**Chapter 2. The Human Challenge of Cloud Native**

Cloud native is more than just a technology or tool set. It is a philosophical approach for building applications that take full advantage of cloud computing. This new paradigm requires not only embracing new technology but also a new way of working.

As this book is being written, we have already helped several dozen companies integrate their cloud native systems. These enterprises have widely varied in size, sector, and background, but all had one thing in common: when they first began migrating their operations to the cloud, they firmly believed that a cloud native transformation meant simply a change in technology.

The thinking seems to go like this: *just start using containers and Kubernetes/microservices/cloud infrastructure, and whatever problems our organization is currently experiencing will suddenly be better*. While this is certainly a possible outcome for most companies moving to cloud native technology, it is also a very unlikely one—unless the organization also addresses the human challenge of cloud native.

We call it a cloud native *transformation* because, in order to make a successful, effective, and above all worthwhile migration to the cloud, the entire organization—not just the tech stack—must change. This means evolving nontechnical aspects such as management practices, organizational structure, and team responsibilities—even your fundamental delivery processes—to work in your new cloud native context. Otherwise, if you simply layer the latest tech on top of the same old way you have always done things, you will create a whole new slew of problems, including but not restricted to, slow delivery, reduced quality, and difficulties in both project and team management.

Quite simply, your organization’s culture needs to change along with the technology. If it doesn’t, you are very likely about to waste a lot of time and money on an ultimately unproductive transformation.

Talking about tech is easy because it’s concrete. We can see it and interact with it and observe direct effects and clear causality: when we change X, then Y happens. Culture, even though we all act and interact within it constantly, is much more abstract and indirect. Invisible, even. Nevertheless, organizational culture plays a crucial role when a company undertakes a cloud native transformation.

This chapter takes a look at the people-centric parts of a cloud native transformation. From an architectural standpoint, all the human aspects of a cloud native transformation fall into the overall category of “culture.” But what does “culture” really mean in this context, and why is it so important?

**Culture Vulture**

*“Culture is not something vague and undefined, but a set of living relationships working together toward a shared goal. Culture is not something you are—it’s something you do.”*

*—Daniel Coyle, Culture Code*

In a business context, the “do” is how the people within your organization interact, communicate, and work with each other. In short, culture is the way your enterprise goes about creating and delivering your service or product. If you have the perfect set of cloud platform, tools, and techniques yet go about using them in the wrong way—if you apply the right tools in an improper way because that is how your culture has always dictated you use any and every kind of tool—it’s simply not going to work. At best, you’ll be functional but far from capturing the value that the system you’ve just built can deliver. At worst, your system simply won’t work at all.

How do we know what we are talking about when we talk about a company’s culture? What does the word even mean inside an enterprise? The answer is surprisingly concrete for such an abstract concept.

Culture is the sum of the daily actions that you take. At the most fundamental level, your regular routines are your culture. For example, if you talk to people, listen to what everyone is saying, and incorporate that into decision making, you have collaborative culture. Needing to seek permission from higher up before trying something new means you have hierarchical culture.

Add up all the things you do in the course of a normal workday and then step back to look at them from the perspective of the collective, yet generally unexamined, assumptions that shape and drive these actions. These assumptions, about how work gets done and how people interact in your company, will reveal your culture. And this is not fixed or immutable—culture is slow to shift, but it can be done.

If you change your actions, you change your culture.

**A (Brief) Field Guide to Organizational Culture**

The three major types of culture within enterprises track closely with the type of product/service delivery process the organization follows: Waterfall, Agile, and cloud native. At least, these are the ones that we’ve identified; there must be more, but those three cover the absolute majority of the companies we see in our work. So we will be framing our discussion in terms of Waterfall, Agile, and cloud native organizations, and we further realize that in the real world there are as many different ways to characterize—and implement—these different approaches as there are companies. Bear in mind that when we say “Waterfall organization” we are talking about a company that mostly, if not entirely, uses the corresponding process.

First, a quick guide to the terms as we will be using them.

*Waterfall organizations* are all about long-term planning. They deliver a single large, solidly built, and thoroughly tested deployment every six months to one year (perhaps even longer). They are risk-averse, with a long decision-making chain and rigid hierarchy. There are a lot of managers, including project managers.

Waterfall organizations tend to have specialist teams handling very specific skills—security, testing, database administration, and so on. Although market estimations (from vendors of Agile tools and solutions) suggest most companies have adopted at least some Agile practices, our estimation (based on our own observations—remember, this is book is drawn from our direct experiences and not intended as any kind of definitive Ten Commandments of cloud native) is that about 80% of software today is still produced using a development approach that is still primarily Waterfall in style.

*Agile organizations* recognize the limitations imposed by monolithic long-term releases and have adapted to deliver faster by using a more iterative, feature-driven approach to releasing in one- to four-week “sprints.” Agile development breaks applications into their functional components, each of which is worked on start to finish by a single team. Instead of handoffs between specialist teams, Agile has cross-functional teams whose members hold all the skills necessary to design, build, and test the service they are responsible for developing. The result: there are wide responsibilities *in* a team, but a very narrow responsibility *for* the team. Scrum is one functional implementation of Agile process. Roles in this system include the teams themselves; Scrum Masters, who facilitate between the different process-owner teams; and Product Owners, who have full ownership of the delivered functionality and the deadlines.

*Agile-ish organizations*are a hybrid breed. Typically they are Waterfall organizations that have made some moves toward Agile, such as adopting Scrum (a common term in the industry is to say they follow a “Scrum-fall” process). For example, teams may work in sprints, but they are still building tightly coupled services on a longer term (six plus months) release schedule. Truly Agile companies are actually pretty rare, because fully implementing Agile across an entire organization is a serious challenge. Agile-ish is a much more commonly found culture type.

WealthGrid, our example enterprise, is an Agile-ish organization. Although they use Scrum and their teams have become somewhat cross-functional, WealthGrid’s culture is still the kind typically found in organizations using the Waterfall software development approach: hierarchical, permission-seeking, focused on proficiency over innovation, and driven by tight, rigid deadlines.

*Cloud native organizations* are built to take optimum advantage of functioning in cloud technologies; the cloud of course will continue to evolve and will look quite different in the future, but we also build to anticipate this. Applications are built and deployed in a rapid cadence by small, dedicated feature teams made up of developers who also know how to build in networking, security, and all other necessities so all parts of the distributed system become part of the application. Meanwhile a platform team sits off to one side, designing and assembling the platform the devs will deploy to. This highly automated platform uses an orchestrator like Kubernetes to manage the complexity.

This organization could also work in an Infrastructure-as-a-Service or a traditional virtualized VMware environment. However, taking maximum advantage of cloud technology means moving to approaches like minimum viable product (MVP) development, multivariate testing, abandoning specialist teams for DevOps, and embracing not just rapid iteration but rapid delivery/deployment (i.e., continuous integration and continuous delivery, even continuous deployment). Some of these are part of Agile, at least when fully practiced, but they are all necessary for cloud native.

Cloud native architecture centers on microservices architecture (i.e., de-composing applications into small, loosely coupled services that operate completely independently from one another). Architecture then influences process: each of these services maps to smaller, independent development teams that release iterative software updates as soon as they are ready. Rapid release of software creates a tighter feedback loop, allowing enterprises to easily adapt and respond to customer demands and desires.

Although cloud native is powerful and can provide a serious competitive edge in many cases, it is not automatically the “right” solution in every case. By the same token, Waterfall is not inevitably wrong or bad. There are times when Waterfall is absolutely the best process for the situation at hand—when, for example, there is a strong product that is stable and will require few future changes. In that case, riding along on Waterfall is effective and efficient. Agile, too, has circumstances where it is the optimal choice. “Good” versus “bad” has little meaning when applied to culture, because culture itself simply *is*.

**“Right” Solutions, “Wrong” Culture**

So, then, what matters is not culture itself, but the *type* of culture. Our daily actions inside an organization are dictated by our collective culture. So when a “right” solution (or, sometimes, a mishmash combination of solutions) gets applied in the “wrong” culture, the solution and the culture conflict, undermine, and ultimately gridlock each other.

It is a little like sitting down behind the wheel of a car when you have spent all your life riding horseback. Fundamentally, both are means of transport. You can be an expert equestrian, but that skill set is of little use in your new automobile. And, actually, driving the machine is not the problem; it’s not hard to figure out that the steering wheel makes the car turn, one pedal makes it go, and the other makes it stop. The truly difficult—not to mention most dangerous—part is figuring out how to take it out on the road, or how to jointly navigate with all the other cars also driving the same route, so nobody collides. The rules and assumptions of good horsemanship simply don’t apply when suddenly you can go 100 mph on a superhighway.

Either mode transports you from point A to point B. But the rules of the road—the collective understanding you share with all the other drivers out there—is suddenly vastly different. When we went from horses to automobiles, our daily actions changed. And this eventually changed the way we worked, where we lived (suddenly, suburbs became possible), how we communicated, and even how we interacted. Everything around us was affected, and our culture shifted in response. Anyone who tried to keep riding horses as their main form of transportation found themselves left behind, no matter how swift their steed.

Today there are laws in most parts of the world against taking a horse-drawn vehicle onto a highway because doing so is perilous for all involved. Conflict arises when new technology gets applied through the old way of doing things. Trying to do cloud native by using methods from your previous paradigm will wreck your initiative. This is in fact the root cause of most of the problems we find when we get called into a company to rescue a stalled cloud native transformation.

**The Culture Clash Conundrum**

In software development, as in real-world geopolitics, the worst problems happen when cultures clash.

The issue is not so dramatic in companies that merge Waterfall and Agile practices. Despite some surface differences, there are many functional similarities that enable such a hybrid approach to function. Both share the very strong presence of Product Owners who decide what should be built. In Waterfall, architects and managers are expected to understand the entire product with all its complexity, while the teams themselves need to understand only the part they are specifically responsible for. Similarly, in Agile, each team only needs to understand the components of the product they’re charged with delivering (which will create strong separation for that part from all other parts, due to Conway’s law wherein software will resemble the communication and organizational structure of the company producing it). But in both cases teams are still not really in charge of *what* they are building; their authority is limited to *how* to build whatever they are assigned.

It is worth noting that companies following truly Agile practices, or especially Agile combined with Lean, display culture very similar to what we are identifying as cloud native here. True Agile is very, very different from Waterfall. However, in our experience very few companies are able to reach completely Agile status, thanks in part to deeply ingrained Waterfall culture (or perhaps it’s because truly Agile companies are able to make the move to cloud native fairly easily, without help from consultants, so we rarely encounter them). The ones that call us for help, however, most often are working with significant baggage in terms of Waterfall culture: risk-averse, predictive, and, above all, slow.

Technology itself doesn’t deliver velocity, though it can certainly help. Changing your culture—the way you work every day—is how an organization gains true velocity.

Furthermore, both Agile and Waterfall require coordination across teams to deliver the separate parts together. The only difference is frequency, with Agile delivering much more frequently than in Waterfall. And while Waterfall teams are building a single monolith, Agile teams are effectively building a number of monoliths. In a typical Agile organization, the number of independent components is usually three to ten, with one team (sometimes two) working on each. This does means that Agile teams can take on three to ten times more complexity compared to Waterfall teams, but they are all still building monoliths. By the same token, in Agile many teams can work on the same, single monolithic component, effectively splitting it up into a smaller waterfall. The result is still central planning and a lot of coordination, including joint delivery of all components every one or few sprints.

So you can see how in some ways the cultural differences between Waterfall and Agile are actually compatibilities, rather than true differences! Working in cloud native, however, requires a completely new and different culture.

The most common culture problem we find when we are called in to help save an attempted cloud native migration gone wrong is where a company has tried to add a single element of cloud native while not changing anything else. “We will keep delivering software the same way we always have, except now we will have DevOps!” So we have a clash between the new cloud native technique, DevOps—where a single team, working independently, first develops and then deploys and maintains an application—and the “old” approach of building a tightly coupled system of apps and delivering everything all at once, every year or so.

Can you do DevOps in a Waterfall organization? Well, sort of. On the Dev side you can apply cloud native practices like continuous delivery to optimize and accelerate your development process. Meanwhile on the Ops side you can automate deployment to make your provisioning and infrastructure faster and more efficient than ever before. This is all great…except for the fact that your beautiful, containerized microservices and their state-of-the-art platform won’t actually get in front of users until the next release cycle is completed, many months from now, because you never changed your delivery process to match. Yes, you are doing cloud native—from inside the linear, sequential, and tightly coupled Waterfall approach. All that speed and efficiency? Simply wasted.

The culture-clash conundrum works backwards, too: you can’t have otherwise full-on cloud native culture but not have microservices. If it takes you six months to deliver, you can’t be truly distributed. There is nothing to be gained in simply re-creating a monolith on the cloud—yet companies try do it all the time.

**Which Brings Us Back to…Culture**

This is why understanding your organizational culture is critical for functioning well in the world—i.e., succeeding as a business. Knowing your culture means being able to choose the path that best fits your organization…or not choosing some hot new tech that, however promising, conflicts with your organization’s fundamental nature.

Cultural awareness also grants the ability to start changing your organization from within, by a myriad of steps all leading in the same direction. This lets you evolve gradually and naturally alongside a rapidly changing world. “Long-term direction” is basically the definition of strategy. It’s a myth that strategy has to be “big.” Egos are big. Taking small steps toward your direction is a solid strategic move. Strategy is incremental: setting a direction and then moving in that direction through intentional, iterative, intelligent moves, each one small, each building upon the last.

So: Your culture is the sum of the practices that add up to and define how you function. You can identify these forces that shape your organization so fundamentally by examining the actions that define your day-to-day operation. Then you can decide the best next steps. This is far from new advice. Long, long before cloud native, Confucian philosophy advised us that, in order to slowly change one action over time, you must first change yourself. Really it all comes down to two simple words from Socrates: know thyself.

A practical tool for creating this organizational self-awareness can be found in [Chapter 5](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch05.html#knowing_thyself_the_cloud_native).