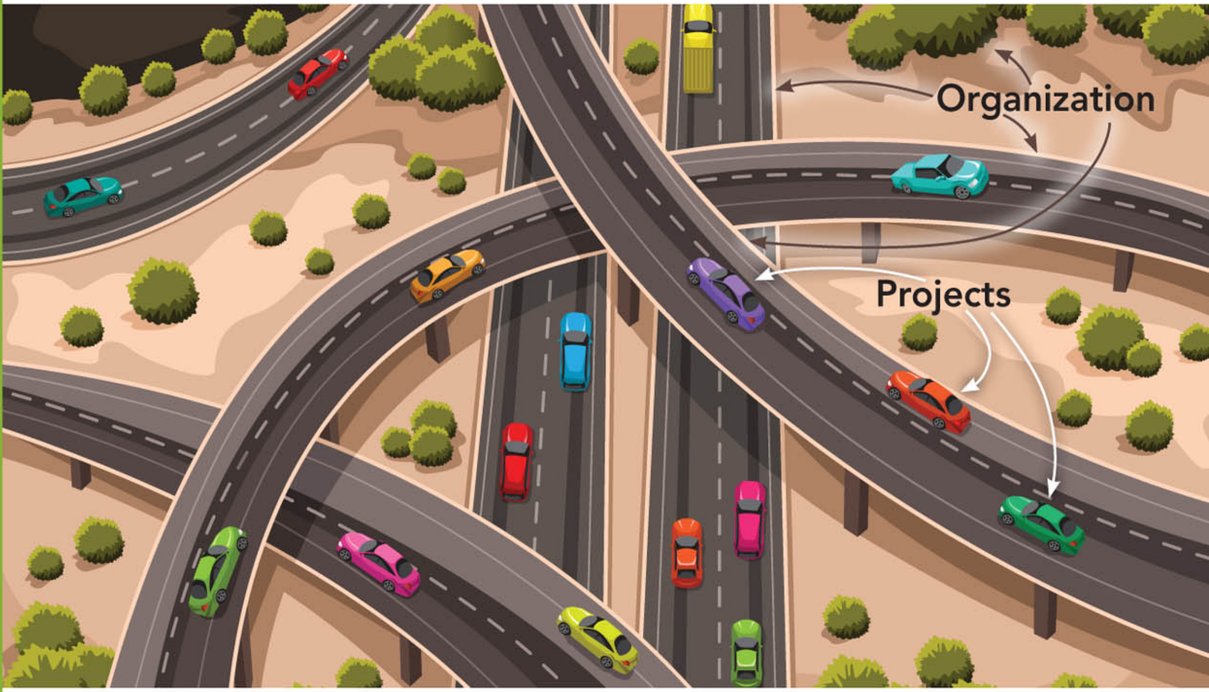
Agile IT Organization Design

For Digital Transformation and Continuous Delivery



Sriram Narayan

Praise for *Agile IT Organization Design*

"Continuous delivery is often described from the perspective of the technicians. This is understandable because that is where it started, but it does the process a disservice. Continuous delivery is a holistic approach. It requires change across the organization and it encourages such change, to the betterment of the groups that practice it. This book addresses that problem and looks at CD from an organizational perspective. It starts from Dan Pink's ideas of intrinsic and extrinsic motivators and describes how to structure an organization for success—how to encourage a focus on autonomy, mastery, and purpose that will motivate your teams and produce high-quality results. This book takes a look at all aspects of organizational design that impact the ability to deliver regular, small, high quality changes. If you follow the advice in this book, your organization will be the better for it."

—*Dave Farley, author of* Continuous Delivery

"A number of years ago, Silicon Valley marketing guru Geoffrey Moore quipped, 'A bank is just a computer with a marketing department.' Today, technologies—cloud, social, big data, the Internet of Things, and mobile—continue to drive this unprecedented digital transformation in organizations. As such, the need for agility has moved from software development to corporate boardrooms. Sriram's book makes the case that to thrive in these fast and uncertain times, enterprise leaders need to rethink how IT, not just software development, is organized, structured, and measured. His book provides guidelines, not prescriptions, which enable innovation, adaptability, and responsiveness at scale."

—*Jim Highsmith, Executive Consultant, ThoughtWorks, Author of* Adaptive Leadership

"Very hands-on and operational book for management of Agile-based development. Provides valuable insight for IT practitioners. A must read for IT professionals."

—*A.V. Sridhar, Founder, President & CEO Digite, Inc.*

"Agile IT Organization Design is an engaging, enlightening, and immensely practical book. While many authors have addressed Agile software development, very few have tackled the wider topic of the more systemic changes necessary to move from Agile software to an agile organization, and onwards to 'digital transformation.' Even fewer have done so at more than a very theoretical level. Drawing heavily upon his substantial practical experience, Sriram Narayan's served by having a single, dedicated outcome owner for the whole team of teams. The cause of autonomy and purpose is served by having a team big and capable enough to own a business outcome (or suboutcome).

5.5 Cross-functionality in Other Domains

The notion of cross-functional organization is also pertinent to disciplines other than IT. It is only fair to indulge in a few interdisciplinary analogies while on the topic of interdisciplinary (cross-functional) teams.

5.5.1 Hospital Pod Teams

A study[16] conducted at the emergency department (ED) of a city hospital corroborates the advantages of moving from an activity-oriented team design to a cross-functional one. Their initial design had three activity-oriented teams of nurses, residents, and attending physicians servicing a value stream that consisted of the following activities:

- Triage incoming patients
- Begin patient care work (nurse)
- Order tests, make decisions about diagnosis, treatment, and disposition (resident)
- Approve or change the orders and decisions (attending physician)

The study notes that back-and-forth discussion was not enabled by this design. Responsiveness was poor—an average of 10% of patients left without being seen because of delays. As a redesign, the ED teams were divided into pods (cross-functional teams). Each pod had the personnel and equipment necessary to treat any type of ED patient; that is, it had the ability to service the entire value stream above. The study found that the pod system delivered a 40% reduction in cycle time (in this case, cycle time is the average amount of time a patient spends in ED) without any significant difference in any other aspect of the quality of care. Note that the rest of the hospital functions can continue with an activity-oriented organization, as they are not directly part of the patient-care value stream.

[16] Valentine, M. A., and A. C. Edmondson. 2014. *Team scaffolds: How meso-level structures support role-based coordination in temporary groups.* Cambridge: Harvard Business School.

5.5.2 A Cross-functional Museum Layout

The Rijksmuseum in Amsterdam is a good example of the power of cross-functional organization. Traditionally, museum galleries have a functional organization—one gallery for sculpture, another for ceramics, a third one for paintings, and so on. Each gallery is managed by a specialist curator—the museum analogue of our function lead. But the new Rijksmuseum has opted for a more integrated or, shall we say, *cross-functional* organization. Each section in now devoted to a different century, and within that section you will find all the artifacts from that period arranged in an integrated holistic display that effectively conveys the story of the age.

An article about the reopening of the museum in *The Guardian*[17] describes the new layout. A Rembrandt gallery, for example, displays some of his early work alongside period-piece furniture, glass and silver artifacts made by people he knew, and a portrait by an art-patron friend. Rijksmuseum's director of collections, Taco Dibbits, says, "You get a sense of the world in which Rembrandt was producing his art." This is similar to how a product analyst gets a sense of the world into which the product is deployed by working in a cross-functional team that includes deployment specialists.

A cross-functional layout is arguably more work for the curators to manage and maintain. It may also nettle expert visitors who may be interested, for example, in sculpture but not ceramics. But from the point of view of majority generalist visitors to the museum (the outcome that matters), a cross-functional layout is probably more meaningful.

5.5.3 Taskonomy

Design guru Dan Norman talks about taxonomy versus taskonomy in the context of human-centered design.[18] Imagine how much less usable a word processor or spreadsheet might be if it only supported main menus, that is, no support for context-sensitive (right-click/pop-up menu) actions. As exemplified by Figure 5-3, the main header menu of an application is taxonomy whereas its myriad context-sensitive menus are taskonomies. Taxonomy is a functional classification or arrangement whereas a taskonomy is a cross-functional arrangement based on the needs of the task at hand. Taxonomies provide navigability—they offer a map of available functionality. Taskonomies provide ease of use and responsiveness—they are responsive to the needs of the user in context. In a user interface, both have their place. In organization design, the

[17] http://www.theguardian.com/culture/2013/apr/05/rijksmuseum-reopens-long-refurbishment-rethink

[18] http://www.jnd.org/dn.mss/logic_versus_usage_.html

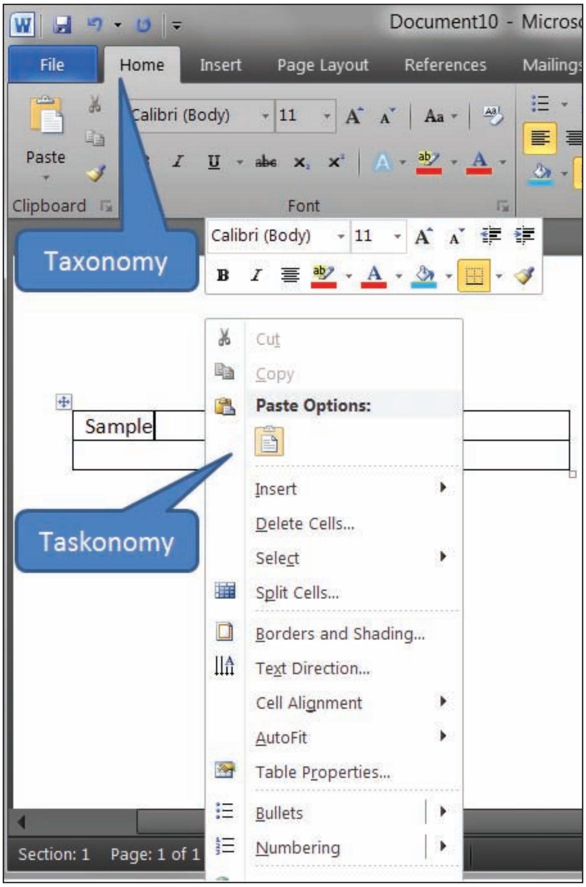


Figure 5-3 *Taxonomy is to taskonomy as functional arrangement is to cross-functional arrangement.*

org chart provides enough taxonomy. For responsive day-to-day work, we need taskonomies, which is what cross-functional teams are.

5.6 Migrating to Cross-functional Teams

It is quite disruptive to move from an IT-B matrix organization (or other functional organization) to self-sufficient, cross-functional teams. Here is one method of doing it gradually:

1. Identify products/capabilities that differentiate the business. You will need as many cross-functional teams as the number of differentiating business products/capabilities.

2. Identify a product/capability for piloting the transition. Ideally, the candidate won't have too many dependencies with other products/capabilities. Make sure there is a full-time outcome owner (Section 4.1.2) available.
3. Have the product owner come up with an initial product roadmap and backlog.
4. Identify people from existing activity teams that will make up the pilot team. Explain to them the rationale for the pilot. Use the penny game[19] to drive home how small batches and inexpensive handoffs help reduce cycle time.
5. Make sure the pilot team has all the skills required to be self-sufficient.
6. Let the new team start working through the backlog.
7. See how it goes for about three months before deciding to spin up another cross-functional team.

This only addresses the structural aspects of migration. Operational, cultural, and political aspects are addressed in the following chapters.

5.6.1 Separation of Duties

Sometimes, IT governance people say that cross-functional teams are not permitted by accounting and investor protection regulations such as SOX and payment regulations such as Payment Card Industry Data Security Standard (PCI-DSS). In particular, they speak of a control called *separation of duties*.[20] In effect, it aims to separate authorization for making changes to an application/system from authorization to release those changes into production. Traditionally, this hasn't been a problem because the production deployment team was different from the development team. However, even if separation of duties requires that the same person not have both authorizations, it does not prohibit two people with the combination of authorizations from working together on the same team.[21]

[19] http://www.leansimulations.org/2014/04/variations-of-lean-penny-game.html

[20] http://en.wikipedia.org/wiki/Separation_of_duties

[21] http://continuousdelivery.com/2012/07/pci-dss-and-continuous-deployment-at-etsy/

5.7 Communities of Practice

We saw earlier that a cross-functional team encourages its members to morph from pure specialists to generalizing specialists. This does not have to come at the cost of mastery in their specialization. A community of practice (CoP) is an alternative solution to nurturing a competency in the absence of a functional organization. A CoP does not require its members to be all part of the same team. It functions like a loose, professional association of specialists with mechanisms for online and offline interaction and knowledge sharing.

A lead is usually elected, nominated, or appointed per CoP. The lead comes from the same specialist background and is someone with people and organizing skills. The CoP lead is by no means a full-time role—she continues to work as a first-class member of some product team while devoting maybe 20% of her time to CoP work. CoP leads sponsor brown bag sessions, training programs, internal conferences, and sponsor members to participate in external conferences. They weigh in on tools and modes of collaboration within the community. They are accountable for the health of the community.

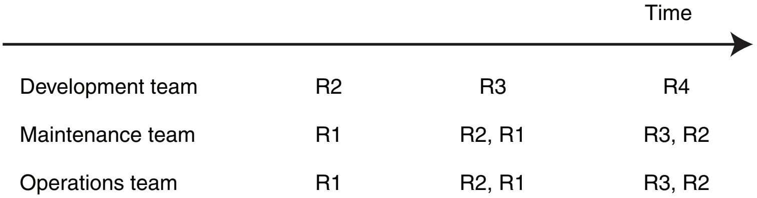
In addition, mastery in IT specialist areas may be sustained by getting involved in groups and activities outside one's organization. Specialist user groups and conferences are thriving in many cities. The Internet has many great resources for specialist skill enhancement. Even just following relevant Twitter hashtags goes a long way toward staying up to date. After all, individual mastery is at least as much the individual's responsibility as the organization's.

5.8 Maintenance Teams

Cross-functional product teams own their product—they shape it, build it, maintain it, and run it. However, many organizations retain separate teams for maintenance (bug fixes and minor enhancements) and IT operations.

Figure 5-4 shows a traditional cycle. Maintenance and IT operations work on what is released while development works on the next release. To cater to users who cannot upgrade to newer releases promptly, there is usually a support window of current minus N releases (N = 1 in Figure 5-4).

There is common but flawed notion in enterprise IT circles that maintenance work requires less skill than full-scale development. As a result, project sponsors looking to reduce cost opt for a different team of lower-cost people for maintenance work. This is false economy. It hurts the larger business outcome



Time

Development team	R2	R3	R4
Maintenance team	R1	R2, R1	R3, R2
Operations team	R1	R2, R1	R3, R2

Figure 5-4 *Typical software release cycle*

and reduces IT agility. When the same product team does development and maintenance, there is no handoff at release time. It is easier to merge bug fixes from release branch to trunk because the team is familiar with the ongoing changes in trunk. What's more, trunk-based development[22]—a branchless technique that is gaining adoption—is nearly impossible with separate development and maintenance teams.

End-of-life support is one situation where a maintenance team might make sense. This team keeps an old app/product running while a new replacement is built. Other than that, it is all about tearing down potential silos such as separate maintenance teams. Even in case of end-of-life support, a capability-oriented IT organization may choose to have the old and new coexist in a single capability team (Section 8.2). A separate maintenance team is a dinosaur in an age of continuous delivery and DevOps.

5.9 Outsourcing

When IT-B work is outsourced, we need to take care that the resulting team design does not violate the conditions of responsiveness, autonomy, mastery, and purpose discussed previously. Otherwise, business outcomes may be at risk. For example, the CapEx-OpEx distinction results in separate contracts/teams/vendors for development and maintenance. Some organizations go a step further and outsource even IT operations to a different team/vendor under a separate contract. The rationale is to stick to core business competency and outsource everything else (let vendors compete with each other for our slice of IT). Depending on how critical an application is for revenue generation, this strategy of "divide-and-conquer IT" can be frustrating at best and suicidal at worst. Internet businesses and ISVs typically outsource little to none of their IT-B. This is simply because having to orchestrate between three teams/vendors for every new feature is a huge drag on the ability to go to market quickly.

[22] http://paulhammant.com/2013/04/05/what-is-trunk-based-development/