Editor: Charles Petrie • petrie@stanford.edu

The Failure of HealthCare.gov Exposes Silicon Valley Secrets

Charles Petrie • *University of St. Gallen, Switzerland*

Your jaw will drop when you see how the "tech surge" team fixed HealthCare.gov!

Well, okay, now that I've got your click, I'll open with the lead that HealthCare.gov didn't really get fixed. And it's even worse than it appeared. Here's the story from my good friend Robert Kennedy, who was part of the team that made this critical, Web-based registration service work well enough that the enrollment period ended successfully, after starting from a complete debacle.

You can see Kennedy's talk on his experience at www.youtube.com/watch?v=GLQyj-kBRdo. He shared some more details of the shockingly ill-constructed system with me over drinks one evening, but let me convey here his high-level points and my take on them. I'll just mention one or two anecdotes that motivate these.

One you can see on the video, but it's just so good I have to repeat it here, as a teaser. Kennedy met with an employee of one of the site contractors who was in charge of software "security." When he asked her which penetration threats she had prepared for, she responded that she didn't think about such things: she was in charge of "risk assessment."

In contractor terms, that meant the risk of government contract nonfulfillment. Her job, as she saw it, was to ensure that a software audit confirmed that the software was "secure" as defined in the contract.

If you're a Silicon Valley engineer (or any other good engineer), your jaw really did just drop. To be fair, perhaps someone else was in charge of penetration testing, but the overall security objective seemed to be ensuring that the contractual terms were met, rather than that the system was actually difficult to hack.

Then, there was the firewall in the wrong place. Oh wait, I can't write about that because the fix would be worse than the problem. I think I can say that something bad probably happened because all of the IP addresses were hard-coded and there was no DNS. That should be enough for you. But there were a *lot* of not just badly designed but guaranteed-to-fail subsystems, often at critical points.

Here's a nice, top-level piece of brokenness: until the tech surge, no monitoring was in place for this nationwide critical distributed system. After the tech surge, there was an operations room for contractor representatives, each with his or her own monitor, because there were no common APIs so that everything could be tied together. To run the system, you had to go to each person and ask if everything was okay.

Kennedy has some good points about how this happened. But I'm going to focus here on just the cultural aspect he also addresses: no one took, or wants to take, responsibility.

Playing It Safe

To illustrate, one day, there was a critical failure, and Kennedy asked the operations room, "Did anyone do anything at 10:48?" Silence. "Is your database getting any requests?" "No." Both the nonanswer and the answer were wrong.

At Google, where Kennedy and many others in the tech surge happen to work when they aren't on unpaid leave being heroes, any engineer in this situation would have answered the first question with, "I reset permissions at 10:48. It looks like I did it wrong. Can someone help me please?" And a Google engineer who was competent but perhaps not fully up to speed on the system would answer the second question with, "I don't know how to tell whether my database is receiving any requests, but it isn't processing any requests. Can someone help me please?" Both would take ownership, and both would get good help right away, from which they could learn. Failure is okay in Silicon Valley, and cooperation and help is normal. Letting a system continue to function poorly is neither okay nor normal.

Published by the IEEE Computer Society 1089-7801/14/$31.00 © 2014 IEEE

In Washington, DC, as in many (but not all) places outside the Silicon Valley bubble, it's just the opposite. You're fired immediately for mistakes, as was the hapless engineer who didn't speak up about what he did at 10:48. The database engineer didn't dare ask for help (even if he'd had the training to ensure he knew the difference between receiving and processing requests), and it probably would have been a long time coming. You don't volunteer to help because you don't take any more responsibility than you have to: you always take the safest way out.

The safest way is often to give a contract to a name-brand corporation for a piece of software that purports to perform the task that's your responsibility. "You don't get fired for buying IBM" was the mantra in the mainframe era. (Actually, I know a counterexample.) You don't care if you purchase a product wrong for the job if it's a brand name and meets software specifications. You happily use an Oracle Lightweight Directory Access Protocol, designed for internal teams with complex permissions, for large-scale, relatively simple external client logins. And you certainly don't care how it fits with everyone else's software.

No one had the job of making everything work together. No one was in charge of seeing that the system was production quality. There were only many people in a vast bureaucracy playing it safe.

Learning from Silicon Valley

Many companies come to Silicon Valley to learn how it works. They rarely do. It's a unique ecoculture that could possibly be replicated, but almost never is. It runs counter to the existing cultures. I had a friend who became CEO of the Silicon Valley institute of a large German automotive company. They hired him to run it like a Silicon Valley company. Every time he implemented a new policy, he was called back to Germany to be chewed out by an executive for doing something counter to company policy. He finally quit.

Many people in the government — most notably, the administration's CIO — weren't consulted about the HealthCare.gov system in the first place; they know what needs to be done and are now taking some measures. They are fighting an established culture. I give them poor odds.

But if they want to succeed, they can take some lessons from Silicon Valley:

- Organize in small tiger teams with overall objectives.
- Reward initiative and risk-taking.
- Reward cooperation.
- Reward information sharing.
- See failure as learning, and even an opportunity.

Many companies come to Silicon Valley to learn how it works. They rarely do. It's a unique ecoculture that could be replicated, but almost never is.

Most important, everyone should realize that form doesn't matter. What matters is whether the whole thing actually works, and realizing what "working" means. This might be too much to ask. Instead, we're getting a kind of emergency topical antibiotic to address the ravages of the disease.

Mikey Dickerson, who led the tech surge in the winter of 2013, has quit Google and has been hired by the administration to be the deputy CIO and leader of the US Digital Services Team.[1] This deliberately small team will try to anticipate issues and rove the government, fixing acute problems. Kudos to Dickerson for taking on this task, and to the administration for taking drastic action, just before the fall HealthCare.gov enrollment period.

The HealthCare.gov site was patched together to run if carefully attended to by the tech surge team. Perhaps the new team can make it work again during the next enrollment period. But the underlying broken architecture and supporting dysfunctional culture will persist for a long time.

Reference

1. M.D. Shear, "White House Picks Engineer from Google to Fix Sites," *New York Times*, 11 Aug. 2014; www.nytimes.com/2014/08/12/us/politics/ex-google-engineer-to-lead-fix-it-team-for-government-websites.html.

Charles Petrie teaches and coaches the topic of innovation in design thinking at the University of St. Gallen, Switzerland (http://dthsg.com/dt-at-hsg/). He retired as a senior research scientist from the Stanford University Computer Science Department. His research topics are concurrent engineering, enterprise management, and collective work. Petrie has a PhD in computer science from the University of Texas at Austin. He was a founding member of technical staff at the MCC AI Lab, the founding editor in chief of *IEEE Internet Computing,* and the founding chair of the Semantic Web Services Challenge. He also manages the Black Rock City Municipal Airport 88NV. Contact him at petrie@cdr.stanford.edu.