Friday, November 23, 2012

Do No Harm - on Problem Solving and Design

What do medicine, politics, and computer programming have in common? Answer: they all involve problem solving in highly complex systems. Less obviously, so do nutrition, parenting, and riding a bicycle. In fact, much of life is about solving problems in complex systems. So then, why do people suck so bad at it?

You may contest that most people manage all right on a bicycle. But I provide bicycle riding as one of the simplest possible complex systems, and will show that relative to its simplicity, in terms of conscious problem-solving approach, people suck at that too. More usefully, I'll show exactly why, and how to do better. And, while we're at it, how to solve problems well in all those other domains too. (By extension, I will effectively show how to *design* quality complex systems that are robust and resilient as well.)

Strikingly, there is a single litmus test which neatly divides what I will call good problem solving from bad. Some of you may, even by the end of this essay, decide that I have those two terms reversed, but I will stick with them here for the sake of simplicity (and because I know who I want on my team). The litmus test is this:

When solving a problem, are you looking for a fix or are you looking for the cause?

A first reaction, no doubt, is that looking for a cause is part of looking for a fix. And while this is true in the most pedantic sense, it is not representative of what usually happens. In fact, people, institutions, and ideologies all tend to have a strong and consistent bias in one direction or the other, creating a sort of dichotomy that cuts across domains: the fixers and the solvers.

You might also recognize this as equivalent to the cliche "are you treating the symptoms or solving the problem?" which it is. But I avoid that phrasing because it's much too easy to conflate the symptom with the problem, or, as the flip side of the first reaction above, to say that treating the symptom *is* solving the problem. E.g.: if you have a headache, the problem is that your head hurts, and an aspirin solves it. Whereas this should be: the problem is that your head hurts, and an aspirin will *fix* it, but it does not explain the *cause* of the problem.

In case it's not obvious yet, I equate the solvers with good, and the fixers with bad, and the rest of this essay is mainly about why--and why it's so much more significant than most people realize.

In order to understand that, we need to look at another important concept, which is the golden rule of complex system design (or, indeed, evolution, since this principle is so important that it is a statistical prerequisite to any working system):

For a complex system to be resilient, it must (in any practical sense) be comprised of a collection of simpler, resilient parts.

The emphasis here is not on the value of modularity, which is commonly understood, but on the vital role of *resilience* in this modularity. Resilience means that when you apply an external force or permutation to a thing, it responds by bouncing back rather than by falling apart. Or, in other words, it means the thing is *self-regulating*.

Nature provides endless examples of this principle at work. The human body alone is a glossary of it. Pick any part at any scale and study it and you will find way after way that it is self-regulating. Each part strives to provide its function as unperturbed as possible over a wide range of conditions. Bones strengthen themselves where they are most stressed. The retina adjusts its own sensitivity in response to light. The heart will keep beating at around 50 beats per minute even if the brain is completely shut down. Every cell is a city of industries devoted to maintaining various equilibria. Blood sugar is kept within a range by a multi-faceted and distributed subsystem. The common theme is this:

The aim of a resilient part is to normalize (make consistent) the things that matter, and to minimize (dampen or hide) the things that don't.

The need for this being that when you put a bunch of parts together, their interactions are combinatorial, or mutually-amplifying. Ten parts with ten states each have ten billion possible states when combined. This explosion of possibilities is useful when it is part of the function, tamed by formulae constraining the states to desired relationships (specifically by factoring into orthogonal variables wherever possible, but that is a topic for another time). But it is death when applied to *exceptions* or variations--that is, to the many ways things can go wrong.

> To illustrate this, consider a simple evening at the opera. The plan is to have a babysitter arrive at six, to drive to the opera, and enjoy. This is the functional design. It is very simple, and it will work just fine--as long as nothing goes wrong. But what if the babysitter comes late? What if the car runs out of gas? What if you forget the tickets? What if the babysitter comes late and the car runs out of gas? What if the babysitter comes early but the car breaks down and you forget the tickets?
> Obviously we can go on like this forever, and some of the combinations will work out ok, but statistically speaking most of them will not. Planning contingencies for each and every possible failed scenario is not a practical solution.

> So what do we do if we absolutely must make it to the opera? Answer: We hire Professionals for each step who will make sure their part is done right whatever it takes. We find a babysitter who is tracked by GPS from a central office that will dispatch an alternate if there's any chance she can't make it on time. We hire a limo

> company that keeps a helicopter ready in case the limo breaks down or gets caught in traffic. We book our tickets electronically with an agency that meets us at the door and holds spares in case of snags. In short, we make each part handle the exceptions *internally* so they can't combine, and so that we can center our plan (design) as much as possible around its positive function. In short, we build the resilient whole from resilient parts.

The good and the bad of this is that the resulting design is deceptively simple. The way it *works* is often reasonably apparent, whereas the way it *fails* (when it fails *despite* all of this modular resilience) often seems to mysteriously arise from complex interactions beyond scrutiny.

The fixer approach, whether consciously or not, is emboldened by the apparently comprehensible functional design to make direct modifications there. Even before failures arise, it seems reasonable to improve things with a little tweak here or there. And when failure of non-obvious cause does arise, the fixer generally declares it beyond our abilities to discern causality within such a complex system, and sees the only sane approach to be controlling the things at the level we do understand. The resilience of the underlying system often accommodates these fixes to a point, initially vindicating the approach.

But here is the quandary: Any modification to the system is altering the *design* of the system, whether intending to or not. If you cannot understand the system well enough to see the cause of failures that are already happening, why do you think you will be able to see the consequences of your changes, let alone predict them in advance? That is:

If you can't solve an existing problem, how do you know you aren't causing more?

The trouble with most direct fixes is that they obviously improve function at the non-obvious expense of resilience. That is, the naive approach to design (the fix) does not explicitly model and contribute resilience, but rather takes it for granted, and in most cases actually *consumes* it, for reasons I will get to. That resilience is in a sense a shared, limited resource: If one part is slightly out of kilter, its resilient neighbors can often pick up the slack. But that leaves them less elbow room, making them less resilient in turn. If the babysitter is late, the limo can drive faster, and you get to the opera on time--but not if there is also heavy traffic (a condition which could normally be handled).

It can be confusing enough when problems propagate like dominoes, but worse is when they fall short of causing immediate failure, but sap global resources through their constant strain on resilience. If the babysitter is regularly late (let's say because you "improved" the system by having her stop at the donut shop on the way) the limo company may have to frequently call in the helicopter to make sure you get to the opera on time. From a functional standpoint, the new improved system is working great--it's all donuts and opera. That is, until it fails catastrophically when your bank balance unexpectedly runs dry. This is a common pattern of perplexing failures: A rogue function producing bad values, causing an error handling routine to fill up your disk with a log file and ultimately crash your computer; a low-fat diet leading to eventual insulin resistance; and so on. A resilient system tries to keep all of its parts in the middle of their preferred operating ranges, in order to maximize flexibility, and minimize strain and rigidity.

Which brings us to why the naive fix almost always consumes resilience: Recall, "the aim of a resilient part is to normalize (make consistent) the things that matter, and to minimize (dampen or hide) the things that don't". Both of those are forms of *absorbing variation*, akin to dissipating energy, which requires the flexibility to move with the variation in the first place. A supple tree moves with the gusts, while a stiff one snaps or uproots.

Resilience invariably relies upon feedback loops, and so the variables involved must be free to move through some critical range or the feedback is broken and the resilience is lost.

The naive fix, aimed at altering some visible behavior, typically in one way or another *adds constraints* to the system, in order to directly inhibit variation or push the system away from its default equilibrium. This is the optimistic approach, in that it focuses (in theory) on the way the system works under ideal conditions, and leads directly to constructive solutions. But it also makes the system more rigid and brittle, so that even if the fix works, the system is now strained and problems are more apt to crop up (here or elsewhere). Furthermore, in practice analysis on this level often simply gets it wrong entirely, because resilient systems generally do not abide by direct manhandling of their variables (resilient systems are self regulating!), so that the obvious fix often gives no improvement at all, yet with many hidden costs. The fixer usually responds to this by doing more of the same--if the nail isn't budging, must be the hammer just isn't big enough... Layer upon layer, the fixer creates a self-fulfilling prophesy: the resulting system, with its rigid connections, no longer obeying the rule of modular resilience, does in fact exhibit behaviors so chaotic and complex as to be beyond scrutiny...

The solver, by contrast, starting from a more pessimistic stance, sees complex systems as being inherently fragile, and so treads as lightly as possible. Do No Harm means more than not adding harm, it means recognizing the harm you are already doing and stopping it. It means attending as much to the resilience of every component of the system as to its function--a balance which requires a great deal of *finesse*. The solver reels at the ham-fistedness of the fixer, knowing in their gut that the fixers' ever increasing latticework of rigid patches will invariably snap and collapse, or at the very best die a slow death of becoming too rigid to serve its intended function at all. The solver sees chaos as a problem of design, and wants to peel back the fixes one by one, to cure the system by reviving its resilience.

Obviously problem solving can't be limited to only undoing what's been done (although in the modern world it seems 95% of the problems we face are best addressed this way). But in the rare constructive case, once the cause is understood (or once the model is understood, if we are working to improve it) the solve is often easier than the fix. Self-regulating systems are ultimately defined by their feedback coefficients, so often it is a matter of finding those, however many levels back they may be, and re-tuning them. (In economics, for instance, incentives are key.) Similarly, extending a system is a matter of solving for the graceful way to change the parameters of interest, and then adding a mechanism that automates that feedback. Which brings us to...

How to steer a bicycle:

Assuming we're coasting gently down hill to keep things simple, let's look at how we steer and how we stay upright. Following the above principles of design, the process is comprised of multiple self-regulating subsystems (implemented in your brain with the aid of your arms and inner ears). The most critical subsystem is the balance module, which works by turning the wheel toward the side we are leaning relative to our desired lean. Next is the steering module, which works by increasing or decreasing the lean (via the balance module) until the desired rate of turning is met. In a sense the balance module is a resilient subsystem of the steering module, which in turn employs it via a self-regulating feedback loop, thus creating a resilient aggregate system. But in any event, each module is relatively simple in its implementation.

The aggregate system's behavior is, by contrast, counter-intuitive at first. If you are heading straight, and you want to turn right, the steering system requests a rightward lean from the balance system, which steers the wheel *left* in order to induce it, reducing the turn as the lean is achieved, and then turns the wheel right enough to maintain equilibrium of that lean vs. the centripetal force induced by the current speed of forward travel. Thus to turn right you start by steering left. Furthermore, each minute adjustment that is made to correct your course begins with counter-steering against that correction. Thus, quite a complex strategy emerges from the combination of just these two very basic modules.

Most people do manage alright on a bicycle, but only because their conscious minds have lost a battle of attrition with their smarter intuition. When they were taught to ride, they were probably told "steer the wheel right to go right", or the equally nonsensical "lean your body to turn" (in fact, leaning your body to turn is possible, as with riding hands-free, but the true mechanism there is bending at the waist, not leaning, and it similarly requires initial counter-leaning). The consequence that follows is a nice illustration of what happens when you "fix" a problem instead of solving it:

Let us assume that with some practice your balance system is working and you are able to stay upright on the bike, but you have not yet learned to go where you want. You apply rational deduction to the most obvious mechanical aspects of the bike and conclude with some certainty that in order to go straight, you need to keep the front wheel straight. Your "problem" seems to be simply that you are (for some reason) turning the wheel somewhat randomly rather than pointing it in the desired direction. The Fix is obvious, so with some extra will power, or if necessary a clamp, you keep that front wheel straight as an arrow. And low and behold, it works! You're going straight finally. Although for some reason you seem to have forgotten how to stay upright... But no matter, there's a fix for that: training wheels. And hey, now you can remove the clamp, and drive around your trike, and everything is working great.

And there we have a fixer outcome, which not only failed to learn (or design) the simple, resilient model for steering (which starts with understanding why you were steering "wrong" in the first place), but actually undermined the working balance system you once had (through a cascade of "fixes"). So what's the problem, you ask? It's a working solution, after all. True, but only within a very narrow domain; it is a *brittle* solution. Try to turn very fast, and over you go, because your system is rigid rather than flexible and self-regulating. Next thing the fixer is building a race track with all of its corners carefully graded with the right lean to accommodate this precarious bike within a very narrow speed range, and then comes the speed regulator to ensure the bike never goes too fast or too slow for the track...

This cascade of consequences is the hallmark of the fixer approach, a sort of whac-a-mole game played with hydras where two heads pop up for every one you knock down. And because the fixers, considering the system too complex to analyze, denounce asking "Why?", their eventual call is "who could have known this would happen?". Better fix it! The solvers, meanwhile, are the derided minority saying "we told you this would happen..."

Note particularly the manner of the first major fixer fail above. The goal is to steer the bike where we want it to go, and the mechanism for doing this is (seemingly!) so clear and obvious. Forcing the result we want *does* work--the bike goes where we steer it. But by disallowing any deviation from our ideal, we have removed the elbow room which another subsystem (balance) was using to self-regulate, and as a consequence that seemingly unrelated subsystem *fails completely*. Resilience starts with the freedom to err, and even if that error is imperceptible in a well-tuned system, it remains vital to its operation. If you have ever had the misfortune to ride your bike too close to a curb and had your front wheel slide up along it, you know what happens when those imperceptible deviations from perfection are eliminated.

The bottom line is: there is no escaping the need to make best effort to understand the whys and wherefores. To skip this and go straight for the fix is not humbly admitting the system is too complex to understand, it is arrogantly assuming we understand it well enough to fix it without breaking it. The humble thing to do is less, not more--to respect the difficulty in keeping any complex system stable, and the degree to which doing so relies upon a best effort by, and freedom of, each component to self-regulate.

Postscript:

I mentioned this issue cuts across domains, and I have noticed some correlations which I think it well explains. Paleo diet, cross-fit, Libertarianism, Sudbury*, EC*, and no-till farming* and related*, are examples of things that, by and large, embody the solver approach, and hence all tend to appeal to solver personalities. Prescription drugs, packaged diets, minimum wage/price controls, prescribed curriculums, GMOs/Roundup, mainstream politics, and pretty much our entire body of laws are all examples of fixes. Fixers are not generally enthralled with the outcomes of their strategy, but they continue to use it (and vote for it) anyway.

I would estimate that 90% of the population are fixers. The world in a nutshell.

[2012/12/2 Addendum: Many folks have sent me pointers to Nassim Nicholas Taleb's book Antifragile: Things That Gain from Disorder which I haven't read but looks very related.]

[2017/1/5 Addendum: See Down Under Minimum Wage]