wonderland’s Cheshire Cat.
* The strangler pattern overcomes a critical problem many of us have faced in applying Agile techniques to modernizations. In the absence of the strangler pattern, we would develop a new system on the modernized architecture and then move the users over to it. The problem is that the users cannot begin using the new system until its capabilities at least match those of the legacy system. Because this usually takes a while, the first release of the new system doesn’t come for quite some time, which works against the Agile principle of delivering value quickly and frequently. Legacy modernizations cannot be done in an Agile way without the strangler pattern.

# Coming up in Part Two

**Enterprise Architecture:** Enterprise Architecture, the domain of the IT bureaucrats, is the place we must look for the solution to our Agile challenges. We shall journey to the land of the template zombies to retrieve our golden asset, careful to carry mirrors to avoid petrification. Good luck, Agile fellows.

**Build Versus Buy:** Everyone knows that in every case under the sun, in any example one can imagine, when rational human beings are making decisions, if an IT product can be acquired “off the shelf,” it is better to do so than to build it. This obvious fact is neat, plausible, and in most cases, wrong.

**Governance and Oversight:** Governance has traditionally been viewed as a filter; a way of allocating scarce IT resources among many competing projects. 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.

**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.

**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.

**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.

**Exhortation and Table Manners:** There’s nothing left to do but end on a message of hope. And that’s what this book is meant to be, really.

# 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.