CHAPTER 4

Busy vs. Done

Confirmation.com is a company created to solve a problem that wasted thousands of hours and millions of trees each year. They took a manual process that was slow and painful and hard and made it electronic, fast, and easy.

What they do is confirm financial data through a huge global network of accounting firms, financial institutions, law firms, and corporations. They think it should be easy to find the truth. And they realize that financial fraud isn't going away, which is why the motto of their founder, Brian Fox, is that they help the good guys catch the bad guys.

Let me give you just one illustrative example. The founder and chairman of Peregrine Financial Group, Russell Wassendorf Sr., defrauded investors out of more than $200 million over a period of years. How did he do it? A little Photoshop and he was able to create real-looking bank statements. The whole thing came tumbling down when Peregrine was forced to begin using Confirmation

.com. Within days the fraud became clear. And Wassendorf still has decades left in jail for his crimes.

For more than a hundred years, the confirmation process was done on paper. An auditor would send confirmation requests by mail to a bank: *Does the institution being audited actually have this much money?* The bank would receive not only this request but thousands of others—even hundreds of thousands—each year. Banks had to have whole groups of people deal with it. Each confirmation request needed to be responded to by manually checking the bank records, writing a letter that confirmed the institution had that much money, and sending it back by mail. Paper. Lots and lots of paper. It took *weeks* for each and every one of the many, many tens of thousands of these requests made each year.

Confirmation.com made all this happen in a matter of moments. When someone makes a request, Confirmation.com directs that request to one of the thousands of connected banks in its network, gets the response, and passes it back to the auditor. The amount of security needed to move all that sensitive financial data around is critical. And that was the hard part in the beginning: getting financial institutions, accounting firms, law firms, and their mutual clients to trust the Confirmation.com system.

Confirmation.com pioneered the idea of electronic confirmations almost twenty years ago—receiving seven patents along the way—and is still dominant in the space by a large margin. It started with one bank and one accounting firm in Nashville, Tennessee, and now sixteen thousand accounting firms, four thousand banks, and five thousand law firms in 160 countries use their platform. They confirm more than a trillion dollars of assets each year.

When the company started back in 2000, at first it was just four people, literally in a garage, cranking out their product, this

new thing that had never been done before, which Brian envisioned and wrote up as his entrepreneurship paper in business school. Eventually big banks started to realize the amount of work they would save and the speed they would gain, and ultimately said they would only accept requests through Confirmation.com's platform—no more paper. Confirmation.com grew rapidly, and they started to add new features and new confirmation types—like legal confirmations—to their platform.

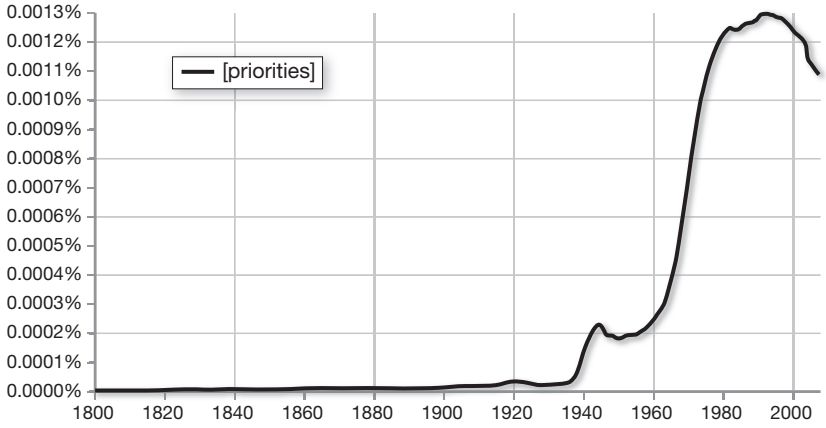
But then something happened. They weren't getting things done quickly enough. They were missing deadlines. The code quality a platform like theirs requires wasn't there. They certainly weren't going to risk that, so they took more time to try to get it right. Oh, they had people working hard. They were busy. One executive there told us he had to keep everyone busy. Hopefully they'd get something done, but if they didn't, he could at least say he tried. But they were simply unable to get stuff out the door. Everyone was busy, but not much was getting done. So they called us.

This is a fairly typical issue in corporations. Some project—it doesn't matter what it is—must get done. Management or sales or somebody has said it's a top priority. Then someone else says something else is also a top priority. And someone else brings in yet another. And, of course, none of these people insisting that their priority is what everyone should focus on talk to one another. They just keep shoving them down on the teams. This is horrifyingly commonplace. Then, with the predictability of gravity, things stop getting done. So management starts putting pressure on the team. They want them to be busy all the time working on the many, many things that are all top priority. They make them work nights and weekends to meet an arbitrary deadline that they promised to someone. And they are mystified when it doesn't happen.

Facts Can Be Stubborn Things

Priority is an old word. It began its life in the late fourteenth century when the French borrowed a Latin word to indicate the state of being earlier—this event happened before that event. It entered English in the early fifteenth century, referring to precedence in right or rank. (Incidentally, it's a singular word. Saying something is a "top priority" is redundant. It's using two words to say one thing. Another linguistic tidbit: Latin words aren't pluralized by putting an *s* on the end, so the word *priorities* is actually nonsensical. It quite literally does not make any sense. It's like saying there were five first-place finishers in a race.)

If you run a Google Ngram search (that's where Google looks at however many thousands of books over the past few hundred years and counts how often words are used) on *priorities*, you get the following result:



Priorities wasn't even a word until about 1940. I'm not dead certain about causation, but there is a certain correlation between the rise of the modern management movement of postwar industry and the birth of a new word that sounds seemingly logical

and rigorous while actually being meaningless. That seems telling to me.

Stop Starting, Start Finishing

When Scrum Inc. goes into a company to assess how Agile they are, we usually find that about 30 percent of the work being done should not be done at all. That work is actually in *opposition* to goals of the business. Stop that. The Standish Group data says, and this is what we find, that 64 percent of the remaining 70 percent are working on features that the customer rarely or never uses. That means 75 percent of people in a company are either actively working against the business interests or are working on things that no one wants. Let that sink in for a second. Three-quarters of your company shouldn't be doing what they are doing.

The reason for this is that people refuse, or don't know how, to prioritize. (Another quick linguistic factoid: Apparently *prioritize* wasn't a word until it was first made up by government bureaucrats in the 1950s and popularized in the presidential campaign of 1972 when political operatives had to choose which voting districts to target. As the *Oxford English Dictionary* put it in 1982, it is "a word that at present sits uneasily in the language." It apparently sits uneasily in people's practices as well.)

Here are the symptoms we see. If you hear or utter any of these phrases, you might want to rethink your approach:

"We have multiple conflicting priorities."

"Our teams are constantly disrupted by new priorities."

"Everything is the number one priority."

The thing that kills me is that everyone knows these are bad things. Not a single person I have ever spoken to thinks that working on five things at once and disrupting even that work for new urgent things is a good idea. No one. We all know it's stupid. But we all do it anyway.

At Confirmation.com, everyone had different priorities. Sales wanted a better Japanese translation so they could sell there, marketing wanted to rebrand the website, and the leadership was concerned with an upstart competitor. So what should the product teams focus on? "I'm always waiting to hear what the new rules are today," one executive told Avi Schneier, Scrum Inc.'s principal on the case. When Avi asked what the company's number one priority was, he was told, "Meeting deadlines." Not what needed to be done, notice, just the deadline itself.

So Avi had them work through the company's real priorities. What are they? What is the most important thing? He helped them see that without choosing those priorities, they ended up with a company adrift, going in a different direction each day. So they made those choices. It can be done, but it requires honest reflection and some tough decisions.

Output vs. Outcome

Let's tease these two ideas apart. Output is how much stuff the Team produces each Sprint, their Velocity. When you start out with Scrum you want that Velocity to double or triple within a few months. If you can't get things out the door, even if they are the wrong things, nothing else matters. Focus on getting the Team's work to *done* and out the door. If it turns out to be the wrong thing, and it probably will be, you will find out quickly rather than wasting millions of dollars and years of your life before finding out no one wants what you are making.

But the good news is, once you know that, you can focus on outcomes. *How do we make happy customers? Save more lives? Bring value to the world?* You need to answer those questions, otherwise your work is in vain. What you want to do is get stuff in front of people so they can tell you what they love, want, and need.

The trick to doing that is getting quick feedback from whoever is getting value from what you are creating. Find out just how valuable it is to them. At the beginning of a project or product, you pretty much guess what things will be most valuable. It's an educated guess, backed up with research and the like, but still just a guess. If you have to wait six months to find out if your guess was right, well, you are planning with hope instead of data.

At Confirmation.com, their biggest problem was their legacy system. It worked, and it worked well. But as it grew over the years, it slowly accreted new features every time a customer asked for one, fixing this system or enhancing that piece. Each bit was tacked on without much thought about architecture and structure. It eventually grew into a mess so big they spent more time fixing old mistakes than building a new system to replace it. What they eventually realized is that while everyone was really busy, they weren't actually making significant forward progress. They couldn't get anything out the door efficiently. By being focused on making sure people were busy, the output, they didn't look at the outcome. They really needed a new, more modern system that would allow them to provide their customers a service that was even better than what they thought they wanted. But everyone was working on keeping the old system together—output, not outcome.

Definition of *Done* and the Nature of Architecture

The key to getting to done is defining up front what *done* actually means. When a Team picks up an item from their Product Backlog, they should know the nature of done for that piece, they need to define *done*. Part of that definition has to be how that piece builds on other pieces. Why? Because—and this is critical—the architecture of your product determines what your Definition of Done can be.

Let me give you a couple of examples. One is from the world of hardware, the other from software, but the thinking is exactly the same.

There is a private space company that will only let me call them the Stealth Space Company. That's what they called themselves on their LinkedIn page, in any of the very unwanted press coverage, and have repeatedly driven into everyone's mind. *We don't brag, we don't talk, we just do.* They're based in an abandoned naval air station sitting on the edge of San Francisco Bay. Military bases like this one do vary somewhat from location to location and service to service, but they all share one thing in common: a brutal, uncompromising architecture of function over form.

Chris Kemp is the CEO. He has blond hair, wears mainly black T-shirts, black jackets, and black pants. His mantra is speed. From his email announcing their first launch attempt:

> On Sunday, we will attempt to launch a rocket that was designed from scratch by a team that did not exist 18 months ago. We have done this five times faster, and with five times less capital, than ever before. This is the first in a series of test launches that will allow us to iterate towards orbit as we build our team and incorporate what we learn from each attempt.

He looks at Elon Musk's SpaceX and sees a target that not only can be beaten but can be beaten soundly—five times as fast at a fifth of the cost. And he is using Scrum to do it. His goal is to be the FedEx to space, launching small payloads daily into low orbit. The military needs a constellation of spy satellites over a new trouble zone? No problem; it'll be there in thirty minutes instead of three years.

Speaking with his people, you can feel their drive to succeed. One of his leaders spent his career in the space business: SpaceX, Virgin Galactic, Boeing. He said some of his team felt the whole Scrum framework is only for software.

"This is all new to me, J.J.," he told me. "But I've seen just how bad the old way of doing things is. I tell these new engineers that they don't know how good they have it, and I'm all in. I have made it very clear they're all in too, or they need to find another job."

A rocket, I learned there, is really three systems: an engine, which turns fuel into force; avionics, which directs where the rocket goes; and structures, the wrapper that holds everything together. In the first iteration of the rocket, those pieces were tightly coupled, both within each system and between systems. The reason for this is that they are trying to get rid of any excess weight, so each interface is custom-designed with custom pieces and custom connectors. Which makes sense when you are thinking only about weight. But the tricky part is when you want to fix something.

Let me give you a simple example. In their first rocket the avionics system was controlled by a series of specialized circuit boards individually connected to each other and to the rocket as a whole with switches made from some ultra-rare unobtanium material. If one board failed, you had to pull out all the boards and then redo hundreds of connections by hand with these incredibly expensive materials. At one point, the rare-earth elements used in those connections simply went off the market: Apple and Samsung had gob-

bled up the world stockpile for use in their next generation of phones. It took twelve weeks to get more. Kemp's reaction to this news was both profane and exasperated: "Three months for an Ethernet switch? This is the kind of [stuff] that will kill us!"

My colleague Joe Justice sat down with Ethan, the head of avionics, and talked through the problem. *First,* said Joe, *you have all of these boards with all these special connectors, each one different from the last, each carrying different information. And you need to unravel the complexity, replace them with better designs. But if you pull one, you break all the others. So let's put a stable interface between the avionics and the rest of the rocket. Let's overengineer it so that it can carry all sorts of data, more than you need, but it will use common connectors you can buy off the shelf for pennies. Let's encapsulate the problem, making a firewall of sorts that we know won't change, and let's make sure the rest of the rocket engineers know their systems only have to connect to this interface with one side of this connector and the avionics engineers know they just have to connect to the other. That way you can change anything you want on either side; as long as that interface remains the same, it doesn't matter. What you are trying to do is modularize the problems. You want it to be like Lego. You can snap the pieces together and snap them apart easily.*

This approach makes the Definition of Done easy: it has to work, and it has to fit into this known stable interface. Then you can start knocking out your problems one by one. The extra weight from the interface itself? You can iterate on that later, once you've fixed your other problems.

Now let's take an example of Agile Architecture from software. It's exactly the same pattern. Spotify is a music streaming service. Their goal, like the goal of the rocket company, is speed. When they were a start-up, the CEO of Spotify, Daniel Ek, once told Scrum Inc., "Listen, Apple, Google, and Amazon want to kill us. And they are smart, large companies with lots of skill. The only way we can survive is speed. We have to be faster than them."

So Spotify is split up into different modules, just like a rocket ship. There's the player, there's the recommendation engine, there's the playlist functionality, there's the mobile app, and so on. And just like at the Stealth Space Company, they've developed stable interfaces between each of those pieces. The teams working on the playlists can innovate as much as they want, change as much as they want, as long as it still fits within the right size box, has the same data going back and forth, and doesn't break anything else. That way they can go fast without worrying about breaking other parts of the system.

They don't have to change the whole system to change their piece of the system. That interface spares them a huge amount of pain. In many systems, the dependencies between the various pieces are so great that making any changes becomes nearly impossible, and the speed of development slows to a crawl as engineers have to use ever more duct tape and baling wire to hold an increasingly rickety and rigid system together.

Most defects, no matter what kind of product or service or process you are creating or doing, come when two perfectly good parts of the system are integrated, and to fix either one you have to break them both. It becomes exhausting.

The Fix

So what do you do? You have dozens of projects, hundreds of priorities, and they all have to get done, or so you've told yourself.

The first step, as always, is to admit you have a problem. If your strategy is to say nonsensically that everything is a priority, then what you are really saying is your strategy will be decided by the most junior person in the organization when that person, given zero guidance on what is actually most important, decides what to do next.

In using Scrum, the first thing is to make sure that every Team has a clear and ordered backlog for every single Sprint and that they understand the relative business value of everything they are being asked to do. This requires a mechanism, which I'll get into in later chapters, that takes the big goals of your organization and breaks them down into actionable items for the Team.

At Confirmation.com, Avi and another colleague, Alex Sheive, sat everyone down in a room and got all the things they wanted to do up on a wall. They worked with management to create a clear, ordered backlog for the Teams. They convinced them to keep the Teams stable, not to move people around, and they sent that message throughout the organization. Management was behind Scrum and was willing to do the hard work. They were taking a hard look at the list of things they wanted to accomplish and deciding what they were *not* going to do in order to get done what actually needed to get done.

That's the first step, admitting that you aren't going to get everything done right away. You have to make a choice. It can be hard at times. There are often a lot of competing interests. But if leadership isn't aligned with what must be done and in what order, the Teams will have no idea what to do. The leadership at Confirmation.com was able to pull it off. They restructured, they streamlined, and they made sure each Team had a clean, prioritized backlog for each and every Sprint. And it dramatically changed what they were capable of doing.

The Importance of No

The root cause of this lack of willingness to prioritize comes down to this, the unwillingness to say no. Just as every Team has a Velocity, every organization does as well: how much can we create in how much time? It is very easy to say yes to customers, to leader-

ship, to bosses. *You want that done? No problem—we'll put it on our ever-growing backlog of things to do. Oh, and it's really important? Well, I'll just cram it in here at the top.* Then someone else asks about a project of theirs. And the answer is yes—again, always yes, until the team or the organization implodes.

Corporate strategy almost always focuses far too much on what a company will do rather than what a company won't do. Let me illustrate this. There is a global materials company that does huge amounts of research and then turns that research into everyday products manufactured at scale, with millions and millions of units going out the door. But they have a problem. This company spends years doing R&D, coming up with a new product with novel materials, and once they introduce the product, a group of what are known as "fast followers" replicate it quickly, sometimes in a matter of months. So they have to keep coming up with new products.

One particular division simply couldn't get new products into the market quickly enough. They had grand ideas but it just wasn't happening. So they called up Scrum Inc., and in early 2016 a mild-mannered guy by the name of Steve Daukas walked into that building to see what he could do.

He got the lab leadership into a room and said, "To start with, let's just talk about what you are doing. Let's get every single project on Post-it notes and on a wall so we can take a look."

A quarter hour of furious scribbling later, they had the projects on a wall. There were over a hundred of them.

"Okay," said Steve, "just for fun, let's put names on all those projects. Who is actually working on them?"

Even at only a few people a project, they ran out of names somewhere around number seventy.

"Why are you working on so many things?" asked Steve.

The answer he got was that corporate was telling them to get more products into market, so they had to do a bunch of products at once to start monetizing them quickly.

"And are any of those actually getting done?"

There was complete silence.

I have heard variants of that conversation with everyone from parents of small children to start-ups to Fortune 500 companies. They have to get so much stuff done, so they start a whole myriad of projects. In their minds, they have to; there is just so much to do. And then they can report, "Look at all of the things we are doing! We are so busy, it's crazy! We're working on all of these top priorities."

What they are actually doing is *starting* a whole bunch of things. What they aren't doing is *finishing* them. It is amazing to me to ask a company how much stuff they have in flight and they brag about it. They seem to think I will be impressed by all they are doing.

And when we ask them what impact they are having, their faces fall.

In that boardroom Steve forced the group to acknowledge that not everything was going to get done. And not only was not everything going to get done, if they continued down the path they were on, *nothing* would get done.

So they took those hundred-plus projects and started making the hard choices of what they were not going to do. What did they want their lab Teams to focus on? What would actually make a difference in the market? They argued. They lobbied. They killed people's pet projects. And they worked hard doing what leadership really has to do, which is to make choices. It is so easy to say yes to another project, so easy to agree with someone else that their idea should be pursued, so easy not to have the hard conversation. So easy not to say no.

Eventually they got down to twelve. Then Steve had the leadership of the group make clear backlogs for all of those projects. Not extremely detailed, but at a high level that communicated the "commander's intent": not how the Teams would accomplish those projects, but what and why. Next the twelve people who would be Product Owners got up in front of the whole group of scientists, a couple hundred people, and presented their backlogs—what they were trying to accomplish, why it was important, and what skills they needed to deliver it. And then the Product Owners and management left the room, telling everyone, *You're smart—you figure out who should be on which Team to accomplish those twelve goals.*

Thirty minutes later, they went back in. All of the backlogs had a Team or Teams, except for one. That one involved some tedious government compliance documentation updates they had to do but no one wanted to do it. Eventually one brave soul raised his hand and said, *Fine, I'll do it. It has to be done—let's see how fast we can get this done.*

In ten weeks they doubled their productivity, discovered revenue opportunities they never would have before, and crushed some fifty-three impediments raised by the Teams getting in their way. A structural reorganization emerged as the goal shifted from output (making sure everyone was busy) to outcome (getting to done). And the results were remarkable. Their typical development life cycle was two and a half years; with Scrum they had two new products in six weeks. And they had customers, big ones, who wanted to buy their product right away. That's what focus can do. They went from a hundred projects not getting done to twelve that did—twelve projects that changed the fate of the division and impacted the stock price of their multibillion-dollar corporate parent.

Focusing on getting things *done* has impact. Just say no.

Don't Be Busy, Be Done

The human brain quite literally cannot multitask. An especially apt example is that daily practice of multitasking: driving and talking on a cellphone. The research is very clear on this. People who drive while talking on cellphones—even those of the hands-free variety—get into more accidents than people who don't. The problem is especially alarming when you consider that, according to the National Highway Transportation Safety Administration, at any given moment 8 percent of people on the road are talking on their cellphone.

That is what multitasking has bequeathed us. Here's a quote from my favorite paper on the subject:

> Even when participants direct their gaze at objects in the driving environment, they often fail to "see" them when they are talking on a cell phone because attention has been directed away from the external environment and toward an internal, cognitive context associated with the phone conversation.[1]

People will actually look at an object—the car they're about to rear-end or the tree they're about to wrap their car around—and not see it. Yet we persist in driving and talking on the phone.

Whenever you attempt to do more than one thing at a time, you lose a huge amount of your productive capability to what's called context switching loss. There are studies that show that just answering an email can derail your brain for half an hour before you can get back in the right headspace for the work you are doing.

Consider this: How many times have you been interrupted while reading this book? How about while reading this chapter? Heck, did someone text you while you were reading this page? Did

you just look for your phone after reading that? Did you read the messages you've missed while reading this chapter? We live in a society that expects us to respond immediately to interruptions. If you don't respond to that email or Slack message or text right away, you are insulting the person who reached out and poked you, demanding your attention. How many times yesterday did you stop what you were doing with people right in front of you to respond electronically to someone not in the room? And at the end of the day, what you began still isn't finished, you still haven't gotten back to that other really important email, and there's that thing you were going to do that was really important but now you can't remember what it was. And then you turn to the task you were supposed to be working on but now you have no idea what you were thinking or where you were headed. And don't you need to go pick up the kids now anyway?

There's actually been quite a bit of research on this—people quite literally cannot multitask. Put people in fMRI machines and ask them to do a few things at once, and you see that their brains just can't handle it. But by making choices, by saying no, by prioritizing clearly, you can change your fate. That global manufacturer did. Confirmation.com did. The Stealth Space Company launched its first rocket. There are ways of surviving and even thriving in this rapidly changing world we live in. But it does require some real insights, and some real choices. The first one, as I'll explain in Chapter 5, is asking whether you are controlled by your fear or by your hope. *You* get to decide. Even if you think things are out of control. Even if you believe the forces acting on you and on your organization are immutable. Even if you're looking at the onrushing edge of a cliff, you get to decide what you're going to do about it.

It isn't always easy. But as you'll see, fear really is the mind-killer. And the antidote is connection.

THE TAKEAWAY

Admit you have a priority problem. If your strategy is to say nonsensically that everything is a priority, then what you are really saying is your strategy will be decided by any person in the organization who, given zero guidance on what is *actually* most important, decides what to do next.

Get to done. The key to getting to done is defining up front what done actually means. When a Team picks up an item from their Product Backlog, they should know the nature of done for that piece, but also how that piece builds on other pieces. Your architecture determines what your definition of done *can* be.

Don't confuse being busy with being done. Focusing on utilization as the metric keeps people busy, but it doesn't lead to anything actually being delivered. Don't focus on outputs; focus on outcomes.

Know the power of no. Corporate strategy typically focuses far too much on what a company will do rather than what a company won't do. Choices have to be made.

BACKLOG

- Write down all your priorities and put them on a wall in a single column. Order them in terms of value, risk, and effort. But remember, it is a single list and a single column. If you have multiple priorities at any stage, reprioritize your list.
- What is your definition of done? Write it down. Put it on a wall where you can see it every day.
- Come up with three ways you could better measure outcomes versus outputs. Find at least one person to ask about the impact your work is having or the amount you are doing.
- What is the architecture of your product? Is it tightly coupled or modular? Where could you insert a known stable interface so that breaking one thing doesn't break another?